Advances in Intelligent Systems and Computing 511

Jan Janech
Jozef Kostolny
Tomasz Gratkowski *Editors*

# Proceedings of the 2015 Federated Conference on Software Development and Object Technologies

Springer

# Advances in Intelligent Systems and Computing

Volume 511

**Series editor**

Janusz Kacprzyk, Polish Academy of Sciences, Warsaw, Poland
e-mail: kacprzyk@ibspan.waw.pl

*About this Series*

The series "Advances in Intelligent Systems and Computing" contains publications on theory, applications, and design methods of Intelligent Systems and Intelligent Computing. Virtually all disciplines such as engineering, natural sciences, computer and information science, ICT, economics, business, e-commerce, environment, healthcare, life science are covered. The list of topics spans all the areas of modern intelligent systems and computing.

The publications within "Advances in Intelligent Systems and Computing" are primarily textbooks and proceedings of important conferences, symposia and congresses. They cover significant recent developments in the field, both of a foundational and applicable character. An important characteristic feature of the series is the short publication time and world-wide distribution. This permits a rapid and broad dissemination of research results.

*Advisory Board*



More information about this series at http://www.springer.com/series/11156

Jan Janech · Jozef Kostolny
Tomasz Gratkowski
Editors

# Proceedings of the 2015 Federated Conference on Software Development and Object Technologies

Springer

*Editors*
Jan Janech
Faculty of Management Science and Informatics
University of Zilina
Zilina
Slovakia

Jozef Kostolny
Faculty of Management Science and Informatics
University of Zilina
Zilina
Slovakia

Tomasz Gratkowski
Institute of Metrology, Electronics
Zielona Góra
Poland

ISSN 2194-5357 ISSN 2194-5365 (electronic)
Advances in Intelligent Systems and Computing
ISBN 978-3-319-46534-0 ISBN 978-3-319-46535-7 (eBook)
DOI 10.1007/978-3-319-46535-7

Library of Congress Control Number: 2016959397



Printed on acid-free paper

This Springer imprint is published by Springer Nature
The registered company is Springer International Publishing AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

# Preface

The second annual international scientific conference SDOT 2015 (Federated Conference on Software Technologies and Object Technologies) was held at the premises of the University of Žilina, Faculty of Management Sciences and Informatics, Slovak Republic in November 19, 2015. The conference was established as a joint event of two predecessor local conferences—Software Development (formerly Programming) and Objects conferences.

Let us briefly recall their history. Previously for a longer time both of them were part of Software Development conference, which reached its 41st year this year; however, it should be recalled that Software Development Conference is directly connected to its predecessor—Programming Conference. The Programming Conference reached top with maximum participants between 1985 and 1995. The years 1995 to 2005 were associated with the conversion of the original practice-oriented conference, i.e. from IT data centers attended mainly by experts to academic conference attended mostly by scientists and Ph.D. students from universities. In such form this continued until 2013, and then the joint conference SDOT began.

Objects Conference was founded in 1996. The topics of this conference were directly associated with the beginning and development of object-oriented programming. For the first seven years, this conference was held at the premises of the Live Sciences University—Prague, with interested participants from both academia and industry. Since 2003, the conference has been organized by organizers from various universities, both in the Czech Republic and Slovakia.

This year's host of SDOT conference was the University of Žilina, Faculty of Management Sciences and Informatics. Several factors can be considered as a great success this year. First, the conference has become truly an international event both in terms of the amount of foreign contributions and in terms of composition of steering and scientific committees. Furthermore, the number of participants was twice more than last year; hence, it was possible to select only high-quality contributions.

Finally, I would like to thank the team of organizers from the University of Žilina who worked under the direction of Ján Janech for the excellent preparation

of the conference and its organization. Also thanks to all others who contributed to organizing the conference, particularly members of the steering and scientific committees. I believe that all conference participants enjoy this year's SDOT 2015 conference and that their participation in it will bring an inspiration in their scientific and professional works.

On behalf of steering, scientific and organizing committees,

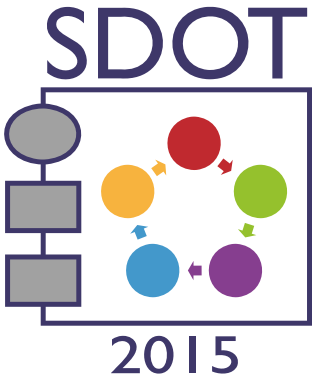
November 2015

Martin Molhanec
Chair of steering committee of SDOT 2015

# Organization

**Proceedings of the Federated Conference on Software Development and Object Technologies 2015**

**November 19–20, 2015, Žilina, Slovakia**



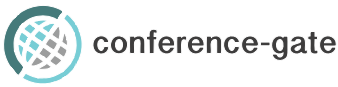

**Organized by**


## Under Patronage


## Scientific Partners








## Steering Committee of the Federated Conference on Software Development and Object Technologies 2015

### Chairman

| | |
|---|---|
| Martin Molhanec | Czech Technical University in Prague, Czech Republic |

### Members

| | |
|---|---|
| Michal Bejček | College of Information Management, Business Administration and Law, Prague, Czech Republic |
| Ján Janech | University of Žilina, Slovakia |
| Branislav Lacko | Brno University of Technology, Czech Republic |
| Jan Voráček | College of Polytechnics, Jihlava, Czech Republic |

## Scientific Committee of the Federated Conference on Software Development and Object Technologies 2015

### Chairman

| | |
|---|---|
| Ján Janech | University of Žilina, Slovakia |

### Members

| | |
|---|---|
| Anna Bobkowska | Gdańsk University of Technology, Gdańsk, Poland |
| Alena Buchalcevová | University of Economics, Prague, Czech Republic |
| Vladimír Burčík | Robert Morris University, USA |
| Jerzy Buriak | State University of Applied Sciences in Elblag, Poland |
| Constantina Costopoulou | Agricultural University of Athens, Greece |
| Alexandre Fonte | School of Technology of the Polytechnic Institute of Castelo Branco, Portugal |
| Hartmut Fritzsche | Hochschule für Technik und Wirtschaft Dresden, Dresden, Germany |
| František Huňka | University of Ostrava, Czech Republic |
| Igor Jurisica | University of Toronto, Canada |
| Wojciech Korneta | PWSIP Lomza, Poland |
| Tomáš Kozel | University of Hradec Králové, Czech Republic |
| Emil Kršák | University of Žilina, Slovakia |
| Russell Lock | Loughborough University, UK |
| Vojtěch Merunka | Czech Technical University, Czech Republic |
| Boris Misnev | Transportation and Communication Institute Riga, Latvia |
| Paul Newton | IBM Software Development, USA |
| Maria Ntaliani | Technological Educational Institute of Sterea Ellada, Greece |
| Artur Opalinski | Gdańsk University of Technology, Gdańsk, Poland |
| Rudolf Pecinovský | University of Economics, Prague, Czech Republic |
| Robert Pergl | Czech Technical University in Prague, FIT, Czech Republic |
| Tomáš Pitner | Masaryk University, Brno, Czech Republic |
| Krešimir Pripužić | University of Zagreb, Croatia |
| Jaroslav Ráček | IBA CZ, s.r.o., Czech Republic |
| Karel Richta | Czech Technical University in Prague, FEL, Czech Republic |
| Tomáš Richta | College of Polytechnics, Jihlava, Czech Republic |
| Victor Romanov | Russian Plekhanov University of Economics, Russia |
| Radi Romansky | Technical University of Sofia, Bulgaria |
| Petr Šaloun | VŠB-Technical University of Ostrava, Czech Republic |

| | |
|---|---|
| Antonín Slabý | University of Hradec Králové, Czech Republic |
| Václav Snášel | VŠB-Technical University of Ostrava, Czech Republic |
| Michal Valenta | Czech Technical University in Prague, FIT, Czech Republic |
| Miroslav Virius | Czech Technical University in Prague, FJFI, Czech Republic |
| Michal Vopálenský | College of Polytechnics Jihlava, Czech Republic |
| Kaloyan Yankov | Trakia University, Stara Zagora, Bulgaria |
| Krzysztof Zieliński | AGH Kraków, Poland |

## Scientific Committee of Workshop on Design and Analysis of Embedded Systems

### Chairman

| | |
|---|---|
| Tomasz Gratkowski | University of Zielona Góra, Poland |

### Co-chairmen

| | |
|---|---|
| Michał Doligalski | University of Zielona Góra, Poland |
| Jacek Tkacz | University of Zielona Góra, Poland |

### Members

| | |
|---|---|
| Grzegorz Borowik | Warsaw University of Technology, Poland |
| Arkadiusz Bukowiec | Orbis Software, UK |
| Luis Gomes | Universidade Nova de Lisboa, Portugal |
| Wolfgang Halang | University of Hagen, Germany |
| Zbigniew Huzar | Wrocław University of Technology, Poland |
| Ka Lok Man | Jiaotong-Liverpool University, China |
| Joao Monteiro | University of Minho, Portugal |
| Ann Pławiak-Mowna | University of Zielona Góra, Poland |
| Alfredo Rosado-Muñoz | University of Valencia, Spain |
| Rybski Ryszard | University of Zielona Góra, Poland |
| Bernd Steinbach | Freiberg University of Mining and Technology, Germany |

## Organizing Committee of the Federated Conference on Software Development and Object Technologies 2015

### Chairman

| | |
|---|---|
| Viliam Tavač | University of Žilina, Slovakia |

### Members

| | |
|---|---|
| Iveta Belošovičová | University of Žilina, Slovakia |
| Miroslav Gábor | University of Žilina, Slovakia |
| Tomasz Gratkowski | University of Zielona Góra, Poland |
| Ján Janech | University of Žilina, Slovakia |
| Jozef Kostolný | University of Žilina, Slovakia |
| Emil Kršák | University of Žilina, Slovakia |
| Matej Meško | University of Žilina, Slovakia |

# Contents

# IoT-Based Smart Monitoring System Using Automatic Shape Identification

Stanisław Deniziak, Tomasz Michno, and Paweł Pieta(✉)

Department of Information Systems Division of Computer Science,
Kielce University of Technology, Kielce, Poland
{s.deniziak, t.michno, p.pieta}@tu.kielce.pl

**Abstract.** In conventional monitoring systems video previews from stationary cameras are overseen only by a human supervisor, who may easily overlook alarming events recorded by a camera. Because surveillance system must be reliable, its capabilities can be improved by applying computer vision algorithms to a video signal in order to detect objects in an automated fashion. Also its autonomy can be extended by the use of mobile robots capable of monitoring tight and occluded areas and by the use of smart cameras with integrated embedded systems. In this paper we introduce an architecture of the autonomous monitoring system based on object shape detection. Our approach is conformable with the concept of Internet of things. It consists of the set of smart objects with video sensors, controlled by the Shape Identification Cloud. Our work is aimed at building the real-time system efficient at reliable recognition of objects on the basis of their approximate shape and with the option to be used as a web service in a cloud. To monitor the environment the system uses mobots equipped with video sensors as well as surveillance cameras capable of remote position control. For object identification task we use the Query by Shape (QS) method which decomposes objects into simple graphical primitives like lines, circles, ellipses etc. and then it identifies them in a shape database.

## 1 Introduction

Traditional monitoring systems mainly consist of a large number of video cameras deployed within an entire environment. These cameras are wired to a central control room where previews from all of them are available for inspection by an operator who supervises such a system. But the process of analyzing simultaneously multiple video feeds in real-time is a very challenging and discouraging task for a human being. The operator might easily overlook an important event recorded by a camera due to temporary distraction or inattention caused by an external factor, exhaustion, or simply because of concentration on the image from a wrong camera etc. The system constructed that way, with the human supervisor trusted with a task of identification of alarming events, might be highly unreliable.

Let's consider a few methods of improving monitoring system capabilities. First of all, computer vision algorithms can be applied to a video signal. Individual video frames can be analysed by means of digital image processing and a fully automated detection of new objects in the environment can be implemented (e.g. intruders,


J. Janech et al. (eds.), *Proceedings of the 2015 Federated Conference on Software Development and Object Technologies*, Advances in Intelligent Systems and Computing 511,
DOI 10.1007/978-3-319-46535-7_1

burglars, hazardous events such as fire etc.). For every detected object an alert is sent to the operator to his terminal, which enormously simplifies and enhances his work. The whole system becomes significantly more reliable, provided it is well calibrated and algorithms used for object detection and recognition are of high precision. Secondly, in order to extend autonomy of the monitoring system, different kinds of sensors and devices can be used, for example mobile robots and smart cameras. The high cost of the system and its maintenance comes mainly from the amount of hardware needed to be deployed in the environment (e.g. cameras and wiring), especially when the area being monitored is cluttered, which would require a significant number of conventional cameras to be employed. Instead of installing more cameras to ensure that the environment is left without blind spots, which would considerably increase the price of the system, the mobots can be used to substantially expand system ability to detect objects in tight areas. Mobile robots equipped with video sensors can be sent out on patrol, and because they can move freely through the environment, they can drive directly to those occluded areas and perform object detection on-the-go. Then they can communicate their findings to the control system through a wireless network. Another example of distributed processing is the use of smart cameras which are equipped with integrated embedded systems capable of automatic motion detection.

Lastly, the monitoring system might also be implemented as a web service in a cloud, which entails several new capabilities. Such a system could be used simultaneously by many different users from several remote locations through a simple web interface. For instance, it could be utilized by smaller companies which own some imaging devices capable of monitoring the environment, but do not necessarily possess nor want to invest at the time in the infrastructure needed to build a proper surveillance system with object detection and identification modules. In addition, the cloud provides easy scalability of the system, also in terms of the cost. Moreover, web interface can also be implemented as an application available from a mobile device.

The monitoring system presented in this paper is a generalization of the system proposed in [1] and it is focused on services in the cloud for Internet of things. Also this paper builds upon our previous work regarding shape identification problem [2].

The paper is organized as follows: in Sect. 2 the research done in the area of autonomous surveillance systems as well as object detection and querying of multimedia databases is presented. Section 3 summarizes the goals of our work and highlights the main characteristics of the proposed system. Section 4 introduces a concept of the autonomous monitoring system and the Shape Identification Cloud. Section 5 describes the main idea of our approach regarding the object recognition task and Sect. 6 presents the experimental results of our method. Section 7 concludes the paper and indicates further development directions.

## 2 Related Works

During the last two decades the task of building and assessing performance of autonomous surveillance systems has been an active area of research involving many scientists. The continuous increase in computational power has allowed them to deal with the problem of designing reliable monitoring systems through the use of more and more

advanced computer vision algorithms. An overview of technical development of intelligent surveillance systems is presented in the work done by M. Valera and S.A. Velastin in [3], another review is shown in the paper of P. Kumar et al. in [4]. The first generation of monitoring systems consisted of analogue CCTV (closed-circuit television) cameras placed in several remote locations and connected to a set of monitors located in a single control room. Human supervisor's role was only limited to investigating events which have already happened, provided they have been recorded. The invention of digital cameras and high performance computers have allowed to construct the second generation systems. Computer vision algorithms were applied to analyze individual video frames and semi-automatic systems, capable of real-time proactive detection and tracking of alarming events, were developed. The third generation focused on improving precision by acquiring more accurate information from multiple different sensors (e.g. fixed, active and smart cameras, mobile robots etc.) and integrating it by means of the technique known as Data Fusion. It was also aimed at the design of huge, distributed and heterogeneous systems for vast environments. Because of a fully automated understanding of a video signal, as well as distribution of processing capacities over the network and the use of embedded signal processing devices, a single human operator was finally able to efficiently supervise such large and complex areas.

Work done by A. Hampapur et al. in [5] characterizes three types of architectures for smart monitoring systems. The first one, which was described there, is the Basic Smart Surveillance Architecture (BSSA). A video signal from fixed video cameras is digitally recorded by the Digital Video Recorder and simultaneously processed by the Smart Surveillance Server, producing a rich video index and real-time alerts (which types and parameters are user configurable). The cameras are wired to a central location because all processing is centralized. The second type of the monitoring system is the Active Smart Surveillance Architecture (ASSA). It extends the previous one by adding the Active Camera Server which investigates images from active cameras. It not only tries to understand the situation in the scene, but it also attempts to selectively pay more attention when and where it is needed. Every time an event of interest is detected, Active Camera Server is able to reposition (move or zoom) active cameras in order to monitor it closely. The third characterized architecture is the Distributed Smart Surveillance Architecture (DSSA). In order to minimize the cost of system installation and its infrastructure, smart cameras can be deployed in the environment. They can analyze a video signal with an on-board embedded system and are able to communicate with a Distributed Camera Coordinator through a wireless network. Substantial cost savings come from reduced components needed to build the system, i.e. less wiring required to transfer video signals to the main server. A large number of examples of commercial monitoring systems are depicted in [3, pp. 5–9].

Building blocks of an autonomous surveillance system and functions that they perform are characterized in [3–6]. Video processing pipeline can be decomposed into several main components:

1. object detection,
2. object recognition,
3. object tracking,
4. behavior and activity analysis.

Object detection can be further fragmented into more specific, specialized parts:

1. data fusion,
2. background modelling,
3. foreground/background (FG/BG) detection,
4. object extraction.

Workload analysis of a video monitoring system performed in [6] by T.P. Chen et al. provided a few interesting and important conclusions. The most computationally intensive module proved to be the FG/BG estimation which holds up to 95% of execution time. The next most significant was the object tracking component which occupies up to 20% of CPU time.

The object detection problem has been a subject of many researches that result in two groups of methods. The first group comprises methods based on algorithms for detection of geometrical shapes. They are mostly focused on very fast algorithms for detection of circles and lines like [7]. In this work a two step algorithm is described, known as Hough Transform, which consists of converting an image into parameter space and voting. The first step is based on the line (Hough Line Transform) or circle (Hough Circle Transform) properties. In the second step, the line or circle candidates with the highest number of votes are chosen. Another example is the algorithm described by Guido M. Schuster and Aggelos K. Katsaggelos [8]. They used the mean square error estimator and weights for detecting a circle, solving an optimization task.

The second group of researchers are focused on detecting more complex objects. Recent algorithms are based mostly on learning systems. One of the most significant works in this group was done by Viola and Jones [9]. They presented an algorithm based on cascades of Harr-like features. First, it uses more general ones, then the precision is increased in the following steps. As a learning classifier the AdaBoost was used. This algorithm is designed for the face and human body detection. The example of an algorithm which tries to detect different classes of objects could be the [10]. In the proposed object detection system, the multiscale and deformable part models are used. As a representation of the model object, a set of star-like HOG (Histogram of Oriented Gradients) based features are used.

All of the learning based algorithms in the second group provide ones of the most precise results, but need a long time to create learning set and learning process. Also some of them are time consuming and cannot be used in real-time. Moreover, adding a new class of objects is very complicated and may require performing the learning process again.

The alternative to learning base algorithms is the approach based on a database of images of all known objects. The object identification may be performed by querying the database using the image of the object. The system may be easily extended to identify new types of objects, simply by adding new images to the database. The querying of the multimedia database was also a subject of many researches. As far as it concerns the Content Based Image Retrieval (CBIR) methods, the low level and high level algorithms may be distinguished [11].

The low level algorithms refer to image features like lightness or color histogram, e.g. using L1 Norm for color histogram comparisons [12]. Sometimes a spatial domain representation is used [13] or a normalized color histogram [14]. Sivakamasundari et al.

proposed a CBIR framework which can be applied to retinal image retrieval for the identification of diabetic retinopathy [15]. As a preprocessing step, the Kirsch template is applied to image in order to detect blood vessels. Then the following image features are computed: energy, entropy, contrast, homogeneity, maximum probability, standard deviation and ratio of vessel to vessel free area. There were also attempts to use Error-Diffusion Block Truncation Coding for the image representation [16].

Because all this algorithms computes features for the whole image, they are not very effective. Most often objects are placed on different background in different surroundings. One of the idea to overcome this problem was based on dividing image into smaller parts, called regions, which connect similar pixels. Regions are connected into graphs and are compared with input image graph, e.g. using Maximum Likelihood estimation [17]. As an improvement of region-based graphs, the algorithm called iPURE [18] includes a new color image segmentation algorithm. It is based on region-growing approach, which incorporates elimination or modification of region edges as a result of testing contrast, gradient and shape of the region boundary. There is also a method which uses DCT for region representation and can be used for compressed images [19].

All mentioned CBIR algorithms require an image as an input during querying a database. This can be a problem because sometimes a user does not have any good sample image, or even do not know what kind of object he is looking for. This problem may occur in case when the operator will want to verify if a given type of object already exists in the database. One of the algorithms which tries to overcome this problem was presented by Kato et al. [20]. It is based on querying the database by a rough sketch drawn by a user. First, images preprocessed by edge detection method are converted into $64 \times 64$ pixels and then they are compared using $8 \times 8$ pixels blocks. Another algorithm was described in [21]. The work detects features like corners, symmetric arcs, ellipses and parallelograms in order to extract regions from grayscale fixed-resolution images. After detection, region map is created, storing each left and right links for each region. There are many differences with our method, for example as a query can be given any resolution color image, even without all relevant details. Also shape primitives are used for objects skeleton building, not extracting regions.

There are also algorithms which try to join both Keyword Based Image Retrieval (KBIR) and CBIR advantages in order to supply the semantic search, for example methods based on automatic image annotation and region based inverted file [22]. Firstly, to create the inverted file, all training images are segmented into regions containing objects, creating visual dictionary with annotations. Then a dictionary tree is being built and learned. Next, the file is created by assigning to each object images in which it is present. The algorithm also stores the relations between objects existing in the same images. As a query, the user gives the composition of annotation names e.g. a ‘red car’.

## 3 Motivation

The main aim of the monitoring system proposed in Sect. 4 is to recognize objects and then optionally perform some actions. Mobile robots, which are part of the system, have very limited computational capabilities. Therefore they cannot perform recognition step themselves, but they should delegate it to the remote server. Moreover, they may operate in an environment without reliable and fast network connection, which does not allow them to send whole frames for processing. On the other hand, the stationary cameras which are also a part of the system, are only able to send images. Therefore there is a need of a relatively fast object recognition method which will be able to handle both types of devices and communication schemas. In order to do so, we propose a dedicated multimedia database which as a result of the query will return object's class.

One of the most important problems is the proper representation of the searched objects. Most often there is a semantic gap between the low-level features of the detected image and its high-level semantic meaning [11]. The Keyword Based Image Retrieval algorithms represent image as a textual description, e.g. keywords. Most often they are not effective because they strongly rely on a subjective human based annotations and are very hard to use without human interaction. According to the proverb "a picture is worth a thousand words", much better results provide Content Based Image Retrieval algorithms [11]. Most often they are efficient for automatic image queries where a sample image is given as a query. Despite their good performance, they are still problematic for situations where there is no full knowledge about searched object.

We expect that a fully-fledged multimedia database should support CBIR queries in order to be used as a part of object recognition system. Because of the limited mobile robots resources, the searched pattern may not be available, thus the most suitable method for specifying the query will be a simple sketch, extracted from the captured image frame. Sketch retrieval methods usually are based on exact matching of complex sketches [20, 23, 24] or images. Due to limited computational power of mobile robots, queries should be represented by the simplified or approximated shape of the searched object. Summarizing, an efficient CBIR method should reliably find images containing objects specified as a simplified sketch. It should also recognize objects that are slightly distorted or incompletely specified.

The main motivation of our research is to provide a new CBIR-based object recognition method which is able to provide results for both an image and an object sketch. Since the method will search objects specified as an approximate shape, the result classification reliability should be defined by the precision coefficient which can be easily used to determine if it is correct, even automatically. Moreover, the results should be obtained with regard of a limited mobile robots environment, e.g. without too much data transfer.

To summarize, we motivate our work as follows:

- the surveillance system should represent the third generation of monitoring systems and should incorporate different kinds of hardware like smart cameras and mobile robots; also in order to reduce its cost and extend its autonomy, the DSSA architecture should be implemented,

- computer vision algorithms used for image processing should be relatively simple, i.e. mainly based upon edge detection, because mobile robots have limited computational capabilities,
- extracted objects should be represented with the use of approximated geometrical shapes like lines, circles, ellipses and rectangles,
- object recognition should be adjustable by some level of similarity,
- efficient work of the monitoring system in real-time is our first priority, even at the cost of accuracy; false positive matches are permitted, but a human supervisor must intervene in such situations to help the system with the object identification task.

## 4 Monitoring System Overview

Proposed architecture of the monitoring system is depicted in Fig. 1. From a hardware point of view it is composed of stationary surveillance cameras capable of remote position control (some of which may be smart cameras with integrated embedded systems carrying out automatic motion detection), mobile robots equipped with video sensors, a cluster server with a database and a terminal for a human supervisor. The cameras and mobots monitor the environment and for each video frame object extraction is performed (we are currently investigating the best algorithms suitable for this task, see Sect. 7). Mobile robots extract objects on their own, but for the cameras it can be done in the cluster. When a new object which does not belong to the known environment is detected, it is decomposed into features as described in Sect. 5 and it is sent to the cluster through a wireless network. In the cluster, the object identification is performed using the Query by Shape (QS) method [2] (for details also see Sect. 5). Each node in the cluster is devoted to a single mobot or camera, so all operations are being executed in parallel. The cluster is using two databases, one which stores information about familiar shapes, and the second one with new objects. If shape identification fails, the object is added to the second database and more accurate learning methods can be used to identify it, e.g. Support Vector Machines (SVMs). Also each time when a new object is detected, the notification with a picture of that object is sent from the cluster to the supervisor's terminal. Moreover, a special alert is raised when the object has not been identified. Despite the possible use of more advanced machine learning, an unidentified object can also be recognized and described by the operator and added to the database with familiar shapes. Furthermore, the human supervisor is able to make some decisions and communicate them through the cluster back to the cameras or mobots, e.g. he or she can reposition a particular camera, or order a mobile robot to perform some action (for instance drive somewhere, extinguish a fire or tease an intruder) [1].

The Monitoring System may be used in the Internet of things environment as a cloud service. Figure 2 shows the main idea. Mobots and cameras are defined as things and operators are clients. The proposed architecture extends greatly the possible usage of such a system, because it may be used for monitoring environment e.g. when a company have only cameras or other devices which are able to capture images and does not have a proper objects identification system or need it only occasionally. Moreover,

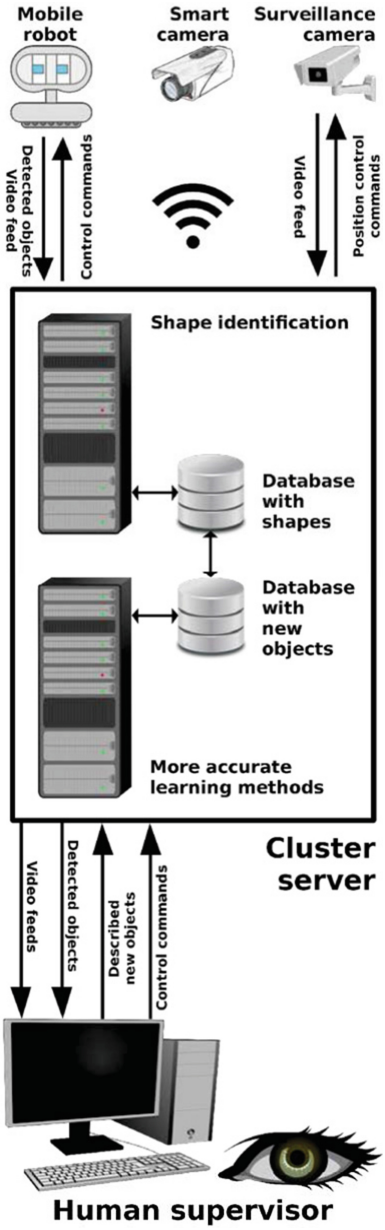



**Fig. 1.** Architecture of the monitoring system

there are situations when the number of capturing devices should be extended, e.g. during mass events, which implies the need of much higher computational capabilities of servers. In that situations the cloud system allows managing the cost of such systems more efficiently, which is very important for many companies. The cloud service may

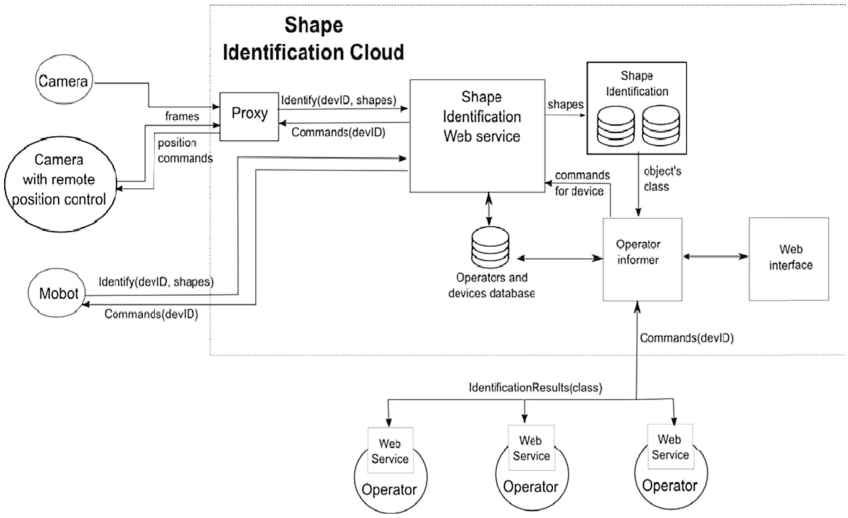



**Fig. 2.** The shape identification cloud overview

also be used by private users which need e.g. to monitor their homes when they are abroad. Because the cloud identification system allows to process shapes which are send to it, it may be used not only to monitor areas, but also to recognize objects on photos or pictures. In order to improve the efficiency, for some clients there also may be added dedicated databases, or it may be run as a private cloud.

The Shape Identification system as an interface may offer a web service, e.g. providing a method with list of shapes to detect as a parameter. The usage of such an interface allows easy access from any place with network connection. Moreover, the usage of web services simplifies adding new devices or applications, because they only need to invoke its methods. Because some devices are able only to send frames without any shape extraction, there can be added a proxy module as a part of the system, which will perform that process and then execute command from the cloud web service. The operators may be available as a web applications or desktop applications. Because they have to be informed immediately when new image appears into detection, they may also implement a web service interface which may be invoked by the Shape Identification Cloud. The Cloud may have a database of video devices and operators, which are connected in order to send proper detection results to the operators, and then send back commands for devices if they are able to execute them.

The example system execution may be as follows. Firstly, the device captures a frame which is send to the proxy if shape extraction cannot be performed (cameras), or processed in order to extract shapes (mobots). Next, the Identify web service method is executed, sending extracted shapes and device identifier to the cloud. Next, the cloud sends data to the Shape Identification module which was described in the first part of this chapter. After processing, information about recognized objects is transferred into Operator Informer module which communicates with operators. Each operator may

perform some actions or send back commands to the device (change camera position or execute commands on a mobot).

The IoT monitoring system working as a web service in the cloud may find a wide range of applications. Lets consider a few examples. Such a system may prove to be very helpful for a seriously ill person who due to illness is not able to move independently. When this person is left unattended, he or she has very limited capabilities to react whenever an alarming event happens. By using the system via a mobile device and a simple web interface, for example after hearing a strange noise, he/she could send a robot on patrol in order to verify what has happened, e.g. whether an intruder has entered the house. Mobot could also perform more specific actions, for instance after uploading a photo of a book, it may identify it lying on a shelf and bring it back to the patient. Instead of a book, one can easily imagine another objects like a glass of water or a medication. Obviously the system is not able to replace medical care provided by a professional nurse, however it can substantially increase comfort of the patient by providing him/her with a certain degree of autonomy and a sense of independence, which may also influence his/her healing process in a positive way.

Finding lost items could also be performed on the basis of an approximate shape of an object, entered through the application available for a mobile touch device such as a tablet. For example, before leaving a house someone is not able to find keys to a car. After drawing the sketch of the object, mobot starts to scan the environment in search of the keys. After finding them, the user receives an indication about this fact on the mobile device through the web interface along with the picture of their location.

## 5 Object Recognition Algorithm

The object recognition algorithm is based on the Query by Shape method which consists of three elements: object representation, matching algorithm and database structure. The processing pipeline is shown in Fig. 3.

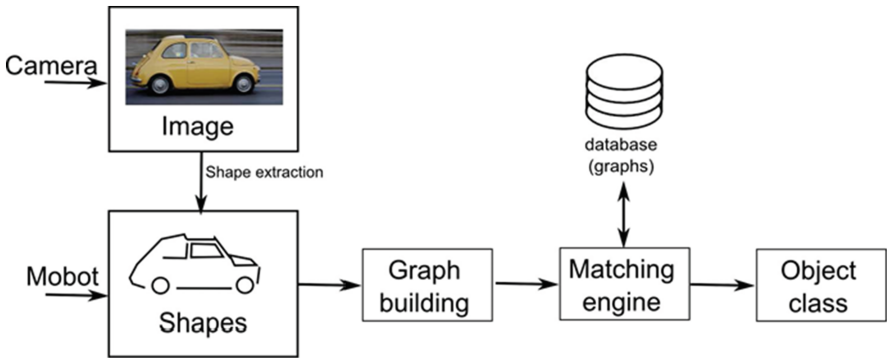



**Fig. 3.** The object recognition algorithm overview.

The main idea of the object representation is based on decomposing an object into features. The features may consists of shapes, colors or textures. Relations between features are represented by a graph associated with the object. After query image decomposition, all detected shape object features should be surrounded by the bounding box. As a shape features, a line and an ellipse is used, but other primitives (e.g. triangles or rectangles) may also be taken into account. Each shape feature has the following attributes:

- for ellipse feature, a ratio between its radii to the diagonal of the bounding box (relative sizes, if both diameters are equal, store only one),
- for line feature, a ratio between its width and height (the line slope computed by dividing line height by line width),
- (optional) shape's average color.

The color feature for the object may be defined as a three most frequent color values. The texture features may be extracted from the interior of any shape feature. For shape detections different algorithms may be used, e.g. Linear and Circular Hough Transform or even artificial intelligence methods. Because during experiments for real photos many detected lines were split, we added line merging stage. It compares the distances between endings of the lines and if it is smaller than chosen threshold value, they are merged. The example graph is showed in Fig. 5. The graph building algorithm is as follows:

1. Detect shapes (e.g. lines, circles, ellipses) and store their position and size (for lines – the ending points, for circles – the center and radius, etc.).
2. For lines perform line merging:
   (a) find lines for which the distances between their ending points is smaller than given threshold,
   (b) for each found line pairs check if the difference between their slopes is smaller than given threshold, if yes, merge them.
3. Find the bounding box which surrounds all shapes.
4. For each shape compute its attribute value.
5. Construct the graph:
   (a) from each detected shape create nodes,
   (b) if the distance between two shapes is smaller than minimal distance coefficient, create a link (an edge) between them.

Because features detection may produce some inaccuracies when building a graph, the minimal distance is used as a threshold. The example result of feature detection (limited to shape features) is presented in Fig. 4.

The next step incorporates classical graph matching problem using modified and simplified approach described in [25, 26]. In the future work, more efficient algorithm should be taken into consideration.

During matching, the comparison between graphs is made in two steps: graph's nodes level and features level (using two thresholds: $\in_N$ for nodes step and $\in_F$ for features step). The parameters values were chosen experimentally ($\in_N = 0.7$, $\in_F = 0.4$). As the first step a test is executed, which checks if the numbers of nodes are the same and if nodes are of the same type. If the similarity is equal or greater than $\in_N$,

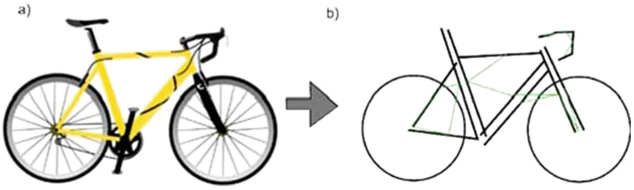



**Fig. 4.** The example of detection of shape features. As result, a graph is created. (a) The input object image. (b) The created graph. The connection between shape features are marked as a green line. Bike image source: openclipart.org

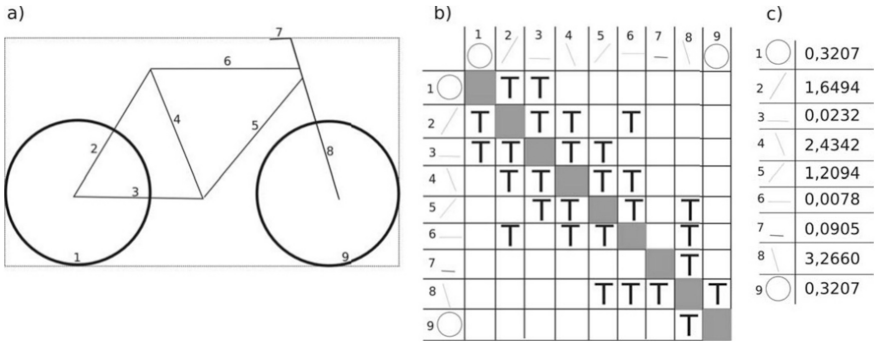



**Fig. 5.** The example bicycle graph: (a) the object after shape detection (the dotted rectangle shows object's bounding box), (b) shape graph adjacency matrix (note that node 5 is connected with 6 because of a minimal distance threshold), (c) parameters for each primitive [2].

the next step is performed for the same subgraphs. For ellipse nodes the algorithm checks if the difference between their sizes is equal or smaller than $\in_F$. For line nodes the line angles are compared in the same way. If tests performed in both steps are successful, then the subgraphs are considered as the same. In order to achieve better accuracy, for each node the relative locations of the connected other nodes to it are stored for X-axis ("left" or "right" value) and for Y-axis ("top" or "bottom" value). The similarity coefficient is defined as a ratio of correct nodes to all nodes. In order to achieve more precise detection results, two thresholds are defined as follows:

- $\in_H$ – above this level the objects are treated as the same,
- $\in_L$ – under this level the objects are treated as different.

If the similarity coefficient has value between $\in_H$ and $\in_L$, then the object is reported as similar (but not the same). For the graphs containing only shape features, we developed the matching algorithm which is as follows:

1. Clear the similarity coefficient value for the query graph $g_q$: *sim* := 0.
2. For each i-th node of $g_q$ $(g_{qi})$:
   (a) clear the variables which are used for storing the value and node number of the most similar node in $g_m$: *nodeSim* := 0 and *nodeNum* := 0,

(b) for each j-th node of the database graph $g_m$ $(g_{mj})$:
  (i) if $g_{mj}$ is marked as matched, check next $g_m$ node,
  (ii) *nodeLevelSim* := 0,
  (iii) if $g_{qi}$ shape type is the same as $g_{mj}$ then *nodeLevelSim* := 1, else go to 2b and check another node,
  (iv) divide the number of $g_{qi}$ links by the number of $g_{mj}$ links; if the number is greater than 1, saturate it to 1,
  (v) add the number from the previous step to *nodeLevelSim*,
  (vi) divide *nodeLevelSim* by 2 to normalize the value,
  (vii) if *nodeLevelSim* < $\in_N$ then go to 2b and check another node,
  (viii) *featureLevelSim* := 0,
  (ix) compare parameters values for $g_{qi}$ and $g_{mj}$: if the difference is smaller than F then go to 2b, else *featureLevelSim* := 1 − *difference*,
  (x) for each $g_{qi}$ link match the corresponding $g_{mj}$ link; compute their parameters difference, check if they are located in the same direction and add to *featureLevelSim* in the same way as in the previous step,
  (xi) divide *featureLevelSim* by the number of $g_{qi}$ links + 1,
  (xii) if *nodeLevelSim* ∗ *featureLevelSim* ≥ *nodeSim* then *nodeSim* := *nodeLevelSim* ∗ *featureLevelSim* and *nodeNum* := *j*,
(c) mark $g_{mnodeNum}$ node as matched,
(d) add *nodeSim* to *sim*.

3. Divide *sim* by number of $g_q$ nodes.
4. Return *sim* as the similarity coefficient value for $g_q$ and $g_m$ objects comparison.

The $g_q$ denotes the graph which is used as a query, the $g_m$ the graph from the database with which a comparison should be done. Currently the query graph is compared with each graph in a database with the matching algorithm.

For CBIR database query the results after comparing all images are then grouped into two sets: certain (with *sim* > = $\in_H$) and rough (*sim* < $\in_H$ AND *sim* > $\in_L$). We assume that if the query object cannot achieve any results grater than $\in_L$ it can be added to the database as a new object class.

For CBIR-based object recognition, which is the aim of this article and the research, after each comparison the resulted sim value is compared with the stored maximum sim value. If it is higher, then it is stored as a new maximum. Moreover, the graphs in the database store not only the nodes and edges, but also the object class name which may be used e.g. for making an annotation on the input image.

All of the matching algorithm parameters values are chosen experimentally. The dependency between their values and the precision of the results will be examined during further research. The algorithm is designed to detect objects which are placed on the different backgrounds than examples in the database.

The third part of the Query by Shape method is the database structure. Because it should store only graphs, we propose a tree structure, which should allow faster querying. Moreover, a more general comparisons could be prepared in the higher tree levels and more precise in the lower levels, similarly to some of the learning algorithms [9, 10]. Also in order to improve the performance of the system, a distributed database structure should be taken into account, for example using Scalable Distributed Data

Structures [27]. In [28] we proposed the preliminary approach to the database structure which is based on them.

## 6 Experimental Results

The proposed image storage method and matching algorithm was initially tested with real life images of a bicycle, a motorbike and a car. As a query image, a bicycle object drawn by a human was given. The algorithm parameters had the following values: $\in_H = 0.6$, $\in_L = 0.4$, $\in_N = 0.7$, $\in_F = 0.4$. During detection stage, only shape features were used. The test results are presented in Table 1.

**Table 1.** The results for the initial test, using a bicycle object (with 11 graph nodes) as a query. Each line shows the $n_s$ or $f_s$ coefficient value for the comparison of queried bicycle graph node and corresponding node in tested object from database. Obtained precision and recall is 1.

| Node | A bicycle | A motorbike | A car | A car |
|---|---|---|---|---|
| 1 | 1 | 0 | 0 | 0,25 |
| 2 | 1 | 0 | 0,6666666667 | 0,6666666667 |
| 3 | 1 | 0,6666666667 | 0 | 0 |
| 4 | 1 | 0,5 | 0,3333333333 | 0,5 |
| 5 | 1 | 0,5 | 0,5 | 0 |
| 6 | 1 | 0,5 | 0,25 | 0 |
| 7 | 1 | 0 | 0,25 | 0 |
| 8 | 1 | 0,3333333333 | 0,25 | 0,25 |
| 9 | | 1 | 0 | 0 |
| 10 | | 0,5 | 0 | |
| 11 | | 0,5 | 0 | |
| *sim* = | 1 | 0,125 | 0 | 0 |
| add to query results? | yes | no | no | no |

The results show that the highest *sim* values (near 1) were obtained for the most similar objects. If objects are different, the values of *sim* are significantly lower. The *sim* value could be used to classify objects, choosing the proper threshold. Moreover, it could give information about the detection reliability.

Another experiments were performed using a C++ application and OpenCV library for image handling and processing. For shape detection the Circular and Linear Hough Transform algorithms provided by OpenCV were used. As a database of images, the total number of 1240 images were used including:

- 101 images from web-crawled database of cars, bicycles, motorbikes, scooters, chairs and tanks were used (Fig. 6),
- 921 Caltech101 images (cars side and motorbikes sub-sets),
- 215 cars and bicycles images from The TU Darmstadt Database.

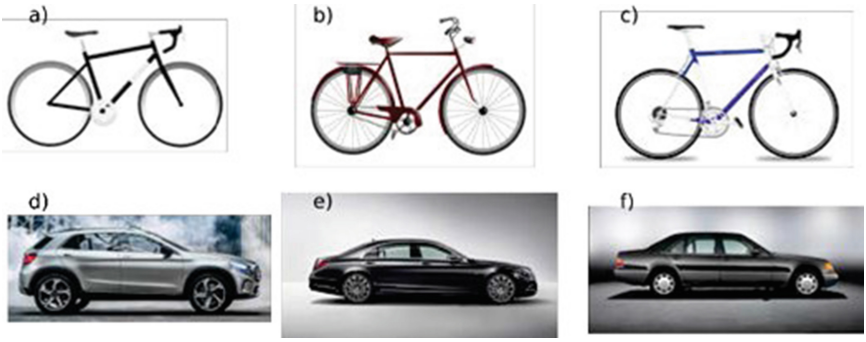



**Fig. 6.** Some of the images used in the C++ application test. Source: (a), (b), (c) from OpenClipart.org, (d), (e), (f) from mercedes-benz.pl

All images from the web-crawled subset had an uniform background and a side view of objects was used. The rest of the test images had different background and object orientations. The summarized results are presented in Table 2. The algorithm results were compared with the implementation of Color Histogram CBIR algorithm described in [12]. As a performance evaluation the precision coefficient was used, defined as follows [15]:

$$precision = \frac{number\ of\ relevant\ result\ images}{total\ number\ of\ result\ images}$$

The results shows that the Query by Shape algorithm is much more precise than Color Histogram based [12] one. The Caltech and TUD data sets were very demanding for the QS implementation because all images were in a very low resolution which was problematic for Circular Hough Transform. For many images it was unable to detect any circles, decreasing sim value and graph size. Also Linear Hough Transform sometimes had problems. Despite that fact, the algorithm was successful for many images with other objects in the background. Moreover, the sim coefficient value could be used as a measurement of the certainty of the recognition, and if it is small, other methods could be applied for the image. Choosing more precise and efficient shape detection algorithms should highly increase the precision value.

**Table 2.** The precision value for query by shape and color histogram algorithms

| Data set | Number of images | QS precision | CH precision |
|---|---|---|---|
| Web-crawled own data set | 104 | 0.8076923077 | 0.1826923077 |
| Caltech 101 cars side | 123 | 0.7317073171 | 0.1788617886 |
| Caltech 101 motorbikes | 798 | 0.4930434783 | 0.3045112782 |
| TUD cars | 100 | 0.72 | 0.3 |
| TUD motorbikes | 115 | 0.4695652174 | 0.2347826087 |
| | Total: 1240 images | | |
| | Average: | 0.6444016641 | 0.2401695966 |

## 7 Conclusions

In this paper the idea of an IoT-based cloud monitoring system was proposed which is able to manage many devices and operators. Moreover, as a part of the system, the shape identification algorithm was described with more details. It uses the representation of an object as a graph of associated features, e.g. shapes, colors or textures. Also the matching algorithm was presented. Because of the usage of shapes like lines, circles, triangles or rectangles, the method would be especially suitable for human designed, artificial objects. The initial tests showed that the method gives high number of positive results despite being at a very early stage of development. The initial tests showed that the method gives high number of positive results despite being at a very early stage of development.

The future research should include further development of matching algorithm to incorporate color or texture features during the first algorithm stage. Furthermore, more advanced shape comparison methods will be considered, e.g. based on decision-making problem proposed in [29]. As a result of the research we would like to obtain a reliable and precise matching algorithm. Moreover, more work should be done in order to achieve more accurate results for objects with different orientations. The database structure should be chosen after some tests in order to produce the best results. The scalability of the database should also be taken into account.

Future work also involves finding the most suitable algorithm for the object extraction component of the monitoring system video pipeline. As the first step, simple morphological operations like erosion and dilation can be used to preprocess images in order to reduce noise and the amount of detail [3–5, 30]. Method based on Top-Hat Transform can be applied to crop individual objects form the image [30]. Segmentation of the image into regions based on colors may also be performed [31, 32]. After that, these regions may be modeled by a mixture of Gaussian distributions and then shapes can be localized by minimizing the corresponding Gibbs field [32]. Moreover, by utilizing a 3D map of the whole environment stored directly in the mobots, background subtraction task can easily be realized.

Another field of research should incorporate developing other parts of the cloud and whole system. We believe that as a final result we will be able to obtain the cloud method for fast object recognition that may be applied in a wide range of applications like image retrieval from multimedia databases monitoring systems and other systems based on object detection and object recognition. Also in order to increase system reliability, a synthesis of real-time cloud applications for Internet of things may be used, which provides a high quality of service [33].

## References


1. Deniziak, S., Michno, T., Pięta, P.: Autonomous monitoring system based on object shapedetection. Measur. Autom. Monit. **61**, 349−351 (2015). In: 18th Conference on Reconfigurable Ubiquitous Computing.
2. Deniziak, S., Michno, T.: Query by shape for image retrieval from multimedia databases. In: Kozielski, S., Mrozek, D., Kasprowski, P., Małysiak-Mrozek, B., Kostrzewa, D. (eds.) BDAS 2015. CCIS, vol. 521, pp. 377–386. Springer, Heidelberg (2015). doi:10.1007/978-3-319-18422-7_33
3. Valera, M., Velastin, S.A.: Intelligent distributed surveillance systems: a review. IEE Proc. Vis. Image Signal Process. **152**, 192–204 (2005)
4. Kumar, P., Mittal, A., Kumar, P.: Study of robust and intelligent surveillance in visible and multimodal framework. Informatica (Slovenia) **32**(1), 63–77 (2008)
5. Hampapur, A., Brown, L., Connell, J., Pankanti, S., Senior, A., Tian, Y.: Smart surveillance: applications, technologies and implications. Inf. Commun. Signal Process. **2**, 1133–1138 (2003)
6. Chen, T.P., Haussecker, H., Bovyrin, A., Belenov, R., Rodyushkin, K., Kuranov, A., Eruhimov, V.: Computer vision workload analysis: case study of video surveillance systems. Intel Technol. J. **9**, 109–118 (2005)
7. Duda, R.O., Hart, P.E.: Use of the hough transformation to detect lines and curves in pictures. Commun. ACM **15**(1), 11–15 (1972). http://doi.acm.org/10.1145/361237.361242
8. Schuster, G., Katsaggelos, A.: Robust circle detection using a weighted MSE estimator. In: International Conference on Image Processing, vol. 3, pp. 2111–2114, October 2004
9. Viola, P., Jones, M.: Rapid object detection using a boosted cascade of simple features. In: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, vol. 1, pp. I–511–I–518 (2001)
10. Felzenszwalb, P., Girshick, R., McAllester, D., Ramanan, D.: Object detection with discriminatively trained part-based models. IEEE Trans. Pattern Anal. Mach. Intell. **32**(9), 1627–1645 (2010)
11. Wang, H.H., Mohamad, D., Ismail, N.A.: Approaches, challenges and future direction of image retrieval. J. Comput. **2**(6), CoRR, abs/1006.4568 (2010)
12. Sharma, J.S.N., Rawat, P.: Efficient CBIR using color histogram processing. Signal Image Process. Int. J. **2**(1), 94 (2011)
13. Shih, T.K.: Distributed multimedia databases. In: Shih, T.K. (ed.) Distributed Multimedia Databases, pp. 2–12. IGI Global, Hershey (2002)
14. Mocofan, M., Ermalai, I., Bucos, M., Onita, M., Dragulescu, B.: Supervised tree content based search algorithm for multimedia image databases. In: 2011 6th IEEE International Symposium on Applied Computational Intelligence and Informatics (SACI), pp. 469–472, May 2011
15. Sivakamasundari, J., Kavitha, G., Natarajan, V., Ramakrishnan, S.: Proposal of a content based retinal image retrieval system using kirsch template based edge detection. In: 2014 International Conference on Informatics, Electronics Vision (ICIEV), pp. 1–5, May 2014
16. Guo, J.-M., Prasetyo, H., Chen, J.-H.: Content-based image retrieval using error diffusionblock truncation coding features. IEEE Trans. Circ. Syst. Video Technol. **25**(3), 466−481 (2014)
17. Li, C.-Y., Hsu, C.-T.: Image retrieval with relevance feedback based on graph-theoretic region correspondence estimation. IEEE Trans. Multimedia **10**(3), 447–456 (2008)
18. Aggarwal, G., Ashwin, T., Ghosal, S.: An image retrieval system with automatic query modification. IEEE Trans. Multimedia **4**(2), 201–214 (2002)

19. Belloulata, K., Belallouche, L., Belalia, A., Kpalma, K.: Region based image retrieval using shape-adaptive DCT. In: 2014 IEEE China Summit International Conference on Signal and Information Processing (ChinaSIP), pp. 470–474, July 2014
20. Kato, T., Kurita, T., Otsu, N., Hirata, K.: A sketch retrieval method for full color image database-query by visual example. In: 11th IAPR International Conference on Pattern Recognition. Conference A: Computer Vision and Applications, vol. I, pp. 530–533, August 1992
21. Lee, H.-C., Fu, K.-S.: Generating object descriptions for model retrieval. IEEE Trans. Pattern Anal. Mach. Intell. PAMI **5**(5), 462–471 (1983)
22. Zhang, D., Islam, M.M., Lu, G.: Structural image retrieval using automatic image annotation and region based inverted file. J. Vis. Commun. Image Represent. **24**(7), 1087–1098 (2013)
23. Del Bimbo, A., Pala, P.: Visual image retrieval by elastic matching of user sketches. IEEE Trans. Pattern Anal. Mach. Intell. **19**(2), 121–132 (1997)
24. Daoudi, M., Matusiak, S.: Visual image retrieval by multiscale description of user sketches. J. Vis. Lang. Comput. **11**(3), 287–301 (2000)
25. Ullmann, J.R.: An algorithm for subgraph isomorphism. J. ACM **23**(1), 31–42 (1976)
26. Cordella, L., Foggia, P., Sansone, C., Tortorella, F., Vento, M.: Graph matching: a fast algorithm and its evaluation. In: Proceedings of the Fourteenth International Conference on Pattern Recognition, vol. 2, pp. 1582–1584, August 1998
27. Lukawski, G., Sapiecha, K.: Balancing workloads of servers maintaining scalable distributed data structures. In: 19th Euromicro International Conference on Parallel, Distributed and Network-Based Processing, pp. 80–84, February 2011
28. Greenspan, H., Dvir, G., Rubner, Y.: The scalable distributed two-layer content based image retrieval data store. In: Federated Conference on Computer Science and Information Systems, pp. 839–844 (2015)
29. Sitek, P., Wikarek, J.: A hybrid approach to the optimization of multiechelon systems. Math. Prob. Eng. **2015**, 1−12 (2015). Article ID 925675
30. Tajeripour, F., Saberi, M., Fekri-Ershad, S.: Developing a novel approach for content based image retrieval using Modified local binary patterns and morphological transform. Int. Arab J. Inf. Technol. **12**, 574−581 (2014)
31. Greenspan, H., Dvir, G., Rubner, Y.: Region correspondence for image matching via EMD flow. In: IEEE Workshop on Content-Based Access of Image and Video Libraries, pp. 27–31. IEEE (2000)
32. Destrempes, F., Mignotte, M., Angers, J.-F.: Localization of shapes using statistical models and stochastic optimization. IEEE Trans. Pattern Anal. Mach. Intell. **29**(9), 1603–1615 (2007)
33. Bąk, S., Czarnecki, R., Deniziak, S.: Synthesis of real-time cloud applications for internet of things. Turk. J. Electr. Eng. Comput. Sci. **23**(3), 913–929 (2015)

# Memory Analysis and Performance Modeling for HPC Applications on Embedded Hardware via Instruction Accurate Simulation

Alexander Ditter[1(✉)], Dominik Schoenwetter[1], Anton Kuzmin[1], Dietmar Fey[1], and Vadym Aizinger[2]

[1] Chair of Computer Science 3 (Computer Architecture), Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU), Erlangen, Germany
{alexander.ditter, dominik.schoenwetter, anton.kuzmin, dietmar.fey}@fau.de
[2] Chair of Applied Mathematics (AM1), Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU), Erlangen, Germany
aizinger@math.fau.de

**Abstract.** The efficient usage and development of embedded multi- and many-core systems is an important field of research for years and decades. In the last decade, utilizing embedded and especially low-power architectures for high performance computing (HPC) applications became an important part of research. The reason for this are the constantly increasing energy costs along with an increasing awareness of energy consumption in general. As suitable low-power HPC architectures are not yet available at a larger scale, simulation of new and appropriate architectures becomes an important factor for the success of low-power systems and clusters. In order to speed up simulation, at the cost of accuracy, different levels of abstraction were introduced. Currently the class of instruction accurate simulations seems to yield the best trade-off between speed and precision, especially when simulating complex multi- and many-core systems. In this paper we present our investigations about the accuracy of the state-of-the-art instruction accurate embedded multi- and many-core simulation environment Open Virtual Platforms (OVP) in comparison to an actual embedded hardware system from Altera. Our investigations include the actual usage of the same operating system running on both, the target hardware and the simulation as well as serial and parallel software benchmarks. We analyze the current accuracy of the simulation environment with respect to a performance model, based on the execution time of the simulation and the actual embedded hardware system. Using the instruction accurate simulation technology from OVP is for the simulation of embedded/low-power HPC hardware architectures and applications. Furthermore, we point out first steps towards further possibilities for obtaining a better performance model by the use of our simple memory model.

## 1 Introduction

Single-core processors have been replaced by multi-core, and in many areas of *high performance computing* (HPC) to a significant extend by many-core processors, such as *general purpose GPUs* (GPGPUs) or Intel's *many integrated core* (MIC) system.


J. Janech et al. (eds.), *Proceedings of the 2015 Federated Conference on Software Development and Object Technologies*, Advances in Intelligent Systems and Computing 511,
DOI 10.1007/978-3-319-46535-7_2

While this development has progressed for well over a decade it has just reached its slow pick-up phase in the field of embedded low-power systems during the last years. Yet, the same restrictions and requirements, e.g., the thermal power wall along with decreasing structures in the manufacturing process and that have been driving this conversion for desktop, server and HPC systems become more and more important for many applications in the embedded domain. The increasing cost for single-core systems, due to the low remaining stocks and production rates, but also and more importantly their lack of performance, becomes more and more of a hindrance, e.g., for driving assistance systems, where computational power is indispensable. Thus, the transition to multi-core systems was necessary and unavoidable. Currently, as more and more functionality is added to applications running on the embedded devices, as well the steadily increasing interest in utilizing them for HPC applications, the additional computing power is needed to satisfy the increasing performance requirements. This, in turn, leads to the development of interconnected multi-core architectures and cluster systems, mostly based on conventional and HPC network fabrics, as embedded hardware developers abandon the use of the more traditional bus systems. Even if such systems are currently not yet an established standard they will play an important role in most embedded low-power application domains in the future. Due to the increasing requirements for power and energy efficiency for HPC systems and architectures, a significant amount of research has been put into the utilization of the afore mentioned conventional multi- and many-core low-power architectures. Yet, as suitable low-power architectures for HPC systems currently are still only scarcely available, accurate simulations of these architectures are a key instrument for verification, memory analysis, performance modeling, prediction and evaluation. It may even be used to guide to development of new embedded hardware towards a more suitable design for HPC applications. Historically, simulation techniques can be categorized in different abstraction levels (cf. Table 1) [1], where the level of abstraction differentiates two main aspects: (i) the speed of the simulation and (ii) the accuracy of the results. In this classification the functional model corresponds to the highest level of abstraction, resulting in the lowest simulation accuracy, but the highest simulation speed. The lowest level of abstraction is the physical level, where even physical effects, e.g., of transistor gate delays, are considered. Naturally, the physical level has the worst simulation speed, yet the highest accuracy of simulation results. The functional level does not distinguish between hard- and software. Thus, this level is too abstract for evaluating combined soft- and hardware aspects. Yet, the instruction accurate level does distinguish between hard- and software. Furthermore, it is even possible to obtain information about non-functional properties like time and energy. At the same time the simulation speed of instruction accurate simulations is very fast, as compared to all more accurate modeling techniques (cycle accurate, RTL and physical level). In terms of simulation speed, the cycle accurate simulation level is very slow in comparison to the instruction accurate one. Weaver and McKee [2] showed that discrepancies of hours up to days are possible and the simulation results are not mandatory more accurate. As a consequence, cycle accurate simulations are not an option for the simulation of multi- or large many-core systems. Ideally the simulation of large embedded multi-core systems, has to be both: accurate and fast. Due to the large number of processors, peripherals and network interconnects in such a system,

**Table 1.** Level of abstraction decreases from top to bottom, while accuracy increases.

| Modeling technique | Modeling language |
|---|---|
| Functional Model | MATLAB/Simulink |
| Instruction Accurate Model | Instruction Set Simulator (ISS) |
| Transactional Level without Time (Programmers View without Time) | Un-Timed SystemC |
| Transactional Level with Time (Programmers View with Time) | Timed SystemC |
| Cycle Accurate Model | C model |
| Register Transfer Level (RTL) | HDL like VHDL |
| Physical Level | SPICE |

especially the simulation speed is an important aspect. A state-of-the-art instruction accurate simulation technology is called *Open Virtual Platforms* (OVP) and provided by Imperas [3]. This simulation environment shows excellent properties concerning to simulation speed. Also, the scalability of the simulation results for different numbers of cores was verified [4].

In this paper we investigate the accuracy of results from the state-of-the-art instruction accurate simulation environment OVP along with their applicability for the verification and evaluation of embedded HPC architectures and algorithms. Furthermore, we have developed an instrumentation mechanism for OVP that allows us to track, record and trace memory access patterns within the simulation of whole applications as well as functions and instructions. Using this technique, we can provide a better understanding for memory access patterns to analyze and compare different algorithms and hardware systems and advice software and hardware developers about potential improvements in their respective designs, allowing for an overall better hardware-software co-design. We extend this capability with a first simple memory model, allowing us to use this approach for a more accurate performance modeling than conventional instruction accurate simulation techniques. For our analysis and evaluation we compare the results of OVP to those obtained from our actual reference hardware, the Altera Cyclone V *sytem on chip* (SoC) (cf. Sect. 3.2), as we emulated the Altera SoC using OVP.

The rest of the Paper is structured as follows. Section 2 provides a comprehensive survey of related work in the field. Section 3 gives an in depth overview of the OVP simulation environment as well as the actual hardware system we compare our results against. Section 4 explains our instrumentation for tracing memory access in OVP along with the memory and performance modeling. Section 5 introduces the applications and benchmarks used for our investigations. Section 6 presents our findings and quantitative results along with a conclusion and an outlook on future work in Sect. 7.

## 2 Related Work

There is a significant body of research in the field of utilizing low-power architectures for HPC and in the optimization of energy efficiency for HPC applications. Rajovic et al. investigated the usage of low-power ARM[1] architectures and SoCs as means to reduce the cost of HPC [5]. They conclude that low-power ARM-based SoCs have promising characteristics for HPC. In 2013, Goeddeke et al. presented a paper on energy-to-solution comparisons between different architectures for different classes of numerical methods for partial different equations [6]. They showed that energy-to-solution and energy-per-time-step improvements up to a factor of three are possible when using the ARM-based Tibidabo cluster based on a setting with 96 ARM Cortex-A9 dual-core processors [7], instead of an x86-based cluster. The x86 cluster used for the reference measurements was the Nehalem sub-cluster of the LiDOng machine provided by TU Dortmund [8]. A study comparing the performance as well as the energy consumption of different low-power and general-purpose architectures was published by Castro et al. [9]. Based on the Traveling-Salesman problem [10], they investigated time-to-solution and energy-to-solution for an Intel Xeon E5-4640 Sandy Bridge-EP with 8 cores, 16 threads, i.e., hyper-threading support, a low-power Kalray MPPA-256 many-core processor consisting of 256 user cores and 32 system cores [11], as well as for a low-power CARMA board from SECO [12], consisting of a NVIDIA Tegra 3 and a NVIDIA Quadro 1000M GPU. The results show, that the CARMA board and the MPPA-256 many-core processor achieve better results than the Xeon 5 measured in terms of energy to solution. With regard to the time-to-solution metric, the Xeon 5 performed better than the CARMA board but not as good as the low-power MPPA-256 many-core processor. A work considering low-power processors and accelerators in terms of energy aware HPC was published in 2013 [13]. There, a number of different HPC micro-benchmarks was used to determine the energy-to-solution. The architectures evaluated were NVIDIA Tegra 2 [14] and Tegra 3 [15] SoCs. The results show that drastic energy to solution improvements are possible on the newer Tegra 3 SoC in comparison to the Tegra 2 SoC (reduction of 67% on average). Furthermore, the authors conclude that the usage of integrated GPUs in low-power architectures, such as Tegra 2 and Tegra 3, can improve the overall energy efficiency. All presented investigations emphasize, that lowpower hardware architectures have promising characteristics for HPC. There is a range of tools that allow memory modeling and simulation, e.g., Gem5 [16], DiskSim [17] or DRAMSim2 [18]. The cycle accurate simulation environment Gem5 allows a comprehensive set of building blocks, ranging from caches, crossbars, to full-blown DRAM controllers. DiskSim is an accurate and highly-configurable disk system simulator. It was developed to support research into various aspects of storage subsystem architecture, including modules that simulate intermediate controllers, buses, device drivers, request schedulers, disk block caches, and disk array data organizations. DRAMSim2 is a cycle accurate memory system simulator. The goal of DRAMSim2 is to be an accurate DDR2/3 memory system model which can be used in tracebased and full system simulations.

[1] http://www.arm.com/.

# 3 Environment

## 3.1 Simulation Environment

The simulation technology from Open Virtual Platforms was developed for high performance simulation. The technology enables debugging applications, which run on the virtual hardware, as well as analysis of virtual platforms containing multiple processor and peripheral models. The OVP simulation technology is extensible. Furthermore, it provides the ability to create new processor models and other platform components by writing C/C++ code that uses the *application programming interface* (API) and libraries supplied as part of OVP [19]. The API defines a virtual hardware platform which is called ICM (Innovative CPUManager Interface). This API includes functions for setting up, running and terminating a simulation (*icmInitPlatform, icmSimulatePlatform, icmTerminate*), defining components for the simulation (e.g., *icmNewProcessor)* and loading application's executable (*icmLoadProcessorMemory*). Figure 1 shows an overview about how an OVP simulation with minimal effort works. Minimal effort means, that one processor has to be defined for the simulation. The example uses the language C for the hardware platform as well as the application to run on that platform. The simulator included in OVP is an instruction accurate simulator. This means, that the functionality of a processor's instruction execution is represented without regard to artifacts like pipelines. Instruction accurate simulation cannot make a clear statement about time spent during pipeline stalls, due to cache misses and other things that are not modeled, so any conversion to time will have limited accuracy compared to actual hardware. OVP multi-processor platforms are not working simultaneously. For efficiency, each processor advances a certain number of instructions in turn. So in multi-processor simulations a single processor is simulated until it has signaled that it has finished its quantum. The quantum is defined as the time period in which each component in turn simulates a certain number of instructions [19]. Simulated time is moved forward only at the end of a quantum. This can create simulation artifacts, for example where a processor spends time in a wait loop, while waiting for the quantum to finish. To avoid this the quantum has to be set very low value (potentially having a significant impact on the simulation performance) so that the measurements will not be affected by this simulation artifacts. This can be adjusted in the simulator settings [20]. The simulation environment can only provide the total amount of instructions that were executed. Assuming a perfect pipeline, where one instruction is executed per cycle, the instruction count divided by the processor's instruction rate, in million instructions per second (MIPS), yields the run time of the program. The OVP simulator provides the possibility for measuring instruction counts within a program. As a consequence, the instruction counts for specific code snippets can be recorded. On singlecore platforms, assuming that no time-controlled peripheral models are invoked, there is no need to set the quantum to one because the multi-core scheduling algorithm does neither affect nor intervene the simulation.

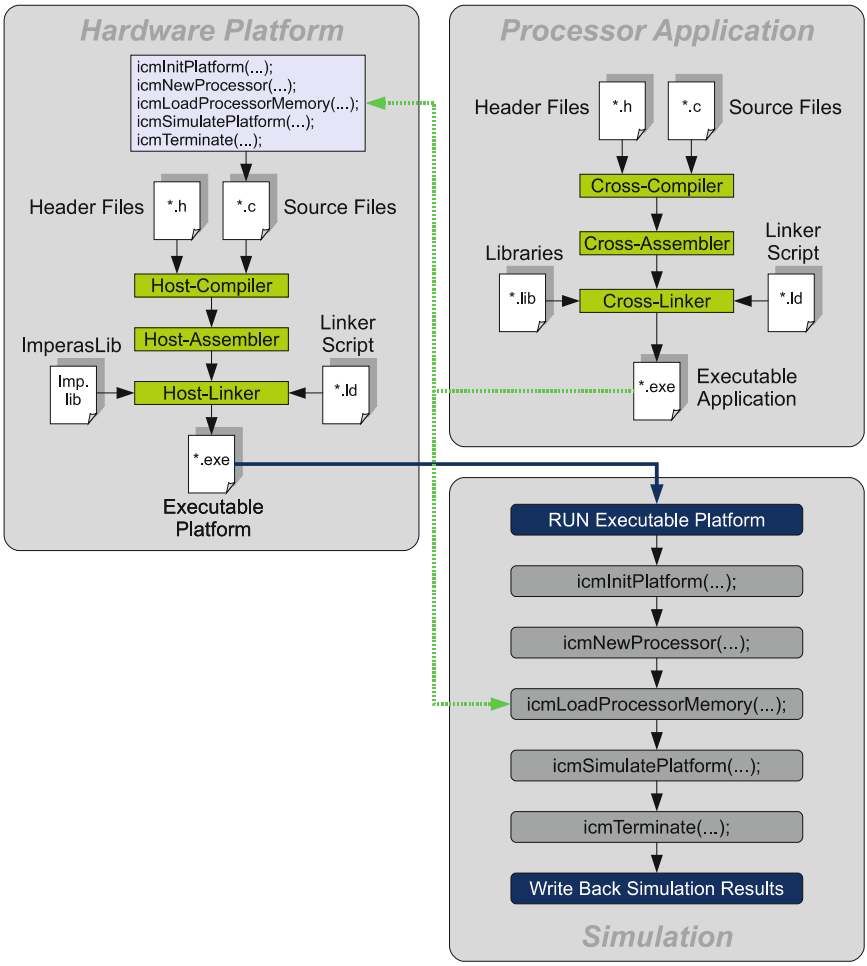



**Fig. 1.** Operating principle of open virtual platforms simulations.

### 3.2 Reference Hardware

Embedded system developers have to satisfy multiple requirements such as a high computational performance, support for a wide variety of communication interfaces and protocols, execution of complex signal processing algorithms in realtime, low power consumption. All these requirements have to be fulfilled with a very limited amount of resources. Given a long lifetime of the deployed systems and ever changing environmental conditions, the in-field support and upgrade is one of the most crucial requirements. A common solution to these demands is the extensive use of *field-programmable gate arrays* (FPGAs) for the hardware part and on the software side, relying on a processor architecture with a well-established and active ecosystem. Just a few years ago this approach implied at least two separate complex chips to be used in a single system. However the situation has recently changed and several FPGA vendors

came to market with integrated devices combining high-performance ARM CPUs, a fast memory controller along with a rich set of peripheral devices and a programmable logic unit. The integration of all these units into a single SoC provides developers with multiple benefits and thus, we expect such systems to become established in many embedded devices. For this reason we we chose Altera's development kit board [21] with a Cyclone V SX SoC-FPGA as our reference hardware platform to carry out our benchmarking. This SoC-FPGA includes a *hard processor system* (HPS) consisting of *multiprocessor subsystem* (MPU), multiport SDRAM controller with support for *double data rate 2* (DDR2), DDR3 and low-power DDR2 memory, a set of peripheral controllers and a high-performance interconnect. The memory controller supports command and data reordering, *error correction code* (ECC) and power management. On the board 1 GiB of DDR3 SDRAM is connected to the memory controller via a 40-bit data bus operating at 400 MHz for a total theoretical bandwidth of over 25.6 Gbps. The multiprocessor subsystem of the HPS includes two ARM Cortex-A9 MPCore 32-bit processors with a NEON SIMD co-processor and double-precision floating point unit (FPU) per processor, 32 KiB instruction and 32 KiB data level 1 (L1) caches and *memory management unit* (MMU) per processor, ARM level 2 (L2) cache controller and shared 512 KiB L2 cache. The cache controller has a dedicated 64-bit master port connected directly to the SDRAM controller and a separate 64-bit master port connected to the system level 3 (L3) interconnect. All blocks of the HPS are connected with L3 multilayer AXI interconnect structure. Low-speed peripheral controllers reside on the level 4 (L4) AXI buses working in separate clock domains for efficient power management. Programmable logic part of the SoC is a high-performance 28 nm FPGA. The HPS and FPGA part of the chip are connected via high-bandwidth (> 125 Gbps) on-chip interfaces. All the benchmarks presented in this paper use only the HPS part of the SoC-FPGA, while the FPGA part is not used. The Cortex-A9 MPCore runs a Linux kernel version 3.16.0 and the user space software is an ARM Arch Linux distribution utilizing a rolling release model.

### 3.3 Virtual Hardware

The virtual hardware platform implements just a part of the actual Altera Cyclone V SoC [22]. Specific and not needed hardware parts, e.g., the FPGA block, are not implemented. Anyway, all required components for running a Linux kernel and guarantee correct hardware functionality for our test cases are available. Figure 2 shows the subset of implemented hardware components. The virtual hardware allows to boot the same Linux kernel (3.16.0) as the actual hardware does. As a consequence, the virtual and the actual hardware are binary compatible.

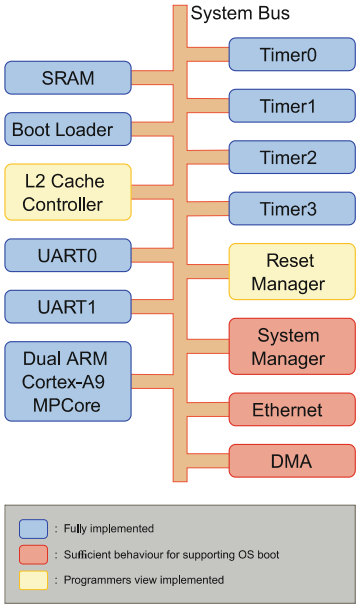



**Fig. 2.** Virtual Cyclone V SoC

## 4 OVP Instrumentation and Modeling

OVP, being an instruction accurate simulation environment, allows to track and trace each individual instruction in the program flow, we utilize this functionality to capture each memory access. For this purpose we designed a light weight library that allows to start and stop the recording of memory access instruction from within the respective application. This can simply be achieved by linking against our library. Since many HPC applications are either written in C/C++ as well as Fortran, especially for legacy application and as in our case the benchmark set, we have designed the library in C, allowing to interface with both programming languages without any additional requirements. The library offers the possibility to start and stop the recording of memory accesses, thus, allowing to restrict the data acquisition to the respective region of interest. This may be an entire application as well as a function call or an individual instruction. Currently the library is capable to record the total number of read and write accesses. We have also experimented with a more detailed recording, which is currently not incorporated any more, as it slows down the entire simulation by one to two orders of magnitude and creates memory access log files of several GiB within just a few minutes of host run time. Nonetheless, we plan to extend the current capabilities by employing techniques to reduces the memory consumption of trace files analogous to [23].

Using the results of our memory access recording library we obtained we designed a first very basic memory model for OVP, which considers and distinguishes the amount of read and write access to memory and weighs them with different factors. This makes sense, as it is much less likely for a write access to be delayed as much as a

read access, in case the data is not available in the cache. We use these factors to better estimate the performance, i.e., run time of the applications and benchmarks in Sect. 5.

## 5 Benchmarks and Applications

Our investigations are based on one real world HPC application, one real world but artificial problem and an extensive artificial benchmark set. All these individual benchmarks resemble typical computational fluid dynamics (CFD) applications and provide a wide range of typical HPC characteristics, such as compute and memory bound kernels. We evaluate our approach for the well known 3D shallow-water solver UTBEST3D, which is a typical high performance computing application and described in detail in Sect. 5.1 as well as the computation of mandelbrot sets. Additionally we use the NAS Parallel Benchmark suite (NPB). [24], i.e., the eight original benchmarks specified in NPB 1, consisting of five kernels and three pseudo applications. We cross-compiled the applications and benchmarks for both, the real hardware as well as the OVP simulation to ensure binary equality and thus, best possible comparability of the obtained results. For this, we used gfortran-arm-linux-gnueabihf for the Fortan based and gcc-arm-linux-gnueabihf for the C based benchmarks (both in version 4.8.2). The same binaries were used in the real and the virtual environment. The individual characteristics of UTBEST3D, the mandelbrot set and the NAS benchmarks are described in the following.

### 5.1 UTBEST3D – U3D

The numerical solution algorithm in the 3D shallow-water solver University of Texas Bays and Estuaries Simulator - 3D (UTBEST3D) considers the system of hydrostatic primitive equations with a free surface [25, 26]. A prismatic mesh is obtained by projecting a given triangular mesh in the vertical direction to provide a continuous piecewise linear representations of the topography and of the free surface. The vertical columns are then subdivided into layers. If a bottommost prism is degenerate, it is merged with the one above it. Due to the discontinuous nature of the approximation spaces, no constraints need to be enforced on the element connectivity. Hanging nodes and mismatching elements are allowed and have no adverse effects on stability or conservation properties of the scheme. This flexibility with regard to mesh geometry is exploited in several key parts of the algorithm: vertical mesh construction in areas with varying topography, local mesh adaptivity and wetting/drying. The discontinuous Galerkin (DG) discretization is based on the local discontinuous Galerkin method [27] that represents a direct generalization of the cell-centered finite volume method, the latter being just the piecewise constant DG discretization. One of the features of this method is a much smaller numerical diffusion exhibited by the linear and higher order DG approximations compared to the finite difference or finite volume discretization. The method guarantees the elementwise conservation of all primary unknowns including tracers, supports an individual choice of the approximation space for each prognostic and diagnostic variable, demonstrates excellent stability properties, and possesses

proven local adaptivity skills. UTBEST3D is written in C++ to provide clean interfaces between geometrical, numerical, computational, and communication parts of the code. The object-oriented coding paradigm is designed to enable a labor efficient development lifecycle of the model. The programming techniques were carefully chosen and tested with the view of guaranteeing a smooth portability to different hardware architectures, operating systems, compilers, and software environments.

### 5.2 Mandelbrot Set – MB

A mandelbrot set is defined as the set of complex numbers in the complex plane where the sequence $c$; $c^2 + c$; $(c^2 + c)^2 + c$; … does not approach infinity, even if the iteration counter tends to infinity [28]. Due to the characteristic of the sequence, it can be defined as a complex quadratic polynomial of the form $z_{n+1} = z_n^2 + c$; with $z_0 = 0$ Mandelbrot sets are often visualized by a mapping from the complex plane into an image representation. For this purpose the imaginary and real part of each complex number is considered as an image coordinate. Depending on how rapid the sequence and quadratic polynomial, of each pixel diverges, the corresponding pixel gets a defined color. If the sequence converges, the pixel is colored black. As each pixel can be computed independently and requires few memory accesses for every iteration, the mandelbrot set is a compute bound application.

### 5.3 NAS Parallel Benchmarks

The kernels considered in our benchmarking are CG (Conjugate Gradient with irregular memory access and communication), MG (Multi-Grid on a sequence of meshes, long- and short-distance communication), FT (discrete 3D fast Fourier Transform containing all-to-all communication), EP (Embarrassingly Parallel) and IS (Integer Sort with random memory access). LU (Lower-Upper Gauss-Seidel solver) are the defined pseudo applications, BT (Block Tri-diagonal solver) as well as, SP (Scalar Penta-diagonal solver).

*Conjugate Gradient Benchmark* – CG: This benchmark computes an estimate of the largest eigenvalue of a symmetric positive definite sparse matrix using the conjugate gradient method [29]. The run time of this benchmark is dominated by the sparse matrix vector multiplication in the conjugate gradient subroutine. Due to the random pattern of nonzero entries of the matrix this requires a high number of memory accesses, leading to a low computational intensity of this memory bound benchmark.

*Multi Grid Benchmark* – MG: The MG benchmark is based on a multigrid kernel, which computes an approximative solution of the three dimensional Poisson problem. In each iteration the residual is evaluated and used to apply a correction to the current solution. The most expensive parts of this algorithm are evaluation of the residual and application of the smoother, both of which are stencil operations with constant coefficients for the specified problem. The update of a grid point require the values of neighboring points. Thus, even with an optimal implementation this requires in

between four and eight additional memory access operations per grid point. For constant stencil coefficients the run time is dominated by memory access and not the computational effort, meaning that the MG benchmark is memory bound.

*Fourier Transform Benchmark* – FT: The FT benchmark solves a partial differential equation by applying a Fast Fourier Transform (FFT) to the original state array and multiplying the result by an exponential. Then an inverse FFT is used to recompute the original solution. Finally, a complex checksum is computed to verify the result [29]. The FFT is dominating the run time of this benchmark. As the implementation in the benchmark uses a blocked variant of the Stockham FFT. This procedure is bound by memory operations, but due to blocking the limiting factor is not directly the memory but rather the cache bandwidth.

*Embarrassingly Parallel Benchmark* – EP: EP is an embarrassingly parallel kernel, which generates pairs of Gaussian random deviates and tabulates the number of pairs in successive square annuli, a problem typical of many Monte Carlo simulations [29]. The EP benchmark is computationally expensive, complex operations such as computation of logarithms and roots make up a big portion of the total run time whereas only very few memory operations are necessary for both random number generation and calculation of the Gaussian pairs. The EP benchmark is compute bound.

*Integer Sort Benchmarks* – IS: This benchmark is able to sort N keys in parallel. The keys are generated by a sequential key generation algorithm. The sorting operations performed in this kernel are important in particle method codes. Both, integer computation speed as well as communication performance are under test [29]. IS requires ranking of an unsorted sequence of N keys. The initial sequence of keys will be generated in a defined sequential manner, but the initial distribution of the keys can have a significant impact on the performance of this benchmark.

*Lower Upper Benchmark* – LU: LU uses a Gauss-Seidel solver for lower and upper triangular systems (regular-sparse, block size $5 \times 5$). It solves flows in a cubic domain, and implements several real-case features, e.g., a dissipation scheme. This benchmark represents the computations associated with the implicit operator of implicit CFD algorithms [29].

*Diagonal Block Matrix Benchmark* – SP and BT: The SP benchmark solves multiple, independent systems of scalar, non diagonally dominant, pentadiagonal equations. BT, however, solves multiple and independent systems of non diagonally dominant, block tridiagonal equations with block size $5 \times 5$. SP and BT are representative of computations associated with the implicit operators of CFD codes. Both, SP and BT, are similar in many respects, the essential difference is the communication to computation ratio [29].

## 6 Results

We have carried out extensive measurements for all NAS benchmarks, the mandelbrot set as well as UTBEST3D and have obtained the following results.

As described, we have analyzed all benchmarks with respect to their plain run time behavior. This means that we compare the results from test runs on the real hardware platform with the run times in the simulation environment. The observation is that generally the prospective run time in the simulation is faster than the execution on the real hardware. This is due to the fact that the simulation does not contain a proper model for cache and memory access penalties as is the case for real hardware. The simulator assumes a constant and too low latency for each memory access and thus yields a lower run time. The degree of deviation is directly coupled to the total amount of memory access within the individual benchmarks. As can be seen in Fig. 3 the total run time can be as much as 4.9 times slower than the prospected run time of the simulation. In our benchmark set this was the case for the CG benchmark of the NPB suite, i.e., a benchmark with low computational complexity. The high deviation becomes not directly visible, as we chose to display the ordinate in log scale to compensate the varying run times across the whole of our benchmarks. We also scaled the benchmarks to obtain an impression about impact on the relative error of the simulation and the execution on real hardware. As the NPB suite offers several problem size classes we analyzed the two smallest classes. This is on one hand due to the fact that the simulation requires about on order of magnitude more to finish than the execution on the real hardware. On the other hand the memory available on the SoC is limited and cannot fit certain benchmarks beyond this class into memory. The results for both classes were consistent with respect to number of memory accesses and the total run times.

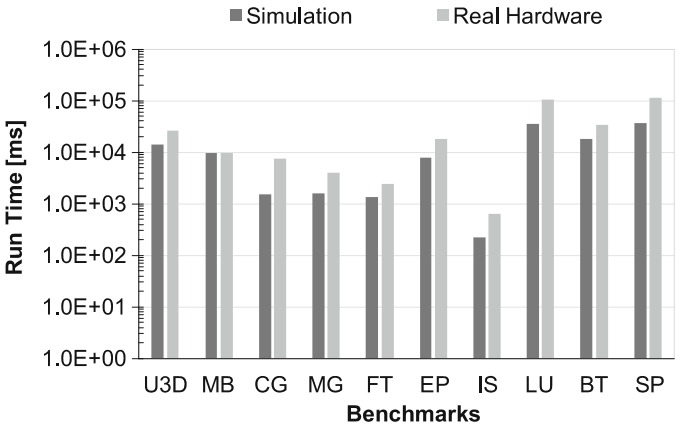



**Fig. 3.** Comparisons of benchmark run times on real hardware and in simulation; ordinate in log scale.

Furthermore, we carried out the benchmarks in parallel using OpenMP and observed a consistent speedup for both, the simulation as well as the execution on the real hardware. The prognosis of the speedup was slightly overshooting in seven out of ten cases, which was to be expected as the run times in the simulation are consistently faster than on the real hardware.

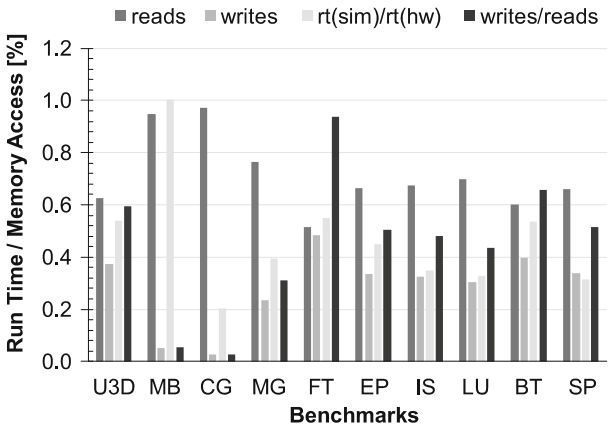



**Fig. 4.** Percentage and relation of read and write operations for each of the benchmarks, along with the quotient of the predicted run time of the simulation, rt(sim) and the actual run time on the hardware, rt(hw).

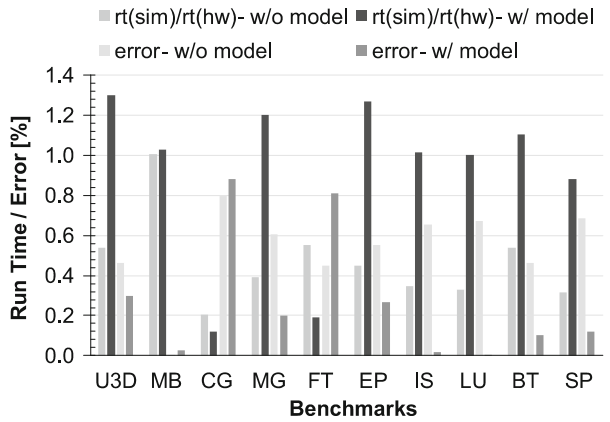



**Fig. 5.** Quotient of simulation run time, rt(sim) and run time on real hardware, rt(hw), in relation with the relative error with and without our memory model.

As we observed, the predicted run time in the simulation to be heavily dependent on the type of kernel and thus the amount of memory access, we instrumented the simulation model in a way that allows to measure the amount of reading and writing memory access of the benchmark runs. Based on our observation (cf. Fig. 4), we derived a memory model improving the prediction accuracy of the run time on the real hardware in seven out of ten cases (cf. Fig. 5) using function (1), which we derived on the basis of our data.

$$rt(hw) = \frac{e^{(1/m)^2}}{ln^{(1/m)^m}} \cdot rt(sim) \tag{1}$$

where: $m$ is the m quotient of the #writes/#reads, $rt(hw)$ is the run time on real hardware, which is to be predicted from, $rt(sim)$, the run time obtained from the simulation.

Using this fitting function, the error margin reduces from a standard deviation of close to 0.4 in the case with no memory model to 0.2, when our model is used for the approximation of the run time on the real hardware.

## 7 Conclusions and Future Work

### 7.1 Conclusion

We carried out a comprehensive analysis of benchmarks on real and simulated hardware in order to analyze the accuracy of simulations with respect to the execution on real hardware platform. We found that a naive simulation model is not sufficient and must be extended with a proper memory model to match the anticipated results for real hardware runs. We developed such a model that allowed us to reduce the inaccuracy in the prediction of the run time on the real hardware based on the simulation. Yet, we show that the combination of an instruction accurate simulation is well suited as a basis for abstract simulations of multi-core embedded systems. Simulations can be carried out in an acceptable time frame and yield sufficiently accurate results.

### 7.2 Future Work

In addition to our first evaluation of one HPC application and benchmark a broader range especially of applications has to be evaluated with respect to their computational and memory requirements. Using these applications and benchmark information we want to extend our memory model with information about parallel execution using OpenMP as well as cache hierarchy its related latency, energy consumption and their interaction. Furthermore, we will expand our models and investigations by the ARMv8 architecture (64-bit). Because instruction accurate simulation itself can never provide precise timing information, beyond modeling it with one or several instructions, such as L1-/L2-Cache accesses, displacement, cache miss rates and page faults we want to introduce concepts of statistical memory modeling. An interesting concept about statistical memory modeling was published by Davy Genbrugge and Lieven Eeckhout in 2009 [30]. They extend statistical simulation methodology to model shared resources in the memory subsystem of multi-processors as shared caches, off-chip bandwidth and main memory. In a next step, we will examine how well suited this approach is for our work and if there is an additional benefit. An interesting article about energy modeling was published by Kerrision and Eder in 2015 [31]. They examine a hardware multi-threaded microprocessor and discusses the impact such an architecture has on existing software energy modeling techniques. Their multithreaded software energy model used with Instruction Set Simulation can yield an average error margin of less

than 7%. This model could be also of benefit for us, even if the ARM architecture we use does not support hardware multi-threading, because the base for the multi-threaded energy model was a single threaded one.

## References


1. Köhler, C.: Enhancing Embedded Systems Simulation: A Chip-Hardware-in-the-Loop Simulation Framework. Vieweg+Teubner research. Vieweg+Teubner Verlag, Wiesbaden (2011)
2. Weaver, V.M., McKee, S.A.: Are cycle accurate simulations a waste of time? In: Proceedings of the 7th Workshop on Duplicating, Deconstructing, and Debunking, June 2008
3. Imperas Software Limited. Official Open Virtual Platforms Website. http://www.ovpworld.org/. Accessed 27 April 2015
4. Schoenwetter, D., Schneider, M., Fey, D.: A speed-up study for a parallelized white light interferometry preprocessing algorithm on a virtual embedded multiprocessor system. In: ARCS Workshops (ARCS), pp. 1–6, February 2012
5. Rajovic, N., Carpenter, P.M., Gelado, I., Puzovic, N., Ramirez, A., Valero, M.: Supercomputing with commodity CPUs: are mobile SoCs ready for HPC? In: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis, SC 2013, pp. 40:1–40:12. ACM, New York (2013). http://doi.acm.org/10.1145/2503210.2503281
6. Göddeke, D., Komatitsch, D., Geveler, M., Ribbrock, D., Rajovic, N., Puzovic, N., Ramirez, A.: Energy efficiency vs. performance of the numerical solution of PDEs: an application study on a low-power ARM-based cluster. J. Comput. Phys. **237**, 132–150 (2013). http://dx.doi.org/10.1016/j.jcp.2012.11.031
7. Rajovic, N., Rico, A., Puzovic, N., Adeniyi-Jones, C., Ramirez, A.: Tibidabo: making the case for an ARM-based HPC System. Future Gener. Comput. Syst. **36**, 322–334 (2014). http://www.sciencedirect.com/science/article/pii/S0167739X13001581
8. ITMC TU Dortmund. Official LiDO Website. https://www.itmc.uni-dortmund.de/dienste/hochleistungsrechnen/lido.html. Accessed 26 March 2015
9. Castro, M., Francesquini, E., Nguele, T.M., Mehaut, J.-F.: Analysis of computing and energy performance of multicore, NUMA, and manycore platforms for an irregular application. In: Proceedings of the 3rd Workshop on Irregular Applications: Architectures and Algorithms, IA3 2013, pp. 5:1–5:8. ACM, New York (2013). http://doi.acm.org/10.1145/2535753.2535757
10. Applegate, D., Bixby, R., Chvátal, V., Cook, W.: The Traveling Salesman Problem: A Computational Study: A Computational Study. Princeton Series in Applied Mathematics. Princeton University Press, Princeton (2011). http://books.google.de/books?id=zfIm94nNqPoC
11. KALRAY Corporation. Official KALRAY MPPA Processor Website. http://www.kalrayinc.com/kalray/products/#processors. Accessed 31 March 2015
12. NVIDIA Corporation. Official NVIDIA SECO Development Kit Website. https://developer.nvidia.com/seco-development-kit. Accessed 31 March 2015
13. Rajovic, N., Rico, A., Vipond, J., Gelado, I., Puzovic, N., Ramirez, A.: Experiences with mobile processors for energy efficient HPC. In: Proceedings of the Conference on Design, Automation and Test in Europe, DATE 2013, pp. 464–468. EDA Consortium, San Jose (2013). http://dl.acm.org/citation.cfm?id=2485288.2485400

14. NVIDIA Corporation. Official NVIDIA Tegra 2 Website. http://www.nvidia.com/object/tegra-superchip.html. Accessed 27 March 2015
15. NVIDIA Corporation. Official NVIDIA Tegra 3 Website. http://www.nvidia.com/object/tegra-3-processor.html. Accessed 27 March 2015
16. Binkert, N., Beckmann, B., Black, G., Reinhardt, S.K., Saidi, A., Basu, A., Hestness, J., Hower, D.R., Krishna, T., Sardashti, S., Sen, R., Sewell, K., Shoaib, M., Vaish, N., Hill, M. D., Wood, D.A.: The gem5 simulator. SIGARCH Comput. Archit. News **39**(2), 1–7 (2011). http://doi.acm.org/10.1145/2024716.2024718
17. Bucy, J.S., Schindler, J., Schlosser, S.W., Ganger, G.R.: The DiskSim Simulation Environment Version 4.0 Reference Manual, May 2008
18. Rosenfeld, P., Cooper-Balis, E., Jacob, B.: DRAMSim2: a cycle accurate memory system simulator. Comput. Architect. Lett. **10**(1), 16–19 (2011)
19. Imperas Software Limited, OVP Guide to Using Processor Models, Imperas Buildings, North Weston, Thame, Oxfordshire, OX9 2HA, UK, January 2015, version 0.5, docs@imperas.com
20. Imperas Software Limited, OVPsim and Imperas CpuManager User Guide, Imperas Buildings, North Weston, Thame, Oxfordshire, OX9 2HA, UK, January 2015, version 2.3.7, docs@imperas.com
21. Altera Corporation. Cyclone V SoC Development Kit User Guide. https://www.altera.com/content/dam/altera-www/global/enUS/pdfs/literaure/ug/ugcvsocdevkit.pdf. Accessed 07 May 2015
22. Imperas Software Limited. Description of Altera Cyclone V SoC. http://www.ovpworld.org/library/wikka.php?wakka=AlteraCycloneVHPS. Accessed 29 April 2015
23. Janapsatya, A., Ignjatovic, A., Parameswaran, S., Henkel, J.: Instruction trace compression for rapid instruction cache simulation. In: Proceedings of the Conference on Design, Automation and Test in Europe, DATE 2007, pp. 803–808. EDA Consortium, San Jose (2007). http://dl.acm.org/citation.cfm?id=1266366.1266538
24. Hardman, J.: Official NAS Parallel Benchmarks Website. http://www.nas.nasa.gov/publications/npb.html. Accessed 23 April 2015
25. Dawson, C., Aizinger, V.: A discontinuous Galerkin method for three-dimensional shallow water equations. J. Sci. Comput. **22**(1–3), 245–267 (2005)
26. Aizinger, V., Proft, J., Dawson, C., Pothina, D., Negjusse, S.: A three-dimensional discontinuous Galerkin model applied to the baroclinic simulation of Corpus Christi Bay. Ocean Dyn. **63**(1), 89–113 (2013). https://www.math.fau.de/fileadmin/am1/users/aizinger/AizingerPDPN2013.pdf
27. Cockburn, B., Shu, C.-W.: The local discontinuous Galerkin method for time-dependent convection-diffusion systems. SIAM J. Numer. Anal. **35**(6), 2440–2463 (1998). http://dx.doi.org/10.1137/S0036142997316712
28. Branner, B.: The mandelbrot set. Proc. Symp. Appl. Math. **39**, 75–105 (1989)
29. Bailey, D.H., Barszcz, E., Barton, J.T., Browning, D.S., Carter, R.L., Dagum, L., Fatoohi, R. A., Frederickson, P.O., Lasinski, T.A., Schreiber, R.S., et al.: The NAS parallel benchmarks - summary and preliminary results. In: Proceedings of the 1991 ACM/IEEE Conference on Supercomputing 1991, pp. 158–165. IEEE (1991)
30. Genbrugge, D., Eeckhout, L.: Chip multiprocessor design space exploration through statistical simulation. IEEE Trans. Comput. **58**(12), 1668–1681 (2009)
31. Kerrison, S., Eder, K.: Energy modeling of software for a hardware multithreaded embedded microprocessor. ACM Trans. Embed. Comput. Syst. **14**(3), 56:1–56:25 (2015). http://doi.acm.org/10.1145/2700104

# Model Checking in Parallel Logic Controllers Design and Verification

Michał Doligalski(✉), Jacek Tkacz, and Tomasz Gratkowski

Institute of Metrology, Electronics and Computer Science, University of Zielona Gora, Zielona Gora, Poland
{m.doligalski, j.tkacz, t.gratkowski}@imei.uz.zgora.pl

**Abstract.** The parallel logic controllers (PLC) developing process can be both, simplified and improved by means of formal methods.

The paper presents approach based on Petri nets specification and model checking techniques used for formal verification, synthesis and implementation. Interpreted petri net (IPN) is selected as a formal behavioural model for parallel logic controllers. It is proposed to use graphical modelling tools for formal behavioural PLC modelling, authors ICPN is such example. The use of common standard, like Petri Net Markup Language (PNML) enables integration with formal analysis tools. The model is simplified (optimised) by means of formal reasoning system (Gentzen). The transformation of the simplified model is made automatically into VHDL description and NuSMV model. The resulting VHLD model can be used for simulation and next for synthesis and implementation. The reliability of the PLC is improved by formal verification. The paper presents the application of the additional specification described in the temporal logic. Model received from reasoning system can be verified by such specification. The formal verification enables to locate deviations from the specification. Proposed approach is useful especially in PLC rapid prototyping approach. The changes in the specification can be verified immediately, towards general specification requirements. Discrepancies between specification and the prototype are localised and can be removed before next iteration. Proposed approach improves visual analysis and fast modifications, ensuring high reliability of the constructed logic controller. The formal methods increases the reliability and quality of the parallel logic controller.

## 1 Introduction

The rapid approach to logic controllers developing process requires tools which will improve the quality of the system and reduce time to market. The reduction of mistakes or defects increase the quality, the precision and early defects detection will decrease time to market. The application of the formal models enables automatic verification and optimisation of the controller. Unified Modelling Language (state machines, activity diagrams) or Petri Nets are popular and acceptable models for behavioral logic controllers specification [10, 16].

Typically the general design flow should be divided into two paths: design and verification (Fig. 1). At each stage, the design should be verified: manually, by simulation, using formal methods or model checking techniques. The feedback from the


J. Janech et al. (eds.), *Proceedings of the 2015 Federated Conference on Software Development and Object Technologies*, Advances in Intelligent Systems and Computing 511,
DOI 10.1007/978-3-319-46535-7_3

verification path to design path enables the defects correction. Typically, each path should be carried out by independent teams or persons. The good example is the situation during model checking, where there is no guarantee that one person (or team) will not make the same mistakes describing the model and model requirements. Especially when the general requirements are too general or ambiguous.

The model requirement should not be limited to project requirement but it should also be extended to technology, security or domain requirements. Especially domain requirements - the knowledge obtained during previous projects is very important. Such requirements are not delivered with the project requirements directly, but as a results of general rules of typical problem or previous solved tasks.

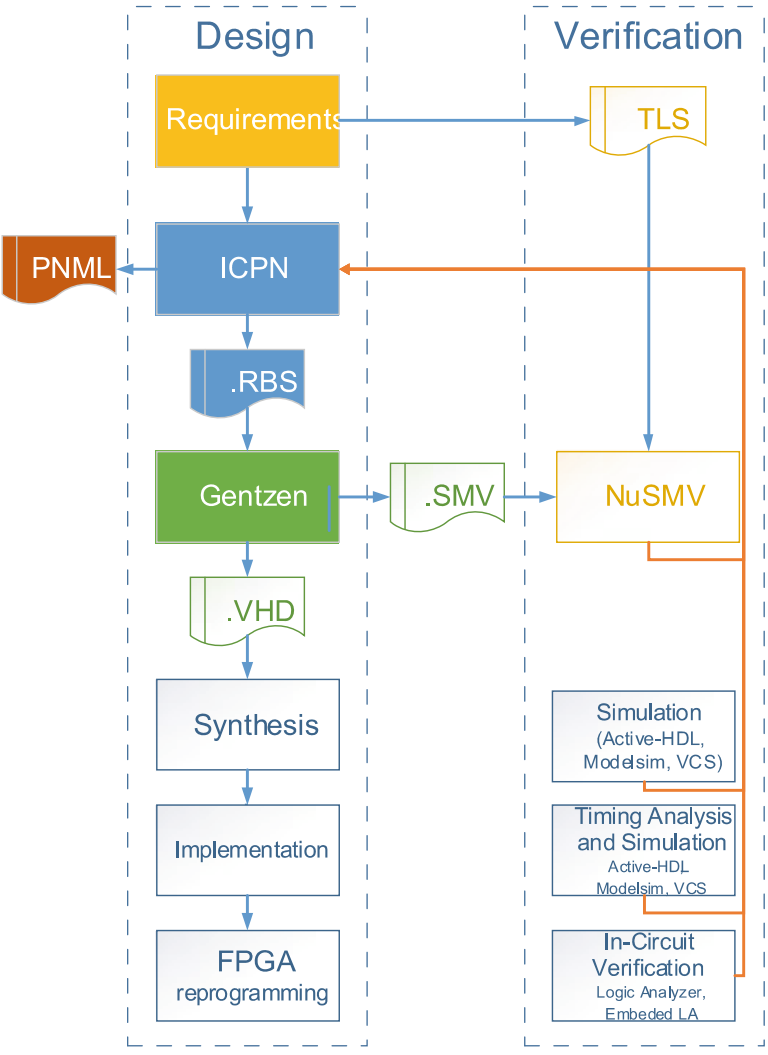



**Fig. 1.** Design flow towards formal verification, optimisation and implementation

The paper presents the formal approach with automatic model and indirect model generation. The designer uses proposed (iCPN) or other PNML compliant tool to describe controller behaviour. The rule based specification (RBS) is generated and loaded by Gentzen system. The input to model checker (SMV file for NuSMV) is generated automatically. The SMV model is verified towards requirements - temporal logic specification (TLS). If all requirements are met, the hardware description language definition (VHDL, Verilog) is also generated automatically. Logic controller description in HDL language can be implemented in the FPGA devices [2, 3, 13, 14]. The other techniques like: exceptions handling [10], optimisation [15], that improve the quality of the controller are also important and closely related to the presented approach.

## 2 Theoretical Introduction

The paper presents a formal approach to parallel logic controllers developing process, where the formal behavioural specification is based on informal requirements. The informal requirements are used for formal specification with Petri Net model, which is one of the most popular formal model for behavioural specification. The application of the PNML (Petri Net Markup Language) standard enables the use of graphical modelling, model checking, optimisation and synthesis tools. The proposed approach use three tools: iCPN, Gentzen and NuSMV. The automatic model transformation enables the use of the proposed approach in rapid parallel logic controllers specification.

### 2.1 Petri Nets

The petri nets are considered as formal model for control system. A simple Petri net [8, 12] is defined as a triple:

$$PN = (P, T, F), \tag{1}$$

where:

$P$ is a finite non-empty set of places, $P = \{p_1, \ldots, p_M\}$

$T$ is a finite non-empty set of transitions, $T = \{t_1, \ldots, t_S\}$

$F$ is a set of arcs from places to transitions and from transitions to places:

$$F \subseteq (P \times T) \cup (T \times P),$$
$$P \cap T = \emptyset.$$

Sets of input and output transitions of a place $p_m \in P$ are defined respectively as follows:

$$\bullet p_m = \{t_s \in T : (t_s, p_m) \in F\},$$
$$p_m \bullet = \{t_s \in T : (p_m, t_s) \in F\}.$$

Sets of input and output places of a transition $t_s \in T$ are defined respectively as follows:

$$\bullet t_s = \{p_m \in P : (p_m, t_s) \in F\},$$
$$t_s \bullet = \{p_m \in P : (t_s, p_m) \in F\}.$$

A marking of a Petri net is defined as a function:

$$\underline{M} : P \rightarrow N$$

It describes a number of tokens $M$ ($p_m$) situated in a place $p_m$. When a place, or a set of places, contains a token it is marked. A transition $t_s$ can be fired if all its input places are marked. Firing of a transition removes tokens from its input places and puts one token in each output place. From automata theory the initial marking $M_0$ is taken. In this case the Petri net is defined as a tupe:

$$PN = (P, T, F, M_0). \tag{2}$$

An interpreted Petri net is enhanced with a feature for information exchange with environment [8]. This exchange is made by using binary signals [9]. Two types of interpreted Petri nets can be distinguished: Moore type and Mealy type.

The Boolean variables occurring in the interpreted Petri net can be divided into three sets:

$X$ is a set of input variables, $X = \{x_1, \ldots, x_L\}$,
$Y$ is a set of output variables, $Y = \{y_1, \ldots, y_N\}$,
$Z$ is a set of internal communication variables, typically it is not used and $Z = \emptyset$.

The interpreted Petri net has a condition (guards) $\varphi_s$ associated to every transition $t_s$. The condition $\varphi_s$ is defined as Boolean function of the input or internal variables form sets $X$ and $Z$. In particular case, the condition $\varphi_s$ can be equal to 1 (always true). The guarded (labeled) transition $t_s$ can be fired if all its input places ($\bullet t_s$) are marked and current value of corresponding Boolean function $\varphi_s$ is equal to 1. In case of Moore type interpreted Petri net, $\psi_m$ is an elementary conjunction of some output variables from the set $Y$. Each of such conjunction $\psi_m$ is associated to some of places $p_m$. If the place $p_m$ is marked the output variables from corresponding conjunction $\psi_m$ are being set otherwise they are being reset. When transition $t_s$ is fired, variables from corresponding conjunction $\psi_s$ are being set if their affirmation belongs to this conjunction and they are being reset if their negation belongs to this conjunction. The value of non-used variables in corresponding conjunction $\psi_s$ remains unchanged. Figure 2 presents example of interpreted Petri net. The simple net will be used in next sections to present Gentzen calculi (formal method of specification), calculi normalization (Sect. 2.4 and RBS generation Sect. 2.3).

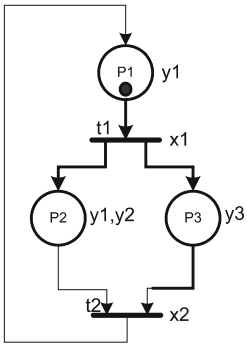



**Fig. 2.** The interpreted Petri net example

## 2.2 iCPN System

The iCPN tool was developed because there was a need for system dedicated to advanced Petri nets analysis, coloring algorithms and decomposition into automata components in particular. The marked analysis shown that academic and commercial system dedicated to Petri Nets are limited to simple functionality. The Tina and WoPed software is such an example. There is possibility of net simulation and basic properties analysis. Also it is possible to determine concurrency graphs as well as marking graphs but that software tools operates on Petri nets simple class. The software adaptation to logic controllers developing process is rather hard. The Jensen's CPN software is the most advanced tool, where the analysis of the interpreted Petri nets is possible. The biggest disadvantage of the CPN software is the interface, which differs significantly from the generally accepted standards.

The functionality and interoperability of the iCPN software is presented on Figure 3. The PNML standard is used for data exchange with other system. The standard defines main graph nodes like: place, transition, arc, initial marking. Also the layout of the model can be described. PNML standard doesn't contain any element for interpreted net description however, the standard is extensible and dedicated extensions can be provided. The special marker (node) <*toolspecific*/> enables extension specific for particular CAD tool. Such an approach guarantees the possibility of extensions and at the same time compatibility with other tools. The information included in special marker, will be read and analysed only by tools which will need it. Thanks to that there is a possibility of the Petri net modelling including all extensions required for logic controllers. The PNLM net description with additional data can be loaded into other tolls. Additional information will be omitted but still toll-specific functionality will be possible. The good example is dynamic simulation or validation provided by WoPed or Tina software.

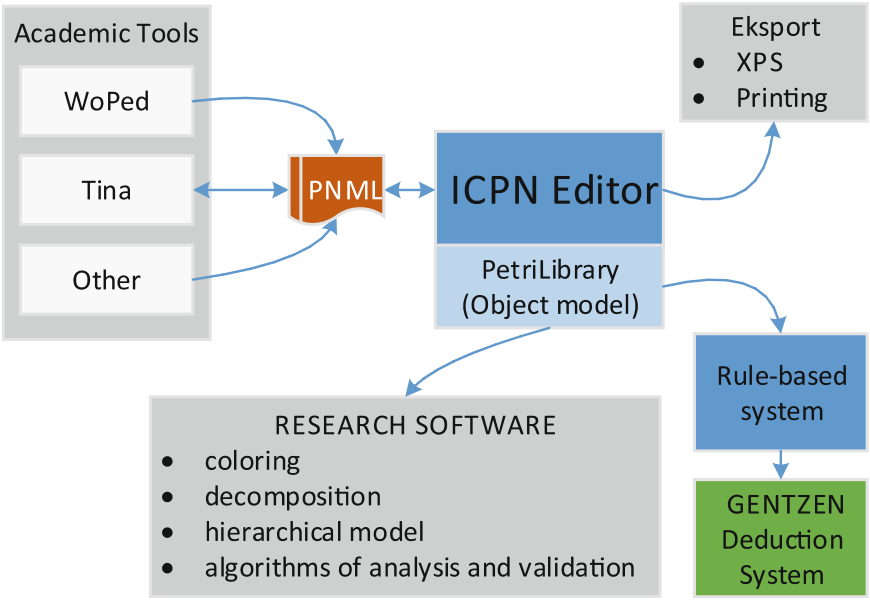



**Fig. 3.** iCPN software functionality and integration

### 2.3 Rule-Based Specification

The GENTZEN system accepts rule-based specification in the form of sequent calculi as input data. The transition-based description is used for Petri net specification. The sequent predecessor will define the product of the transition input places and transition condition, the successor product of the places sum with generated outputs (3).

$$\prod_{i=1}^{n}(p_i) \wedge condition \vdash \prod_{j=1}^{k}\left(p_j \vee \sum_{l=1}^{m}(y_l)\right) \tag{3}$$

The automata design requires the register signal definition (REG) which corresponds to Petri net places. Also the incitation of the initial places and transition are required (INIT). The example of the Petri net specification according to Eq. (3) is presented in Listing 1.

```
IN: x1, x2;
OUT: y1, y2, y3;
REG: p1, p2, p3;
INIT: p1;

p1*x1|-(p2+y1+y2)*(p3+y3);
p2*p3*x2|-p1+y1;
```

**Listing 1.** Rule Based Specification example

The RBS description is used as an input into GENTZEN system which generates both SMV specification and VHDL description.

### 2.4 Gentzen System

The sequent is a formalized statement used for deduction and calculi [4]. In the sequent calculus, sequents are used for specification of judgment that are characteristic to deduction system. The sequent is defined as a ordered pair ($\Gamma$, $\Delta$), where $\Gamma$ and $\Delta$ are finite sets of formulas, and $\Gamma = \{A_1, A_2, \ldots, A_m\}$, $\Delta = \{B_1, B_2, \ldots, B_n\}$. Instead of ($\Gamma$, $\Delta$) it is used notation with use of turnstile symbol $\Gamma \vdash \Delta$.

$\Gamma$ is called the antecedent and $\Delta$ is a succedent of the sequent. The sequent $\Gamma \vdash \Delta$ is satisfiable for the valuation $v$ iff for the same valuation $v$ the formula $\bigwedge_{i=1}^{m} A_i \rightarrow \bigvee_{j=1}^{n} B_j$ is satisfied.

The Gentzen system is a CAD tool dedicated to Petri nets verification and optimisation. It implements sequent logic and rules of elimination of logic operators. For each operator (negation, disjunction, conjunction, implication and equivalency), there are defined two rules of its elimination. First rule is used when the operator is located in antecedent and the second one when it is located in succedent. For each logic operator there are two rules of the elimination. As an example the rule of the disjunction operator elimination will be presented (4), it is one of the ten potential rules that can be used. If the main logical operator in sequent is disjunction located in succedent then two sequents will be produced. First sequent will contain the first comma separated argument of disjunction in succedent, and the second sequent will contain the second argument of disjunction also in succedent (4).

$$\frac{\Theta,\ \boldsymbol{\Phi} \vee \boldsymbol{\Psi},\ \Gamma \vdash \Pi}{\Theta,\ \boldsymbol{\Phi},\ \Gamma \vdash \Pi \quad \Theta,\ \boldsymbol{\Psi},\ \Gamma \vdash \Pi} \tag{4}$$

If the main logical operator in sequent is disjunction located in antecedent then in the next step only one sequent is produced and a logic operator is replaced by the comma in antecedent. Both of arguments of disjunction are separated by the comma in antecedent (5).

$$\frac{\Lambda \vdash \Theta,\ \boldsymbol{\Phi} \vee \boldsymbol{\Psi},\ \Gamma}{\Lambda \vdash \Theta,\ \boldsymbol{\Phi}\ \boldsymbol{\Psi}} \tag{5}$$

The elimination process is repeated, while only normalized sequents are received (Fig. 4). The normalized sequent is the sequent without any logical operators. The sequent is a tautology iff it has the same formula in an antecedent and a succedent. The located tautology sequents could be removed from further normalization. Iff all normalized sequents are tautologies the analysed root sequent is also a tautology. When one of leafs in deduction tree (Fig. 4) is not a tautology it means that it is a counterexample for analyzed sequent. The consensus method is included into formal deduction (Gentzen cut rule).

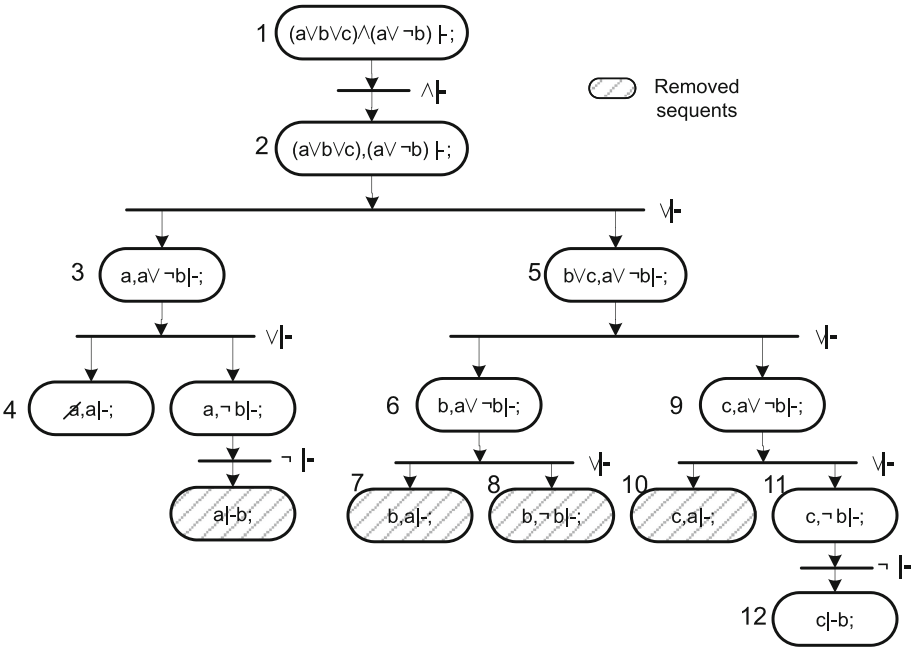



**Fig. 4.** Example of sequent normalization

The current version of implementation of Gentzen system "GENTZEN v6.7.2" accepts many types of logical operators, taken from Palasm, VHDL, Verilog, Linear logic and NuSMV [13]. Thanks to the PNML standard, the system enables integration with other tools. The results of the developing process can be transformed to NuSMV model. Also the implementation to VHDL, VERILOG description is possible. The connection of the verification and synthesis in one tool accelerates the developing process.

The example of the normalisation of the rule-based specification from Listing 1 which describes interpreted Petri net from Fig. 3.

$$\begin{array}{l} p_1, x_1 \mid - p_2, y_1, y_2; \\ p_1, x_1 \mid - p_3, y_3; \\ p_2, p_3, x_2 \mid - p_1, y_1; \end{array} \tag{6}$$

The redundant logic formula can be removed or partially reduced as a result of the standardization process. As the presented example is simple, the logic formulas were only normalised into the group of simple sequent, devoid logic conjunctions.

The particular sequent can be easily found in a new, normalised specification. The normalisation is necessary for next steps - transformation into NuSMV model and HDL synthesizable specification.

As an addition to presented process, the linear logic can be used to generate formal description of the parallel logic controller. The linear logic specification [5, 11]

describes the same Petri net model (Eq. 3), and is coherent with the presented one but it is more transparent. The specification in the form of sequent linear logic was presented in Eq. 7 (the REG and INIT definition was intentionally omitted).

$$\begin{gathered} \vdash (p_1 \otimes x_1) - \circ((p_2 \oplus y_1 \oplus y_2) \otimes (p_3 \oplus y_3)); \\ \vdash (p_2 \otimes p_3 \otimes x_2) - \circ(p_1 \oplus y_1); \end{gathered} \tag{7}$$

The presented symbolic deduction system was used for solving the combinatorial digital technology problems [15]. As an alternative to the approach presented in this paper, the Gentzen system can be used to formal verification by checking the dependency between minimal traps and syphons. As a result the information about liveness is returned with the counterexample in case of net fault. The counterexample can be used for failure localisation and correction. Also it can be used for the specification reduction, logic simulation and verification of the implementation relative to full or partial specification. As a result of carried out research the other applications of the Gentzen in the area of concurrent logic controllers design were proposed [1].

### 2.5 NuSMV System

The controller specification can be verified by means of model checking techniques [6]. The paper presents approach based on Petri nets specification but other formal models like UML diagrams can be verified formally as well [7]. The model verification enables early project verification. From the designer point of view it enables fast changes to the project and verification if the provided changes are compliant with the requirement or not. Especially the model verification techniques is useful when the model for NuSMV tool will be generated automatically. It also eliminates mistakes, possible when the transformation is made manually. The model verification at the specification phase limits its propagation into next phases. Thanks to that the design is more reliable and the developing process time can be reduced. The paper presents the example of automatic NuSMV model generation and verification (Sect. 3.2).

The verification of the model is done in relation to the requirements. The logic controller is specified in the NuSMV format. There are two methods of specification: lineal temporal logic (LTL) and computation tree logic (CTL). The requirements are based on informal specification. It can describe overall requirements based on specification, technical requirements related to particular technical facilities or domain requirements. The model checking techniques compare model and requirements and if the requirements are not fulfilled then counterexample returns as a result. The counterexample presents the place of inconsistency, the guide to a designer. The verification is carried out only for specified requirements, the other ones that were unspecified will be not verified. It means that the in the case of incomplete requirements the positive result of the verification will not guarantee the proper work of the controller.

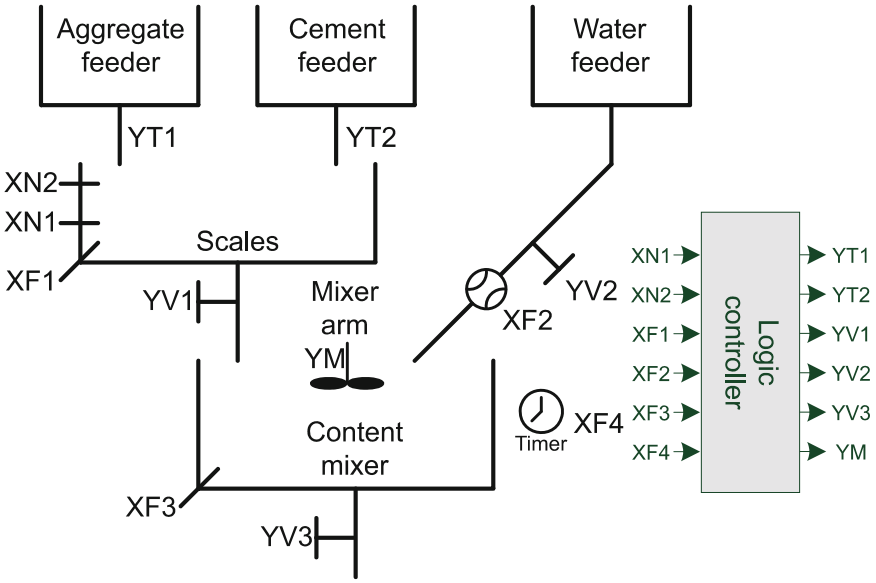



**Fig. 5.** Mechanical part of discrete control system

## 3 Formal Approach to PLC Design

This section presents practical approach to parallel logic controller design by means of formal models and method. The example of behavioural specification will be transformed automatically into the rule-based specification (RBS). The specification will be then verified by means of NuSMV tool. After successful verification automatic synthesis will result logic controller HDL description.

### 3.1 Control System

The controlled plant (Fig. 5) consists of three feeders, scales and content mixer. The aim of the plant is to produce concrete with cement, aggregates and water. Scales is used to measure required amount of friable ingredients. The proper amount of water is delivered by means of water feeder and flow meter $XF$ 2.The components are mixed according to the recipe for a limited period of time, timer $XF$ 2 steers the mixing time. The logic controller has six inputs $\{XN_1, XN_2, XF_1, XF_2, XF_3, XF_4\}$ and six outputs $\{YT_1, YT_2, YV_1, YV_2, YV_3, YM\}$.

The example of control-interpreted Petri net, which describes the behavioural of control system is presented on Fig. 6. The Petri net places $\{P_1, P_2, \ldots, P_{11}\}$ stand for the local states of concurrent state machine. The transitions $t_1 \ldots t_9$ describe events in terms of local changes inside the Petri net state space. Boolean expressions $XN_1 \ldots XF_4$ called guards give the external conditions for transitions to be enabled and fired.

**Table 1.** List of transitions and guards

| Transition | Guard | Interpretation of guard |
|---|---|---|
| $t1$ | XN1 | Required value of aggregate is reached |
| $t2$ | 1 | Always true |
| $t3$ | XF1 | The scale is empty |
| $t4$ | XN2 | Required value of cement is reached |
| $t5$ | XF1 | The scale is empty |
| $t6$ | 1 | Always true |
| $t7$ | XF4 | Ingredients are intermixed |
| $t8$ | XF3 | Cement mixer is empty |
| $t9$ | XF2 | Required value of cement is reached |

The colored coordination places $P_{12}$ and $P_{13}$ in Fig. 6 are optional. The guarded events are strongly related to transitions of the net (Table 1). The Moore type outputs $Y\,T_1$ … $YM$ are attached to places (Table 2).

### 3.2 Formal Verification

Parallel logic controller Behavioral specification in the form of interpreted Petri net presented in Fig. 6 was edited in iCPN tool and next saved into rule-based format Listing 2. One of the most important advantage of the presented approach is that the rule-based specification is generated automatically. There is no need to manually describe the specification. Alternative approach [7] requires the use of an intermediate format. There is no possibility of automatic conversion from commonly acceptable formats like PNML.

**Table 2.** List of places and outputs

| Place | Output | Interpretation of place |
|---|---|---|
| $P1$ | YT1 | First dozing of cement |
| $P2$ | – | Waiting |
| $P3$ | – | Waiting |
| $P4$ | YV1 | First emptying the scale |
| $P5$ | YT1 | Second dozing of cement |
| $P6$ | YV1 | Second emptying the scale |
| $P7$ | – | Waiting |
| $P8$ | – | Waiting |
| $P9$ | YV2 | Dosing of water |
| $P10$ | YM | Mixing of compounds |
| $P11$ | YV3 | Emptying the mixer |

```
IN: XN1, XN2, XF1, XF2, XF3, XF4;
OUT: YT1, YT2, YV1, YV2, YV3, YM;
REG: P1, P2, P3, P4, P5, P6, P7, P8, P9, P10, P11;
INIT: P1, P2, P9;

P1*XN1|-P2;
P2*P3|-P4+YV1;
P4*XF1|-P5+YT2;
P5*XN2|-P6+YV1;
P6*XF1|-(P1+YT1)*P7;
P7*P8|-P10+YM;
P10*XF4|-P11+YV3;
P11*XF3|-P3*(P9+YV2);
P9*XF2|-P8;
```

**Listing 2.** Rule Based Specification for control system

The Gentzen system generates logic controlled model by means of temporal logic (Listing 3). The model will be verified towards design requirements fulfilment. Model describes petri net structure. Places, input and output signals are treat as boolean variables. The transitions and are described using *next() := case…esac*; construct.

In NuSMV tool model is verified towards design requirements meet. The requirements for the control system were presented on Fig. 4. There are three groups of requirements. Firs group checks the state reachability. If it's not possible to reach the particular state, the net has a structural defect. Second group checks the negation of a conjunction of states marking, states that should never be active at the same time are verified. For example, it is forbidden to empty and fill of the tank at the same time. It is also forbidden to run the mixer when the vat is empty. Third group describes sequence of events. The relationship of cause and effect between the occurrence of the event and its aftermath is verified. An example is being checked if the tank valve is closed after unloading.

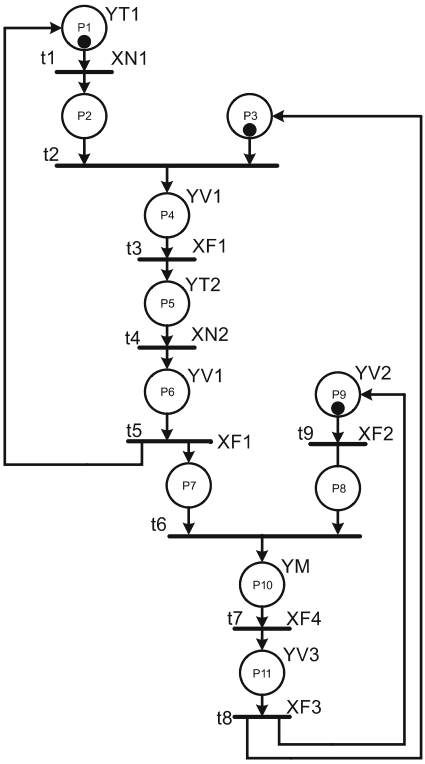



**Fig. 6.** Petri net model

```
MODULE main
VAR
     XF1 : boolean; (...) XN2 : boolean;
     YM : boolean;    (...) YV3 : boolean;
     P1 : boolean;    (...) P11 : boolean;
ASSIGN
     init(P1) := TRUE;   init(P2) := FALSE;
     (...)                 init(P9) := TRUE;
     init(P10):= FALSE;  init(P11):= FALSE;
     init(XF1):= FALSE; (...) init(XN2):= FALSE;
     init(YM) := FALSE; (...) init(YV3):= FALSE;

next(P1) := case
  P1 & XN1  : FALSE;
  P6 & XF1  : TRUE;
  TRUE : P1;
esac;
```

**next**(P2) := **case**

```
    P2 & P3 : FALSE;
    P1 & XN1: TRUE;
    TRUE : P2;
esac;
(...)
next(P11) := case
    P11 & XF3 : FALSE;
    P10 & XF4 : TRUE;
    TRUE : P11;
esac;

next(XF1) := case
    P4 | P6 :{FALSE,TRUE};
    TRUE : FALSE;
esac;
next(XF2) :=case
    P9 : {FALSE,TRUE};
    TRUE : FALSE;
esac;
(...)
next(XN2):=case
    P5 : {FALSE,TRUE};
    TRUE : FALSE;
esac;

next(YM):=case
    P10 : TRUE;
    TRUE : FALSE;
esac;
next(YT1) := case
    P1 : TRUE;
    TRUE : FALSE;
esac;
(...)
next(YV3):=case
    P11 : TRUE;
    TRUE : FALSE;
esac;
```

**Listing 3.** NuSMV model generated by means of GENTZEN system

The list of assertions can describe the requirements at the general level or it can go deep in details. The result of the formal verification is boolean. Each assertion is verified and TRUE or FALSE result is given. IF the result is FALSE, the counterexample is given. It's a very useful feature, because it directly indicates the place of failure. A designer can correct the incompatibility with the specification at an early stage without the need of additional simulation.

```
CTLSPEC  EF  P1;
(...)
CTLSPEC EF  P11;
CTLSPEC AG !(YT1 & YV1 );
CTLSPEC AG !(YT2 & YV1 );
CTLSPEC AG !(XF1 & XN1 );
CTLSPEC AG !(XF1 & XN2 );
CTLSPEC AG !(YV1 & YV3 );
CTLSPEC AG !(YM  &  XF3);
CTLSPEC AG !(YT1 & YT2 );
CTLSPEC AG !(YV3 & YV2 );
CTLSPEC AG (XN2 -> AF !YT2);
CTLSPEC AG (XF1 -> AF !YV1);
CTLSPEC AG (XF2 -> AF !YV2);
CTLSPEC AG (XN1 -> AF YT1);
```

**Listing 4.** Requirements based on informal specification

### 3.3 Synthesis to HDL

The normalised specification in the form of sequent is automatically transformed into hardware description language VHDL (Listing 5), the Verliog specification is also possible. From the specification definition section the input, output signal and internal register signal are generated (*P*1 … *Pn*). The *RESET* signal initialises the module according to Petri net initial marking. This data is also based on the specification section. The architecture of the module describes transition between Petri net Places. The transitions are fired in accordance with the rising edge of the clock signal.

The complex logic formula is based on the normalised specification. The reverse deduction based on logic conjunction is performed. This process is reversed in relation to the process described in Sect. 2.4.

```
library IEEE ;
use IEEE.stdlogic 1164.all;

entity controller is
port(
  CLK,RESET: in  STD LOGIC;
  XF1: in STD LOGIC;
  XF2: in STD LOGIC;
  XF3: in STD LOGIC;
  XF4: in STD LOGIC;
  XN1: in STD LOGIC;
  XN2: in STD LOGIC;
  YM: out STD LOGIC;
  YT1: out STD LOGIC;
  YT2: out STD LOGIC;
  YV1: out STD LOGIC;
  YV2: out STD LOGIC;
  YV3: out STD LOGIC
) ;
```

end controller;

```
architecture controller_arch of controller is
signal P1,P10,P11,P2,P3,P4,P5,P6,P7,P8,P9:STD LOGIC;
begin
FF : process (CLK, RESET)
begin
if RESET = '1 ' then
  P1<='1'; P10<= '0'; P11<='0'; P2<='0'; P3<='1';
  P4<='0'; P5<='0'; P6<='0'; P7<='0'; P8<= '0';
  P9<= '1';
elsif rising_edge (CLK) then
  P1<=P1 ; P10<=P10 ; P11<=P11 ; P2<=P2 ; P3<=P3 ;
  P4<=P4 ; P5<=P5 ; P6<=P6 ; P7<=P7 ; P8<=P8 ; P9<=P9 ;
  if P1 = '1 ' and XN1 = '1 ' then
  P1<= '0 '; P2<= '1 ';
  end if;
  if P2 = '1' and P3 = '1' then
    P2<= '0'; P3<= '0'; P4<= '1';
  end if;
  if P4 = '1' and XF1 = '1 ' then
    P4<= '0'; P5<= '1';
  end if;
  if P5 = '1' and XN2 = '1' then
    P5<= '0'; P6<= '1';
  end if;
```

```
  if P6 = '1' and XF1 = '1' then
    P6<= '0'; P1<= '1'; P7<= '1';
  end if ;
  if P7 = '1' and P8 = '1' then
    P7<= '0'; P8<= '0'; P10<= '1 ';
  end if ;
  if P10 = '1' and XF4 = '1' then
    P10<= '0'; P11<= '1';
  end if ;
  if P9 = '1' and XF2 = '1' then
    P9<= '0'; P8<= '1';
  end if ;
  if P11 = '1' and XF3 = '1' then
    P11<= '0'; P3<= '1'; P9<= '1';
  end if ;
  if P9 = '1' and XF2 = '1' then
    P8<= '1'; P9<= '0';
end if ;
end if ;
end process ;
YM<=P10; YT1<=P1; YT2<=P5; YV2<=P9;
YV3<=P11; YV1<=P4 or P6;
end controller_arch;
```

**Listing 5.** The Petri net description in VHDL language

## 4 Conclusion

The idea of rapid parallel logic controllers prototyping by means of formal methods was presented in this paper. The application of the additional specification with the requirements specification for model checking enables fast and constant verification of the implementation model. Any move beyond the specifications, or lack of specification of the particular item will be immediately captured by the model checking tools. As a result, the counterexample will indicate exactly the places where a conflict occurs. The proposed approach use accepted standard for Petri net specification (PNML), there is a possibility of omitting PNML specification and using logic sequent which specifies the controller directly. It's proposed that other formal models can be transformed using model transformation methods (QVT) into Petri net model, which will be intermediate model transparent for a designer. The automatic rule-based specification generation limits the errors and makes the presented method usable, especially in RSP techniques. Such information greatly improves the process of designing and prototyping concurrent logic controllers and will ensure the compliance of the implementation with the specification.

## References


1. Adamski, M., Tkacz, J.: Formal reasoning in logic design of reconfigurable controllers. In: Proceedings of 11th IFAC/IEEE International Conference on Programmable Devices and Embedded Systems PDeS 2012, Brno, Czech Republic, pp. 1–6 (2012)
2. Doligalski, M.: Behavioral specification of the logic controllers by means of the hierarchical configurable Petri nets. In: Proceedings of 11th IFAC/IEEE International Conference on Programmable Devices and Embedded Systems - PDeS 2012, Brno, Czechy, pp. 80–83 (2012)
3. Doligalski, M., Adamski, M.: UML state machine implementation in FPGA devices by means of dual model and Verilog. In: 2013 11th IEEE International Conference on Industrial Informatics (INDIN), pp. 177–184 (2013)
4. Gallier, J.H.: Logic for Computer Science: Foundations of Automatic Theorem Proving. Harper & Row Publishers, New York (1985). http://www.cis.upenn.edu/∼jean/gbooks/logic.html
5. Girault, F., Pradin-Chezalviel, B., Kunzle, L., Valette, R.: Linear logic as a tool for reasoning on a Petri net model. In: INRIA/IEEE Symposium on Emerging Technologies and Factory Automation, ETFA 1995, vol. 1, pp. 49–57 (1995)
6. Grobelna, I.: Formal Verification of Logic Controller Specification by Means of Model Checking. Lecture Notes in Control and Computer Science. University of Zielona Góra Press, Zielona Góra (2013)
7. Grobelna, I., Grobelny, M., Adamski, M.: Model checking of UML activity diagrams in logic controllers design. In: Zamojski, W., Mazurkiewicz, J., Sugier, J., Walkowiak, T., Kacprzyk, J. (eds.) Proceedings of the Ninth International Conference on DepCoS-RELCOMEX. AISC, vol. 286, pp. 233–242. Springer, Heidelberg (2014)
8. Karatkevich, A.: Dynamic Analysis of Petri Net-Based Discrete Systems. Lecture Notes in Control and Information Sciences, vol. 356. Springer-Verlag, Berlin (2007)
9. Kozłowski, T., Dagless, E., Saul, J., Adamski, M., Szajna, J.: Parallel controller synthesis using Petri nets. In: IEEE Proceedings – Computers and Digital Techniques, vol. 142, no. 4, pp. 263–271 (1995)
10. Leroux, H., Andreu, D., Godary-Dejean, K.: Handling exceptions in Petri net based digital architecture: from formalism to implementation on FPGAs. IEEE Trans. Ind. Inform. **99**, 1 (2015)
11. Lincoln, P., Shankar, N.: Proof search in first-order linear logic and other cut-free sequent calculi. In: Symposium on Logic in Computer Science, LICS 1994, pp. 282–291 (1994)
12. Murata, T.: Petri nets: properties, analysis and applications. Proc. IEEE **77**(4), 541–580 (1989)
13. Pereira, F., Gomes, L.: Automatic synthesis of VHDL hardware components from iopt Petri net models. In: Industrial Electronics Society, IECON 2013 - 39th Annual Conference of the IEEE, pp. 2214–2219, November 2013
14. Rawski, M., Tomaszewicz, P., Borowik, G., Łuba, T.: 5 Logic synthesis method of digital circuits designed for implementation with embedded memory blocks of FPGAs. In: Adamski, M., Barkalov, A., Węgrzyn, M. (eds.) Design of Digital Systems and Devices. LNEE, vol. 79, pp. 121–144. Springer, Heidelberg (2011)

15. Tkacz, J., Adamski, M.: Logic design of structured configurable controllers. In: Proceedings of IEEE 3rd International Conference on Networked Embedded Systems for Every Application NESEA 2012, Liverpool, United Kingdom, p. 6 (2012)
16. Wiśniewski, R., Stefanowicz, Ł., Bukowiec, A., Lipiński, J.: Theoretical aspects of Petri nets decomposition based on invariants and hypergraphs. In: Park, J.J.(J.H.), Chen, S.-C., Gil, J.-M., Yen, N.Y. (eds.) Multimedia and Ubiquitous Engineering. LNEE, vol. 308, pp. 371–376. Springer, Heidelberg (2014)

# Fuzzy Logic for Optimized Path Establishment in Optical Networks

Miroslav Dulik and Gabriel Cibira(✉)

Faculty of Electrical Engineering, Institute of Aurel Stodola,
University of Zilina, Liptovsky Mikulas, Slovakia
{dulik, cibira}@lm.uniza.sk

**Abstract.** This article brings new approach to optical network optimization. When routing data in fast optical networks, defined parameters of each path are taken into account and evaluated to find optimized path for data routing. Our fuzzy logic approach implements fuzzy weighed parameterization to achieve optimal path setup in decision process, over WDM network structure. Complex path evaluation and dynamic routing using fuzzy logic can provide better results and faster decisioning for data transport in optical networks.

## 1 Introduction

Routing in WDM optical networks has been widely solved for different types of optical networks. New solutions and approaches are based on measuring and evaluation of several parameters, like:

- eye diagram (Q-factor)
- evaluation of optical signal-to-noise ratio (OSNR)
- power and spectrum dissipation
- evaluation of channel state
- bit error rate (BER)

By default, fiber networks provide node-to-node connections over long distances. They include optical fibers, optical transmitters/receivers and amplifiers (if necessary). If routing on higher layer is necessary, some optical-electrical and optical-electrical converters must be deployed.

Cisco forecasted increasing data traffic rate of about 21% per year as well as continually growing large number of mobile non-PC devices [1]. In WDM systems, bandwidth is met by employing multiple carrier wavelength channels over a fiber. It respects transmission limits [2], depending on route components [3], modulation methods, operational modes and wavelength routing control abilities. Dynamic flexible control approaches bring better bandwidth agility [4–10]. The most used routing and wavelength assignment algorithms implement various methods to improve overall throughput, like: a shortest path choosing within user-specified or network-specified constraints, alternate routing creation, etc. Modern systems embrace highly sophisticated automated processes, re-routing, advanced modulations [11, 12], wavelength optimization [8, 10, 13–17] backup linking etc.


J. Janech et al. (eds.), *Proceedings of the 2015 Federated Conference on Software Development and Object Technologies*, Advances in Intelligent Systems and Computing 511,
DOI 10.1007/978-3-319-46535-7_4

In mesh networks, fuzzy control can provide these functions: wavelength reconfiguration (based on defined parameters) [6, 9, 18], routes updating [19] and load balancing to improve overall throughput. Cognitive light path optimization using OSNR is proposed in [20, 21]. Such an important parameter must be considered during fiber optics planning process [22, 23], too.

In this paper, we focus on data transportation routing optimization in WDM optical network containing several transfer nodes and paths, deliberating various path parameters. We used it for near future status estimation, over real network. We bring data routing process implementing fuzzy logic evaluation and decision processing, by link state parameters.

## 2 Model of Real Network

To achieve preset aim, we have considered SANET (Slovak Academic NETwork) topology [24] that covers educational and government sectors among 23 Slovak towns. SANET consists of two inter-linked main routes providing full redundancy with 5 ms maximum delay. The national connectivity is linked to Slovak Peering Center SIX placed in Computer Center of the Slovak Technical University (CVT STU) in Bratislava, by Ethernet links at 10 Gbps speed. SANET infrastructure is based on leased dark fibers terminated by Cisco Catalyst gigabit switches. The foreign connectivity is realized through leased dark fiber at the speed 10 Gbps to ACOnet node in Vienna (AT), CESNET node in Brno (CZ), Pioneer node in Bielsko-Biala (PL) and local link to GEAT PoP in Bratislava (1 Gbps) and GTS PoP in Bratislava (2 Gbps) [24].

Current topology of the SANET network is shown in Fig. 1 [24]. As we follow fuzzy optimal data routing aim among main nodes placed in Bratislava, Trnava, Nitra, Zvolen, Ruzomberok, Presov and Kosice, the initial topology is simplified to its


**Fig. 1.** SANET network [24]

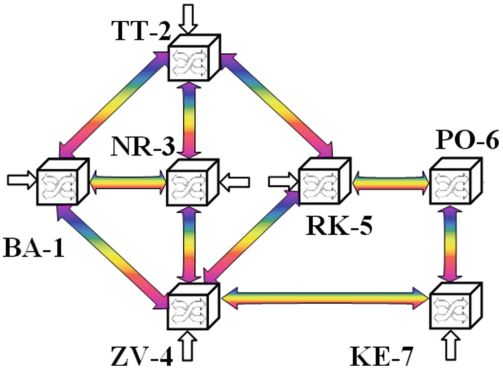



**Fig. 2.** SANET network – main nodes

topological skeleton, Fig. 2. Main nodes are responsible enough for correct routing among covered towns.

## 3 Optical Network Parameters

When evaluating and calculating optimal path, several parameters describing state of the path are taken into account.

### 3.1 Bit Error Rate, BER

When transmitting in any binary communication channel, true or false LOG1 or LOG0 detections occur at the channel output. False detections are often caused by signal loss/damping, inter-channel interferences, or degradation processes within the link. Thus, conditional probability P(1|0) is stated when false decision LOG1 is accepted or, conditional probability P(0|1) is stated when false decision LOG0 is accepted at the channel output.

$$\begin{aligned} BER &= p(1)P(0|1) + p(0)P(1|0) \\ &= \frac{P(0|1) + P(1|0)}{2} + \frac{erfc\left(\frac{I_1 - I_{THR}}{\sigma_1\sqrt{2}}\right) + erfc\left(\frac{I_{THR} - I_1}{\sigma_0\sqrt{2}}\right)}{4} \end{aligned} \quad (1)$$

where

| | |
|---|---|
| p(*1*) | is detection probability of LOG1, |
| p(*0*) | is detection probability of LOG0, |
| $I_1$ | is mean diode photocurrent by LOG1, |
| $I_0$ | is mean diode photocurrent by LOG0, |
| $I_{THR}$ | is decision threshold level of photocurrent, |
| $\sigma_1$ | is photocurrent standard deviation by LOG1, |
| $\sigma_0$ | is photocurrent standard deviation by LOG0 |

### 3.2 Optical Signal – to – Noise Ratio, OSNR

The OSNR parameter defines ratio between the signal power and the noise power in a given bandwidth. This parameter is measured with an optical spectrum analyzer.

### 3.3 Link Power, Power

The link power parameter defines signal output power level at the link end. Usually, it is measured by terminal equipment while it can optimize link power level.

## 4 Control System for Path Evaluation

As the control parameters are defined in the previous part, we use them to obtain the optimal path. Proposed complex optimization algorithm contains several steps to find the most appropriate path between chosen input and output nodes, among list of transport links. Control system properties are:

- Basic parameters (BER, OSNR, Power) measurement is required for each path. Besides of basic parameters (system inputs) use, the system uses its mean value and standard deviation for optimal path fuzzy calculation.
- These basic parameters are processed by fuzzy logic control subsystem and its relationships assessed for each particular path. Here, membership functions and fuzzy rules are defined by experienced user to obtain optimal links evaluation (current cost along <0,100>).
- Fuzzy logic subsystem creates available paths full list between chosen input and output nodes. As a result, final data transport path is selected from starting node to end node through selected nodes.

## 5 Fuzzy Logic

Measured parameters, BER, OSNR and Power, are continuously fed by each node into central fuzzy logic control subsystem. Its centralized diagnostics evaluates these parameters, calculates its statistical means during measuring interval. Then, fuzzy logic processing sequentially executes following typical processing steps, in pre-defined blocks:

Measured parameters, BER, OSNR and Power, are continuously fed by each node into central fuzzy logic control subsystem. Its centralized diagnostics evaluates these parameters, calculates its statistical means during measuring interval. Then, fuzzy logic processing sequentially executes following typical processing steps, in pre-defined blocks:

1. Fuzzification – in this step, measured “crisp values” of BER, OSNR and Power are evaluated using input membership functions. Pre-defined membership functions intervals and shapes follow user experience. The result of this process is a fuzzy set.

2. Fuzzy decision making – based on fuzzy rules set and current input fuzzy set comparison in inference engine, it produces output fuzzy values.
3. Defuzzification – in this step, based on output membership function and output fuzzy values comparison, output crisp parameters are obtained and crisp final values are formulated. Ranked final values lead up to path choosing process.

Instead of numerical values, fuzzy logic evaluates numerical values coupled with linguistic expressions, describing defined states and situations. Linguistic expressions are divided into intervals, where a parameter of a membership functions represent degree of truth by statements like "parameter occurs at very low … medium … high level". Some values and parameters can be "blurred" if they are uncertain. Examples of input membership functions setup over input measured intervals are shown in Figs. 3, 4 and 5.

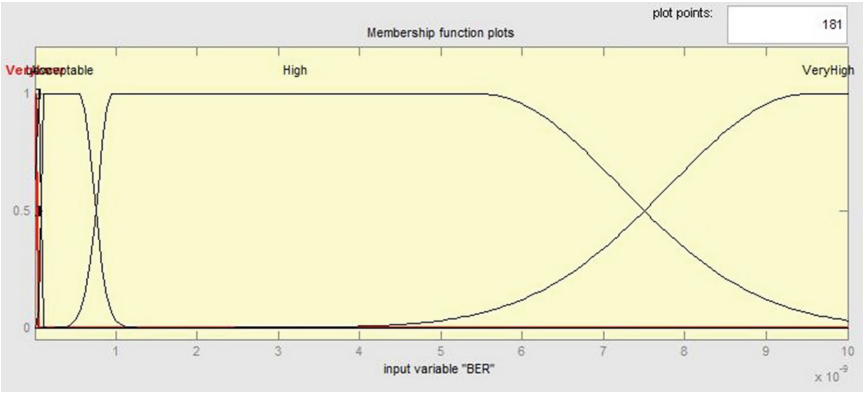



**Fig. 3.** BER fuzzy membership functions

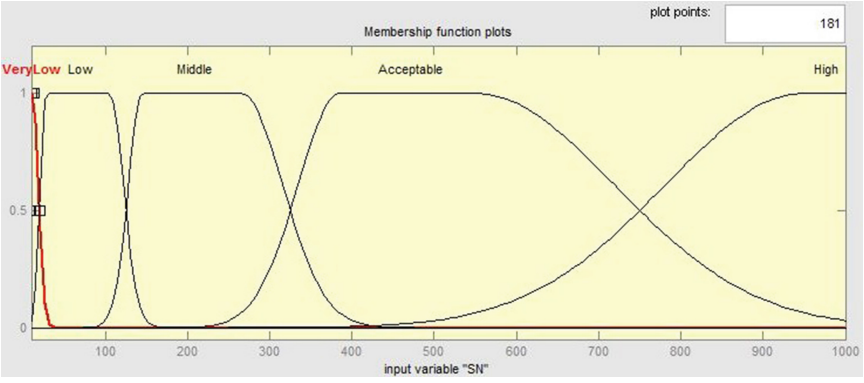



**Fig. 4.** OSNR fuzzy membership functions

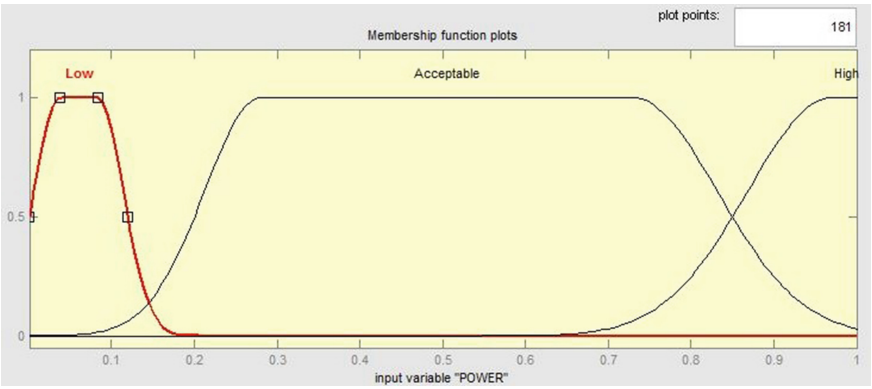



**Fig. 5.** Power fuzzy membership functions

As it was mentioned above, we defined basic input parameters with their ranges, with assigned appraisal. BER numerical value spans from 10e-12 to 10e-8, its linguistic expressions for this parameter are Very Low, Low, Acceptable, High and Very High.

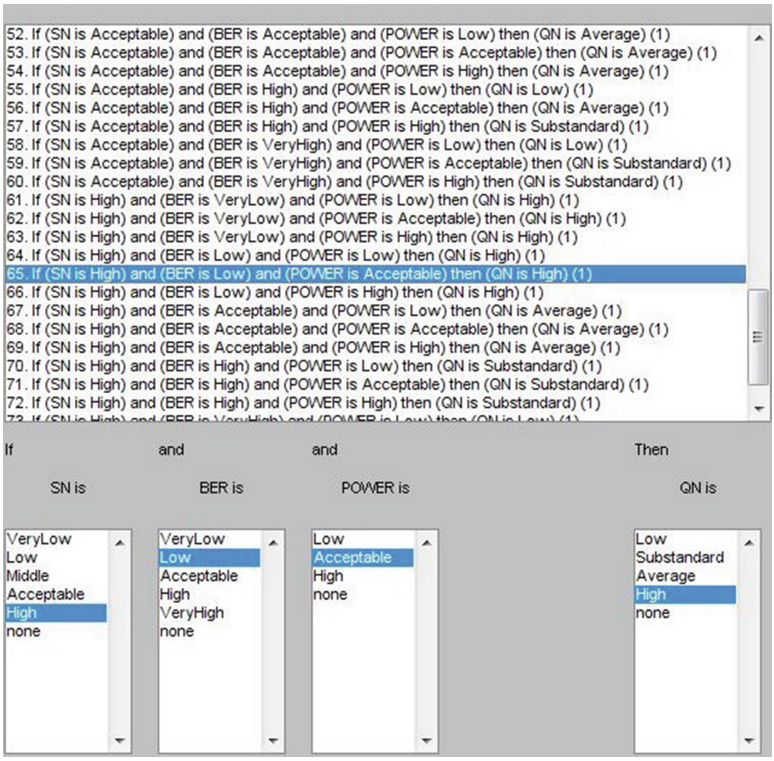



**Fig. 6.** Fuzzy rules setup among input and output parameters

OSNR numerical value spans from 10 to 1000 following linguistic expressions Very Low, Low, Middle, Acceptable and High. Link power numerical value spans from 0,001 W to 1 W; linguistic expressions for this parameter are Low, Acceptable, and High.

An example of defined fuzzy rules is shown in Fig. 6. Here, relationships between input parameters (SNR, BER and Power) and output parameter (QN, quality of particular path) are defined to obtain output fuzzy values.

To obtain particular path cost, we defined output crisp parameter QN representing linguistic expressions Low, Substandard, Average and High, spanning along interval <0,100>, Fig. 7. The QN parameter represents overall quality and suitability of each particular path.

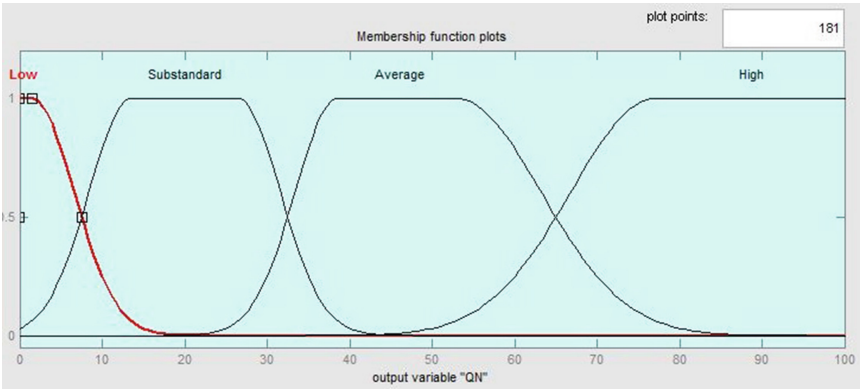



**Fig. 7.** QN fuzzy membership functions

To prove membership functions setup and fuzzy rules setup, surface views of input/output parameters relationships are available, Figs. 8, 9 and 10.

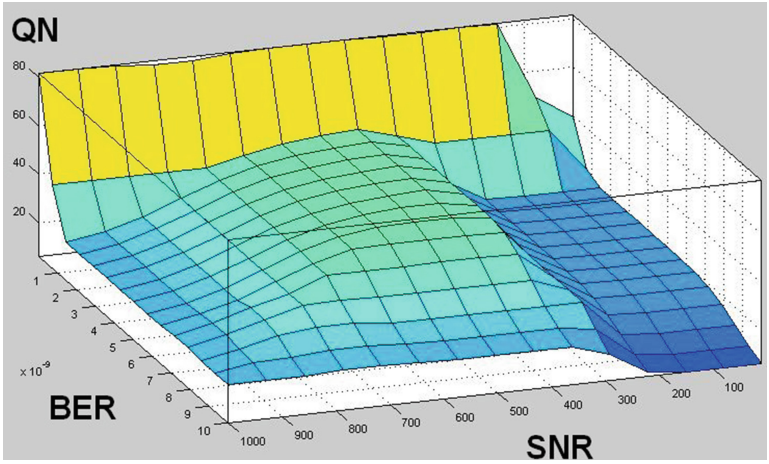



**Fig. 8.** SNR and BER relationship to QN, at constant Power

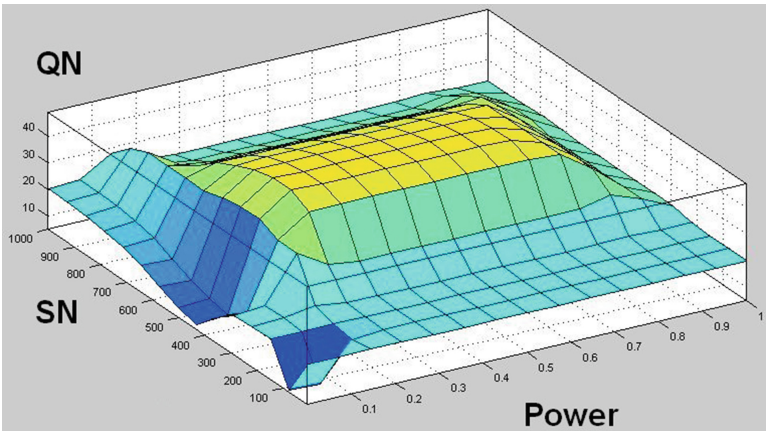



**Fig. 9.** SNR and Power relationship to QN, at constant BER

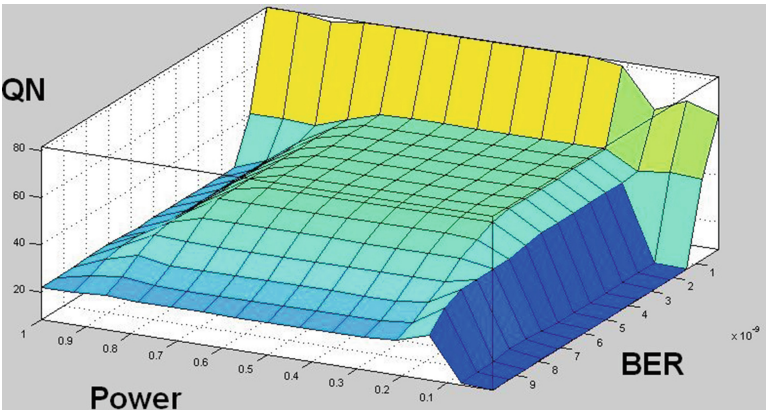



**Fig. 10.** Power and BER relationship to QN, at constant SNR

## 6 Paths Evaluation and Route Ordering

After obtaining QN cost for each particular path, all possible routes located between chosen input and output nodes are evaluated to specify routes stability and reliability. Thus, two important statistical parameters are calculated over each available route: μ mean value and σ standard deviation over incorporated paths QN costs. These μ and σ parameters are computed using Eq. 2. Finally, the most appropriate route is chosen using the highest μ and, if its equality, by the lowest σ.

$$\mu = \frac{\sum_{i=1}^{n} QN_i}{n}, \quad \sigma = \sqrt{\frac{\sum_{i=1}^{n} (QN_i - \mu)^2}{n}} \tag{2}$$

To demonstrate evaluation of all paths, we simulated data transport from Bratislava to Kosice (Figs. 1, 2) using randomly balancing OSNR, BER and Power values. Using above described fuzzy logic processing, each path is assessed by its own cost. After paths costs list completion (see example in Table 1), list of available routes between input and output nodes is generated and weighed using Eq. (2), see Table 2, thus the most appropriate route and alternate routes are ordered.

**Table 1.** Path cost between neighbouring nodes

| Input node – Output node | 1 - 2 | 1 - 3 | 1 - 4 | 2 - 3 | 2 - 5 | 3 - 4 | 4 - 5 | 4 - 7 | 5 - 6 | 6 - 7 |
|---|---|---|---|---|---|---|---|---|---|---|
| Path cost [-] | 20 | 10 | 30 | 15 | 45 | 65 | 85 | 75 | 35 | 40 |

**Table 2.** Mean value and standard deviation for all available routes, input node 1 to output node7

| Route (Input node – Transfer nodes – Output node) | μ | σ [-] |
|---|---|---|
| 1 - 2 - 3 - 4 - 7 | 43.7500 | 30.6526 |
| 1 - 2 - 3 - 4 - 5 - 6 - 7 | 43.3333 | 26.9568 |
| 1 - 2 - 5 - 4 - 7 | 56.2500 | 29.5452 |
| 1 - 2 - 5 - 6 - 7 | 35.0000 | 10.8012 |
| 1 - 3 - 2 - 5 - 4 | 46.0000 | 33.9853 |
| 1 - 3 - 2 - 5 - 6 | 29.0000 | 15.5724 |
| 1 - 3 - 4 - 7 - 0 | 50.0000 | 35.0000 |
| 1 - 3 - 4 - 5 - 6 - 7 | 47.0000 | 28.8531 |
| 1 - 4 - 7 | 52.5000 | 31.8198 |
| 1 - 4 - 3 - 2 - 5 - 6 - 7 | 38.3333 | 16.6333 |
| 1 - 4 - 5 - 6 - 7 | 47.5000 | 25.3311 |

## 7 Discussion

Non-fuzzy algorithms don't allow complex multi- parameters paths evaluation to assign real route cost based on parameters fusion. Even if the fuzzy logic implementation might evocate concerns about routing stability but, as shown in the example, the multi-input approach allows more precise routing to achieve stated data delivery time even if some WDM network paths might be overloaded. Current real time parameters determine the most appropriate route choice along the network and help to find optimal transportation route.

## 8 Conclusion

In this article we have presented a new approach to find optimal route in fast optical WDM network. Multi-input multi- path fuzzy logic control system allows real paths parameters evaluation and all-paths assessment. Presented algorithm automatically

finds alternate routes. Such approach brings future optimizing opportunities for WDM networks data transfer processes. Proposed fuzzy controlled routing system shows an efficiency increase option in WDM optical networks.

**Acknowledgment.** This work was supported by VaV projects of the operational program "Centre of excellence of power electronic systems and materials for the components", project code 2008/2.1/01-SORO, ITMS 26220120003, "Centre of excellence of power electronic systems and materials for the components II.", ITMS 26220120046. Project is co-funded from EU funds.

This paper was realized with the support of the APVV-0888-11 Project – The research of new passive process structures on silicon basis.

This paper was realized thanks the support of the APVV-0025-12 Project – Mitigation of stochastic effects in highbitratesall-optical networks.

This paper was realized with the support of the VEGA 1/0853/13, Study of micro structural, electrical and optical properties of semiconductor-dielectric systems.

This work was supported by project ITMS: 26210120021, co-funded from EU sources and European Regional Development Fund.

## References

1. http://www.cisco.com/c/en/us/solutions/collateral/service-provider/ip-ngn-ip-next-generation-network/white_paper_c11-481360.html
2. Essiambre, R.-J., Tkach, R.W.: Capacity trends and limits of optical communication networks. Proc. IEEE **100**(5), 1035–1055 (2012). ISSN 0018-9219
3. Leuthold, J., Koos, C., Freude, W., Alloatti, L., Palmer, R., Korn, D., Pfeifle, J., Lauermann, M., Dinu, R., Wehrli, S., Jazbinsek, M., Gunter, P., Waldow, M., Wahlbrink, T., Bolten, J., Kurz, H., Fournier, M., Fedeli, J.-M., Yu, H., Bogaerts, W.: Silicon-organic hybrid electro-optical devices. J. Sel. Top. Quantum Electron. **19**(6), 13 (2013). ISSN 1077-260X
4. Zapata-Beghelli, A., Bayvel, P.: Dynamic versus static wavelength–routed optical networks. J. Lightwave Technol. **26**(20), 3403–3415 (2008). ISSN 1558-2213
5. Gringeri, S., Basch, B., Shukla, V., Egorov, R., Xia, T.J.: Flexible architectures for optical transport nodes and networks. IEEE Commun. Mag. **48**(7), 40–50 (2010). ISSN 0163-6804
6. Gandhi, S.I., Vaidehi, V., Fernando, X.: Modified dynamic online routing algorithm and regenerator placement in WDM networks. J. Commun. Technol. **2**(1), 246–254 (2011). ISSN 2229-6948
7. Leiva, A.L., Machuca, C.M., Beghelli, A.Z.: Upgrading cost modelling of capacity-exhausted static WDM networks. In: 16th International Conference on Optical Network Design and Modeling, Colchester, UK, 17–20 April 2012, p. 6 (2012). ISBN 978-1-4673-1441-1
8. Elbers, J.-P., Autenrieth, A.: From static to software-defined optical networks. In: 16th International Conference on Optical Network Design and Modeling, Colchester, UK, 17–20 April 2012, p. 4 (2012). ISBN 978-1-4673-1441-1
9. Zhao, Y., Li, X., Li, H., Wang, X., Zhang, J., Huang, S.: Multi-link faults localization and restoration based on fuzzy fault set for dynamic optical networks. Opt. Express **21**(2), 1496–1511 (2013). ISSN 1094-4087
10. Wang, M., Li, S., Wong, E.W.M., Zukerman, M.: Performance analysis of circuit switched multi-service multi-rate networks with alternative routing. J. Lightwave Technol. **32**(2), 179–200 (2014). ISSN 1558-2213

11. Von Lindeiner, J.B., Ingham, J.D., Wonfor, A., Zhu, J., Penty, R.V., White, I H.: 100 Gb/s uncooled DWDM using orthogonal coding for low-cost data communication links. In: Optical Fiber Communications Conference and Exhibition, San Francisco, CA, 9–13 March 2014, p. 3 (2014). ISBN 978-1-5575-2994-7
12. Wei, J., Cheng, Q., Cunningham, D.G., Penty, R.V., White, I.H.: 100-Gb/s hybrid multiband CAP/QAM signal transmission over a single Wavelength. J. Lightwave Technol. **33**(2), 415–423 (2015). ISSN 1558-2213
13. Frisken, S., Poole, S.B., Baxter, G.W.: Wavelength-selective reconfiguration in transparent agile optical networks. Proc. IEEE **100**(5), 1056–1064 (2012). ISSN 0018-9219
14. Koonen, A.M.J., Chen, H.S., Sleiffer, V.A.J.M., Van Uden, R.G.H., Okonkwo, C.M.: Compact integrated solutions for mode (de-)multiplexing. In: Optical Fibre Technology, OptoElectronics and Communication Conference, Melbourne, VIC, 6–10 July, pp. 164–166, (2014). ISBN 978-1-922107-21-3
15. Chou, J., Lin, B.: Adaptive re-routing over circuits: an architecture for an optical backbone network. In: Conference on Computer Communications, San Diego, CA, 15–19 March 2010, p. 5 (2016). ISBN 78-1- 4244-6739-6
16. Wang, S.-W., Wen, C.-Y.: Lightpath-level active rerouting algorithms in all-optical WDM networks with alternate routing and traffic grooming. In: International Conference on Information Networking, Bali, Indonesia, 1–3 February 2012, pp. 42–46 (2012). ISBN 978-1-4673-0250- 0
17. Kamamura, S., Shimazaki, D., Mori, H., Sasayama, K., Koizumi, Y., Arakawa, S., Murata, M.: Optimization of light-path configuration order in IP over WDM networks using fast traffic matrix estimation. In: Optical Fiber Communications Conference and Exhibition, San Francisco, CA, 9–13 March 2014, p. 3 (2014). ISBN 978-1-5575-2994-7
18. Bhardwaj, N., Gupta, N.: A novel technique to minimize gain-transient time of cascaded EDFA using fuzzy logic controller. Int. J. Emerg. Technol. Comput. Appl. Sci. **1**(4), 569–573 (2012). ISSN 2279-0055
19. Sardar, A.R., Sing, J.K., Sarkar, S.K.: Fuzzy logic based alternate routing scheme for the minimization of connection set up time and blocking rate in WDM optical network. Int. J. Soft Comput. Eng. (IJSCE) **3**(1), 222–228 (2013). ISSN 2231-2307
20. Borkowski, R., Caballero, A., Arlunno, V., Zibar, D., Monroy, I.T.: Robust cognitive-GN BER estimator for dynamic WDM networks. In: European Conference on Optical Communications, Cannes, France, 21–25 September 2014, pp. 607–609 (2014). ISBN 978-1-4799-3066-1
21. Jiménez, T., Aguado, J.C., De Miguel, I., Durán, R.J., Angelou, M., Merayo, N., Fernández, P., Lorenzo, R.M., Tomkos, I., Abril, E.J.: A cognitive quality of transmission estimator for core optical networks. J. Lightwave Technol. **31**(6), 942–951 (2013). ISSN 1558-2213
22. Freude, W., Schmogrow, R., Nebendahl, B., Winter, M., Josten, A., Hillerkuss, D., Koenig, S., Meyer, J., Dreschmann, M., Huebner, M., Koos, C., Becker, J., Leuthold, J.: Quality metrics for optical signals: eye diagram, Q-factor, OSNR, EVM and BER. In: 14th International Conference on Transparent Optical Networks, Coventry, England, 2–5 July 2012, p. 4 (2012)
23. Pastorelli, R., Bosco, G., Nespola, A., Piciaccia, S., Forghieri, F. Network planning strategies for next-generation flexible optical networks. In: Optical Fiber Communications Conference and Exhibition, San Francisco, CA, 9–13 March 2014, p. 4 (2014). ISBN 978-1-5575-2994-7
24. SANET – Slovak Academic Network. http://samon.cvt.stuba.sk

# Providing Extensible Mobile Services to Car Owners Based on On-Board-Diagnostics

Richard Hable[1(✉)] and Gerhard Brugger[2]

[1] evolaris next level GmbH, Graz, Austria
richard.hable@evolaris.net
[2] Porsche Informatik GmbH, Salzburg, Austria
gerhard.brugger@porscheinformatik.at

**Abstract.** Based on data fetched via the On-Board- Diagnostics interfaces of cars, a comprehensive solution was created to provide functionality like automatically maintained driver's logbooks within a mobile phone application. Advanced on-board hardware modules and a dynamic client/server architecture allowed the design and implementation of an extensible system with optional integration of third-party software and infrastructure. This includes a mobile phone application which can be extended with new functionality according to customer configuration settings and payments. The project posed challenges like time and location independent availability, privacy and security issues, the dynamic extension of mobile applications with platform-independent code, and also economic restrictions. This paper contains a description of these challenges, the technologies and concepts explored and utilized, and the final system architecture and implementation decisions taken during product development. First experiences gained during deployment and commercialization already show the aptitude of the system and the impact of the design decisions taken.

## 1 Introduction

Modern cars contain an standardized On-Board-Diagnostics (OBD) socket which can be used to communicate with the car's internal electronic devices [1]. Commercial hardware modules are available which plug into this socket and can be used to read information about the current car status and also to make changes to the car's configuration.

Simple solutions require a mobile device to be connected to the module, either via cable or a wireless protocol like Bluetooth, and to send commands to the car and evaluate its responses. This means that a laptop or mobile phone has to be in the proximity of the car all the time, if all available car information shall be utilized.

Advanced OBD modules, however, provide additional features like internal data caching and long-distance data transfer via a built-in GSM module, enabling permanent remote access to the car. It is then possible to stay connected with the car during its entire operation time and provide features based on long-term information collection and real- time control. Centralized collection of diagnostic car information can then be used, for example, to centrally manage a fleet of cars, supporting maintenance and repair tasks [2].


J. Janech et al. (eds.), *Proceedings of the 2015 Federated Conference on Software Development and Object Technologies*, Advances in Intelligent Systems and Computing 511,
DOI 10.1007/978-3-319-46535-7_5

This paper shows the design and implementation of a comprehensive client/server solution based on automatically fetched car information, which enables car owners to get information about their cars at any time and any place via their mobile phones. A flexible mobile application was created which supports a large variety of features, from simple things like checking if the doors are closed and the lights are switched off to advanced functionality like an automatically maintained driver's logbook. Also server software was developed, performing administrative tasks and efficiently providing the mobile application with all required data.

Support for efficient commercialization of the product was a major factor in the design of the system: Instead of a monolithic application with fixed built-in features, the customers should be proposed a flexible solution, allowing them to individually subscribe to different functionality with different payment options. Also, third-party companies should be enabled to take part in the development of new features for the mobile application. Therefore, dynamic integration of third- party services and additional mobile application components had to be possible.

This led to a unique client/server architecture and a specially designed mobile application structure, taking advantage of both native platform-specific development and cross-platform web technology. Based on this design, a commercial product was created for a large automotive company, proving that the chosen concepts work in practice and also facilitate meeting general requirements like robustness, security, and usability.

The following chapters start with a description of the requirements and challenges faced in the project. Then, an overview of available technologies and their applicability is given. Based on these preconditions, the final system architecture and implementation decisions taken can be described and motivated. In conclusion, the results of the project are summarized, assessing the properties of the created product and giving an outlook to possible future extensions and improvements.

## 2 Requirements and Challenges

Users should be able to access car information from their mobile phones. Thus, software and hardware had to be made available to support this task. Hardware would be installed in the user's car by professionals, software should be provided via the usual distribution channels: a special mobile application should be downloadable via the app stores of the phone manufacturers.

### 2.1 Availability and Distribution

Technically, an ongoing stream of data would have to be fetched from the car in order to be able to provide up-to-date information about a car's current state and position. This should also work when the mobile phone is located far away from the car, does not have network connectivity, or is completely switched off. Inversely, the phone also should be able to access car information while the car is at a remote position or currently does not provide data via its interfaces. The collected car data itself is quite

simple, consisting mainly of small integer values indicating, for example, the current driving speed in kilometers per hour. However, due to a potentially large number of users and the need for always available up-to-date information, data has to be exchanged in high frequency and large accumulated quantities.

Also the distribution of the application's functionality posed challenges: instead of a fixed feature set for all users, different functionality should to be provided based on user preferences and payments. Even completely new commercial features should be possible without the user having to download new application versions. Therefore, the application had to be highly configurable and extensible.

### 2.2 Privacy and Security

The mobile application would only read data about the current state of the connected cars and thus never interfere with their normal operation. Nevertheless, there were severe privacy and security concerns: the current location, speed, and other usage information of a car provides a lot of private information about a car owner's behavior. Even more conclusions could be drawn from information collected over a longer time span.

Research has shown that diagnostic information transmitted by a smart phone over a number of days can be used to identify users with high accuracy [3]. The diagnostic information available from the multitude of sensors built into a modern car would allow similar approaches, even when it is carefully avoided to transmit information about the car owner's identity.

Even more, unauthorized modification of transmitted and stored data could lead to severe trouble if, for example, it is the basis for a driver's logbook.

Therefore, it was important to make sure that only authorized parties may access car and user information, and transmission and storage of data is performed in a secure way.

### 2.3 Economic Restrictions

Last but not least, implementation effort and required skills of the software developers had to be considered. The application had to be available on different mobile phone platforms: based on the market share of different mobile phone operating systems, it was decided to immediately support phones with iOS and Android operating systems, with the option to add support for other systems like Windows Phone later. It was not acceptable to have to create separate implementations for each supported OS each time a new feature should be added.

## 3 Available Technology and Solutions

### 3.1 Hardware

Off-the-shelf hardware modules are available which plug into the OBD socket of a car and allow short-range wireless data transfer to a smart phone via the Bluetooth protocol. The smart phone has to send commands to such a module, which forwards them

to the car and returns response data. Of course, this only works as long as both the smart phone application is running and the car is in a nearby position.

More complex modules are equipped with GSM hardware [4], enabling them to transfer data to a remote server as soon as they are supplied with electrical power from the car. This allows ongoing data transfer independently of the mobile phone's state and location. The server can collect the data and make it available to the mobile phone via a defined service interface. Ready-to-use solutions providing both hardware modules and server support are commercially available.

### 3.2 Server and Network Software

The state of the art in platform independent communication on the Internet is to provide REST (Representational State Transfer) services via the HTTP protocol [5]. This allows interchanging data both between two servers and between a mobile phone and a server. It also has the advantage that standards for encryption and authentication based on this technology are readily available.

Depending on the capabilities of the installed hardware modules, the car information can be transferred to a server on the Internet directly or after being fetched by the mobile phone.

HTTP services can be programmed with arbitrary technology and development tools. For processor and operating system independence, the Java Platform, Enterprise Edition (Java EE) standard [6] can be used as a basis.

### 3.3 Mobile Phone Software

Manufacturers of mobile phones generally support two ways of application development: creating web applications hosted on a server, which use web technologies like HTML and JavaScript within the mobile phones' built-in web browsers, and native applications distributed via an app store, which can access the capabilities of the devices using their built-in framework APIs (Application Programming Interfaces). Generally, users prefer downloadable native applications to web solutions [7].

In both cases, however, tight control of the operating system over resources like processor time and I/O facilities imposes restrictions on what the application can be programmed to do. Also, especially in the case of iOS devices, applications distributed in the app store are limited by strict guidelines with regards to functionality like code reloading and to the accepted content.

In addition to the manufacturers' development environments, third-party solutions are available, which try to combine the advantages of native and web-based development.

**(1) Pure Web Applications**

A pure web application is hosted on an independent server and thus not downloadable via the manufacturers' app stores. However, this also means that these applications can be changed centrally at any time. If users are logged in to a web application, the server software can check their identities and automatically provide customized features

according to their payments and preferences. It is also easy to make new functionality available by extending the server software: all the users have to do is to reload the application in their web browsers.

Web applications use vendor-independent, standardized technology (HTML, JavaScript, CSS) for all platforms, and thus the same implementations can be used on different platforms. However, the applications are limited by the capabilities of the web browsers they are running in. Although browsers with HTML5 [8] support, as used in current smart phones, provide access to built-in hardware like the camera and the GPS sensor, direct access to the framework functions of the mobile phones' operating systems is not possible. As a consequence, even pure web applications which take advantage of advanced browser functionality, may show unsatisfactory performance. For example, a pilot study [9] on the mobile application described in [10] showed that even re-development can be necessary in such cases.

**(2) Wrapped Web Applications**

Tools like Apache Cordova [11] support embedding web applications in a native application frame and thus create stand- alone applications which can be distributed in the app stores. In addition, they provide JavaScript APIs which allow web applications to access some native framework functions of the operating systems.

However, with such a solution the central configurability and extensibility of pure web applications gets lost: a fixed set of web resources is included in the app which cannot be modified and extended without forcing the user to download a new version.

**(3) Native Applications**

With native applications the full power of the operating system frameworks can be utilized. However, the frameworks, programming languages, and also development environments provided by the vendors are very specific, forcing the programmer to create separate implementations for each supported platform. Also, technical restrictions and app store guidelines allow adding new code to existing applications only by providing new application versions which the user has to download from the app store.

**(4) Cross-Platform Development Tools**

With third-party development tools like Xamarin [12] and Appcelerator [13] native applications can be created for different mobile platforms based on common source code, thus supporting at least partial re-use of code. Applications are written in a common programming language, like C# and JavaScript, and then automatically transformed to code required by the target platforms. Wrapper APIs provide, to a varying degree, unified access to the different operating system frameworks.

While it is possible to reduce implementation effort this way, cross-platform tools also add an additional layer of complexity to the implementation, and compromises concerning the application quality have to be made if only functionality common to all supported operating systems shall be used.

**(5) Comparison**

There is no single application type which fulfills all requirements of the mobile application, as can be seen in Table 1.

**Table 1.** Application types and features

| | Application type | | |
|---|---|---|---|
| | App store | Code reuse | Extensibility |
| Pure Web | no | yes | yes |
| Wrapped Web | yes | yes | no |
| Cross-Platform | yes | yes | no |
| Native | yes | no | no |

In addition to these primary criteria, it is also important to take into consideration the different development skills required and the achievable quality and performance of the implementation [14].

Generally, more developers with experience in native and web development are available than developers for specific cross-platform tools. The best user experience is possible with native applications, followed by cross-platform and wrapped web applications [15]. Pure web applications are severely limited concerning both functionality and performance.

Thus, a combination of different application types and technologies would have to be found for the mobile application, carefully accounting for the advantages and disadvantages of the applied tools and methods.

## 4 Architecture and Implementation

### 4.1 Client/Server Architecture

The overall architecture of the system was derived straight- forwardly from the general requirements: A hardware module (the on-board unit) is provided for each car and plugged into its OBD socket. It then regularly transmits information via a built- in GSM module to a central server. The server collects this information and makes it available via web services. The mobile application uses these services to fetch information needed to provide the functionality requested by the user.

In order to avoid having to develop low-level communication services between the on-board units and custom server software, a ready-to-use solution of hardware modules and central servers was selected to cover the information collection requirements. However, in addition to these existing servers it was also necessary to operate an own server to handle configuration information, like subscriptions and payments. Based on this information different features can be delivered to different users. This server also provides a unified service interface, allowing the mobile application to fetch both user information and car information required for the implementation of its functionality.

For independent development of additional application functionality, third-party servers can also take part in this server structure, communicating both with the mobile application and the servers providing car and configuration information.

An overview of the complete infrastructure can be seen in Fig. 1.

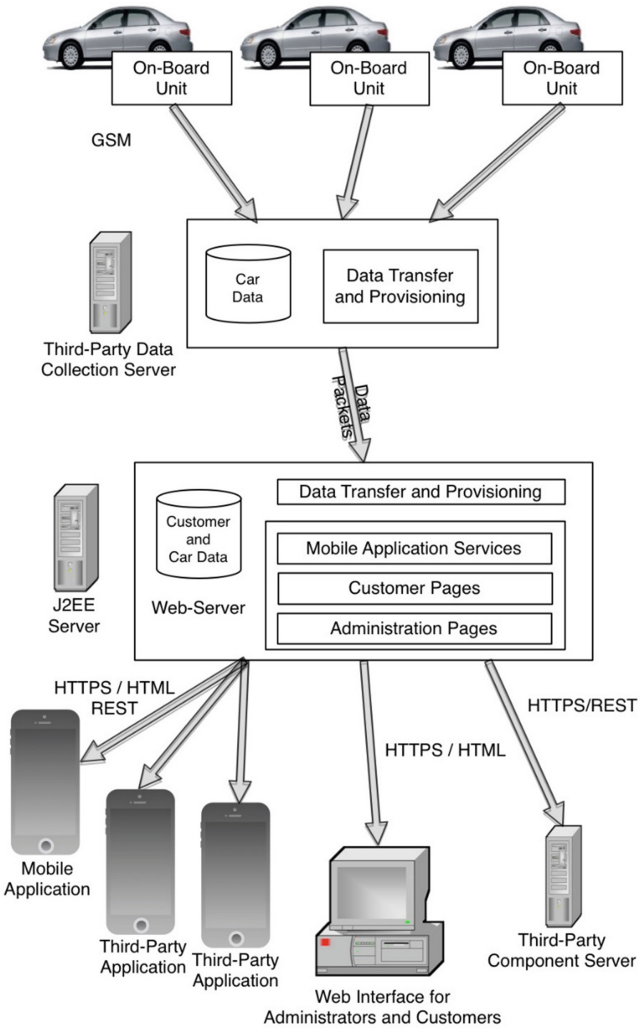



**Fig. 1.** Client/Server infrastructure

### 4.2 Authentication and Authorization

Confidential data is transmitted to and exchanged between different servers, including those of third-party vendors, and mobile devices. It is also stored in databases and processed in various ways before it is transmitted to the customers' mobile devices.

As a comprehensive security solution for authentication, the OAuth 2.0 standard [16] is used both on the server and on the client side, and all data transfer is performed via secure HTTPS connections.

Therefore, the following guidelines had to be followed during development of both client and server software:

- All communication is encrypted (HTTPS) with validated certificates proving the identity of all communication partners.
- User names and passwords are never stored locally and only used to initially request access and refresh tokens.
- All further services require authentication via an access token before sending user-specific data.
- Third-party servers are only allowed to fetch car data after third-party authorization via the mobile application.

Thanks to built-in OAuth 2.0 support of the utilized communication frameworks, this did not require much additional implementation effort. However, a lot of testing and security auditing of communication routines was necessary to make sure that all security demands were met.

Access tokens are stored permanently within a safe area of the mobile phone, thus allowing the user to start and stop the application at will without having to provide user credentials each time. However, for security reasons, both access and refresh tokens expire after a configurable amount of time, then requiring the user to provide user name and password again.

In order to protect the identity of the users, no personal information about users is transferred to third-party servers. They may access car data services after authorization via the mobile application according to the permissions granted to the third-party component, but cannot combine them with registration information of the users.

### 4.3 Server Software

**(1) Fetching and Storing Car Data**
Due to the selected hardware solution with vendor-provided servers no direct communication between mobile phones and on-board units had to be implemented. However, it was necessary to fetch collected data regularly from the vendor servers via a pre-defined web service interface.

A web service client based on the Java API for XML Web Services (JAX-WS) was therefore created, fetching car information and storing it in a local database. This local availability of car data improves the response time for requests from the mobile application and allows efficient server-side statistics and calculations.

**(2) Services for the Mobile Application**
For communication with the mobile application, customized REST services based on the HTTP protocol and the JSON data format were implemented and deployed on a web server using Java EE technology. These services provide selected car and configuration information according to the needs of the mobile application and the subscribed components of the customer. This way, the amount of data transferred to the mobile phones could be minimized, and existing libraries for asynchronous HTTP communication and JSON encoding and decoding could be utilized for the mobile application implementation.

Based on the stored configuration information, the server also coordinates access to functionality implemented on third- party servers. It provides information about available components to the mobile application and authorizes access to data required by them.

**(3) Web Pages for Customers**
Web pages were created which enable customers to check and extend their subscriptions to optional components within the mobile application. Similarly to the provided REST services, Java EE technology is used to serve the web pages for the customers and perform lower-level business logic handling subscriptions, payments and configuration of services.

Customers use their account information (user names and passwords) for authentication both on the web pages and in the mobile application. This way, settings concerning the mobile application can also be performed within a web browser on an desktop computer.

**(4) Administration Pages**
Configuration of components and offered subscriptions is also done via web pages hosted on the web server. Users with access rights for administrative web pages can deploy and distribute third-party components. They can provide the information which shall be visible to customers, like component names and descriptions, and technical information, like third-party server credentials and car data dependencies.

### 4.4 Mobile Application

Contrary to the server side, no standard way of mobile application development fulfilling all requirements of the project was available. Therefore, a project-specific combination of existing technologies had to be utilized, and some compromises had to be made to achieve a satisfying solution.

**(1) Finding a Suitable Implementation Method**
The following considerations were important for decision taking:

- An existing prototype of a wrapped mobile web application (using the Cordova framework) showed that web technologies and platform-independent APIs for native functionality were sufficient to implement an attractive user interface for the customer.
- Server support was required not only to provide car information to the application but also to perform monitoring tasks and send notifications to the user even while the application is in the background.
- For efficiency reasons all communication with servers should be performed by asynchronous native code.
- Some functionality visible to all users of the application, like information pages and basic car status information, should be a fixed part of the application in order to improve the user experience and meet the iOS app store directive, which requires applications to provide more functionality than just remote web content.

- The application had to regularly fetch configuration information from the server and dynamically show components with different functionality according to the customer preferences and subscriptions.
- Third-party components had to be prevented from interfering with each other. This required a strict separation of user interface components and a centralized handler of application events like notifications concerning specific components.

It was therefore decided that the mobile application should consist of both built-in native code and dynamically loaded web application code. The Cordova framework, which supports static web code within a fixed, automatically generated native application frame, was used as a basis for this endeavor.

**(2) Adding Flexibility to Cordova Applications**

The Cordova framework creates source code in the project format required by the development environments of the mobile phone vendors. These projects contain natively implemented platform-specific classes and cross-platform web components. Additional native code can be added in the form of Cordova modules, which allow the web components to access native framework functionality via the JavaScript programming language.

The projects can then be opened and built in the development environments in order to create executable applications ready for app store deployment. However, the functionality of these applications is limited by the statically included web components, with no support for modularity and extensibility.

Therefore, after creating such a project for each target platform (iOS and Android), including carefully selected Cordova modules, it was necessary to add the required flexibility:

- The native code had to be extended to provide the built-in features of the application.
- Instead of a single web view containing the entire user interface, separate areas for native functionality and for the dynamically added third-party components had to be implemented.
- Instead of showing web content from within the application, all third-party content had to be loaded dynamically according to the customer-specific configuration information.

This actually required rewriting most of the automatically created native code. The Cordova modules and the interface code between web and native code, however, could be taken over almost without changes.

The included web content was replaced by a static start page, which immediately fetches subsequent pages from a third-party server according to a dynamically retrieved URL. This URL is part of the service response structure received after authentication of third-party components.

**(3) Putting it together**

Figure 2 shows the principal division of dynamic and native content within the app. The main screen contains a mixture of static and dynamic functionality, including

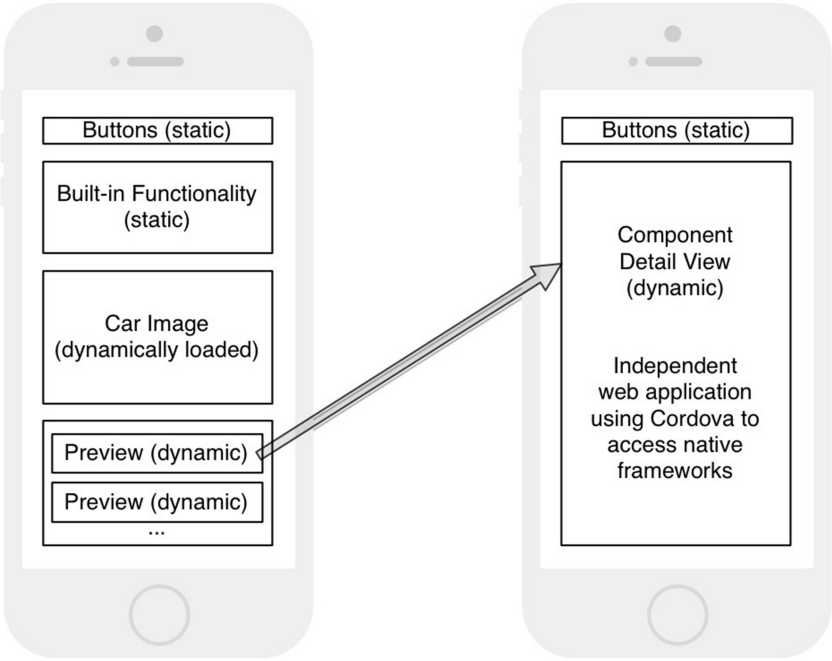



**Fig. 2.** Screen layout with static and dynamic content

a list of dynamically retrieved preview areas for third-party components. Touching one of the preview areas usually leads to a dynamic detail view of this third-party component. Note that these detail views can themselves contain a complete web application with an arbitrary number of subpages and appropriate navigation facilities.

As an alternative to the embedded detail view, third-party components can also provide a reference to a separate native application. Touching the preview area then redirects the user to the app store download area for the application, or immediately launches the application if it has already been downloaded.

The user navigates between the main page, built-in pages for information and configuration, and the third-party detail pages via buttons, list items, and a slide-in menu.

Figure 3 shows a screen shot of the application's main page with an open slide menu. The slide menu overlaps the main page on the right side of the screen, allowing quick navigation to the start page, the detail pages of available components, and a settings page. On the left side of the screen, parts of the main page are visible: a back button for navigation, a natively implemented door status indicator, a dynamically loaded car picture, and preview areas of two components.

The mobile application can be put into background and foreground without losing the current navigational position, allowing the user to work with a specific component for a prolonged amount of time. If, however, the application is completely removed from memory, e.g. because the phone is restarted, the next launch opens the main page. Also, if at any time communication with the servers fails due to expired tokens, the user

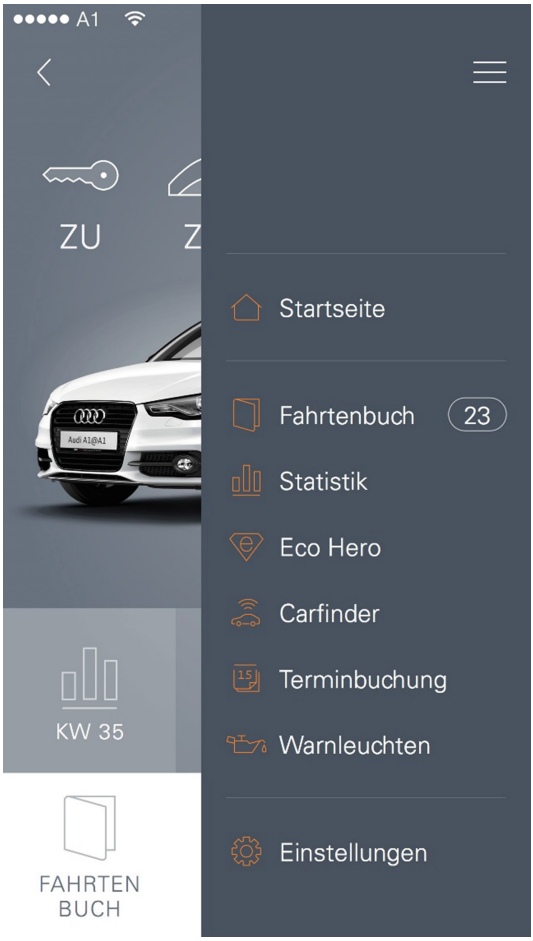



**Fig. 3.** Screen shot of application main screen with slide-in menu

is automatically directed to a login screen, requesting user name and password for OAuth 2.0 authentication.

In order to inform users about component-specific events, the push notification service of the operating system vendors (Apple for iOS, Google for Android) is used. Third-party component providers send application-specific messages via a vendor's server to the mobile phone of a specific user. The mobile phone operating system shows the message to the user, even if the application is currently closed or in the background. If the user opens the notification message, the operating system launches the application or puts it into foreground, if necessary. The application then automatically navigates to the full-screen view of the component responsible for the notification. For this to work, each third-party component is assigned a unique component ID, which has to be included in the push notification message. Based on this ID

and configuration information from the servers, the mobile application can then automatically show the required third-party web application.

**(4) Consequences and Experiences**

The chosen client/server architecture and the dynamic behavior of the mobile application posed challenges concerning reliability and usability:

- Compared to conventional (monolithic) applications, a lot of potentially slow and unreliable server communication is necessary to provide the required functionality.
- The user interface has to be suitable for different subscriptions and preference setting of the customers, thus requiring a dynamic screen design.

The key to meeting these challenges was to perform all network communication in an asynchronous way. This allows several server requests to be processed in parallel, and the user interface can dynamically add screen content according to received data, while staying responsive to user input at all time.

When, for example, the application has to refresh the information on the main screen, it fetches in parallel the current car status (tank filling etc.), the car's image, and the components available to the user. As soon as information is received from a server, this information is updated on the screen, e.g. by updating the door status field or adding a new component preview area. While network activity is in progress, the user can continue navigating within the application via buttons and the slide-in menu, or open the full-screen view of a component whose preview area has already been added.

This asynchronicity also makes the application robust against failure of individual components, allowing the user to utilize most of the application's functionality even if, for example, one of the third-party servers is temporarily not accessible.

Nevertheless, especially in the case of large third-party components, the user of the mobile application may experience clearly noticeable delays during network activity, similar to using a web browser. This behavior was deemed acceptable for the time being, but may have to be improved depending on user feedback in the future, e.g. by means of sophisticated data caching mechanisms.

## 5 Conclusion

Design and implementation of the system showed that offering services to customers based on automatically collected car information requires more than just fetching information from a car and displaying it on a mobile device.

While built-in capabilities of modern cars and available hardware accessories provide a readily available technical base, creating a comprehensive commercial solution posed challenges both concerning the design of the system architecture and the implementation of the required client and server software.

In this paper a flexible solution meeting these challenges was described, which allows the creation and provisioning of services based on car information, including the integration of third-party software and server infrastructure. The solution includes an application which makes these services available to customers on their mobile phones

in a time and location independent way. Innovative implementation methods were devised to provide a solution which allows platform- independent dynamic configuration and extension of the application while taking account of app store restrictions and development costs.

The deployment and commercialization of the system proved that the applied concepts work well in practice. Drawbacks of the chosen system design, like the need for extensive network communication between different clients and servers, were alleviated by measures like the consequent utilization of asynchronicity and an adherence to strict security standards. Nevertheless, the focus on portability and dynamic extensibility caused some limitations concerning the achievable usability compared to monolithic, platform-specific mobile applications.

The server infrastructure is now operational, and the mobile application is already being used by customers. In addition to the built-in functionality providing information like the current tank filling, first optional components created by a third-party company are available, performing diverse tasks like helping the car owner to schedule repair appointments or to just find the location of the car. More optional components will be added soon, including also extended server support in the background.

Future research will require assessing customer acceptance of the offered functionality, guiding the way to the development of new features. The utilization of the mobile application itself will be monitored, with server statistics showing when and how often the application and its optional components are used. Customer feedback concerning the mobile application's usability will tell about possible needs for improvements and changes.

## References

1. You, S., Krage, M., Jalics, L.: Overview of Remote Diagnosis and Maintenance for Automotive Systems. SAE International, Warrendale, PA (2005). http://papers.sae.org/2005-01-1428/. Accessed 21 Aug 2015
2. Lin, J., Chen, S., Shih, Y., Chen, S.: A Study on Remote On-Line Diagnostic System for Vehicles by Integrating the Technology of OBD, GPS, and 3G. World Academy of Science, Engineering and Technology (2009). http://www.waset.org/publications/13356
3. Quattrone, A., Bhattacharya, T., Kulik, L., Tanin, E., Bailey, J.: Is this you?: Identifying a mobile user using only diagnostic features. In: Proceedings of the 13th International Conference on Mobile and Ubiquitous Multimedia, pp. 240–243. ACM (2014). http://doi.org/10.1145/2677972.2677999
4. Cai, J., Goodman, D.: General packet radio service in GSM. IEE Commun. Mag. **35**(10), 122–131 (1997). http://doi.org/10.1109/35.623996
5. Fielding, R.T.: Architectural Styles and the Design of Network-based Software Architectures (2000). http://www.ics.uci.edu/~fielding/pubs/dissertation/fielding_dissertation_2up.pdf
6. Java Platform, Enterprise Edition (Java EE). http://docs.oracle.com/javaee. Accessed 21 Aug 2015
7. Wong, S.H.R.: Which platform do our users prefer: website or mobile app? Ref. Serv. Rev. **40**(1), 103–115 (2012). http://doi.org/10.1108/00907321211203667

8. Pilgrim, M.: HTML5: Up and Running. O'Reilly Media Inc., Sebastopol (2010). 1005 Gravenstein Heighway North, CA 95472
9. Zbick, J., Nake, I., Jansen, M., Milrad, M.: mLearn4Web: a web-based framework to design and deploy cross-platform mobile applications. In: Proceedings of the 13th International Conference on Mobile and Ubiquitous Multimedia, pp. 252–255. ACM (2014). http://doi.org/10.1145/2677972.2678007
10. Zbick, J., Jansen, M., Milrad, M.: Towards a web-based framework to support end-user programming of mobile learning activities. In: 2014 IEEE 14th International Conference on Advanced Learning Technologies (ICALT), pp. 204–208 (2014). http://doi.org/10.1109/ICALT.2014.66
11. Apache Cordova. https://cordova.apache.org/. Accessed 21 Aug 2015
12. Mobile App Development & App Creation Software - Xamarin. http://xamarin.com/. Accessed 21 Aug 2015
13. Mobile App Development Platform – Appcelerator. http://www.appcelerator.com/. Accessed 21 Aug 2015
14. Charland, A., Leroux, B.: Mobile application development: web vs. native. Commun. ACM **54**(5), 49–53 (2011). http://doi.org/10.1145/1941487.1941504
15. Angulo, E., Ferre, X.: A case study on cross-platform development frameworks for mobile applications and UX. In: Proceedings of the XV International Conference on Human Computer Interaction, pp. 27:1–27:8. ACM (2014). http://doi.org/10.1145/2662253.2662280
16. RFC 6749 - The OAuth 2.0 Authorization Framework. http://tools.ietf.org/html/rfc6749.html. Accessed 21 Aug 2015

# A New Architectural Design Pattern of Distributed Information Systems with Asynchronous Data Actualization

Patrik Hrkut(✉), Ján Janech, Emil Kršák, and Matej Meško

Department of Software Technologies, University of Zilina, Zilina, Slovakia
{Patrik.Hrkut, Jan.Janech, Emil.Krsak, Matej.Mesko}@fri.uniza.sk

**Abstract.** Rapid development of miniaturization technologies helps create small and easily portable devices. This enables users to work with their information systems (IS) almost anywhere. The only obstacle, however, may be a poor network connection, which can make the system inaccessible. One of the solutions is creating a distributed information system with asynchronous data update. Then the user operates over the local data, and they do not necessarily need an instant network connection through a server. Later, with a connection available, the local data are merged with the rest of the system. This concept will effectively solve the situations in which two users are editing the same data. Below we present a new architectural pattern for designing such IS.

## 1 Introduction

The current information systems (IS) development clearly shows its priority in the nearest future: mobility. Users want to work from home, when travelling, doing field work or being on a business trip abroad.

Therefore, the multi-client information systems are required to be able to operate fully also without a network connection (offline). The time of the connection unavailability is limited, and at the time of re-connection the data need to be synchronized. So, the client has to be able to work with locally stored data, operate over them and, after re-connection, to synchronize them with the rest of the system.

The current architectural patterns cannot meet the above requirement fully. That is why our work focuses on designing a new architectural pattern to solve the problematic of distributed IS with asynchronous data synchronization.

## 2 Current State

The system views all data changes as part of a transaction. The transaction (also called the business transaction) has to meet the following criteria:

- ACID properties (Atomicity, Consistency, Isolation and Durability)
- Consistency and isolation from other business transactions


J. Janech et al. (eds.), *Proceedings of the 2015 Federated Conference on Software Development and Object Technologies*, Advances in Intelligent Systems and Computing 511,
DOI 10.1007/978-3-319-46535-7_6

If two various users are doing two mutually conflicting business transactions (with a synchronous update), an alignment /conflict of two transactions occurs:

- *Optimistic locking* – it prevents conflicts between concurrent business transactions by means of detecting the conflicts and sending off the transaction status [1]
- *Pessimistic locking* – it prevents conflicts between concurrent business transactions by allowing only one business transaction to access the data [1]

Thus the synchronous synchronization normally allows only one user at a time to change the data. The second user will not be allowed to do so. There are some hybrid techniques [2, 3] which combine some of the approaches mentioned above.

The asynchronous update is more complicated, since the synchronization may occur at a time not known in advance. It is vital to compare the changes of the local and shared data. Merging such forms of data is not an easy task, and there are various techniques how to do that. E.g., in [1], to merge the data, the authors used decision trees.

However, this article focuses on designing an architecture, not on the data-merging algorithm itself.

## 3 A New Design Pattern Proposal

In the systems with the asynchronous data update the situation becomes more complicated by the fact that the user can work all the time without being connected to the server. The IS client can run e.g. on a laptop without the internet connection, or the server can be accessible only via a company intranet network. Therefore the server as a component responsible for organizing the cooperation among clients does not have the information about the running business transactions.

Due to these limitations, it was crucial to determine which basic qualities the IS needs to have so that it can work successfully.

### 3.1 Basic Principles

The proposed new architectural pattern is based on the following assumptions:

- *The user needs to have full access to the information system.* It is not desirable for them to be limited by anything else (e.g. technical limitations) than by the assigned permissions. The user has to have the offline access to all necessary data to be able to edit them.
- *The connection to the computer network may be requested only for a data update.* The user needs to be able to work offline at all other occasions. Unless contacting the server is absolutely necessary, the client must not use the connection to the computer network.
- *The data get updated at the user's request exclusively.* The user has to be in full control of this important operation. Only they know when they finished their work which they want to transmit to the server, and when the connection to the computer network meets their needs.

- *The data administrated by the information system will be structured.* Currently there are only a few examples of information systems working solely with unstructured data.

The above assumptions can be used to derive the following basic principles which will influence the design of our architectural pattern.

*Principle 1.* Since full offline work is required, the architectural pattern has to be based on the architecture of a distributed IS with a fat client. If the domain logic were a part of the server implementation, working without a permanent client-server connection would not be possible.

*Principle 2.* The system has to contain two separated data storages. One of them is shared and accessible only through the server. It stores all data, regardless which user has created them. In addition, every client needs an access to their own local data storage. The user executes and saves all changes in their local storage only. The local and shared databases are synchronized during the data update.

*Principle 3.* To enable a data update on both client and server sides, both storages (local and shared) have to be able to return the list of changes made since the last data update.

With the local storage it is quite simple. It is sufficient to save the information about the nature of the change, i.e., which object has been changed and which object has been created since last data update of the concerned client.

The situation is more complicated with the server. Each client can be doing an update at a different moment, therefore they can each be using a different version of the local data. That is why the server should remember the differences between the current version and any of the previous versions.

*Principle 4.* The above assumptions also show the necessity of solving the alignment of business transactions only during the data update. Only then it is possible to communicate with the server.

The locking used with the IS with synchronous data update (see Sect. 3.3) cannot apply here. Instead, it is necessary to allow the alignment but to merge the changes from various clients during the update.

*Principle 5.* The structured data requirement suggests that the record administrated by the system will be a composite structure. That is why the network transmissions, data merging and similar operations will be performed over these composite structures.

## 3.2 Pattern Structure

The listed principles allow us to derive the structure of the proposed architectural pattern. Figure 1 shows its graphic design:

As we have mentioned in Principle 1, the basis is the IS architecture with a fat client. Even though it is clear the design has been influenced by the traditional n-layered architecture, in few places its basic principles have been violated. As examples we can mention ‘Update layer’ or ‘Local database’ with which we cannot speak about a higher and lower levels.

The layers of the new pattern are described as follows:

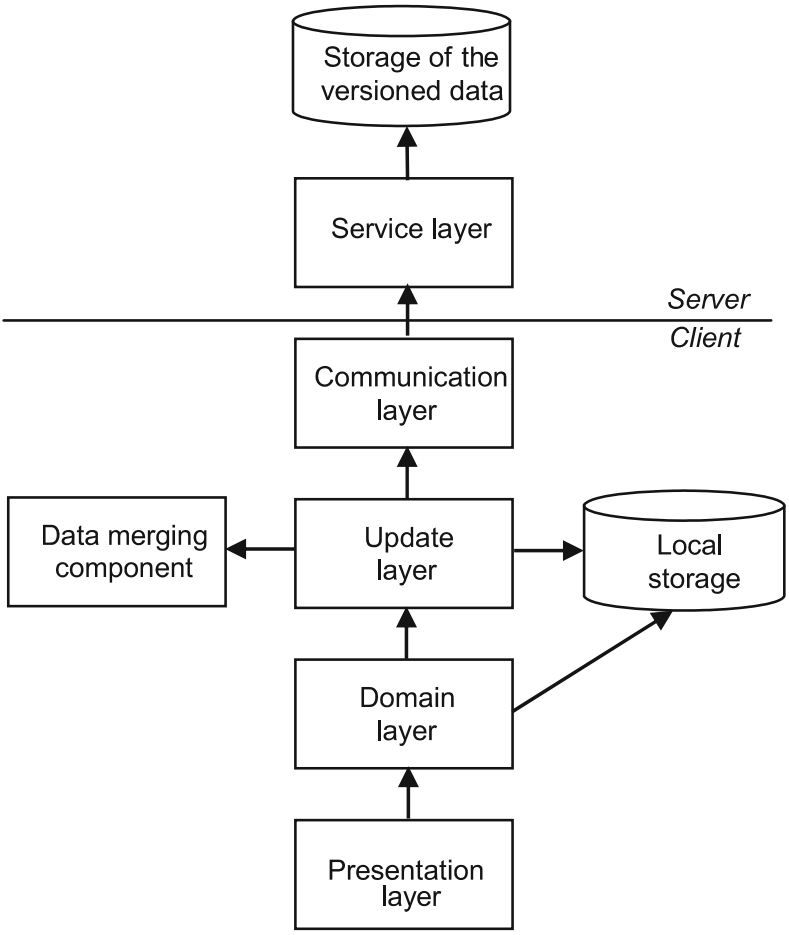



**Fig. 1.** Layers of the design pattern

- *The communication and service layers* enable IS to communicate over the computer network. They are analogous to the same layers of the fat client IS architecture. Nevertheless, since the architecture is used in a specific way, several basic services have been defined which the service layer needs to provide clients with.
- *The update layer* covers the entire data update process on both server and client's sides. It receives the command to run the update from the domain layer. During the update it has to solve all problems related to potentially concurrent business transactions. To do so, it uses the data merging component.
- *The data merging component* deals with solving the alignment problem. As we have mentioned before, a standard solution using data locks is not possible. Even if one of the clients created a lock for a particular record, they would not be able to inform others about it. By the time the lock would arrive to all system users, it might have been changed by somebody else. The data merging principle is shown in Fig. 2.

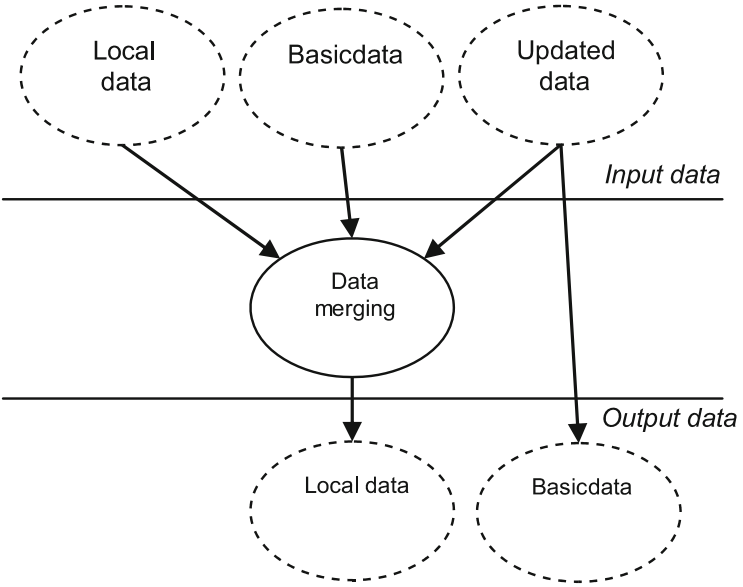



**Fig. 2.** Data merging process

Each data record enters the process in three different versions:

- *Basic data* – the common version of a record for both, the local and shared databases. Both records are based on it. In fact, this is the version of the record that was current at the latest data update.
- *Local data* – the local database record version.
- *Updated data* – the shared database record version.

The data merging process will find changes of the local as well as shared records compared to the basic version, and it will merge them into one record version to be stored in the local database.

- *The local storage layer* needs to provide two separate interfaces. The interface accessible to the application is not expected to meet any specific requirements. The particular IS is responsible for the operation principles of this interface. However, the local database must provide a second interface, against the communication layer. The layer has to be able to identify the local changes list and incorporate the changes from the server.
- *The domain and presentation layers* represent the domain and presentation logic. They run operations demanded by the user and show them the results. These layers need not be designed with a regard to the fact that the way IS works is based on the asynchronous update principle.

### 3.3 Data Update Process

The data update is divided into two phases. During the phase of taking over the changes, the changes on the server are identified and they are transmitted to the client. The changes have to be incorporated into the local database without disrupting the local changes.

*The phase of handing over the changes* constitutes sending the local changes to the server. During the phase it is not possible to deal with the alignment problems, therefore it is assumed that the local database has just been updated recently to the latest version.

Depending on which phase the user decides to use for the update, we can distinguish three ways of updating the data with the server:

- *The data update on the server* – the user has the new changes in the local database, and they want to transfer them into the shared storage. They use only the phase of handing over the changes.
- *The data update on the client's side* – the user wants to retrieve the changes from the shared storage. They use only the phase of taking over the changes.
- *The full data update* – the user wants to update the local data as well as to transfer the local changes to the server. They use both of the phases, starting with the phase of taking over the changes.

Since the first way requires the local data to be updated by the latest version of the shared database, from the user's point of view it may seem complicated. That is why we recommend to implement into IS only the last two update manners.

*The phase of taking over the changes* from the server runs as follows:

1. The user initiates the update. The domain layer requests the update layer to control the process of taking over the changes.
2. Through the communication layer, the update layer connects to the server. It informs the service layer on the latest data version retrieved from the server, and it asks for the list of all changes.
3. The service layer requests the list of the changes from the shared data storage and notifies the communication layer. It also adds the information about the current data version for its future identification at the next update.
4. The update layer incorporates the list of the changes into the local database using the data merging layer. It also saves the current data version.
5. The user is notified about the result of the data update.

Each object from the set of globally changed objects is detected as either new, changed or deleted. If the object is new, it is stored into the local storage as it has arrived from the server, without any modification. The deleted objects are treated similarly – the local database is sent a request to delete them. The changed objects bring along more complications. Before they can be stored in the local database, they need to be checked for local changes. If there are not any, the object can be stored directly, replacing the original version. However, if there have been any local changes, first they need to be connected with the received changes and only then the object can be stored into the local database.

What makes the process of handing over the changes more simple is the fact that all the changes from the server need to be taken over first. The process consists of the following steps:

1. The user initiates the update. The domain layer asks the update layer to control the process of handing over the changes.
2. The update layer connects to the server through the communication layer. It announces the last retrieved data version to the service layer.
3. The service layer compares the versions of the local and shared data. If they are not equal, the process of handing over the changes is terminated immediately.
4. The update layer requests the list of the local changes from the local database.
5. The update layer supplies the list of the local changes to the communication layer which forwards it to the service layer. The service layer provides it to the changed data storage to be incorporated.
6. The changed data storage makes sure the changes are correctly incorporated into the database.
7. The service layer reports the identifier of the new global data version. The update layer saves it.
8. The domain and presentation logic reads the new data version.
9. The user is notified about the result of the data update.

In this case, the process of integrating the changes is very simple because it is sufficient to replace all objects by their new versions. The basic version of the local data and the current version of the global data are identical, therefore no merging is necessary.

## 4 Server-Side Services

As we have mentioned in the introduction, the communication between the client and the server through the computer network can be done only during the update process. According to the processes described in the previous section, we can identify the basic services the service layer has to provide:

- *The Client login service* represents the operation of logging in. The particular service implementation depends on the security requirements of the specific IS. It is even the only one of the described services that is optional, and the service layer does not have to implement it.
- *The Start of taking over the data service* indicates the beginning of the phase of retrieving the changes from the server. The service needs to be implemented as *pull*, i.e. it has to be initiated by the client and the server needs to reply to it. In their request, the client sends the information about the current local data version. If the client has got no data yet (it is the first data take-over), they indicate the fact by a special zero version. The server's reply must contain the confirmation of the successful completion of the operation. If not mistake occurs, the client continues using the *Get changes* service.

- The *Get changes service* returns the list of the changed objects. The service can be implemented as a pull or a push type. This means that the server can send the data without being requested to do so by the client.
- The S*tart of data sending service* denotes the beginning of the phase of handing over the changes in the local database to the server. Having received the information, the server creates a new database version and expects to receive the data for it. The service needs to be implemented as *pull*. In their service request, the client states the current local data version. The server's reply is an error information. In each implementation a specific error may occur: the local and shared databases versions are not identical. If there has been no error, the client continues to use the *Incorporation of changes service*.
- *The Incorporation of changes service* gradually incorporates the given data into the current database version. The service needs to be implemented as pull. The only reply is the information about an error. The changes incorporation can only be carried out by one client at a time. If more clients try to incorporate their changes into the shared database, the server has to allow only the first one of them to do so. The other clients only receive the error notification.
- The *Finish service* denotes the end of reading and recording the changes from/to the database. For the taking over phase, the service can be implemented as push. It returns the current shared database version number to the client.

Other services can be implemented, too, however, the above services are required for IS with the asynchronous data update to function correctly.

## 5 Conclusion

We used the described architectural design pattern to create the railway transportation timetables, IS ZONA. Some of client applications are described in our previous papers [4–6].

A lot of clients work in the system, participating in the process of creating train schedules. The clients often travel and find themselves in the areas without a direct access to servers through a data network. The network is either very slow or of poor quality. The clients usually collected the data in written form and entered them into the information system only after they had returned to their offices. The process generated a lot of errors.

Our task was to make their work more effective and enable them not only to define the basic train parameters but also all other data necessary for the creation of train schedules.

The main parameters required for the new system for the mobile train schedule creators were as follow:

- Recording all train path parameters in a well-arranged form
- Displaying all created train paths, even if they were created by other users, in both tabular and graphical forms
- The client's functionality being independent from a network connection to the server

We created a new information system with a three-layer architecture. We implemented the above described architectural pattern.

### 5.1 Object Granularity Level

A train path consists of a large number of data. One might picture it as a tree structure. The roots represent the basic data on the train (name, number, the carrier involved, services, power supply, breaking…). Then the path with the list of transportation points follows. Each of them contains the data on stops, engines, calendars, notes and plenty of other technical parameters.

First, we needed to define the granularity level so that it was possible to work with the objects as with a whole, so called bunch of grapes. On the grounds of the analysis [7–10] of the data reusability in the object model, we established the following bunches of grapes: train, note, calendar, user, group of users etc.

As we have said above, we implemented the Data update on the client and Full data update. The data update between the client and the server runs on the basis of synchronizing the whole bunches of grapes. The client has the information which bunches of grapes have been modified and which have not. With the full data update only the modified bunches of grapes are synchronized.

The modified *bunches of grapes* are highlighted red (Fig. 3). The user of the application can see clearly which data are to be updated.

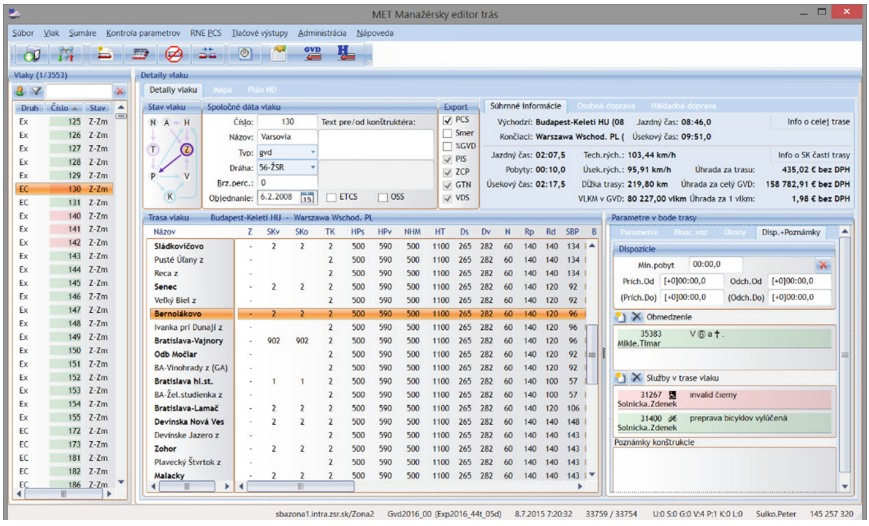



**Fig. 3.** Highlight of changes (red-tinged)

### 5.2 Conflict Solving

Conflicts occur at data update operations. They arise when two users modify the same bunch of grapes. Many of the conflicts are solved automatically by change merging. If it is not possible, e.g. two users have changed one and the same attribute, the conflict has to be solved by the user who was the second to carry out the update (Fig. 4).

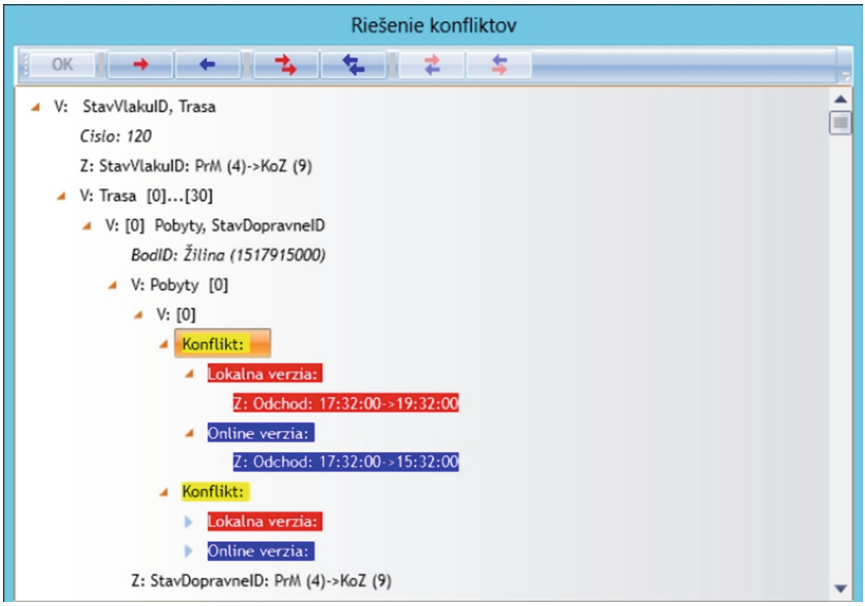



**Fig. 4.** Solving the conflicts

To solve the conflicts we implemented the operations existing in version systems.

**Acknowledgement.** This contribution/publication is the result of the project implementation: Centre of excellence for systems and services of intelligent transport, ITMS 26220120050 supported by the Research & Development Operational Programme funded by the ERDF.

Agentúra Ministerstva školstva, vedy, výskumu a športu SR pre štrukturálne fondy EÚ

Európska únia

Podporujeme výskumné aktivity na Slovensku/Projekt je spolufinancovaný zo zdrojov EÚ

## References

1. Fowler, M., Rice, D., Foemmel, M., Hieatt, E., Mee, R., Stafford, R.: Patterns of Enterprise Application Architecture. Addison-Wesley, Upper Saddle River (2002)
2. Thomasian, A.: Distributed optimistic concurrency control methods for high-performance transaction processing. IEEE Trans. Knowl. Data Eng. **10**(1), 173–189 (2002)
3. Bakura, S.A., Mohammed, A.: Lock-free hybrid concurrency control strategy for mobile environment. In: 2014 IEEE 6th International Conference on Adaptive Science & Technology (ICAST), pp. 1–5, 29–31 October 2014
4. Bachratý, H., Janech, J., Ružbarský, J.: EDYN - new software for timetable construction for Slovak railways. In: EURO - ŽEL 2014, Žilina (2014)
5. Krsak, E., Bachratý, H., Polach, V.: GTN - information system supporting the dispatcher and remote tracks control. In: Communications: Scientific Letters of the University of Žilina, Žilina (2010)
6. Bachratý, H., Ružbarský, J.: Information technologies for creating of railway transport timetables. In: 14th International Symposium EURNEX - Žel 2006 Towards the Competitive Rail Systems in Europe, pp. 59–67 (2006)
7. Tavač, M., Tavač, V.: DBRE and MDA integration. In: Objekty 2011 Proceedings of the 16th International Conference on Object-Oriented Technologies, pp. 52–65, 24–25 November 2011
8. Tavač, M., Tavač, V.: The general algorithm for the design of the MDA transformations models. In: CICSyN2013 Fifth International Conference on Computational Intelligence, Communication Systems and Networks, pp. 171–176, 5–7 June 2013
9. Gábor, M.: Creating of train timetable in system ZONA. In: Proceedings of Scientific Contributions Žilinská univerzita (2007)
10. Andrzejak, A., Langner, F., Zabala, S.: Interpretable models from distributed data via merging of decision trees. In: 2013 IEEE Symposium on Computational Intelligence and Data Mining (CIDM), pp. 1–9, 16–19 April 2013

# The Economics and Data Whitening: Data Visualisation

Radek Hrebik(✉) and Jaromir Kukal

Faculty of Nuclear Sciences and Physical Engineering, Department of Software Engineering, Czech Technical University in Prague, Prague, Czech Republic
Radek.Hrebik@seznam.cz, jaromir.kukal@fjfi.cvut.cz

**Abstract.** The paper deals with principal component analysis and data whitening. The research is done in the area of main economic indicators. This means the data preprocessing problem. The main aim of this paper is to present and discuss the possible ways of data preprocessing. The paper deals with four main approaches. There are compared the results from raw data, absolute differences, relative differences and logarithmic differences. The classic principal component analysis is also used with some improvement, there is described the basement of data whitening. The main aim is to get the good data visualisation. The next aim of such approach can be to identify the similarities between some states and their main trends. For this reason there is presented the comparison of states of Visegrad Group. At this moment there is no aim to deeply discuss the reasons of development in detail. This paper suggests new point of view to time series connected to economic development. The deep analysis of all relationships is the topic for further research.

## 1 Introduction

The contribution deals with principal component analysis (PCA) and the additional data whitening. The main aim is to discuss the possible ways of data preprocessing and present the best way to get easily readable data representation. The main idea of principal component analysis is reduction of dimensionality of some data set that consists of a large number of interrelated variables. The reduction retains as much as possible of the variation present in the data set. The aim is achieved by transforming to a new set of variables called the principal components. These principal components are uncorrelated and ordered so that the first few retain most of the variation present in all of the original variables. [2]

The aim of this research is the reduction to two principal components (PC1 and PC2). This means that all selected indicators used as explanatory variables are reduced to two dimensional space to be easily interpreted from its graphical representation. The results are presented on selected countries. As the representatives were selected Visegrad group countries.


J. Janech et al. (eds.), *Proceedings of the 2015 Federated Conference on Software Development and Object Technologies*, Advances in Intelligent Systems and Computing 511,
DOI 10.1007/978-3-319-46535-7_7

## 2 Methodology

Paper deals with the basic economic data and shows the ways of possible interpretation to serve as input for principal component analysis. The aim is to search the main indicators, monitor the potential trend of concrete objects and find objects having something in common. This approach as itself is nothing new. The data input and expected conclusion are the new thing. The authors expectation is to get the graphic representation of economic indicators in time that will be easily readable and interpretable.

One of the first published use of PCA for the economic time series data was presented in late forties by Stone ([2]). It also goes hand in hand for example with principal component analysis goal defined by Abdi and Williams – extracting the important information from the table to represent it as a set of new orthogonal variables called principal components and to display the pattern of similarity of the observations and of the variables as points in maps. [1]

The study dealing with principal component analysis to forecast a single time series with many predictors was presented by Stock and Watson. [5] The use of principal component analysis in connection with gross domestic product is discussed for example in [6]. Favero deals with comparison of two competing methods to estimate large-scale dynamic factor models based, respectively, on static and dynamic principal components. [7] Principal component analysis as alternative way to predict gross domestic product is presented in [8]. In our case, the question of main trends in development of Visegrad countries will be shortly discussed.

Research is based on publicly available economic indicators. Thanks to dimension reduction it gives a lot of possibilities which data can be used. This contribution presents three basic ways of using principal component analysis to interpret economic data. The way means in this case interpreting the data set as objects. These objects represent different economic indicators through time and states. The prediction of future country development can be also the reason for doing such research. It also opens the door to new way of finding the position of state if we know the basic economic prediction. The main thing is to capture some progress in time and easily read the development in time. The basic approach used as pattern is represented by objects as years. This means the number of explanatory variables can be very high. The ways of preprocessing are discussed later.

The aim of research is to present a big chance to show some new way of economic data interpretation. In this case indicators of Visegrad group play the key role. At this point the presentation of individual components is not the aim. Contribution deals not with the share of individual indicators in components. This will be the topic for our next research to deeply analyse all ties on economics.

## 3 Data Pre-processing, Whitening

It is good to start with the possible kinds of processing of the raw data collected from published statistics. Let $N, m \in \mathbb{N}$ be dimensionality of object description and length of

time series. Let $\mathbf{y_k}$ *in* $\mathbb{R}^{\mathbb{N}}$ be *k*-th sample of original data series for $k = 1, 2, \ldots, m$. Optional preprocessing (if any) will transform the data to time series $\{\mathbf{x_k}\}_{k=1}^{M}$ where $M = m$ or $M = m - 1$ *m* respectively. Omitted preprocessing as crucial case is represented by

$$\mathbf{x}_{\mathrm{k}} = \mathbf{y}_{\mathrm{k}} \tag{1}$$

for $k = 1, 2, 3, \ldots, m$. Therefore, the data are used as measured. This kind means no added value and take just the data published by the institutions.

In the case of positive descriptors i.e. $\mathbf{x}_{\mathrm{k}} > 0$, which is typical for macroeconomical indicators, we can apply relative differences as

$$\mathbf{x}_{\mathrm{k,j}} = \left(\mathbf{y}_{\mathrm{k}+1,\mathrm{j}} - \mathbf{y}_{\mathrm{k,j}}\right)/\mathbf{y}_{\mathrm{k,j}} \tag{2}$$

rather use logarithmic differences

$$\mathbf{x}_{\mathbf{k,j}} = \mathbf{ln}\frac{\mathbf{y}_{\mathbf{k}+\mathbf{1,j}}}{\mathbf{y}_{\mathbf{k}+\mathbf{1,j}}} \tag{3}$$

for $k = 1, 2, \ldots, \mathrm{m} - 1, \mathrm{j} = 1, 2, \ldots, \mathbb{N}$.

The main processing and data visualisation operates only with time series $\{\mathbf{x_k}\}_{k=1}^{M}$.

The main idea of research is focused on dimensionality reduction which is based on Principal Component Analysis (PCA) [2]. Thanks to it there can be a lot of explanatory variables and the way of possible visualization of results stays clear. We calculate mean value vector as

$$\mathbf{x}_0 = \frac{1}{M}\sum_{k-1}^{m}\mathbf{x}_{\mathrm{k}} \tag{4}$$

and covariance matrix estimate as

$$\mathbb{C} = \frac{1}{M-1}\sum_{k-1}^{m}(\mathbf{x}_{\mathrm{k}} - \mathbf{x}_0)(\mathbf{x}_{\mathrm{k}} - \mathbf{x}_0)^{\mathrm{T}} \tag{5}$$

Eigen-Value Decomposition (EVD) is based on equation

$$(\mathbb{C} - \boldsymbol{\lambda}\mathbb{I})\mathbf{v} = 0 \tag{6}$$

with constrain

$$\|\mathrm{v}\| = 1 \tag{7}$$

where $\mathbb{I} \in \mathbb{R}^{\mathbb{N}\mathrm{x}\mathbb{N}}$ identifies matrix, $\lambda \geq 0$ is eigenvalue and $\mathbf{v} \in \mathbb{R}^{\mathbb{N}}$ is corresponding eigenvector. Solutions of EVD can be ordered as $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_{\mathrm{N}} \geq 0$ with corresponding eigenvectors $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_{\mathrm{N}}$.

### 3.1 Principal Component Analysis

Traditional PCA of order $D \in \mathbb{N}$ is based on formula

$$\mathbf{p}_{\mathrm{k}} = \mathbb{W}^T(\mathbf{x}_{\mathrm{k}} - \mathbf{x}_0) \in \mathbb{R}^D \tag{8}$$

where

$$\mathbb{W} = (\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_{\mathrm{D}}) \in \mathbb{R}^{N\mathrm{x}D} \tag{9}$$

### 3.2 Whitening

Data Whitening (DWH) [4] is a little bit improved process which guarantees unit covariance matrix of resulting vector

$$\mathbf{y}_{\mathrm{k}} = \mathbb{L}^{-1/2}\mathbb{W}^T(\mathbf{x}_{\mathrm{k}} - \mathbf{x}_0) \in \mathbb{R}^D) \tag{10}$$

where

$$\mathbb{L} = \mathrm{diag}(\lambda_1, \lambda_2, \ldots, \lambda_{\mathrm{D}}) \in \mathbb{R}^{D\mathrm{x}D} \tag{11}$$

under supposition $\lambda_{\mathrm{D}} > 0$.

## 4 Explanatory Variables

The input data set plays the key role in this research based on principal component analysis. It is based on some economic time series. Used economic data has been selected from Statistical Annex of European Economy presented by European Commission in autumn 2014 [3].

The selection of the indicators and period was the first task for the authors. There has been selected time series representing data of nine descriptors. The selection of the descriptors can be made as of the author of analysis needs. This is the advantage to other model approaches. Thanks to dimensionality reduction there can be added more indicators and result of analysis will be still same readable.

The input data set for this research is represented by the thirty five countries from the whole world, majority are the European countries. The observation take place in years 1993 to 2014. Selected indicators are the total population, unemployment rate, gross domestic product at current market prices, private final consumption expenditure at current prices, gross fixed capital formation at current prices, domestic demand including stocks, exports of goods and services, imports of goods and services and gross national saving. So totally nine indicators are monitored.

The first kind of data interpretation is by objects representing calendar year. So there are only twenty one objects in this case so that $N = 21$. As the number of objects is very low, the number of descriptors is quite high, so the total number of indicators

$m = 35 \times 9$, so each object is created by number of countries mal number of indicators. The number of properties is totally over three hundreds. There is also possible to use such representation on the each state. So the number of indicators is 9. This kind of interpretation is used to demonstrate the data pre-processing in case of Czech republic.

In second case of possible use of principal component analysis the objects are represented by each state. So the properties are made of indicators in selected years. The number of objects is thirty five. The number of objects is small. So the representation will be very simple and it will be clear which states are closed to each other. From graphic representation is expected to easily notice the groups of states. One point representing one state gives a unique chance to identify the groups of states with similar type of economy. In this case $N = 9 \times 21$ representing each macroeconomic indicator in each year and $m = 35$ representing each of 35 states.

As third possible interpretation of the data set the object is represented by a state in a given year. So the number of objects is relatively high. The total number of object is in this case seven hundred and eighty, it represents number of states multiplied by the number of observed years. As the number of object is high, the origin data set dimensionality is relatively small. In this case $N = 9$ representing each macroeconomic indicator and $m = 21 \times 35$ representing 21 years in each of 35 states. There are selected just Visegrad countries to keep the figures readable.

## 5 Data Analysis in Period 1993–2014

The data analysis is presented for all three basic approaches mentioned before representing the given economic data as objects. The input objects are represented by years, states and states in given years. In the first case of analysing the years as objects there is presented the role of preprocessing. The selected approach to preprocessing is then used for the rest representations.

### 5.1 Years as Objects – Role of Preprocessing

Last 21 years were selected for the analysis to show the main trends and try to identify the main milestones in this period. The next aim is to compare the development of Czech Republic with others. As it is seen from Table 1 the best explanation of used data gives the raw data.

**Table 1.** Principal Components

| | PCA1 | PCA2 |
|---|---|---|
| PCA of raw data | 0,9281 | 0,9967 |
| Absolute differences | 0,9334 | 0,9740 |
| Relative differences | 0,6965 | 0,8645 |
| Logarithmic differences | 0,6652 | 0,8533 |

The same conclusion can be seen in graphical representation in Fig. 1. The research is based on searching the clear data interpretation. Comparing four possible ways it is clear that easily read can be raw data with whitening. The whitening is already doing some preprocessing and comparing the results, the using of raw data gives the highest level of explanation. The term easily read means the clear identification of trend based on graphic representations of principal component analysis of objects represented as years. So the intuitive trend identification is seen in case of whitening using raw data set. For example development in time in horizontal ow vertical way.

### 5.2 Years as Objects

The Visegrad group countries were selected as example for data interpretation. The result is captured in Fig. 2. There are counted always new principal components for each state.

There is easily seen the trend of development in years 1993 to 2006 in the case of Czech republic. This trend is typical with the movement down. In this time there is one change around year 2000 which is represented as change from trend right and down to the trend left and down. There is a big change from the year 2006 in trend. The last five years are represented very close to each other.

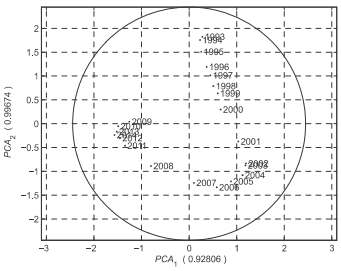
Two main break points are identified in case of Slovakia, the year 2000 and 2012. This means, that in the known year of crisis the economics was in the trend which started already in 2000. The following research will explain these main changes in trends. The aim of this paper is at first present the new way of data representation.

Poland shows the one big break point near year 2008. The last five years are as in case of Czech republic represented close to each other. So the last development gives the same marks as in case of Czech republic.

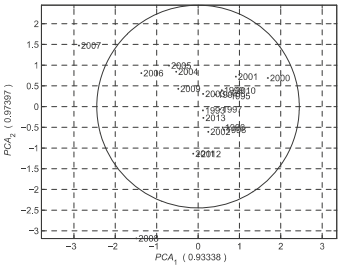
Hungarian break point is represented by year 2000. The rest of development is represented by some blocks of four two five years. It can represents for example the election cycle. This will be also the aim of future research, but the graphic representation gives interesting results already now.

Almost in all cases the last years after starting the debt crisis in Europe are represented close to each other without any significant trend in these years. As already said, this means only first way of interpretation based on graphic representation. The relevant analysis based on deep knowledge of all descriptors is the topic for further research. Maybe already this graphic representation can lead to the ideas of coming lost decade.

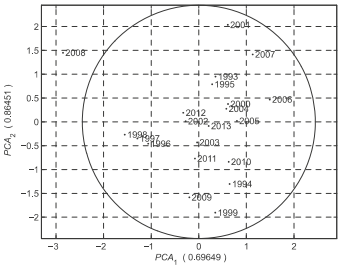
The advantage of this approach is that it gives the new way to identify the main trends in the development of each country or group of countries in time. The disadvantage is that it is not the best way to make the comparison of the states each other because of the other principal components. The next kind of object representation as states in given years serves better for such comparison of states each other.




(b) Abs. diff. – Whitening




(c) Rel. diff. – Whitening




(d) Log. diff.– Whitening

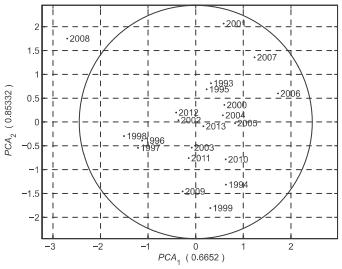
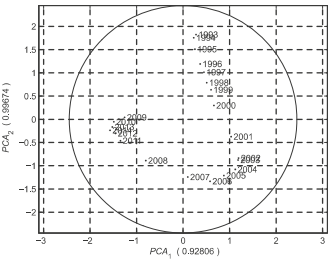
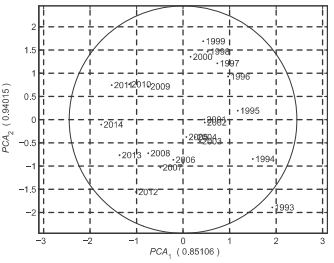



**Fig. 1.** Year as Object of Macroeconomic Investigation

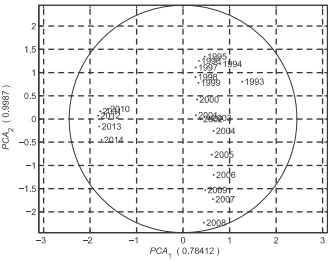
(a) Czech Republic




(b) Slovakia




(c) Poland




(d) Hungary

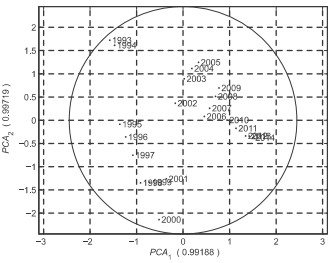
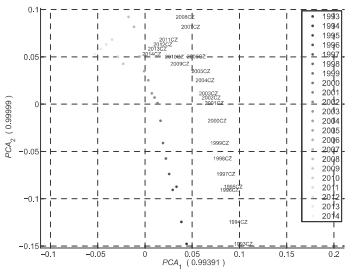
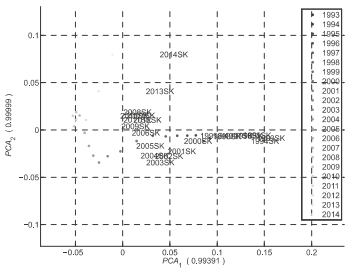


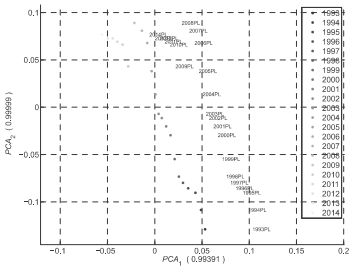

**Fig. 2.** Year as Object - Visegrad Group



(b) PCA – State in Year as Object – Slovakia



(c) PCA – State in Year as Object – Poland



(d) PCA – State in Year as Object – Hungary

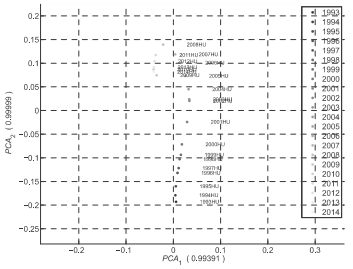


**Fig. 3.** PCA – State in Year as Object

### 5.3 States as Objects

The principal components are counted in this case from nearly two hundred indicators. So the reduction of dimensionality is quite high in this case. The indicators are created by the nine economy indicators in twenty one years. As in the previous case of using principal component analysis also here the biggest weights are on gross domestic product and population. In case of first principal component there is the population values included with bigger weight than in case of gross domestic product. Second principal component is preferring the values of gross domestic product in years. The values of first principal component are in most cases very close to zero, following the weights that implies that the population is without big changes having affect to component values. Second principal component is mostly counted from gross domestic product values. The results are captured in Fig. 4.

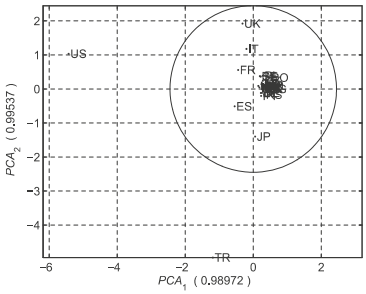



**Fig. 4.** PCA – States as objects

In this case the analysis ends with some outliers. Even if the outliers where omitted in the new run of analysis with reduced number of indicators ended with the similar results. In the second run of the analysis the United States and Turkey were omitted. This led only to other group of outliers. The third attempt to attend the outliers ended with the nearly same result. So objects defined in this way gave no clear result as expected. This is the reason why this way is at this time not to be recommended.

### 5.4 States in Years as Objects

In this case the results are again presented on group of Visegrad four. The results as graphical representation are presented in Fig. 3. It is the selective interpretation of the whole analysis. The analysis was done with all the countries and to keep the figure readable there are selected only these fours. Comparing to the previous representation states as object, the same wights for each state are used. The development of Czech republic, Poland and Hungary is represented by some trend. The trend is typical with mainly vertical way of representation. In case of Slovakia there are some differences.

## 6 Conclusion

Firstly the basic pre-processing methods were presented. The principal component analysis was used in case of main economic indicators. The data whitening as some upgrade on classic principal component analysis seems to be the best way for the aim of data visualisation. The countries of Visegrad group were selected as the example of data visualisation. It was shown that principal component analysis can be also very useful in interpreting the economic data. It represents some other way of interpreting time series and shows the states position in comparison to others. To fully interpret the results there is need to study the weights of principal components to know what stands behind the components values. The main of this paper was to use not a common way to get good graphic representation of economic data set. The graphic representation is intuitive, descriptive and easily interpretable in the most cases. So the main trends are easily recognized from the graphic representation. This way serves mainly to interpret the main development trends in each country. The identification of some breakpoints of development in each country was discussed. The intuitive and easily readable approach of development of Visegrad group countries was presented mainly in case of states in years. This approach gives a possibility to compare directly the countries. Czech republic, Poland and Hungary have the similar trend recognizable from its graphic representation. The one country of Visegrad, Slovakia, has the other trend. This can be in connection with its euro area membership. The deeper analysis based on deep economics knowledge is the aim for further research.

**Acknowledgment.** The authors would like to acknowledge the support of the research grant SGS14/209/OHK4/3T/14.

## References

1. Abdi, H., Williams, L.J.: Principal Component Analysis (2010). http://www.utdallas.edu/~herve/abdi-awPCA2010.pdf. Accessed 03 Feb 2015
2. Jolliffe, I.T.: Principal Component Analysis, 2nd edn. Springer, Heidelberg (2002)
3. European Comission. Statistical Annex of European Economy: Autumn 2014. Economic and Financial Affairs (2015). http://ec.europa.eu/economy_finance/publications/european_economy/2014/pdf/statistical_annex_autumn_2014_en.pdf. Accessed 15 Jan 2015
4. Eldar, Y., Oppenheim, A.V.: MMSE whitening and subspace whitening. IEEE Trans. Inf. Theor. **7**, 1746–1851 (2003)
5. Stock, J.H., Watson, M.W.: Forecasting using principal components from a large number of predictors. J. Am. Stat. Assoc. **97**(460), 1167–1179 (2002)
6. Schumacher, C.: Forecasting German GDP using alternative factor models based on large datasets. Bundesbank Discussion Paper 24/2005 (2005)
7. Favero, C., Marcellino, M., Neglia, F.: Principal components at work: the empirical analysis of monetary policy with large datasets. J. Appl. Econometrics **20**, 603620 (2005)
8. Chamberlin, G.: Forecasting GDP using external data sources. Econ. Labour Market Rev. **1** (8), 18–23 (2007)

# Kopenograms and Their Implementation in BlueJ

Marek Chadim and Rudolf Pecinovský(✉)

Department of Information Technologies, University of Economics, Prague, Prague, Czech Republic
marek.chadim@seznam.cz, rudolf@pecinovsky.cz

**Abstract.** Although currently the bulk of the most common algorithmic tasks is included in libraries of programming languages, it is necessary to realize, that upon completion of object oriented design of application, we still do not avoid of using more complex algorithmic constructions. For its visual projection and easier understanding several graphic languages are used. Architecture First methodology for its purpose prefers kopenograms, as one of the most suitable method of displaying structured algorithm. This paper deals with the tool, which was added to IDE BlueJ in order to improve support of Architecture First methodology by this IDE and which allows students to show kopenogram of selected method in a simple manner in interactive mode.

## 1 Introduction

Object oriented programming languages are currently the most widely used around the world. Simultaneously, everything indicates, that this situation will persist. It is therefore natural, that the most widespread methodology of teaching programming is Objects First. For this purpose educational IDE BlueJ was developed.

This IDE allows to start teaching programming in really objective way. It means that teaching starts in interactive mode, which allows a better understanding of object-oriented principles for students. Unfortunately, this methodology quickly abandones its premise and interpretation is starting to go towards the syntax an algorithmic constructions. So, even though the Objects First methodology is based on the excellent idea, untapped potential is obvious at first sight.

Architecture First methodology, developed on the Department of Information Technologies at the University of Economics in Prague, is trying to eliminate these drawbacks. This methodology claims, that if students have to learn how to make really good object oriented architecture of application, they have to come into contact with this from the beginning, in order to have enough time to acquire thinking in objects. Therefore students are on the layer of architecture since the first lessons. This ensures that students are not unnecessarily distracted by syntax and programming construction, so they can fully concentrate on architecture design of application.

All the code is created by code generator, which allows working in the interactive mode [2]. Such generator is currently also integrated in BlueJ. However this code generator is very simple, and therefore we developed its enhanced version in order to


J. Janech et al. (eds.), *Proceedings of the 2015 Federated Conference on Software Development and Object Technologies*, Advances in Intelligent Systems and Computing 511,
DOI 10.1007/978-3-319-46535-7_8

better meet the needs of the Architecture First methodology. The textbook [4] can serve as an example of how teaching is designed in accordance with this methodology. However it is important to say, that in time, when this book was written, IDE BlueJ was not modified for the needs of Architecture First methodology, so the book does not use the advanced code generator features, which were added later.

Though the Architecture First methodology tries to postpone the explanation of algorithmic construction as much as possible, it is obvious, that students will sooner or later come into contact with it. After the object analysis is completed, it is often inevitable to design more complex algorithmic construction.

In order to allow students to better absorb interpretation of algorithm, several method of visualization of their structure are used. According to the Architecture First methodology the best way is to use kopenograms. The reasons are described in the next section.

## 2 Kopengrams

### 2.1 History of Kopenograms

Currently the most common form, which is used for representation of algorithm, is flowchart. The disadvantage of flowcharts lies in the fact, that they do not coerce users to construct algorithm in accordance with the principles of structured programming.

One of the responses to elimination of this drawback was the emergence of Nassi-Schnneiderman diagrams that came with it, but brought another problem. For the representation of the condition it uses oblique lines, which were difficult to display on alphanumeric displays used, in that time. In the eighties this shortage motivated the creation of kopenograms, which use for its representation colored rectangles.

### 2.2 Syntax of Kopenograms

The basic structural element of kopenogram is a block, which presents specific element of the algorithm. This element is represented by a rectangle, filled with a color derived from the meaning of the displayed element. Individual elements can contain other elements in its bodies. It means that elements are nested into each other, which shows structure of algorithm. The form of algorithmic block varies, depending on their meaning. Individual elements can contain following parts [3]:

**Header** – forms the upper section of the block and is tinged with a darker shade of its color. It also contains a text representing the name of this element or another form of description of its meaning.

**Body** – it is the biggest part of the element and is filled with lighter shade of appropriate color. It can contain another blocks. They are always displayed on the given level bellow each other, which clearly show their sequential execution.

**Footer** – optional lower part of the block, which represents the end of the cycle body. It uses a darker shade of the block color.

These parts are separated by horizontal parting line. Moreover some elements consist of several separate parts, which are horizontally or vertically connected (which means not in the sense described above). An example of a kopenogram is shown on Fig. 1.

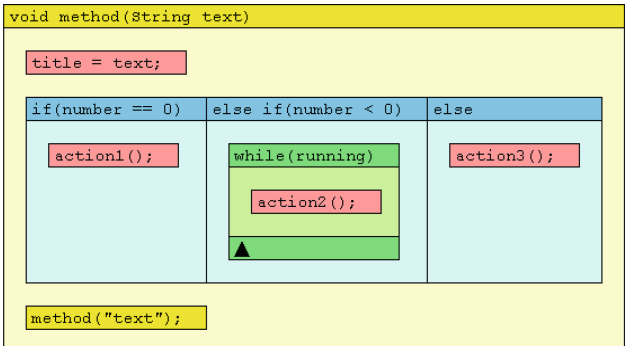



**Fig. 1.** Example of kopenogram

### 2.3 Basic Blocks

As was mentioned, one of the important characteristic of the kopenogram blocks is their color. It carries the advantage of easy orientation in algorithm, even from the distance, which means without having to read the labels of individual elements. The base consists of following four colors [1]:

- **Yellow**, which shades illustrate methods (procedures/functions) and recursive calls.
- **Green** is used for representation of cycles.
- **Blue** shows conditional commands and switches.
- **Red** is used to tint blocks representing individual elements, except that in the form of recursive call, as was mentioned.

### 2.4 Other Blocks

Blocks represented on Fig. 1 are basic, because they fully comply with structured writing of algorithm. However, sometimes situations can occur, in which it is better to violate principles of structured design of algorithm. Therefore were added representations of commands, which on the one hand violate principles of structured programming, however on the other hand, they can often simplify the whole algorithm [1].

The premature termination of loop (command **break** or **continue**) and premature termination of whole algorithm (command **return**) are two of them. Representation of these elements in kopenogram shows Fig. 2, where the red rectangle with white triangles oriented to the right represents the break command, for leaving endless while loop in case of running is set to false.

The continue statement is usually drawn similarly, except that the white triangles are facing upwards.

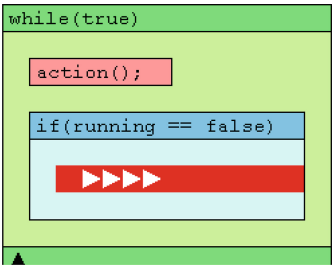



**Fig. 2.** Representation of premature loop termination

### 2.5 Exception Handling

With the advent of Java, work with exceptions has become a regular part of beginner programming courses. Therefore, it was necessary to take this into account in kopenograms [1]. An exception-handling mechanism can be viewed as a special composite block, consisting of a part, in which it can be expected, that exception will be thrown, and of one or more block, which this exception catch and process. The kopenogram showing the exception handling mechanism is on Fig. 3.

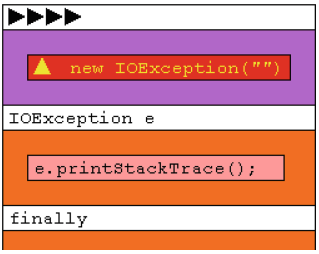



**Fig. 3.** Representation of exception throwing and catching

The block, in which we can expect exception throwing, is colored purple. Its header is white, with black triangles, illustrating the entry into the body of this block, which contain statements, potentially throwing exception. The deep red block with yellow text means, that the exception is thrown, which is processed in an immediately downstream orange block with white header.

## 3 Implementation in BlueJ

As was indicated in the introduction, for the purpose of improving support of the Architecture First methodology in BlueJ IDE, functionality was added, which allows users to show the algorithm of the selected method by a kopenogram. This should support especially the last part of teaching, in which the more complex algorithmic

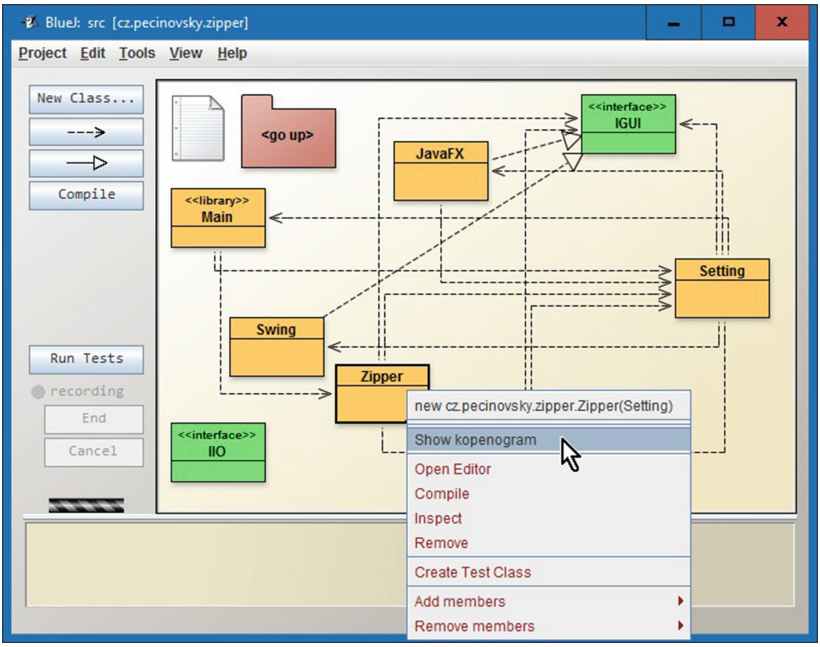



**Fig. 4.** Class context menu with the **Show kopengram** menu item

constructions that exceed the abilities of integrated code generator are designed. For better support of the Architecture First methodology, this tool is integrated into BlueJ.

For this purpose, class and object context menus were enriched with the Show kopenogram item (see Fig. 4), which invokes the dialog box, for selecting the method, which kopenogram should be shown (see Fig. 5).

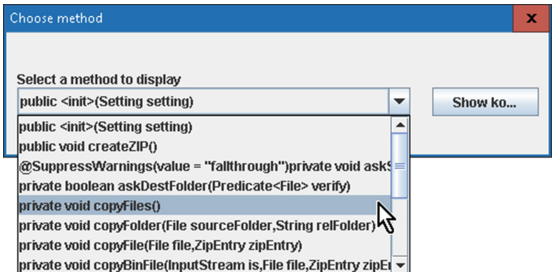



**Fig. 5.** **Choose method** dialog

After selection, user can display kopenogram of the selected method by pressing the Show kopenogram button. Each kopenogram is depicted in a separate window. It allows comparing more algorithms, if necessary (for example in case of editing code of displayed method).

Kopenograms are created during compilation. Invocation dialog box from local menu of data types in the class diagram and objects in object bench differs in the set of offered methods: objects show only instance methods, whereas classes show all. For that reason it is not necessary to create instance of appropriate class, in case of need to show kopenogram of its instance method.

Regarding colors, their meaning is given, however it is possible to define own colors for individual elements by editing configuration file bluej.defs.

In addition to elements described above, this tool can also display blocks, labels and static blocks (which are in their principle the same as methods). An example of complex method, which contains also empty blocks and a label, is shown on Fig. 6.

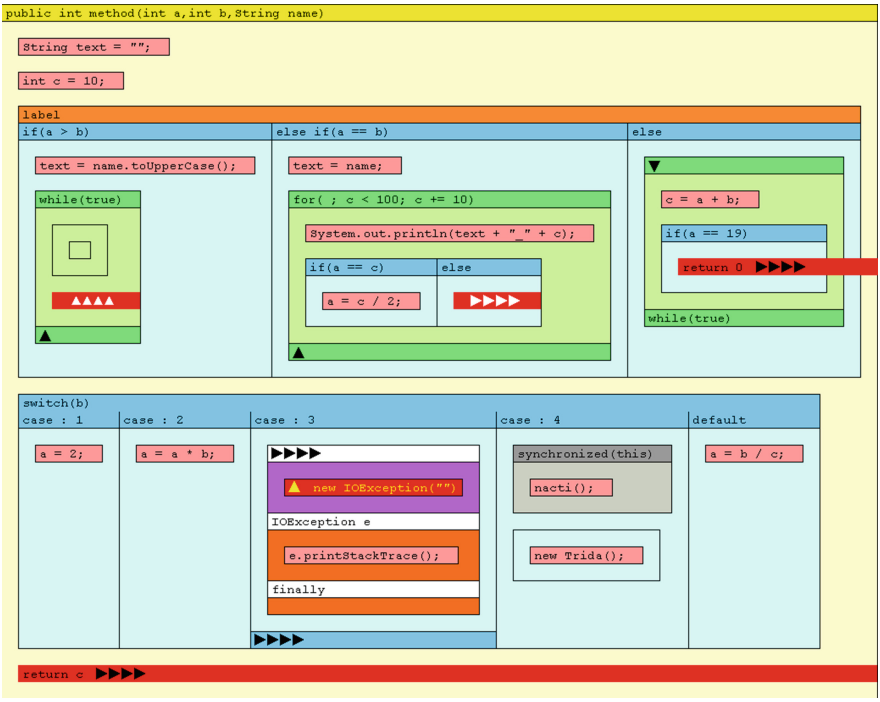



**Fig. 6.** Kopenogram of a complex method

All of the included images were created in described tool and they are also an example of its output.

The logic of building a kopenogram is based on the abstract syntax tree offered by the compiler. It means that each element (besides element representing method) keeps a link to its parent in itself and also to all its subelements. The width of the element is in the base deducted of its title (in case of element without title default width is set). Default height is sum of head height and body height of element. If the element contains subelements in its body, it is necessary to reflect it in its dimensions. Before

drawing of a kopenogram to the window, the right dimensions of each block have to be set. The algorithm of calculating the size of kopenogram works as follows:

- If the element contains one or more subelements in its body, then it asks its first descendant to prepare dimensions. This process is repeated until there is no subelement left.
- Then determines whether or not is the descendant larger than parent. If so, parent element will assume its size, plus the size of betting on both sides.
- Since all elements on the same layer are displayed below each other, they affect the height of their parent, regardless of their size (it is because even in case of a smaller element, its indentation exceeds the base height of the body of the parental element).

However the knowledge of dimensions is not enough to draw the kopenogram. It is necessary to prepare the position of each element. For adjusting position, the element which is to be depicted has to know previous depicted element. It is public information, so the element only has to determine, in what relationship with this element is, and on the basis of it prepare its own position. The painting of elements is performed in the reverse order than adjusting the dimensions. So the element, which sets its dimensions last, is drawn first. This ensures that the elements do not overlap each other.

## 4 Future Plans

The kopenograms generator, described in previous section, works reliably, however there is still place for several improvements. One of the first things, that should be resolved, is that in case of long line of code, the final kopenogram is too large to fit on the screen.

This shortage is currently most significant in case of streams. It is because kopenograms are based on parsing of abstract syntax tree, which means, that even if user writes individual statements bellow each other, the tool represents them as one long command and does not deal with the fact, that user wrote it differently. However, even after this treatment, still a situation can occur that final kopenogram will be unable to fit the screen. Solution for this could be to allow export kopenogram in form of picture, which is than possible to adapt the screen.

Another feature, which will be good to implement, is possibility of settings, allowing to choose whether to display full syntax of the blocks and statements, or whether to show only simple description of meaning of individual elements. It would be useful in the time, when students should not be distracted by syntax.

However, the most important future improvement is to allow program debugging using the kopenograms. It means that particular element, which is executed, would be highlighted. So the user would have much more vivid idea, how exactly the particular algorithm works. And even without the knowledge of syntax.

## 5 Conclusion

The first section of this paper describes the reason of creation of Architecture First methodology and mentions its main principles. It also reminds that we still do not avoid design of more complex algorithm. The next section is about the history of kopenograms and explains the reason for its creation. In the rest of this capture the syntax of kopenograms is explained. The third section describes the tool, which was integrated into BlueJ and which allows users to display kopenogram of the selected method. The last capture suggests the future changes and improvements, which can be expected.

## References

1. Pecinovský R.: Kopenograms and their implementation in NetBeans. In: Proceedings of the 38th International Conference on Software Development, Ostrava 2012. ISBN:978-80-248-2669-1
2. Pecinovský R.: Principles of the methodology architecture first. In: Objekty 2012 – Proceedings of the 17th International Conference on Object-Oriented Technologies, Praha 2012. ISBN:978-80-86847-63-4
3. Kofránek, J., Pecinovský, R., Novák, P.: Kopenograms – graphical language for structured algorithms. In: Foundations of Computer Science, FCS 2012, pp. 90–96. CSREA Press, Las Vegas (2012). http://world-comp.org/proc2012/fcs/papers.pdf ISBN:1-60132-211-9.
4. Pecinovský, R.: OOP – Learn Object Oriented Thinking and Programming. Eva&Tomas Bruckner Publishing, Czech Republic (2013). http://pub.bruckner.cz/titles/oop. ISBN 80-904661-8-4

# Simulation of Hydrological Processes by Optimization Algorithm Using Continuous Function

Martin Chlumecky(✉)

Faculty of Electrical Engineering, Department of Computer Science,
Czech Technical University in Prague, Prague, Czech Republic
chlummal@fel.cvut.cz

**Abstract.** The aim of hydrological models is to represent physical processes of catchments. The models provide information about hydrological processes. Evapotranspiration is a main model output. The quality of the model output is highly dependent on a model calibration which is not a simple process. SAC-SMA is one of rainfall-runoff models. In SAC-SMA, evapotranspiration is defined by 12 discreet values. Optimization algorithms do not return valid calibrations for specific basins or time periods because model parameters are correlated or also, because the optimal solution found by the algorithm is not applicable in hydrology. Some model parameters can be interpreted by a continuous function. It provides higher precision of evapotranspiration and a faster optimization run than it is provided by the definition of evapotranspiration by 12 discreet values. The main goal of this article is to determine a continuous function which can replace the 12 values definition of evapotranspiration. It should make the optimization process faster because the continuous functions are described by less than 12 parameters. Evapotranspiration is very similar to the Gaussian function. Genetic algorithm has been modified to SAC-SMA model optimization which uses the continuous functions for a description of evapotranspiration. This modification has brought interesting results. The continuous definition of evapotranspiration provides better results for specific time periods. The optimization speed can be up to a one third faster. The next step will be to confirm the approach by another catchments and time periods.

## 1 Introduction

Hydrological knowledges and skills are gained and improved by mankind for hundreds of years. In the 19th century, a bigger expansion of hydrology was noticed. Hydrodynamical models have been established at this time but they had been enough exact before. Nevertheless, computation of the models and data collection was very time-consuming.

Everything has changed when computers appeared. It caused a huge development of mathematical and hydrodynamical models. Their broad use in practice became real mainly due to personal computers which enabled to work with models more consistently and operatively. Although, we could see a huge development of hydrological models for last 35 years. Their common and practical use has just been dawning.


J. Janech et al. (eds.), *Proceedings of the 2015 Federated Conference on Software Development and Object Technologies*, Advances in Intelligent Systems and Computing 511,
DOI 10.1007/978-3-319-46535-7_9

The aim of hydrological models is to represent physical processes within each basin of interest. The models create streamflow hydrographs at the basin outlet that reproduce the appropriate observed hydrographs. The model, which we are interested in, is generally called rainfall-runoff model.

Model components estimate basin states (e.g. snow pack liquid water equivalent, snow pack energy properties, soil water volume and channel water volume) and outputs (rain plus snow melt, runoff, and channel flow). The rainfall-runoff model estimates states and outputs at each step of a model simulation.

The main output of the simulation is evapotranspiration. Evapotranspiration (ET) is the sum of evaporation and plant transpiration from the Earth's land and ocean surface to the atmosphere. Evapotranspiration is a significant water loss from drainage basins whereby a flow of a basin can be predicated [10].

It is not easy at all. It is necessary to calibrate the models in an effective way to be usable in practice. Results of a simulation should correspond to the reality as much as possible. The models are calibrated with historical data of a specific basin. The successfully calibrated model is used for many purposes which are mentioned below, e.g.:

1. One of the main purposes are *hydrological forecasts*. Thanks to them, it is possible to estimate behaviour of a basin during various precipitations and to estimate the runoff. It is possible to detect probability of floods and to determine size of a flood wave.
2. *Environment influences* a basin behaviour. The model outputs provide useful information about stocks of groundwater. This allows to determine for instance, how quickly vegetation of a basin can grow or probability of floods.
3. The model is also able to predicate *the influence of vegetation cover* changes to capacity of groundwater stocks. As an example we could mention a dry standing spruce forest remained after the bark beetle calamity on Šumava. The forest change influenced the groundwater stocks [19] and it caused the flood in 2002 [20].

The model calibration is very demanding. Huge number of calibrated parameters is needed. Users of models need to have large knowledge of hydrology and calibrated basin. Also, it depends a lot on experience and intuition. "The model calibration is more an art than a science" [21].

## 2 Problem Statement of the Model Calibration

There are many rainfall-runoff models available. The Institute of Hydrodynamics of the Academy of Sciences of the Czech Republic (IH) uses the Sacramento Soil Moisture Accounting (SAC-SMA) Model. This model is composed by three sub-models: snow accumulation and ablation (SNOW-17), basin rainfall-runoff model (SAC-SMA) and Unit-hydrograph channel routing procedures (UNIT-HG). These sub-models are described more precisely in [22–25].

The model is parameterized by approximately 70 values. Distribution of parameters between the sub-models is following: SNOW-17 is scaled with 24 parameters, SAC-SMA with 38 parameters and UNIT-HG with 3 up to 20 parameters. The total count of the UNIT-HG parameters is dependent on a simulated basin.

The goal of the calibration is to extract the information contained in calibrated data effectively [11]. It follows that the main aim is to calibrate SAC-SMA model as efficiently as possible. More precisely, the simulated flows should be as close as possible to the observed flows. The simulated flow is a general output of the model. Evapotranspiration is another important model output. The more precisely the model is calibrated, the more accurate evapotranspiration is simulated [12].

### 2.1 Optimization Problems

The effective calibration brings a lot of attendant problems, not only hydrological problems but also optimization ones.

The first problem is to find an acceptable compromise between optimization speed and calibration quality. The more iterations are processed by an algorithm, the more time it takes [1]. Count of interactions depends on simulated basin and/or time period. The root cause can be a variety of surface watershed or the weather. In case of too high variety, an algorithm needs more time for computation to find the best calibration.

The second problem is a validity of the calibration output. The algorithm is able to find the calibration successfully and deterministically. However, the calibration could contain model parameter settings not completely applicable in hydrological scope. Even though the calibration is found by the algorithm, it is not usable at all.

The third problem is to determine a fitness function. The fitness function identifies quality of the found solution (model calibration). This function uses several statistical indicators for deviation measurement of simulated and observed flows. Root Mean Square Error (RMSE) or Correlation Coefficient (CC) are the most common used.

The fourth problem is purely an optimization issue. Some parameters of SAC-SMA model are in close correlation. It is important to note that the close correlation of parameters is not connected with CC. It is completely different.

The presence of an experienced hydrologist is needed for all problems described above. The hydrologist will decide about quality of the resulting calibration.

### 2.2 Optimization of Evapotranspiration

The original implementation of SAC-SMA model defines evapotranspiration discretely through 12 monthly values. For each month, a value is determined to indicate evapotranspiration in the given month. For each day, daily evapotranspiration is computed as a linear interpolation according to the formula [13]:

$$ET_{actual}(m,d) = ET_{m-1} + d\frac{ET_0 - ET_m}{f(m)} \tag{1}$$

$f(m)$ – count of days in the month $m$.

$ET_{\mathrm{i}}$ – evapotranspiration value of the month $i$.

This discreet description of evapotranspiration can often cause unreal variations in a waveform which should reflect a real evapotranspiration [18]. Then the waveform with the variations described by 12 values does not correspond to the reality of the simulated basin.

## 3 Related Works

The total number of optimized parameters is really large. It is not possible to search the whole state space of all solutions [14]. The largest representation of rainfall-runoff model optimization is the Genetic algorithm (GA) [17].

The article [15] is aimed at a GA application on SAC-SMA model. It describes the basic idea and derives the fitness function based on the sensitivity analysis of model parameters. It uses two main statistical approaches for the fitness function. One of them is the RMSE with regards to the median which allows a good agreement of the peak flow with respect to time and volumes. The genetic algorithm has been successful in the context of searching, optimization and teaching [2]. GA can also be applied to the parameter calibration of conceptual rainfall–runoff models [3, 4]. The simple GA consists of four steps: fitness, reproduction, crossover and mutation steps. These steps allow flexibility, but make difficult to obtain the optimal model parameters. Additionally, they require more computation time in a complicated system associated with a number of parameters. Numerous investigations are proposed to modify GA to fast yield a more accurate optimum solution at order of magnitude [5, 6].

The results of the comparison of three different algorithms are described in [16]. There are compared the following optimization algorithms: Genetic Algorithm, Pattern Search and Shuffled Complex Evolution. The analyses were conducted using a conceptual rainfall-runoff model applied both to a single basin and to a complex basin. In the real world case, its solutions were stable but characterized and produced a very unstable set of parameters.

Duan in [17] deals with a global optimizing of general watershed models. He describes local and global optimizing methods. The local methods were used in past when the computing capacity was very limiting. Whereas, the global methods have brought a great potential. The article discusses thirteen methods for model optimization by the use of brute force and/or heuristic methods.

The aforesaid articles do not mention any procedure how to effectively optimize model parameters for basins with inhomogeneous environment, unstable vegetation or weather.

## 4 Approach

The challenge is to find and test a procedure for better and more realistic optimization of evapotranspiration on inhomogeneous basins. There is an effort to minimize a count of optimized SAC-SMA parameters. The main parameter which influences the quality of calibration is evapotranspiration. It is defined by twelve discrete real values which resemble the Gaussian curve with a slight deformation by their annual cycle.

The main objective is to identify the best continuous function which reproduces a desired shape of evapotranspiration specified by 12 values in the best way. The issue is evapotranspiration shape which is usually a bit different for each river basin or time period. The Gaussian curve briefly described above and diffusion function are applicable for this issue.

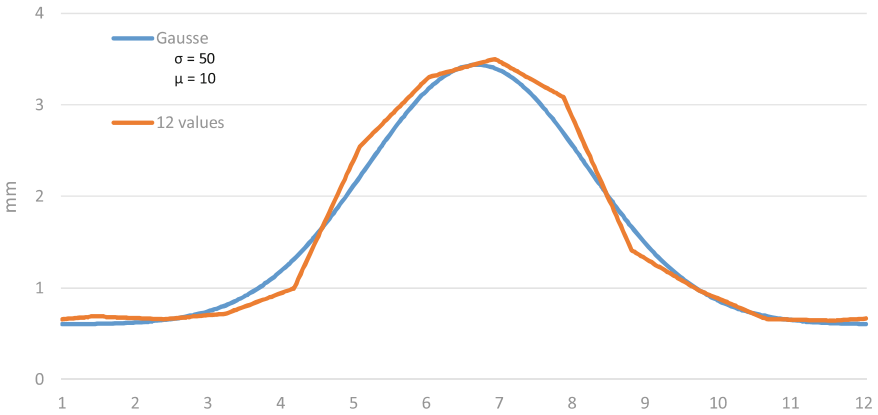



**Fig. 1.** The Gaussian function and the 12 values definition which define evapotranspiration

The benefit of these functions is their continuum which eliminates the linear interpolation (1). The definition of the Gaussian function prevents the unreal deflections which can be caused by 12 values. It is described for Liz basin in [18]. The disadvantage of the functions is ET shape unreality for specific time periods. The Fig. 1 presents a course of evapotranspiration which is defined by 12 values and the Gaussian function.

The Gaussian function is defined analytically as follows:

$$f_{gauss}(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{(x-\mu)^2}{2\sigma^2}} \tag{2}$$

The parameter σ is its standard deviation with its variance then $\sigma^2$. The μ is the mean or expectation of the distribution.

Diffusion function is similar to the Gaussian function with a symmetrical difference. The symmetry can be deformed easily as illustrated in Fig. 2.

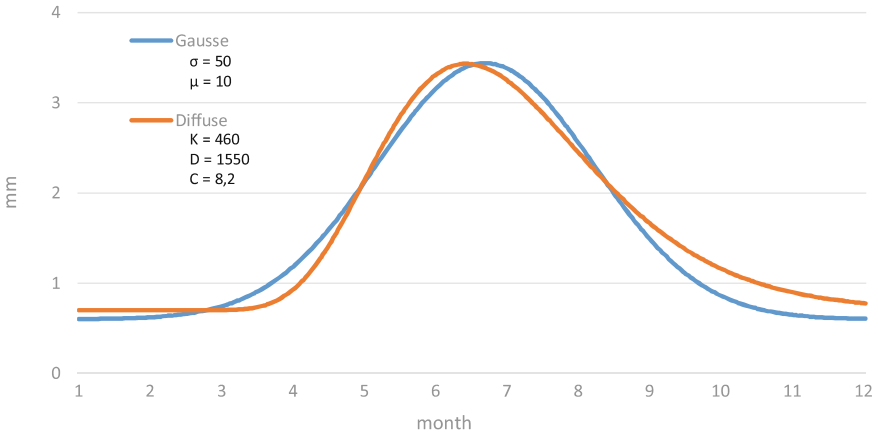



**Fig. 2.** Comparison of the Gaussian and the diffuse functions

The Diffuse function is defined analytically as follows:

$$f_{diff}(x) = \frac{1}{4\pi K}\frac{D}{\sqrt[3]{x^2}}e^{\frac{(Cx-D)^2}{4Kx}} \tag{3}$$

where $K$ is the density of the diffusing, $C$ is the collective diffusion coefficient.

From the functions mentioned above, it is evident the both functions can be scaled by 2 respectively 3 variables. It is less than the legacy definition of evapotranspiration using 12 values. The analytic forms are in the normalized form. Two additional variables have to be added due to field values. The variables change field values of functions (2) and (3) form $<0, 1>$ interval to $<0, x>$ interval, where $x$ is a maximal evapotranspiration value of a simulated basin.

The primary candidates for the function which could represent evapotranspiration are the Gaussian and the diffuse functions.

The analytic form of evapotranspiration has several advantages. The count of optimized parameters is smaller and therefore the optimization algorithm is able to search state space of possible solutions faster. Evapotranspiration values are more precise because the linear interpolation (1) is not used. The linear interpolation is always discreet essentially as the values are computed from discreet values.

The next step is to modify the Genetic algorithm which has been used and described in [14]. The main variation is an input of evapotranspiration and a modification of the fitness function. The article [14] uses Correlation coefficient (CC) as the fitness function. CC is not suitable to be used in this way as indicated in [17].

Data of Elbe basin will be used to confirm results. Experiments with the Gaussian function, the diffuse function and the 12 values ET definition will be performed on 1946–1965 time period. Only evapotranspiration will be optimized because it is required to confirm benefit of evapotranspiration defined by the continuous function. The remaining model parameters will not be optimized. Their values have been derived from previous simulations described in [8].

### 4.1 Optimized Parameters

The (1) and (2) functions are optimized including the additive (a) and multiplicative (m) variables which extend field values. The resulting optimized functions are as follows:

$$f_g(x) = a + m\frac{1}{\sigma\sqrt{2\pi}}e^{\frac{(x-\mu)^2}{2\sigma^2}} \tag{4}$$

$$f_d(x) = a + m\frac{1}{4\pi K}\frac{D}{\sqrt[3]{x^2}}e^{\frac{(Cx-D)^2}{4Kx}} \tag{5}$$

For each parameter of Eqs. (4) and (5), it is necessary to determine the range of optimized values used for the algorithm. Also, it is necessary to determine the default values for the parameters. These default values and the range of the parameters are determined of the evapotranspiration shape which is derived from [8].

### 4.2 Quality of Optimization

The RMSE focusing on the median is selected as a fitness function for a new version of Genetic algorithm. It is defined as follows:

$$RMSE(x) = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left[(x_i - \bar{x}) - (y_i - \bar{y})\right]^2} \tag{6}$$

The fitness function is advantageous especially for unstable time periods and basins. It is discussed in [7]. This is the exact requirement for this purpose because it reflects the median deviation of simulated and observed flows.

Another measure of the approach is sum of smaller differences between an observed flow ($Q_{\text{obs}}$) and simulated flows ($Q_{\text{sim}}$), where $Q_{\text{sim}}$ are simulated flows computed by using (4) and (5). The exact definition of measurement is following:

$$\Delta ET = \sum_{i=1}^{n} f(Q_{obs}(i), Q_{sim}(i), Q_{gauss}(i)) \tag{7}$$

$$\Delta ET = \sum_{i=1}^{n} f(Q_{obs}(i), Q_{sim}(i), Q_{diff}(i)) \tag{8}$$

$$f(o,s,x) = \begin{cases} 1 & |o-x| < |o-s| \\ 0 & other \end{cases}$$

Other metrics are based directly on SAC-SMA model described in [13]. The metrics are focused on another time intervals such as daily and monthly intervals.

### 4.3 Optimization

The optimization will be performed on Elbe basin using the following evapotranspiration definitions:

1. *The 12 values definition*
2. *The Gaussian function*
3. *The Diffuse function*

Results of the optimized simulations will be used to evaluate the approach which defines evapotranspiration using the continuous function.

Basic parameters of the new Genetic algorithm are following:

POPULATION SIZE defines the count of solutions created during one iteration. GENERATION LIMIT determines the count of created iterations. The other parameters which scale the algorithm are described in [9].

Beyond the (6), (7) and (8) metrics, also purely optimization metrics are monitored. One of the metrics is a speed of calculation and a count of searched solutions.

# 5 Results

## 5.1 Statistical Results of SAC-SMA Model

The following Table 2 summarizes the results of the performed simulations. The major statistical data representing the error models, respectively error between $Q_{obs}$ and $Q_{sim}$ are monitored.

The Fig. 3 demonstrates deviations of simulated and observed discharges by (7). It is an optimization defined by the 12 values definition and the Gaussian function (4). The metric (7) returned a result 3680 which is 53% smaller count of differences in comparison with the simulation using the 12 Values definition.

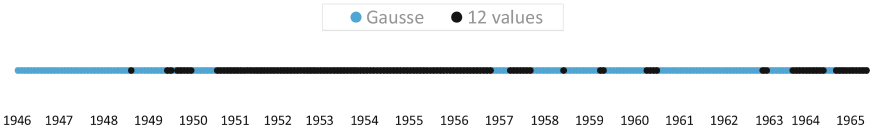



**Fig. 3.** Deviations of the Gaussian function and the 12 values definition

The Fig. 4 is similar to Fig. 3 but it is using the metric (8). It is an optimization using the 12 Values definition and the diffuse function. The metric (7) returned a result 3497 which is 47% smaller count of differences in comparison with the simulation using the 12 Values definition.

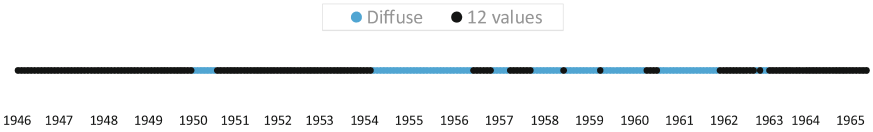



**Fig. 4.** Deviations of the diffuse function and the 12 values definition

## 5.2 Optimization Algorithm

Calculation Speed defines time required for completion of the algorithm. It means processing of all generations, see Table 1. Approach speed to optimum defines how

**Table 1.** Settings of genetic algorithm for SAC-SMA model optimization

| Parameter | Value |
|---|---|
| POPULATION SIZE | 100 |
| GENERATION LIMIT | 1000 |
| Tournament Selector | 60 |
| One point crossover | 0.39 |
| Simple mutator | 0.33 |
| Elitism | 5 |

**Table 2.** Comparison of the simulations by evapotranspiration definitions

| Metric | 12 Values | Gaussian | Diffuse |
|---|---|---|---|
| Daily RMS error | 9.295 | 9.246 | 9.352 |
| Daily AVG ABS error | 86.404 | 85.487 | 87.464 |
| AVG ABS monthly volume error | 2.289 | 2.045 | 4.713 |
| RMSE | 9.534 | 11.348 | 24.740 |
| Correlation coefficient | 0.8196 | 0.8220 | 0.8149 |

**Table 3.** Comparison of genetic algorithm metrics

| Metric | 12 Values | Gaussian | Diffuse |
|---|---|---|---|
| Calculation speed [ms] | 1713414 | 1837857 | 1668371 |
| Approach speed to optimum [iterations] | 870 | 120 | 230 |
| Highest score | 133.4 | 318 | 223 |
| Avg score | 20.07 | 30.2 | 50.3 |
| Calculation speed [ms] | 1713414 | 1837857 | 1668371 |

long the algorithm must operate until the fitness function (6) is approaching to the limit solution. Highest and Avg Score are statistical data for highest and average values of all individuals which are generated across the whole algorithm run (Table 3).

## 6 Discussion

Overall, optimization results are interesting of the hydrological as well as of the software perspective. Both the continuous functions (2) and (3), which define evapotranspiration, are valid and useful from the viewpoint of hydrology. The results of SAC-SMA model, which use the continuous functions, are usable in real.

### 6.1 Comparison of Evapotranspiration Definitions

It is not possible to determine the most effective definition of evapotranspiration clearly. The Gaussian function (2) seems to be the best solution. It is confirmed by statistics in Table 2. There is an exception for Monthly volume RMS error that defines a deviation worse for 1.8 compared to the best solution. In the terms of statistics, it is a minor deviation because it is not at order of magnitude.

The diffusion function (3) seems to be the worst variant of evapotranspiration definition. The deviation is acceptable at order of magnitude. The reason of greater RMSE is a slight ET deformation in the months 9 up to 12. It follows that the realistic shape of evapotranspiration is most similar to the Gaussian curve (2). It illustrates Fig. 1.

The Figs. 3 and 4 confirm that different definitions of ET are preferred for different time periods. Evapotranspiration defined by the Gaussian function gives the most favorable results. The second ranking is the 12 Values definition and the last one is the

diffuse function. This implies that a modification of the algorithm would be beneficial. The modification would apply to evapotranspiration calculating. The use of ET definition would adjust to a specific simulated time period. This would lead to intersection of the best solutions.

### 6.2 Quality and Speed of Optimization

Both definitions of evapotranspiration do not affect the calculation speed at order of magnitude. The calculation speed is approximately 28 min.

Approach speed to optimum is a more interesting metric. Evapotranspiration by the 12 values definition provides the worst results. The algorithm has to execute more iterations at order of magnitude than the other evapotranspiration definitions. On the other hand, the Gaussian definition or diffuse definition need a smaller number of iterations. This implies that it is possible to reduce the GENERATION LIMIT parameter almost for one third. It should accelerate the overall calculation time. In other words, less time should provide the same quality of simulations.

Highest Score and Avg Score are in the opposite status. Individuals with a huge error are created if the Gaussian or diffuse definitions are used. It is caused by the multiplicative variable. Whereas, probability of large amplitude curve creation is larger than the 12 values definition. However, because these simulations provide great results, high values of these metrics are not so important.

## 7 Conclusion

SAC-SMA model uses a large number of calibration parameters. Evapotranspiration defined by the symmetric continuous function reduces number of parameters which are used for its definition.

Evapotranspiration defined by the Gaussian or the diffuse function provides better results compared to the legacy definition in specific time periods. The continuous functions are more effective from the viewpoint of quality indicators. Differences between the indicators are not at order of magnitude so the continuous functions do not deteriorate simulation results. A significant advantage of continuous ET definition is more natural description of the natural process of evapotranspiration. Unreal evapotranspiration values caused by the linear interpolation (1) are eliminated. Evapotranspiration defined by the continuous function is more efficient from the viewpoint of the optimization process. Approach speed to optimum is up to 80% faster. The continuous definition allows to search faster in the state space of solutions. Optimized calibrations provide valid results. Evapotranspiration defined by the Gaussian function returns better results in comparison to the diffuse function.

### 7.1 Future Work

Definition of evapotranspiration by the continuous function appears to be a correct way to improve and accelerate the optimization of SAC-SMA model. The approach is

generally applicable to all optimization problems whose input can be interpreted by the continuous function.

The next step will be to confirm the approach on another catchments and other time periods than Elbe catchment area. It will be necessary to check whether the simulations using the new optimizing technique will be beneficial for different climate conditions as well. Simulations on dry or flood plains and time periods might also be very interesting.

In case it is established that different evapotranspiration definitions are effective for various time periods, then the algorithm will be modified so that it will choose an optimal definition for each time period. Lastly, the optimal evapotranspiration computation will be defined by the 12 values definition, the Gaussian function or the diffuse function for each time period.

## References

1. Okdem, S.: A simple and global optimization algorithm for engineering problems: differential evolution algorithm. Turk. J. Elec. Eng. **12**(1), 53–60 (2004)
2. Golberg, D.E.: Genetic Algorithms in Search, Optimization, and Machine Learning. Addion wesley, Boston (1989)
3. Wang, Q.J.: The genetic algorithm and its application to calibrating conceptual rainfall-runoff models. Water Resour. Res. **27**(9), 2467–2471 (1991)
4. Wang, Q.J.: Using genetic algorithms to optimise model parameters. Environ. Model Softw. **12**(1), 27–34 (1997)
5. Krishnakumar, K.: Micro-genetic algorithms for stationary and non-stationary function optimization. In: 1989 Advances in Intelligent Robotics Systems Conference. International Society for Optics and Photonics, pp. 289–296 (1990)
6. Franchini, M.: Use of a genetic algorithm combined with a local search method for the automatic calibration of conceptual rainfall-runoff models. Hydrol. Sci. J. **41**(1), 21–39 (1996)
7. Madsen, H.: Automatic calibration of a conceptual rainfall–runoff model using multiple objectives. J. Hydrol. **235**(3), 276–288 (2000)
8. Buchtele, J., Tesar, M.: Simulation of a rainfall - runoff process for the evaluation of variability in the river flow regime in small basins with vegetation changes. Folia Geographica **41**, 103–110 (2010). ISSN 0071-6715
9. Whitley, D.: An executable model of a simple genetic algorithm. Found. Genetic Algorithms **2**(1519), 45–62 (2014)
10. Allen, R.G., et al.: Crop evapotranspiration-guidelines for computing crop water requirements. In: FAO Irrigation and Drainage Paper, vol. 56, FAO, Rome (1998). 300.9: D05109
11. Gupta, H.V., et al.: Multiple criteria global optimization for watershed model calibration. In: Calibration of Watershed Models, pp. 125–132 (2003)
12. Wagener, T., Wheater, H.S., Gupta, H.V.: Identification and evaluation of watershed models. In: Calibration of Watershed Models, pp. 29–47 (2003)
13. Burnash, R.J.C., Ferral, R.L., Mcguire, R.A.: A generalized streamflow simulation system, conceptual modeling for digital computers (1973)
14. Chlumecký, M.: Optimizing of parameters in model (SAC-SMA). In: 17th International Student Conference on Electrical Engineering, POSTER 2013, Czech Technical University, Prague, pp. 1–6 (2013). ISBN:978-80-01-05242-6

15. Wu, S.-J., Lien, H.-C., Chang, C.-H.: Calibration of a conceptual rainfall–runoff model using a genetic algorithm integrated with runoff estimation sensitivity to parameters. J. Hydroinform. **14**(2), 497–511 (2012)
16. Franchini, M., Galeati, G., Berra, S.: Global optimization techniques for the calibration of conceptual rainfall-runoff models. Hydrol. Sci. J. **43**(3), 443–458 (1998)
17. Duan, Q.: Global optimization for watershed model calibration. In: Calibration of Watershed Models, pp. 89–104 (2003)
18. Chlumecký, M., Tesař, M., Buchtele, J.: The appraisals of long time series of evapotranspiration using modelling rainfall-runoff with optimized parameters. In: International work-Conference On Time Series: Proceedings ITISE 2014. Granada: University of Granada, pp. 1280–1291 (2014). ISBN:978-84-5814-97-9
19. Pavlásek, J., Máca, P., Ředinová, J.: Analýza hydrologických dat z Modravských povodí. J. Hydrol. Hydromech. **54**(2), 207–216 (2006). (in Czech)
20. Buchtele, J.: Influence of the development of vegetation cover in the basin on the regime of surface water and groundwater resources. The Institute of Hydrodynamics of the Academy of Sciences of the Czech Republic, Prague (2011)
21. Westerberg, I.K., et al.: Calibration of hydrological models using flow-duration curves. Hydrol. Earth Syst. Sci. **15**(7), 2205–2227 (2011)
22. Georgakakos, K.P.: Analysis of model-calculated soil moisture. J. Clim. **9**, 1350–1362 (1996)
23. NOAA: Modification of Sacramento soil moisture accounting heat transfer component (SAC-HT) for enhanced evapotranspiration, 72 s. Technical report NWS, 53 (2010)
24. Anderson, E.A.: National weather service river forecast system: snow accumulation and ablation model. US Department of Commerce, National Oceanic and Atmospheric Administration, National Weather Service (1973)
25. Aron, G., White, E.L.: Fitting a gamma distribution over a synthetic unit hydrograph. J. Am. Water Resour. Assoc. **18**, 95–98 (1982)

# Cache Module for the Dictionary Writing System

Kamil Barbierik, Martin Bodlák, Zuzana Děngeová, Vladimír Jarý(✉), Tomáš Liška, Michaela Lišková, Josef Nový, and Miroslav Virius

The Institute of the Czech Language, Czech Academy of Sciences, Prague, Czech Republic
{kamil.barbierik, martin.bodlak, vladimir.jary, tomas.liska, zjosef.novy}@foxcom.eu, dengeova@ujc.cas.cz, liskova@foxl.cz, miroslav.virius@fjfi.cvut.cz

**Abstract.** This paper focuses on implementation and benefits of the cache module for the dictionary writing system (DWS) Alexis. At first, the DWS Alexis that is being developed at the Institute of the Czech Language is introduced. We shortly present architecture of the system and briefly describe its main modules including list of entries, editing, or output modules. The main emphasis of this contribution is put on the recently developed cache module. We explain design and implementation of the cache, then we present its benefits for the dictionary writing system. At first, we analyze performance improvement of the output module, then we explain its roles in the fulltext search in the web interface of the dictionary and source of offline data for mobile devices. We conclude with a current state of development and plans for the nearest future.

## 1 Introduction

Currently, a new monolingual explanatory dictionary called Academic Dictionary of the Contemporary Czech (Akademický slovník současné češtiny, see [4]) is being prepared at the Institute of the Czech Language of the Czech Academy of Sciences. The dictionary is targeted at the native speakers with secondary education; final version is expected to contain approximately 150 000 lexical units. As a software support for the project, a brand new dictionary writing system called Alexis is being developed.

At first, we will discuss existing DWS and reasons for development of the new system. Then we will overview the architecture of the Alexis and software technologies used during development. We will introduce main parts of the system including output and editorial modules that strongly benefit from the recently implemented cache.

We present design and implementation of the caching facility. Originally, it served for decrease of load times in the output module, therefore we analyze performance improvement gained by caching. We also discuss other possible use cases of the cache module in the context of the dictionary writing system.

Finally, we summarize the current state of the Alexis system and its cache module and present some features planned for the nearest future.


J. Janech et al. (eds.), *Proceedings of the 2015 Federated Conference on Software Development and Object Technologies*, Advances in Intelligent Systems and Computing 511,
DOI 10.1007/978-3-319-46535-7_10

## 2 Motivation

At first, we needed to decide whether we use some existing dictionary writing system or we develop a custom one. We have investigated (see [1]) several commercial and open source system, some of them (e.g. DEB II, see [7]) also available for the Czech language. After considering price required to obtain license, amount of needed adjustments, we have decided for the development of custom system.

This decision allows us to fully respect demands of team of lexicographers; consequently they will work with system that suits their needs. Moreover, we will have control over source code and development cycle. On the other hand, development of custom system involves lot of work.

Currently, approximately 20 000 lexical units have been partially or fully processed and entered into the system. With increasing number of entries in the database, work with several parts of the system such as an output module becomes slower and slower. In this contribution we focus on design and implementation of the cache module that serves mainly for decrease of loading times of the output module. However, the cache is also used to simplify the fulltext search in the public web interface of the dictionary and as a source of offline data for mobile applications.

## 3 Overview of the DWS Alexis

In this section, we will briefly introduce the Alexis system. We will describe the overall architecture, then we will focus on the core parts of the system.

### 3.1 Used Technologies

According to the demands of team of the lexicographers, the system should be used simultaneously by multiple users from multiple places. In order to fulfill this requirement, we have decided to develop Alexis as a multilayer web application. This decision also greatly simplifies the deployment and maintenance of the Alexis.

The data layer is built on top of MySQL server. The database consists of approximately 100 tables centered around the heslo (i.e. lemma in the Czech language) table. Originally, we have used the MyISAM storage engine for its simplicity, however we have recently switched to InnoDB engine that supports ACID compliant transactions and foreign key constraint. Issues solved during migration are described in [5].

The server layer is implemented mainly in PHP language with support of several third party libraries such as snappy (for PDF export) or DiBi (for database access).

Finally, the application layer is based on HTML5 and CSS3 technologies, client side scripting is implemented in JS language with jQuery framework. Furthermore a new applications for tables and smart phones are currently under development.

To simplify the process of development, we have established a repository on the GitHub platform, [8]. We have adopted a branching model described in [6]: we use two main branches: the master branch represents stable code, while the devel branch represents the development version. In case some problem is detected in the master

branch, we create a new hotfix branch, repair the problem, and merge the hotfix both to the master and devel branches.

Furthermore, we use the Mantis issue tracking system (see [9]) to allow lexicographers easily provide information about detected issues within the stable version of Alexis.

### 3.2 Main Modules of the System

Alexis is a very complex system that can be divided into several more or less interconnected modules. We will introduce list of entries and editing module, then we will describe output and editorial modules in more details.

### 3.3 List of Entries

List of entries is the first page displayed when users login into the system. This list represents the macrostructure of the dictionary. Each lexical unit in the list is accompanied with list of its variants, its editor(s), or dates of creation and last modification. Furthermore, a notification is displayed for entries with active correction (see Fig. 1).

As the final version of the dictionary is expected to contain approximately 150 000 lexical units, we have divided the list into pages and implemented navigation between pages. Furthermore, list of displayed lexical units can be filtered using several tools: Quick Search tool and an advanced filter called xFilter. Quick search allows lexicographers to browse through almost any part of the lexical units including variant, word class, meaning definition, or exemplification. Depending on the chosen category, the user enters search term into text field (e.g. variant) with support for automatic completion and wild card convention or selects search value from fixed list of predefined values (e.g. word class). More advanced filters can be constructed using the xFilter tools that allows lexicographers to combine queries over multiple categories using

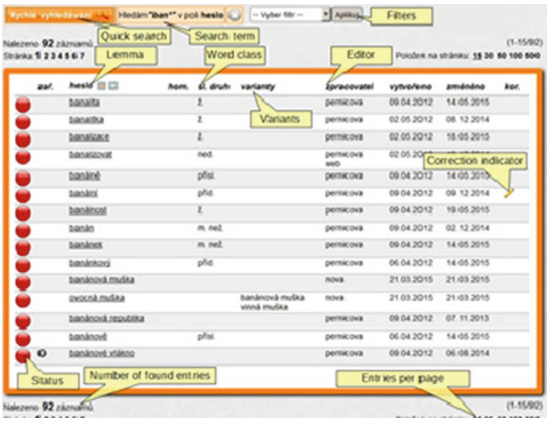



**Fig. 1.** List of entries displaying lemmas starting with *ban**

logical conjunctions. Set of predefined filters is also available - it is possible to show lexical units modified within specified time interval or units with multiple variants. Finally, users may manually select desired units.

From the list of entries, currently filtered units can either be opened in the editing module or sent to the output. When large number of items is currently filtered, generating of the output may take some time. After implementation of cache, time required to generate output decreased by factor of ten as described below.

### 3.4 Editing Module

The editing module represents the microstructure of the dictionary, it allows lexicographers to provide details of the lexical units. From the architecture point of view, it is implemented as a large HTML form. The form can be divided into four sections: header, variants, meanings, and cross reference. The header section holds general information about lexical units such as its editor, state (new, completed,…), type of lexical unit, date of creation, or notes for lexicographers and domain experts. The section also contains list of active corrections on the opened lexical unit.

Below the header section, lexicographers can enter details about variants of the lemma including morphology, word class, origin, thematic scope, or pronunciation. It is possible to create a new variant as a copy of the existing, to change order of variants, or to remove some particular variant. These operations are implemented using the JS language.

Because the Academic Dictionary of the Contemporary Czech will be explanatory dictionary, great emphasis is put on the meanings section. Each meaning can be accompanied with multiple of exemplifications. Panels with meanings can be added or removed on demand. It is also possible to exclude meaning or even particular exemplification from output. Moreover, panels with meanings can be minimized to save space.

Finally, in the last section of the editing form, it is possible to define references and links to other lexical units.

The editing form uses several standard HTML5 input elements such as input fields for entering short textual information (variant, comments), radio buttons, and checkboxes. We also use TinyMCE editor for entering rich texts such as meaning definitions or exemplifications. Furthermore, we have developed custom widget for selections from the fixed item lists. Lexicographers use these widgets to select one or more items from list of values that are managed by administrator of the system. More information about user interface can be found in [2].

In the background, *Central Data Object* (CDO) lies. It is the PHP class that encapsulates all the data related to the lexical unit. CDO also serves for retrieving the data from the database and for saving of the data back into the database. When editor decides to save changes to the opened entry, it needs to be marked as outdated in the cache.

From the editing module, lexicographers may return back into the list of entries or they can send the current lexical unit into the output module.

## 4 Output and Editorial Modules

As already mentioned, the output module can be launched either from the list of entries (in this case, all items in the active selection are displayed) or from the editing module (only currently opened lexical unit is printed). The output module is based on complex rules that define a way in which elements of the microstructure are formatted.

Alexis uses two main types of output: electronic and printed. The electronic output serves mainly as an entry point to the editorial module; correctors use it to propose corrections to the lexical units. This output is implemented as a web page - by clicking on any part of the lemma, a new correction popup dialog is opened. On the other hand, the printed output is implemented as a PDF file which cannot be edited. It shows the lexical units in a same way in which they will be printed in the final paper version of the dictionary. Some parts of the microstructure of the dictionary such as notes or selected exemplifications are excluded from this output. Furthermore, common information from multiple variants is merged in the printed output. We use the *wkhtmltopdf* tool to convert HTML code into the PDF file. Difference between electronic and printed output is demonstrated in Fig. 2.

Furthermore, a recently implemented web interface for general public can be regarded as a special type of output. The web interface consists of two main part - there is simplified list of entries (only variants are shown) together with quick search tool on the left. On the right, a slightly modified version of the printed output is shown. Additionally, the web interface is connected to external lemma processing tools and other existing online dictionaries. Users can export currently displayed lexical unit into the PDF file or use the fulltext search over entire dictionary.

Editorial module has been designed and implemented to simplify the process of corrections. Instead of printing HTML page with output on the paper and putting the

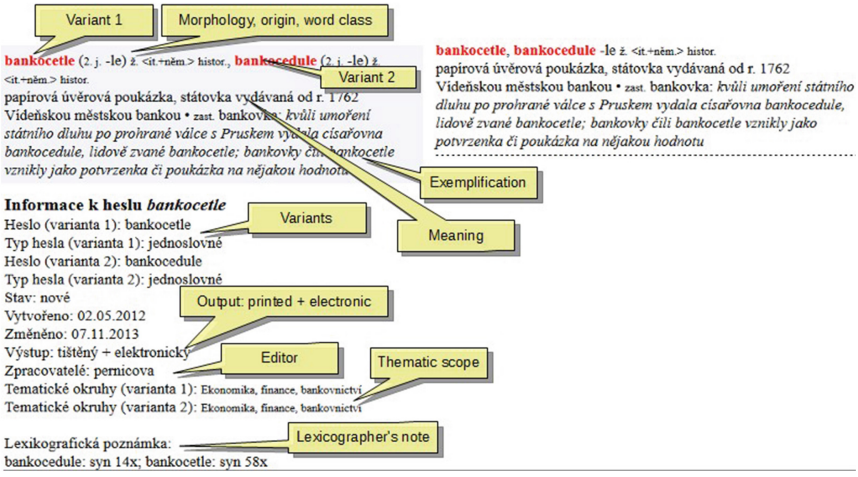



**Fig. 2.** Electronic output (on the left) and printed output (on the right) of the lemma *bankocetle* (i.e. treasure note)

corrections on it, correctors open the editorial module by clicking on problematic element in the electronic output. In the new correction dialog box, corrector sees the original value and enters the proposed new value, optionally with some comment. He or she may also select specific editors who will receive notification about the correction. Some smaller correction (such as typing errors) can be directly applied by corrector.

Active corrections are marked by special icon in the list of entries (see Fig. 1), furthermore, there is list of active correction for given lexical unit in the editing module. While a correction is active on certain field, the field is locked and corresponding correction needs to be either accepted and applied by editor or returned to the corrector. Again, when correction is applied on certain lexical unit, it needs to be removed from cache.

## 5 Cache Module

Lexicographers use the output module on daily basis to preview currently edited lexical units. With increasing number of processed units, displaying larger and larger lists took longer and longer. To decrease loading times and reducing load of the database, we have decided to implement the cache module.

### 5.1 Design and Implementation

In order to display lexical unit in the output module, almost entire database needs to be searched to retrieve all microstructure details. Output module then uses the predefined rules to format the loaded data and produce corresponding HTML code.

We have proposed and implemented cache as PHP class CacheManager. This class provides several main operations: saves lexical unit with given id into the cache, loads given unit from the cache, removes given unit from cache, and checks whether unit is stored in the cache. Cache works in the write through mode, i.e. the data describing given lexical unit are simultaneously stored in the main database and in the cache itself which is also implemented as a database table (see ER diagram in Fig. 3). The *cache* table is connected to the *heslo* (i.e. lemma) by Foreign Key constraint. When some lemma is deleted from the dictionary, this constraint ensures that its cached version is also removed. The *data* column holds the cached version of the lemma, while the *data_bz* contains the plain text version of the lemma which is used for fulltext search as described below. We need to store the time stamp of creation of the cache entry in order to be able to detect if the cache entry is valid (i.e. it has been saved into the cache after last update of the DWS software). Moreover, this time stamp is also important for synchronization with mobile devices (see below). The role, *server*, and *vystup* (i.e. output) columns are used to distinguish appropriate version of cached data for given user role (editor or domain expert), server (production or development), and output (electronic, printed, web).

The CacheManager has been integrated into output, editorial, and editing modules. According to Fig. 4, for each lexical unit in the active list of entries the output module

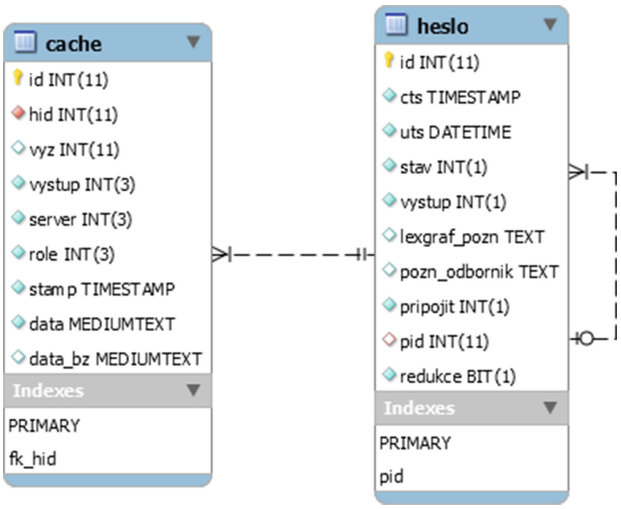



**Fig. 3.** Entity-relationship diagram of the cache

uses CacheManager, to verify whether a unit already exists in the cache (*cache hit*). If the entry is stored in the cache, it is loaded directly from it and displayed in the output. Otherwise (*cache miss*), the database is searched for all microstructure elements of the given lexical unit which are then formatted using the predefined rules. Generated HTML code is displayed in the output and stored in the cache for future use. However, each lexical unit described by its identification number may be present in the cache in several versions: there is difference between electronic, printed, and web outputs as each of these output is based on different formatting rules. Furthermore, the electronic output depends also on the user role (different result is presented to lexicographers and different to domain experts, e.g. Lexicographers see lexicographer's note in the output which is replaced by expert's note for domain expert). Finally, we use production and development servers (see above); each of them uses different formatting rules, therefore cache also needs to distinguish type of server. Each of these variables (type of server, type of output, user role) needs to be passed to methods of the CacheManager class in order to work with correct version of cached value.

It is possible to completely disable the cache module on development or production server via option in the configuration table in the database which is useful for debugging and testing purposes. When the cache is disabled, the entries are generated on the fly and resulting code is only displayed in the output but not saved back in the cache (see Fig. 4).

We need to guarantee that when a lexical unit is modified, it needs to be removed from cache in order to display the most recent version in the output. Therefore, editing form and editorial module use CacheManager to remove lexical unit from cache when it is modified, i.e. saved or when correction is applied. Furthermore, electronic output of lexical unit needs to be removed from the cache when a new correction is proposed or correction is rejected by editor because microstructure elements with active or

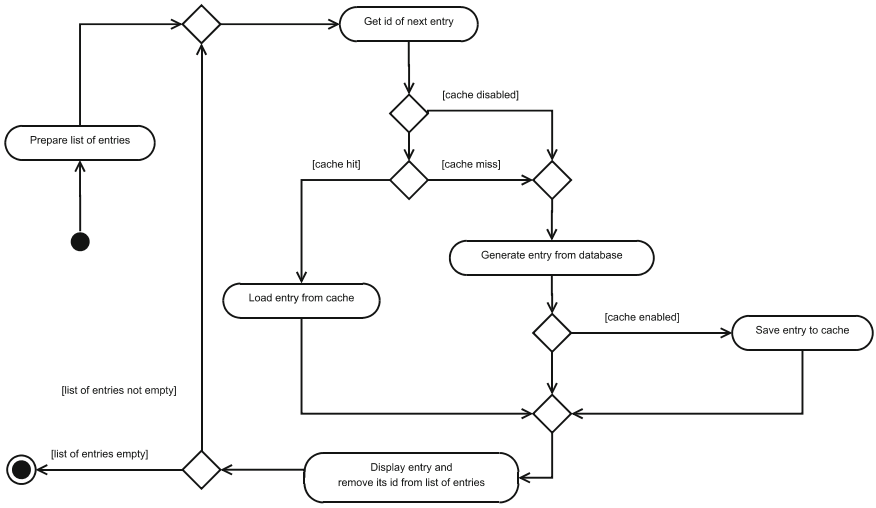



**Fig. 4.** Process of generation of output with cache implemented

rejected corrections are highlighted in the output. However, as a lexical unit may depend on nested items, antonyms, synonyms, or referenced items, all these related units needs to be removed from cache as well. We plan to integrate the Central Data Object (see above) more deeply into the system, so all read and write operations will be handled by it. This will simplify the process of cache update as all write operations will be handled by single function.

The Alexis is still under development and existing modules are being modified according to request from lexicographers: new microstructure elements are being added (and some are being removed) which has impact on editing and formatting rules used by output module. Therefore, when new version of Alexis software is deployed, entire cache must be cleared as new version. CacheManager watches time stamp of the VERSION file which is updated each time when a new version is deployed. When update is detected, the manager automatically clears the cache. However, when cache is empty, loading the output takes longer due to fact that cache needs to be repopulated (see below the test results). Therefore, we are preparing the script that will be used to update the cache table in the background. The CacheManager will call this script instead of clearing the cache. In the meantime, the entries are saved back into the cache when they are loaded into the output (see diagram in Fig. 4).

## 5.2 Benefits of the Cache Module

We have developed the cache primarily to improve performance of the output module. We have performed several tests to measure the performance increase with cache enabled. We present results of two series of tests. In the first test, we have compared time to load lemma *dat* (i.e. to give) which is very complex unit with lot of meanings, nested units, and cross references. Without cache, it took 3.0 s to load the lemma

(combined electronic and printed output). With cache enabled and cleared, the time slightly increased to 3.2 s due to fact that data needed to be written to the cache. However, after reload of the output, time dropped significantly to 0.2 s. Number of database queries needed to load the page decreased from 14033 to only 261.

In the second test series, we have measured time to output all 92 lemmas starting with letters *ban**. Without cache, it took 5.3 s to generate the output. With cache cleared and enabled, the time increased to 18.5 s again due to fact that data needed to be written to the cache. Moreover, it seems that frequent change of read and write operation reduces usage of the internal cache of the MySQL server which consequently increases the load time. The performance decrease is unacceptable, therefore we are developing script that will periodically fill the cache in the background in order to reduce this performance overhead.

On the other hand, when the cache was filled, the time dropped to 0.6 s. This time, number of queries required to load the page decreased from 25487 to 1574. All times in the tests are taken as average of three measurements. Test results are summarized in Table 1.

**Table 1.** Test results (average of three measurements)

| Test | Lemma *dát* | | Lemmas *ban** | |
|---|---|---|---|---|
| Cache | Time to load | Queries | Time to load | Queries |
| Disabled | 2.9 s | 14033 | 5.3 s | 25487 |
| Enabled and cleared | 3.2 s | 14039 | 18.5 s | 25931 |
| Enabled and filled | 0.2 s | 261 | 0.6 s | 1574 |

### 5.3 Other Use Cases for the Cache

Although, we have originally implemented cache only to increase the performance of the output, it is now used also in the other parts of dictionary.

During development of the web interface for the public, it appeared necessary to implement the fulltext search over the dictionary. That would require designing complex queries over multiple tables with variants, meaning definitions, or exemplifications. However, as all the required information is joined together in the cache entries, we have decided to use it as a data source for the search. We only needed to provide new column to the cache table with dictionary data without formatting tags. We also need to guarantee that cache stays updated, which will be implemented by script in PHP language that will be started in the background each time a new version of the dictionary software is deployed.

Finally, new native applications for mobile devices (i.e. iOS and Android platforms) is currently being prepared. Idea is to allow these applications to access the dictionary data in the offline mode (i.e. without connection to the Internet). Therefore, we have decided to use the cache as a source for offline data. To decrease network traffic, we are now developing synchronization tools that will allow to download only items updated since the last synchronization; this is possible because time stamp of the

last update is saved with each cache entry. This use case also requires that cache contains reasonably up to date data.

As the mobile applications use the same set of formatting rules as the public web interface of the dictionary, it is not necessary to keep another separate version of cache. However a different CSS files will be used on mobile platforms due to smaller screen size. Each microstructure element is enclosed with HTML tag associated with appropriate CSS class, therefore it is very easy to quickly change appearance of the dictionary entries.

Furthermore, we plan to implement HTML compression into the cache. The compressed data will be used during synchronization on mobile devices. This will reduce network traffic which will consequently decrease time required to synchronize the cache. However, we need to uncompress and stored the data in the cache when the synchronization is completed as it is used as a source for the fulltext search.

## 6 Conclusion and Outlook

Although Alexis is still under development, it is already used by lexicographers on daily basis. Approximately 20 000 lexical units have already been processed. With increasing number of items in the database, need to optimize speed of output module has appeared.

We have proposed and implemented cache. Tests have proved that if the cache is filled, the time required to generate the output can be reduced by the factor of ten. On the other hand, if the cache is empty, the process of loading output is slowed by generation and saving back into the cache. Therefore, we need to implement tool that runs in the background and keeps cache up to date. Completing the integration of the central data object into editing, editorial, and output modules will allow us to simplify the process of cache update.

Besides the performance improvement in the output module, cache has already been used as a source of offline data for the mobile devices and as a base for fulltext search in the web interface for public. To reduce network traffic, we plan to compress cache data.

**Acknowledgment.** This work has been supported by the grant project of the National and Cultural Identity (NAKI) applied research and development program A New Path to a Modern Monolingual Dictionary of Contemporary Czech (DF13P01OVV011).

## References

1. Barbierik, K., et al.: A new path to a modern monolingual dictionary of contemporary Czech: the structure of data in the new dictionary writing system. In: Proceedings of the 7th International Conference Slovko, pp, 9–26. Slovenska akademia vied, Jazykovedny ustav Ludovita Stura, 13–15 November 2013

2. Barbierik, K., et al.: Development of dictionary writing software. In: Proceedings of the 39th Conference Software Development, 8th November 2013. College of Polytechnics, Jihlava (2013)
3. Barbierik, K., et al.: The dictionary-making process. In: Proceedings of the 16th International Conference EURALEX, 15–19 July 2014, pp. 125–135. Institute for Specialised Communication and Multilingualism at the European Academy of Bolzano/Bozen (EURAC) (2014)
4. Kochov, P., et al.: At the beginning of a compilation of a new monolingual dictionary of Czech (A Report on a New Lexicographic Project). In: Proceedings of the 16th International Conference EURALEX, 15–19 September 2014, pp. 1145– 1151. Institute for Specialised Communication and Multilingualism at the European Academy of Bolzano/Bozen (EURAC) (2014)
5. Barbierik, K., et al.: Versioning module for dictionary writing system. In: Proceedings of the International Conference on Communication Systems and Computing Application Science, Jeju Island, South Korea, 16–17 May 2015 (2014)
6. Driessen, V.: A successful git branching model (2010). nvie.com/posts/a-successful-git-branching-model/. Accessed 30 Sep 2015
7. DEB II: Dictionary Editor and Browser. http://deb.fi.muni.cz/index.php. Accessed 30 Sep 2015
8. Github. https://github.com/. Accessed 30 Sep 2015
9. https://www.mantisbt.org/. Accessed 30 Aug 2014

# Control Process Management by Means of Evolutionary Algorithm

Roman Kielec[1] and Michał Doligalski[2(✉)]

[1] Science and Technology Park, Faculty of Mechanical Engineering, University of Zielona Góra, Zielona Gora, Poland
r.kielec@pnt.uz.zgora.pl
[2] Institute of Metrology, Electronics and Computer Science, University of Zielona Góra, Zielona Gora, Poland
m.doligalski@imei.uz.zgora.pl

**Abstract.** The control process can be understand as the specification of the manufacturing process in factory or specification of the control program for logic controller. The aim of the control process management can be understood quite widely, especially as the resources usage optimisation or time reduction. The paper presents the application of the evolutionary algorithm towards the loops reduction. The unnecessary repeating of the tasks can increase the time and costs of the control process. The UML formal models are popular method of the control process specification. The Proposed approach enables optimisation of the control process specified by means of the UML activity diagrams. Both, manufacturing process and logic controller behaviour can be specified as well.

## 1 Introduction

The paper presents the possibility of the evolutionary algorithm application for control process management. Formal models enables unambiguous specifications and model transformation. Unified Modelling Language (state machines, activity diagrams) or Petri Nets are popular and acceptable models for behavioural logic controller's specification [11, 13], the paper is focused in activity diagrams and state machine diagram. Matrix model are the other example of control process specification, well known and widely accepted in production and project management. The application of the evolutionary algorithm enables matrix model-based specification optimization. The optimization criteria are steered by target function. The paper presents one of the function focused on total time optimisation. The similarities of the activity diagrams and matrix model specification enables models mapping. The presented method enables more accurate and flexible logic control process management by means of activity diagrams and evolutionary algorithm application. The AD are more precise and transparent than matrix model and transformation enables the use of evolutionary algorithm. The algorithm is dedicated to matrix model specification. The application of the activity diagrams specification for control process specification, production process in particular, enables the application of the model checking algorithms in direct way [8].


J. Janech et al. (eds.), *Proceedings of the 2015 Federated Conference on Software Development and Object Technologies*, Advances in Intelligent Systems and Computing 511,
DOI 10.1007/978-3-319-46535-7_11

## 2 Behavioral Specification

The control process can be specified in different ways, UML activity diagrams and state machine diagrams in particular. The graphical form of the specification enables better understanding of the control process scope. This form of the specification is platform-independent which means that implementation method is not enforced. Such control process can be understood as control system or production process specification. The logic controller is a part of the control system and it can be implemented by means of reprogrammable logic (FPGA) by means of hardware description language (HDL) [2, 3].

The production process is a set of task, used to create an object or multiple units of the same item. The production process uses resources like coroll system to execute particular task. The application for both context the same method of the specification simply the documentation. It enables the use (or partial use) the same algorithms for analysis and optimisation.

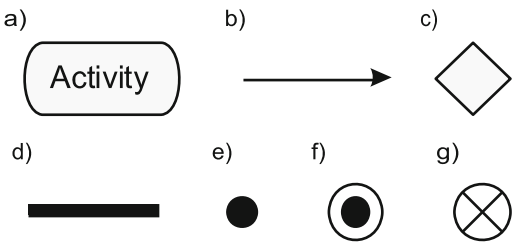



**Fig. 1.** Activity diagrams elementary nodes

## 3 Activity Diagrams

Activity diagrams are good specification method for system behaviour. The system is understood as a group of objects cooperating with one another through indication to tasks that are executed by them and communication that takes place between the objects. System behaviour is modelled with the use of activities, actions and flows that take place between them (Fig. 1). The Action node represents single work item (task), the single means that it is atomic action from the designer point of view. For large, complex system the hierarchical modelling is required. The activity diagrams model provides hierarchy by means of Activity node, the sub-diagram can be assigned to Activity node. There also signal actions: send and receive signal action. Those nodes are responsible for the messages broadcasting and processes synchronisation.

## 4 State Machine

UML state machine diagrams represent a behaviour of a specific (single) object. An object can be defined in a broad sense and considered on a number of abstraction levels: it can represent an information system (at the highest abstraction level), particular system modules or specific objects (class instances). The behaviour is presented through showing different states in which object can exist and transitions between states. Behaviour is described through the specification of states in which a object can exist and transitions between those states. Each of the states can have specified labels of activity: entry, do and exit. They mean an activity stimulated at the state activation

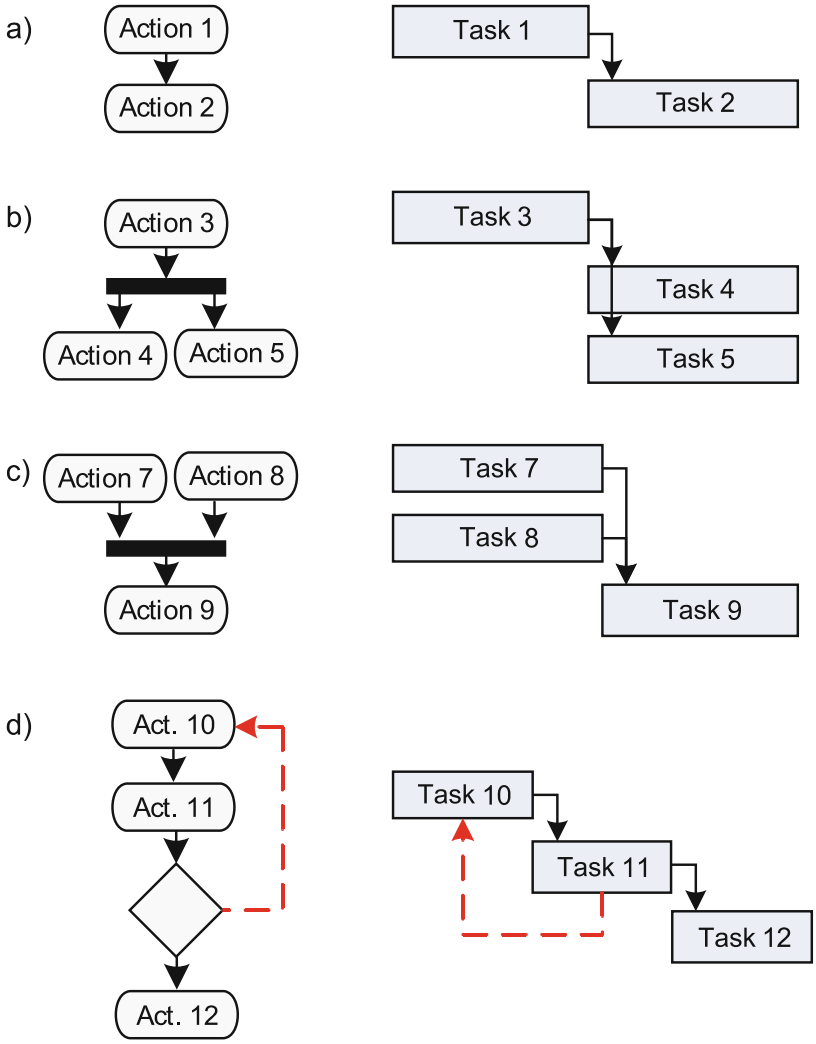



**Fig. 2.** Conversion UML activity diagram to matrix model

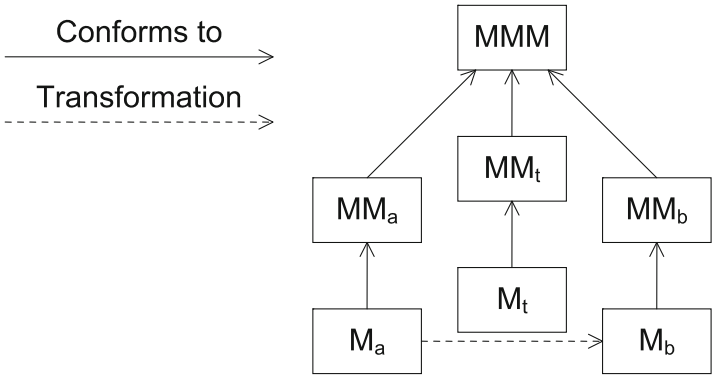



**Fig. 3.** Model transformation pattern

moment, activity performed during a state activity and an activity stimulated at the deactivation moment (state deactivation). The logic controllers can be also specified by means of activity diagrams. This is not optimal way of the specification and it's caused by the way of transitions triggering. It requires decision node for each transition condition.

## 5 Two Level Specification

The main difference with object behaviour specification by means of UML state machine and activity diagrams is transition firing. The transitions firing in activity diagrams are conditioned by activity completion, unlike in state machine diagrams where state change (the state deactivation) is conditioned by transition firing. This difference in causation of transition firing is crucial for control process specification. The two level specification is proposed. First level describes general scope of the control system. It describes the control tasks at the enterprise level by means of the activity diagrams. The control (production) process is modelled using Tasks, graphically represented by bars. The aim of the industrial process is to deliver or produce goods. The application of the decision block enables iterative development. The second abstraction level describes tasks in details. It can be described by means of state machine model. State machine diagram can be implemented in different ways. The direct logic synthesis into hardware description languages is one of the example. The application of dual models simplify partial reconfiguration process and enables the formal verification methods [1, 7].

## 6 Activity Diagrams Conversion

The principles of activity diagrams conversion to matrix model is presented in Fig. 2. The action and activity nodes are transformed into tasks, also the flows are also mapped in matrix model (Fig. 2a). The concurrent paths from activity diagrams are mapped into

concurrent task (Fig. 2b, c). The decision mode is transformed into loopback path in matrix model. Typically there is no timing constraints assigned to Activity diagrams. The time for activity or action is not determined. Matrix model charts are based on timing constrains. Each task has start and end date, the period for the each task is determined. The OCL language or UML timing enables the specification of the timing constraints for the activity diagrams.

The conversion between models can be done automatically by means of model transformation (Fig. 3). The MDA methodology specifies model transformation pattern through the description of conversion method between particular models at the level of metamodels with the use of transformation rules3. Three models: Ma, Mb, Mt are: source model, target model and transformation rule respectively. The models are concordant with their metamodels: MMa, MMb, MMt. The MMM metamodel defines MMa, MMb, MMt metamodels - it is used for those model specification. The model transformation requires metamodels for both specification method: activity diagrams and matrix model are formal models. The source model (Ma) is specified by means of metamodel MMa. The transformation engine use transformation rules to convert source model into resulting model (Mb). The automatic transformation reduce the time and possible errors of the transformation. I

The model transformation rules at the level of metamodels within the MDA methodology are described with the use of model transformation languages. Studies are conducted concerning the software engineering process support in the scope of the information system development automatization stage which is fulfilled, among other things, through model transformation. Atlas transformation language (ATL) and Query View Transformation (QVT) are the most common description formal standards of model transformation rules. Both languages are supported by a Model to Model project (M2M) 1 being the framework including model transformations in a comprehensive way. The M2M project is the component model of the Eclipse Modeling Framework (EMF) project 2 developed within n the Eclipse platform promoting modeloriented technologies. The M2M project includes three transformation engines: ATL, QVT - QVTo operational (QVTo), QVT declarative.

## 7 Evolutionary Algorithm for the Dependency Matrix Optimisation

Evolutionary algorithms (EA) are chains of signs (or symbols) relevant to the considered problem. For each particular EA application appropriate coding [4, 5] is required. The choice of the right code and setting proper algorithm parameters are required for algorithm efficiency. A good coding systems should guarantee easily coding and de-coding, at each stage of the evolution. High computational complexity in coding and decoding will result in an increase in the calculation time and reduce the efficiency of the algorithm. De-coding is important for the evaluation and selection of candidate solutions.

For each particular design process a particular code, called a chromosome is specified. In this problem, partial process tasks are genes of the chromosome. Thus, each chromosome corresponds to a variation in process realisation [6, 12].

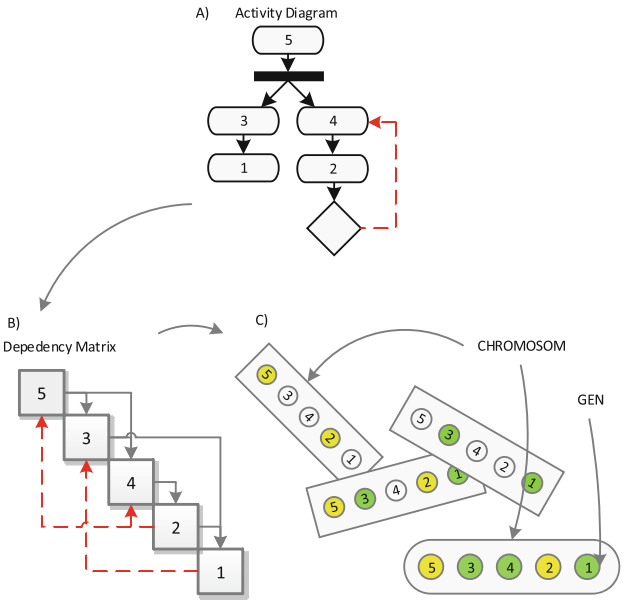



**Fig. 4.** The activity diagrams optimization process

The optimisation of the control process specified by means of activity diagrams using EA is possible as well. The activity diagram specification is transformed into dependency matrix, Fig. 4 presents an example of a dependency matrix, which contains 5 tasks. Each task corresponds to activity from formal specification. It is coded in the form of a chromosome where each task is represented by a gene. The aim of the evolutionary algorithm is to find the most suitable combination of the tasks. First, an

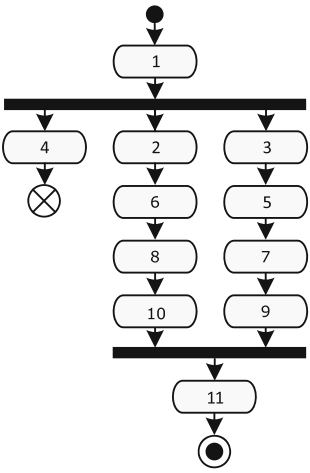
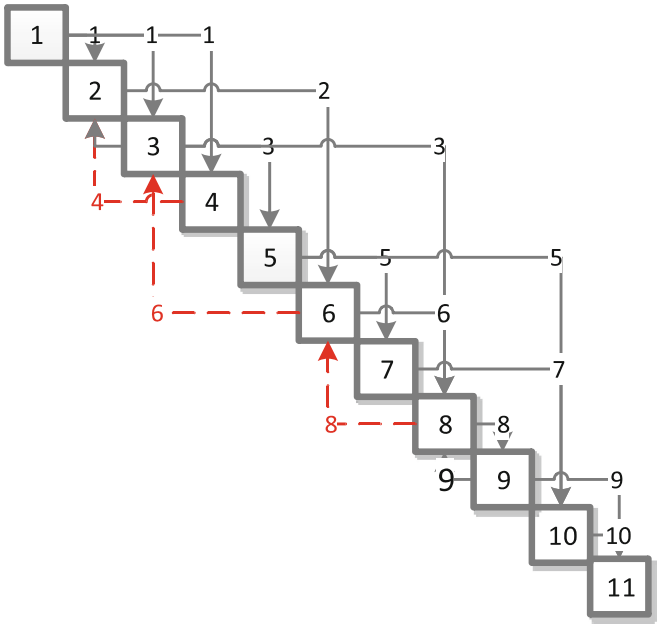



**Fig. 5.** The activity diagrams of the exemplary control process

appropriate coding system has been accepted for genetic representation of any particular realisation process. Second, operators of crossover, mutation and selection have been devised. A special fitness function that expresses the quality of the process structure in terms of its time, cost, and complexity has been formulated. Third, an algorithm for searching the best plan of the realisation process has been set up and, subsequently [6, 12].




| Id | Task name | Start | End | Time | maj 2013 |
|---|---|---|---|---|---|
| | | | | | 12 13 14 15 16 17 18 19 20 21 |
| 1 | Task 1 | 2013-05-13 | 2013-05-13 | 1d | |
| 2 | Task 2 | 2013-05-14 | 2013-05-14 | 1d | |
| 3 | Task 3 | 2013-05-14 | 2013-05-14 | 1d | |
| 4 | Task 4 | 2013-05-14 | 2013-05-14 | 1d | |
| 5 | Task 5 | 2013-05-15 | 2013-05-15 | 1d | |
| 6 | Task 6 | 2013-05-15 | 2013-05-15 | 1d | |
| 7 | Task 7 | 2013-05-16 | 2013-05-16 | 1d | |
| 8 | Task 8 | 2013-05-16 | 2013-05-16 | 1d | |
| 9 | Task 9 | 2013-05-17 | 2013-05-17 | 1d | |
| 10 | Task 10 | 2013-05-17 | 2013-05-17 | 1d | |
| 11 | Task 11 | 2013-05-20 | 2013-05-20 | 1d | |

**Fig. 6.** Matrix model and MPM

## 8 The Optimisation Example

Matrix method is based on representation of the process on the dependency matrix form. The example of that matrix is shown in Fig. 6. The control system example i specified by means of the activity diagram Fig. 5. Fork and join nodes are responsible for parallelism introduction, the control process consists of the three parallel path, one of them is terminated. The control process is rather small, the author's intention was to present overall design flow. Typical real control systems are more complex and the genetic algorithm efficiency will be bigger. The control system is transformed into dependency matric. Typically feedbacks are not included in matrix model but the proposed approach enables such form of specification. Partial tasks in the dependency matrix are indexed on the main diagonal. Connections above the diagonal represent progressive connections, whereas the ones below the diagonal represent feedback connections. Time or cost of the process are one of the possible criteria for matrix method process optimization.

The chances in the tasks realization order have the influence on the process realization [9]. In simple examples good solution can be found with the use of the method of tries and failures using human skills to represent the matrixes. In complex processes with large number of tasks analyzing all combinations without using a computer is very complicated and time consuming. Because of that a special evolutionary algorithm was created. The proposed method will not guarantee that found solution is the best one, but it is optimal solution near the best one.

## 9 Conclusion

Planning and management of the control process i crucial, especially when new products production line or controller are developed or deployed. High competition as well as product quality, cost and realization time requirements caused a radical change in engineering project management. It allows considering factors (which were earlier omitted in these projects) and integration of engineering actions with other realization operations. Nevertheless, in order to obtain such a situation, it was necessary to implement new methods for production processes planning and scheduling as well as their automating in the computer software.

The results of the EA application depends on control process complexity. The he overall product realization time, according to manufacturer's data can be shortened. Tests results shows that it is possible to save up to 46,58 time units [10]. Then presented method are focused on optimization.

The possibility of the activity diagram based specification optimization by means of evolutionary algorithm was presented. However, it has to be remembered that the analyzed process model was idealized and the possibility of direct use of the results depends on the factors not allowed for in this analysis, such as an enterprise organizational structure, human resources availability, machine/technology park, etc.

## References


1. Adamski, M., Tkacz, J.: Formal reasoning in logic design of reconfigurable controllers. In: Proceedings of 11th IFAC/IEEE International Conference on Programmable Devices and Embedded Systems PDeS 2012, Brno, Czech Republic, pp. 1–6 (2012)
2. Doligalski, M.: Behavioral specification of the logic controllers by means of the hierarchical configurable Petri nets. In: Proceedings of 11th IFAC/IEEE International Conference on Programmable Devices and Embedded Systems - PDeS 2012, Brno, Czechy, pp. 80–83 (2012)
3. Doligalski, M., Adamski, M.: UML state machine implementation in FPGA devices by means of dual model and Verilog. In: 2013 11th IEEE International Conference on Industrial Informatics (INDIN), pp. 177–184 (2013)
4. Eppinger, S.D., Joglekar, N.R., Olechowski, A., Teo, T.: Improving the systems engineering process with multilevel analysis of interactions. Artif. Intell. Eng. Des. Anal. Manuf. **28**, 323–337 (2014). http://journals.cambridge.org/article. S089006041400050X
5. Feng, W., Crawley, E.F., de Weck, O.L.: Design Structure Matrix Methods and Applications. MIT Press, Cambridge (2012). ch. BP Stakeholder Value Network, Example 5.5
6. Goldberg, D.E.: Genetic Algorithms in Search, Optimization and Machine Learning, 1st edn. Addison-Wesley Longman Publishing Co., Inc, Boston (1989)
7. Grobelna, I.: Formal Verification of Logic Controller Specification by Means of Model Checking. Lecture Notes in Control and Computer Science, vol. 24. University of Zielona Gora Press, Zielona Gora (2013)
8. Grobelna, I., Grobelny, M., Adamski, M.: Model checking of UML activity diagrams in logic controllers design. In: Zamojski, W., Mazurkiewicz, J., Sugier, J., Walkowiak, T., Kacprzyk, J. (eds.) Proceedings of the Ninth International Conference on DepCoS-RELCOMEX. AISC, vol. 286, pp. 233–242. Springer, Heidelberg (2014)
9. Kielec, R.: Planning of iterative production processes. Przegld Mechaniczny **11**, 22–26 (2009)
10. New method for engineering projects management with the use of evolutionary algorithm. In: 7th International Conference on Research and Practical Issues of Enterprise Information Systems - CONFENIS 2013. Linz, Trauner Verlag, Prague, Czechy, pp. 89–102 (2013). ISBN 9783990330814
11. Leroux, H., Andreu, D., Godary-Dejean, K.: Handling exceptions in petri net based digital architecture: from formalism to implementation on FPGAs. IEEE Trans. Ind. Inf. **99**, 1 (2015)
12. Rogers, J.L.: Demaid/ga - an enhanced design manager's aid for intelligent decomposition. AIAA Paper, pp. 96–4157 (1996)
13. Yakovlev, A., Gomes, L., Lavagno, L.: Hardware Design and Petri Nets. Kluwer, Boston (2000)

# On Parallel Versions of Jumping Finite Automata

Radim Kocman(✉) and Alexander Meduna

Department of Information Systems, Faculty of Information Technology,
Brno University of Technology, Brno, Czech Republic
{ikocman,meduna}@fit.vutbr.cz

**Abstract.** The present paper proposes a new investigation area in automata theory - *n-parallel jumping finite automata*. These automata further extend recently presented jumping finite automata that are focused on discontinuous reading. The proposed modification uses multiple reading heads that work in parallel and can discontinuously read from the input in several places at once. We also define the more restricted version of these automata which only allows jumping to the right. This restricted version is then further studied, compared with n-parallel right linear grammars, and several of its properties are derived.

## 1 Introduction

In the previous century, most formal models were designed for continuous information processing. This, however, does not often reflects the requirements of modern information methods. Therefore, there is currently active research around formal models that process information in a discontinuous way. Most notably, there are newly invented *jumping finite automata* (see [1]) that are completely focused on discontinuous reading. These automata go so far that they cannot even define some quite simple languages (e.g. $a^*b^*$) because they cannot guarantee any specific reading order between their jumps.

The present paper proposes the modification of these automata - n-*parallel jumping finite automata*. This modification presents a concept where the input is divided into several arbitrary parts and these parts are then separately processed with distinct synchronized heads. A quite similar concept was thoroughly studied in terms of formal grammars, where several nonterminals are being synchronously rewritten at once; for example, simple matrix grammars (see [2]) and n-parallel grammars (see [3–7]). However, to the best of our knowledge, no such research was done in terms of automata, where several heads synchronously read from distinct parts on the single tape. When this concept is combined with the mechanics of jumping finite automata, each part can be read discontinuously, but the overall order between parts is preserved; such automaton then can handle additional languages (e.g. $a^*b^*$). Therefore, this modification represents the combined model of discontinuous and continuous reading.

The unrestricted version of jumping finite automata handles a quite unique language family, which has not yet been sufficiently studied and which had no counterparts in grammars; until jumping grammars were introduced (see [8]). Therefore,


J. Janech et al. (eds.), *Proceedings of the 2015 Federated Conference on Software Development and Object Technologies*, Advances in Intelligent Systems and Computing 511,
DOI 10.1007/978-3-319-46535-7_12

we decided to base our initial research on the restricted version of these automata, which use only right jumps. Such restricted jumping finite automata define the same language family as classical finite automata. When these restricted automata are combined with the previously described concept, we get a model which is very similar to $n$-parallel grammars. Such automata utilize jumping only during the initialization, when heads jump to their start positions. After that, all heads read their parts of the input continuously in a left-to-right way. The paper compares these automata with $n$-parallel right linear grammars and shows that these models actually represent the same language families. Consequently, several properties of these automata are derived from the previous results.

## 2 Preliminaries

This paper assumes that the reader is familiar with the theory of automata and formal languages (see [9, 10]). Let $\mathbb{N}$ denote the set of all positive integers. For a set $Q$, *card (Q)* denotes the cardinality of $Q$. For an alphabet (finite nonempty set) $V$, $V^*$ represents the free monoid generated by V under the operation of concatenation. The unit of $V^*$ is denoted by $\varepsilon$. For $x \in V^*$, $|x|$ denotes the length of $x$, and *alph(x)* denotes the set of all symbols occurring in $x$; for instance, $alph(0010) = \{0, 1\}$. For $a \in V$, $|x|_a$ denotes the number of occurrences of $a$ in $x$. Let $x = a_1a_2 \ldots a_n$, where $a_i \in V$ for all $i = 1,\ldots, n$, for some $n \geq 0$ ($x = \varepsilon$ if and only if $n = 0$).

A *general jumping finite automaton* (see [1]), a *GJFA* for short, is a quintuple $M = (Q, \Sigma, R, s, F)$, where $Q$ is a finite set of states, $\Sigma$ is an input alphabet, $Q \cap \Sigma = \varnothing$, $R \subseteq Q \times \Sigma^* \times Q$ is finite, $s \in Q$ is the start state, and $F$ is a set of final states. Members of $R$ are referred to as rules of $M$ and instead of $(p, y, q) \in R$, we write $py \rightarrow q \in R$. A *configuration* of $M$ is any string in $\Sigma^*Q\Sigma^*$. The binary *jumping relation*, symbolically denoted by $\curvearrowright$, over $\Sigma^*Q\Sigma^*$, is defined as follows. Let $x, z, x', z' \in \Sigma^*$ such that $xz = x'z'$ and $py \rightarrow q \in R$; then, $M$ makes a jump from $xpyz$ to $x'qz'$, symbolically written as $xpyz \curvearrowright x'qz'$. In the standard manner, we extend $\curvearrowright$ to $\curvearrowright^m$, where $m \geq 0$. Let $\curvearrowright^+$ and $\curvearrowright^*$ denote the transitive closure of $\curvearrowright$ and the transitive-reflexive closure of $\curvearrowright$, respectively. The language accepted by $M$, denoted by $L(M)$, is defined as $L(M) = \{uv \mid u, v \in \Sigma^*, usv \curvearrowright^* f, f \in F\}$. We also define the special case of the jumping relation. Let $w, x, y, z \in \Sigma^*$, and $py \rightarrow q \in R$; then, M makes a right jump from $wpyxz$ to $wxqz$, written as $wpyxz \ {}_r\curvearrowright wxqz$. We extend ${}_r\curvearrowright$ to ${}_r\curvearrowright^m$, ${}_r\curvearrowright^*$, and ${}_r\curvearrowright^+$, where $m \geq 0$, by analogy with extending the corresponding notations for $\curvearrowright$. The language accepted by $M$ using only right jumps, denoted by ${}_rL(M)$, is defined as ${}_rL(M) = \{uv \mid u, v \in \Sigma^*, usv \ {}_r\curvearrowright^* f, f \in F\}$. Let $w \in \Sigma^*$. We say that $M$ accepts $w$ if and only if $w \in L(M)$. $M$ rejects $w$ if and only if $w \in \Sigma^* - L(M)$. Two GJFAs $M$ and $M$' are said to be equal if and only if $L(M) = L(M')$.

Let $n \in N$. An *n-parallel right linear grammar* (see [3, 4, 5, 6, 7]), an $n$-PRLG for short, is an $(n + 3)$-tuple $G = (N_1,\ldots, N_n, T, S, P)$, where $N_i$, $1 \leq i \leq n$, are mutually disjoint nonterminal alphabets, $T$ is a terminal alphabet, $S$ is the sentence symbol, $S$ not in $N_1 \cup \ldots \cup N_n \cup T$, and $P$ is a finite set of pairs. Members of $P$ are referred as rules of $G$ and instead of $(X, x) \in P$, we write $X \rightarrow x \in P$. Each rule in $P$ has one of the following forms: (1) $S \rightarrow X_1 \ldots X_n, X_i \in N_i, 1 \leq i \leq n$, (2) $X_i \rightarrow a_iY_i, X_i, Y_i \in N_i, a_i \in T^*, 1 \leq i \leq n$, (3) $X_i \rightarrow a_i, X_i \in N_i, a_i \in T^*, 1 \leq i \leq n$. The binary *yield operation*,

symbolically denoted by $\Rightarrow$ , is defined as follows. Let $x, y \in (N_1 \cup \ldots \cup N_n \cup \{S\} \cup T)^*$ then $x \Rightarrow y$ if either $x = S$ and $S \to y \in P$ or $x = a_1X_1 \ldots a_nX_n$, $y = a_1x_1 \ldots a_nx_n$ and $a_i \in T^*, X_i \in N_i, X_i \to x_i \in P$, $1 \leq i \leq n$. In the standard manner, we extend $\Rightarrow$ to $\Rightarrow^m$, where $m \geq 0$. Let $\Rightarrow^+$ and $\Rightarrow^*$ denote the transitive closure of $\Rightarrow$ and the transitive-reflexive closure of $\Rightarrow$, respectively. The language generated by $G$, denoted by $L(G)$, is defined as $L(G) = \{x \mid S \Rightarrow^* x, x \in T^*\}$.

## 3 Definitions and Examples

In this section, we define the modification of jumping finite automata - $n$-parallel jumping finite automata - which read input words discontinuously with multiple synchronized heads. Consequently, we also define the more restricted version of these automata which uses only right jumps.

**Definition 1.** Let $n \in \mathbb{N}$. An $n$-parallel general jumping finite automaton, *an* $n$-PGJFA *for short, is a quintuple*

$$M = (Q, \Sigma, R, S, F),$$

*where $Q$ is a finite set of states, $\Sigma$ is an input alphabet, $Q \cap \Sigma = \emptyset$, $R \subseteq Q \times \Sigma^*$ $Q$ is finite, $S \subseteq Q^n$ is a set of start state strings, and $F$ is a set of final states. Members of $R$ are referred to as rules of $M$ and instead of $(p, y, q) \in R$, we write $py \to q \in R$.*

*A configuration of $M$ is any string in $\Sigma^*Q\Sigma^*$. Let $X$ denote the set of all configurations over $M$. The binary jumping relation, symbolically denoted by $\curvearrowright$, over $X$, is defined as follows. Let $x, z, x', z' \in \Sigma^*$ such that $xz = x'z'$ and $py \to q \in R$; then, $M$ makes a jump from $xpyz$ to $x'qz'$, symbolically written as*

$$xpyz \curvearrowright x'qz' .$$

*Let \$ be a special symbol, $\$ \notin Q \cup \Sigma$. An $n$-configuration of $M$ is any string in $(X\{\$\})^n$. Let ${}_nX$ denote the set of all $n$-configurations over $M$. The binary $n$-jumping relation, symbolically denoted by ${}_n\curvearrowright$, over ${}_nX$, is defined as follows. Let $\zeta_1\$ \ldots \zeta_n\$, \vartheta_1\$ \ldots \vartheta_n\$ \in {}_nX$, so $\zeta_i, \vartheta_i \in X$, $1 \leq i \leq n$; then, $M$ makes an $n$-jump from $\zeta_1\$ \ldots \zeta_n\$$ to $\vartheta_1\$ \ldots \vartheta_n\$$, symbolically written as*

$$\zeta_1\$ \ldots \zeta_n\$ \; {}_n\curvearrowright \; \vartheta_1\$ \ldots \vartheta_n\$$$

*iff $\zeta_i \curvearrowright \vartheta_i$ for all $1 \leq i \leq n$. In the standard manner we extend ${}_n\curvearrowright$ to ${}_n\curvearrowright^m$, where $m \geq 0$. Let ${}_n\curvearrowright^+$ and ${}_n\curvearrowright^*$ denote the transitive closure of ${}_n\curvearrowright$ and transitive-reflexive closure of ${}_n\curvearrowright$, respectively. The language accepted by $M$, denoted by $L(M,n)$, is defined as $L(M,n) = \{u_1v_1 \ldots u_nv_n \mid s_1 \ldots s_n \in S, u_i, v_i \in \Sigma^*, u_1s_1v_1\$ \ldots u_ns_nv_n\$ \; {}_n\curvearrowright^* \; f_1\$ \ldots f_n\$, f_i \in F, 1 \leq i \leq n\}$. Let $w \in \Sigma^*$. We say that $M$ accepts $w$ if and only if $w \in L(M, n)$. $M$ rejects $w$ if and only if $w \in \Sigma^* - L(M,n)$.*

**Definition 2.** *Let $M = (Q, \Sigma, R, S, F)$ be an $n$-PGJFA, and let $X$ denote the set of all configurations over $M$. The binary right jumping relation, symbolically denoted by*

*${}_r\curvearrowright$, over X, is defined as follows. Let w, x, y, z $\in \Sigma^*$, and py $\rightarrow$ q $\in$ R; then, M makes a right jump from wpyxz to wxqz, symbolically written as*

$$wpyxz \; {}_r\curvearrowright \; wxqz.$$

*Let ${}_nX$ denote the set of all n-configurations over M. The binary right n-jumping relation, symbolically denoted by ${}_{n-r}\curvearrowright$, over ${}_nX$, is defined as follows. Let $\zeta_1\$ \ldots \zeta_n\$$, $\vartheta_1\$ \ldots \vartheta_n\$ \in {}_nX$, so $\zeta_i, \vartheta_i \in X$, $1 \leq i \leq n$; then, M makes a right n-jump from $\zeta_1\$ \ldots \zeta_n\$$ to $\vartheta_1\$ \ldots \vartheta_n\$$, symbolically written as*

$$\zeta_1\$ \ldots \zeta_n\$ \; {}_{n-r}\curvearrowright \; \vartheta_1\$ \ldots \vartheta_n\$$$

*iff $\zeta_i \; {}_r\curvearrowright \; \vartheta_i$ for all $1 \leq i \leq n$.*

*Extend ${}_{n-r}\curvearrowright$ to ${}_{n-r}\curvearrowright^m$, ${}_{n-r}\curvearrowright^+$, and ${}_{n-r}\curvearrowright^*$, where $m \geq 0$, by analogy with extending the corresponding notations for ${}_n\curvearrowright$. Let L(M,n−r) denote the language accepted by M using only right n-jumps.*

Next, we illustrate the previous definitions by two examples.

**Example 3.** *Consider the 2-PGJFA*

$$M = (\{s, r, p, q\}, \Sigma, R, \{sr\}, \{s, r\}),$$

*where $\Sigma = \{a, b, c, d\}$ and R consists of the rules*

$$sa \rightarrow p, \; pb \rightarrow s, \; rc \rightarrow q, \; qd \rightarrow r.$$

*Starting from sr, M has to read some a, and some b with the first head and some c, and some d with the second head, entering again the start (and also the final) states sr.*

*Therefore, the accepted language is*

$$L(M, 2) = \left\{uv | u \in \{a, b\}^*, v \in \{c, d\}^*, |u|_a = |u|_b = |v|_c = |v|_d\right\}.$$

It can be easily shown that such a language cannot be defined by any original jumping finite automaton.

**Example 4.** *Consider the 2-PGJFA*

$$M = (\{s, r, t\}, \Sigma, R, \{ss\}, \{s\}),$$

*where $\Sigma = \{a, b, c\}$ and R consists of the rules*

$$sa \rightarrow r, \; rb \rightarrow t, \; tc \rightarrow s.$$

*Starting from ss, M has to read some a, some b, and some c with both heads. If we work with unbound jumps, each head can read a, b, and c in an arbitrary order. However, if we work only with right jumps, each head must read input symbols in the*

*original order; or the automaton will eventually get stuck. Therefore, the accepted languages are*

$$\mathrm{L(M,2)} = \{\mathrm{uv}|\mathrm{u,v} \in \{\mathrm{a,\ b,\ c}\}^*, |\mathrm{u}|_\mathrm{a} = |\mathrm{u}|_\mathrm{b} = |\mathrm{u}|_\mathrm{c} = |\mathrm{v}|_\mathrm{a} = |\mathrm{v}|_\mathrm{b} = |\mathrm{v}|_\mathrm{c}\},$$
$$\mathrm{L(M,2{-}r)} = \{\mathrm{uu}|\mathrm{u} \in \{\mathrm{abc}\}^*\}.$$

*Denotation of language families*

Throughout the rest of this paper, the language families under discussion are denoted in the following way. **REG**, **CF**, and **CS** denote the families of regular languages, context-free languages, and context-sensitive languages, respectively. $_\mathrm{r}$**GJFA**, $_\mathrm{r}$*n*-**PGJFA**, and *n*-**PRLG** denote the families of languages accepted or generated by GJFAs using only right jumps, *n*-PGJFAs using only right *n*-jumps, and *n*-PRLGs, respectively.

## 4 Conversions

In this section, we prove that *n*-PGJFAs with right *n*-jumps and *n*-PRLGs define the same language families.

**Theorem 5.** *For every n*-PRLG $G = (N_1,\ldots, N_\mathrm{n}, T, S1, P)$, there is an *n*-PGJFA *using only* right *n*-jumps $M = (Q, \Sigma, R, S2, F)$, *such that* $L(M,n{-}r) = L(G)$.

*Proof* Let $G = (N_1,\ldots, N_\mathrm{n}, T, S1, P)$ be an *n*-PRLG. Without a loss of generality, assume that $f \notin N_1 \cup \ldots \cup N_\mathrm{n} \cup T$. Keep the same *n* and define the *n*-PGJFA with right *n*-jumps

$$\mathrm{M} = (\{\mathrm{f}\} \cup \mathrm{N_1} \cup \ldots \cup \mathrm{N_n}, \mathrm{T}, \mathrm{R}, \mathrm{S2}, \{\mathrm{f}\}),$$

where *R* and *S*2 are constructed in the following way:

1. For each rule in the form $S1 \rightarrow X_1\ldots X_\mathrm{n}$, $X_\mathrm{i} \in N_\mathrm{i}$, $1 \leq i \leq n$, add the start state string $X_1\ldots X_\mathrm{n}$ to *S*2.
2. For each rule in the form $X_\mathrm{i} \rightarrow a_\mathrm{i}Y\mathrm{i}$, $X_\mathrm{i}$, $Y_\mathrm{i} \in N_\mathrm{i}$, $a_\mathrm{i} \in T^*$, $1 \leq i \leq n$, add the rule $X_\mathrm{i}a_\mathrm{i} \rightarrow Y_\mathrm{i}$ to *R*.
3. For each rule in the form $X_\mathrm{i} \rightarrow a_\mathrm{i}$, $X_\mathrm{i} \in N_\mathrm{i}$, $a_\mathrm{i} \in T^*$, $1 \leq i \leq n$, add the rule $X_\mathrm{i}a_\mathrm{i} \rightarrow f$ to *R*.

The constructed *n*-PGJFA with right *n*-jumps *M* simulates the *n*-PRLG *G* in such a way that its heads read symbols in the same fashion as the nonterminals of *G* generate them.

Any sentence $w \in L(G)$ can be divided into $w = u_1\ldots u_\mathrm{n}$, where $u_\mathrm{i}$ represents the part of the sentence which can be generated from the nonterminal $X_\mathrm{i}$ of a rule $S1 \rightarrow X_1\ldots X_\mathrm{n}$, $X_\mathrm{i} \in N_\mathrm{i}$, $1 \leq i \leq n$. In the same way, *M* can start from an n-configuration $X_1u_1\$\ldots X_\mathrm{n}u_\mathrm{n}\$$, where all its heads with the states $X_\mathrm{i}$ need to read $u_\mathrm{i}$. Therefore part (1), where we convert the rules $\mathrm{S1} \rightarrow X_1\ldots X_\mathrm{n}$ into the start state strings. The selection of a start state string thus covers the first derivation step of the grammar.

Any consecutive non-ending derivation step of the grammar then rewrites all $n$ nonterminals in the sentential form with the rules $X_i \rightarrow a_i Y_i$, $X_i$, $Y_i \in N_i$, $a_i \in T^*$, $1 \leq i \leq n$. Therefore part (2), where we convert the grammar rules $X_i \rightarrow a_i Y_i$ into the automaton rules $X_i a_i \rightarrow Y_i$. The automaton $M$ always works with all its heads simultaneously and thus the equivalent effect of these steps should be obvious.

In the last derivation step of the grammar, every nonterminal is rewritten with a rule $X_i \rightarrow a_i$, $X_i \in N_i$, $a_i \in T^*$, $1 \leq i \leq n$. We can simulate the same behavior in the automaton if we end up in a final state for which there are no ongoing rules. Therefore part (3), where we convert the grammar rules $X_i \rightarrow a_i$ into the automaton rules $X_i a_i \rightarrow f$, where $f$ is the sole final state. All heads of the automaton must also simultaneously end up in the final state or the automaton will get stuck; there are no ongoing rules from f and all heads must make a move during every step.

The automaton $M$ can also start from an $n$-configuration where the input is divided into such parts that they cannot be generated from the nonterminals $X_i$ of the rules $S1 \rightarrow X_1 \ldots X_n$, $X_i \in N_i$, $1 \leq i \leq n$. However, such an attempt will eventually get the automaton stuck because the automaton simulates only derivation steps of the grammar.

**Theorem 6.** *For every n-PGJFA using only right n-jumps $M = (Q, \Sigma, R, S2, F)$, there is an n-PRLG $G = (N_1, \ldots, N_n, T, S1, P)$, such that $L(G) = L(M, n-r)$.*

*Proof.* Let $M = (Q, \Sigma, R, S2, F)$ be an $n$-PGJFA with right $n$-jumps. Keep the same $n$ and define the $n$-PRLG

$$G = (N_1, \ldots, N_n, \Sigma, S1, P),$$

where $N_1, \ldots, N_n$, and $P$ are constructed in the following way:

1. For each state $p \in Q$, add the nonterminal $p_i$ to $N_i$ for all $1 \leq i \leq n$.
2. For each start state string $p_1 \ldots p_n \in S2$, $p_i \in Q$, $1 \leq i \leq n$, add the start rule $S1 \rightarrow p_{1_1} \ldots p_{n_n}$ to $P$.
3. For each rule $py \rightarrow q$, $p$, $q \in Q$, $y \in \Sigma^*$, add the rule $p_i \rightarrow y q_i$ to $P$ for all $1 \leq i \leq n$.
4. For each state $p \in F$, add the rule $p_i \rightarrow \varepsilon$ to $P$ for all $1 \leq i \leq n$.

The constructed $n$-PRLG $G$ simulates the $n$-PGJFA with right $n$-jumps $M$ in such a way that its nonterminals generate terminals in the same fashion as the heads of $M$ read them.

The definition of $n$-PRLGs requires that $N_1, \ldots, N_n$ are mutually disjoint nonterminal alphabets. However, the states of $n$-PGJFAs do not have such a restriction. Therefore, we use a new index in each converted occurrence of a state, this creates a separate item for every nonterminal position. The index is represented by $i$ and is used in all conversion steps.

Any sentence $w \in L(M, n-r)$ can be divided into $w = u_1 \ldots u_n$, where $u_i$ represents the part of the sentence which can be accepted by the head of $M$ with a start state $p_i$ from a start $n$-configuration $p_1 u_1 \$ \ldots p_n u_n \$$, where $p_1 \ldots p_n \in S2$, $1 \leq i \leq n$. In the grammar, we can simulate the start $n$-configurations with the start rules $S1 \rightarrow p_{1_1} \ldots p_{n_n}$,

where the nonterminals $p_{i_i}$ must be able to generate $u_i$. Therefore part (2), where we convert the start state strings into the rules.

During every step of the automaton all heads simultaneously make a move. Likewise, during every non-start step of the grammar all non-terminals are simultaneously rewritten. Therefore part (3), where we convert the automaton rules $py \rightarrow q$ into the grammar rules $p_i \rightarrow yq_i$. The equivalent effect of these steps should be obvious.

The automaton can successfully end if all its heads are in the final states. We can simulate this step in the grammar if we rewrite every nonterminal with $\varepsilon$. Therefore part (4), where we create new empty rules for all final states. These rules can be used only once during the last derivation step of the grammar; otherwise, the grammar will get stuck.

**Theorem 7.** ${}_r n$-**PGJFA** $\subset$ $n$-**PRLG**.

*Proof.* This theorem directly follows from Theorem 6.

**Theorem 8.** $n$-**PRLG** $\subset$ ${}_r n$-**PGJFA**.

*Proof.* This theorem directly follows from Theorem 5.

**Corollary 9.** ${}_r n$-**PGJFA** = $n$-**PRLG**.

## 5 Characterization

**Theorem 10.** *For all* $n \in \mathbb{N}$, ***${}_r n$-PGJFA*** $\subset$ ***${}_r(n+1)$-PGJFA***.

*Proof.* This theorem directly follows from $n$-**PRLG** $\subset$ $(n+1)$-**PRLG** (see [3]).

**Theorem 11.** *For all* $n \in \mathbb{N}$, ***${}_r n$-PGJFA*** *is closed under union, finite substitution, homomorphism, reflection, and intersection with a regular set.*

*Proof.* This theorem directly follows from the same results for $n$-**PRLG** (see [3]).

**Theorem 12.** *For all* $n > 1$, ***${}_r n$-PGJFA*** *is not closed under intersection or complement.*

**Proof.** This theorem directly follows from the same results for $n$-**PRLG** (see [3]).

**Theorem 13.** ***${}_r 1$-PGJFA = ${}_r$GJFA = REG***.

**Proof.** This theorem directly follows from 1-**PRLG** = **REG** (see [3]), and from ${}_r$**GJFA** = **REG** (see [1]).

**Theorem 14.** ***${}_r 2$-PGJFA*** $\subset$ ***CF***.

**Proof.** This theorem directly follows from 2-**PRLG** $\subset$ **CF** (see [3]).

**Theorem 15.** ${}_r n$-**PGJFA** $\subset$ ***CS*** *and there exist non-context-free languages in* ${}_r n$-**PGJFA** *for all* $n > 2$.

**Proof.** This theorem directly follows from the same results for $n$-**PRLG** (see [3]).

## 6 Remarks and Conclusion

The presented results show that the concept of parallel jumping positively affects the model of jumping finite automata. The most significant result is that every additional head increases the power of these automata, which creates an infinite hierarchy of language families. Furthermore, due to the very simple conversions and the similar concepts, $n$-parallel jumping finite automata using only right $n$-jumps can be seen as a direct counterpart to $n$-parallel right linear grammars.

**Acknowledgment.** This work was supported by the European Regional Development Fund in the IT4Innovations Centre of Excellence project (CZ.1.05/1.1.00/02.0070), the TAČR grant TE01020415, and the BUT grant FIT-S-14-2299.

## References

1. Meduna, A., Zemek, P.: Jumping finite automata. Intl. J. Foundations Comput. Sci. **23**(7), 1555–1578 (2012)
2. Ibarra, O.H.: Simple matrix languages. Inf. Control **17**, 359–394 (1970)
3. Rosebrugh, R.D., Wood, D.: Restricted parallelism and right linear grammars. Utilitas Mathematica **7**, 151–186 (1975)
4. Wood, D.: n-linear simple matrix languages and n-parallel linear languages. Rev. Roum. de Math. Pures et Appl. 408–412 (1977)
5. Wood, D.: Properties of n-parallel finite state languages. Utilitas Mathematica **4**, 103–113 (1973)
6. Rosebrugh, D., Wood, D.: A characterization theorem for n-parallel right linear languages. J. Comput. Syst. Sci. **7**, 579–582 (1973)
7. Rosebrugh, D., Wood, D.: Image theorem for simple matrix languages and n-parallel languages. Math. Syst. Theory **8**(2), 150–155 (1974)
8. Meduna, A., Zbyněk, K.: Jumping grammars. Intl. J. Foundations Comput. Sci. **26**(6), 709–731 (2015)
9. Meduna, A.: Automata and Languages: Theory and Applications. Springer, London (2000)
10. Wood, D.: Theory of Computation: A Primer. Addison-Wesley, Boston (1987)

# SD2DS-Based Datastore for Large Files

Adam Krechowicz(✉), Arkadiusz Chrobot, Stanisław Deniziak, and Grzegorz Łukawski

Kielce University of Technology, Kielce, Poland
{a.krechowicz, a.chrobot, s.deniziak, g.lukawski}@tu.kielce.pl

**Abstract.** We are introducing Scalable Distributed Two-Layer Datastore. The system that is an efficient solution while storing relatively big multimedia files. In the article we are focusing on storing high-resolution photos. We are introducing some of the key implementation concepts as well as the careful evaluation. We are comparing our solution with two of the most recognizable data storing systems: MongoDB and Memcached.

## 1 Introduction

Rapidly increasing amount of information processed and stored on the Internet requires effective and reliable data storage systems. Huge data sets are associated with business applications, social networking and multimedia services. Even short periods of data outages can be expensive and may result in loss of productivity, as well as financial consequences, while permanent data loss can be catastrophic. Therefore, reliable and efficient datastores are important components of most IT infrastructures.

Nowadays, more and more applications migrate their data working sets from the conventional relational databases to NoSQL systems [1]. They provide greater efficiency and scalability by the cost of worse aggregation and consistency. Unfortunately most of the NoSQL systems are still optimised for small portions of data. However, much of the data is still stored in the most versatile format, the flat file. The growing collections of multimedia files, text documents, graphics, spreadsheets etc. are created and stored in datacenters. Storing of such a large number of files requires distributed storage systems that provide scalability and fast access. In the documentation of the Cassandra system we can find the information "The practical limit on blob size, however, is less than 1 MB, ideally even smaller" [2]. The users of HBase also discourage to use this system with large records [3]. In the most applications it is recommended to store only metadata in the NoSQL system while the actual data are stored elsewhere. Such procedure only partially resolves the problem of data storage. It complicates aggregation of the data and requires additional effort to manage the whole system.

In this paper we propose an efficient distributed datastore for storing relatively big multimedia files. The architecture of the datastore is based on Scalable Distributed Two-Layer Data Structures (SD2DS) [4] supporting simple and universal key- value access. The SD2DS enable storing data in a distributed environment and they are very efficient for storing especially medium- and big-size files. Recently [5] we showed that for files larger than 1 MB our datastore is more efficient than MongoDB, one of the


J. Janech et al. (eds.), *Proceedings of the 2015 Federated Conference on Software Development and Object Technologies*, Advances in Intelligent Systems and Computing 511,
DOI 10.1007/978-3-319-46535-7_13

most popular distributed datastores. In this paper we continue those research by comparing our solution with another well-known datastore. Memcached was chosen because its efficiency and similarity with our original conception of Scalable Distributed Two-Layer Data Structures. The rest of the paper is organized as follows. Next section reviews the related work. Section 3 presents the motivation for the paper. In Sect. 4 the architecture of our datastore is described. Section 5 contains experimental results. The paper ends with conclusions.

## 2 Related Work

Existing NoSQL solutions implement a variety of data models like: columns [6], key-value [7], documents [8] and graphs [9]. Thus, only some of them can be used to store flat files. Most of them was developed to replace conventional relational databases which becomes inefficient in a massive web applications. Unlike databases, which are universal and are capable of storing data of any type and schema, NoSQL systems are often prepared for special demands. This explains the variety and number of existing systems while the new ones are still developed and introduced to public.

MongoDB [8] is one of the most known document-based datastores. It is widely used by growing number of applications. However, this is a NoSQL system it have many features known from classical databases. It has its own query language for aggregating and creating simple schema for data. Documents are stored in BSON format which is binary version of JSON. MongoDB offers atomic access for a single document. One of the most interesting fact about MongoDB is that it can distribute data on many servers within a cluster by using sharding technique. A shard can be a standard MongoDB instance which is responsible for managing a portion of the whole data. The Config Server is a special MongoDB instance which is responsible for managing the whole set of shards. It allows for splitting the overloaded shard into two separate shards and removing the faulty shards from cluster. The MongoDB client does not have any information about the current state of shards in the cluster. It needs to communicate through the special element called Mongos which uses the information from the Config Servers and routes the requests from clients to appropriate MongoDB instance. In a situation where many clients need to use single Mongos instance it may cause serious bottlenecks. MongoDB stores the data on reliable media like disk drives by utilizing journals but it also uses cache mechanisms to speed up the access to the most recently used data. It uses as much available memory for caching, as it can obtain. So if the data is relatively small and distributed on many nodes it is possible that the whole set is assessed very fast from the main memory. Writes still needs to persist data into permanent memory like hard disks. The main purpose of the MongoDB is to store documents in BSON format. However, it also allows for efficiently storing data of any format using special API called GridFS. The data managed by GridFS are stored in two separate collections. One, which is usually called fs.files, stores the metadata of each element. The other, usually called fs.chunks, stores the actual data in small parts, typically of the size equal to 255 KiB.

Memcached [10, 11] is an open source in–memory caching system that implements Distributed Hash Table (DHT), to store data items. It is used by such companies as

YouTube or Twitter to reduce database load in their services. Memcached uses the key–value concept for data addressing. A unique identifier is associated with each stored data item. For performance reasons the maximal size of a single data item should not exceed 1 MiB. The caching system may be used in a centralized or distributed environment. In the latter case the Memcached servers are located on different nodes of a cluster and they do not communicate with each other. Only the client software knows exact location of servers and it uses the consistent hashing algorithm to assign data items to them. Such a hashing method provides a good load balance but it does not allow the system to scale. Internally, each Memcached server divides its memory into different zones called classes [12]. Each class stores data items of a particular size. If server does not have enough space for a new data item, it uses the LRU algorithm to discard one of the previously stored. The old data are deleted, thus there is no way to access them after the replacement. The performance advantage have also some drawback, thus Memcached can be only used to speed up the performance of another NoSQL system or database. This may be a good solution in caching mechanisms but is unacceptable in case of permanent data storing. Although it is not recommended to store the data larger than 1 MiB it is still possible to configure it to store data up to 20 MiB at the performance expense. The client can not wait for long for the desired data. There exists a timeout which do not allow the client to wait more than 2.5 s for the data. To authors knowledge there is no possible to change this time. There is also known issue that the connections with the clients are delayed in some cases [13].

Memcachedb is a distributed high available key–value data-store [14]. It is compatible with Memcached but unlike the latter it stores data items in a persistent manner. To provide data coherence Memcachedb supports transactions. The fault tolerance and availability is achieved by data replication.

## 3 Motivation

Motivated by our recent success of creating efficient distributed datastore [4, 5, 15] we decided to prepare unique solution for storing high-resolution multimedia files. Existing NoSQL systems are still optimized mostly for storing many small pieces of data. For example the existing works of performance tests of Memcached was developed mostly for small portions of data of hundreds of kilobytes [11] which is not acceptable for the stated assumptions.

We propose to built a distributed datastore for multimedia files, using Scalable Distributed Two-Layer Data Structures [4]. There is no single point in SD2DS that could become a bottleneck for client–server communication, even if many clients make requests simultaneously. The two layer structure allows for better memory utilization when data records of a large size are stored. Short data access time is achieved by keeping the headers buckets in RAMs and applying the distributed linear hashing algorithm for addressing them. The SD2DS–based data store could be easily modified in order to add persistence or throughput scalability.

We believe that the unique features of the SD2DS make it possible to obtain SD2DS-based datastore for multimedia files even of hundreds of megabytes. To verify this expectation we evaluate the performance of our datastore and compare the results

with the results obtained for other commonly used storage systems like MongoDB and Memcached. This allows us not only to assess the advantages of SD2DS-based datastore but also to identify shortcomings that we should address in our future works.

## 4 The SD2DD Data Store

Scalable Distributed Two–Layer Data Structures (SD2DS) were developed to efficiently store large amounts of data in a distributed RAM of a multicomputer. Although the basic idea of SD2DS was inspired by Scalable Distributed Data Structures (SDDS) [16, 17], it overcomes some of the ancestor's disadvantages by using double layered structure. The SDDS LH* is very useful and efficient for management of the first layer of SD2DS, as shown in the paper. Moreover, the basic SD2DS architecture may be expanded for improving load balancing, fault tolerance and so on [4, 15].

### 4.1 Basic SD2DS Architecture

The data stored in SD2DS consist of *components* of constant or variable size. Each component is uniquely identified by an unique key, thus SD2DS is suitable for building a *key–value* type datastore. Components stored in SD2DS are split into two parts:

- *Header*, consisting of the key and so called locator (memory pointer, URL/IP address, etc.), pointing where the second part of the component (body) is stored. Generally, the header stores the metadata of a component,
- *Body*, consisting of the key and the component data.

Headers are stored in the first layer of SD2DS, called the file, while bodies are stored in the second layer, called the storage. Both layers are managed independently and many different algorithms may be applied for the first and the second layer. Moreover, headers and bodies may be supplemented with additional information such as checksums, multiple locators, counters and so on.

A single instance of SD2DS contains at least three types of elements:

- *Buckets*, containing both, the headers and bodies. De- pending on a particular implementation, there may be separated buckets for storing headers and bodies, or a bucket may store both layers simultaneously, as presented in the paper.
- *Clients*, processing the data stored in buckets. A client may be supplemented with additional mechanism for addressing of the first layer of SD2DS, such as SDDS LH* file images. This implies that there is no need for any element that is required for addressing so there is no risk of bottlenecks or a single point of failures.
- *Split Coordinator*, responsible for managing state of both layers, crucial for ensuring scalability of the SD2DS. Depending on the implementation, there may be multiple Coordinators or none, if layers manage their state and scalability on their own.

### 4.2 Management of the First Layer

Scalable Distributed Data Structure (SDDS) is an idea of storing data in a distributed RAM of a multicomputer [16]. The data is divided into fixed-size records uniquely identified with keys. Records are stored in buckets distributed among multicomputer nodes. The whole set of buckets is called a file. There are many already developed SDDS variants using different algorithms for addressing of the data. Basic SDDS architectures use Range Partitioning (RP*) [18], Linear Hashing (LH*) [19], trees/tries [20] and more. SDDS LH* is especially useful for efficient management of data and is very useful for the management of the first layer of SD2DS, thus its details are given below.

SDDS LH* uses a hasing function for record addressing, such as a simple modulo division:

$$h(\mathrm{C}) = C \bmod 2^i \tag{1}$$

Where $C$ denotes a key and $i$ is a so called *file level*, which increases along with expansion of the file. Because there is no need that the file must consist of buckets which number is a division of two, there may be at most two consecutive levels ($i$ and $i + 1$) used simultaneously.

An empty file consists of a single bucket with a level $i = 0$. As more and more data are inserted into the file, the lonely bucket splits and a new bucket is attached to the file. After a split, a bucket increases its level $i$, just as its offspring. Thus the level of the file increases to $i = 1$. The process continues and always the leftmost bucket using the lower level $i$ is the next to be split. If all buckets increase their level $i$ to $i + 1$, the higher value ($i + 1$) replaces $i$, and the process restarts from the first bucket constituting the file (bucket number 0). As the whole process of expansion must be precisely controlled, there is another component of the file called the Split Coordinator (SC), keeping track of the next bucket to split. There is a special split pointer denoted n, holding the number of the next bucket to be split.

The SC is the only place where current values of $i$ and $n$ are stored. These values are required for determining where a record with a given key is stored within the file. As there may be many (hundreds, thousands) of clients simultaneously accessing the file, the SC must not be accessible by clients, as it would quickly become a bottleneck limiting the scalability of the file. For overcoming the problem, each client uses its own copy of these values, denoted $i'$ and $n$' and called a file image. Using its image, a client may calculate where a desired record is stored, using Algorithm 1.

**Algorithm 1.** LH* bucket addressing (client side)

```
a ← h_i' (C);
if  a < n'   then
    a ← h_i'+1 (C);
end if
```

The resulting value $a$ is the number of a bucket which is the most probable destination for record with key $C$. As the client's file image may be out of date, a client may commit an *addressing error*. In such case, the first incorrectly addressed bucket sends an Image Adjustment Message (IAM) to the client, updating his local image. This way, a client will never commit the same addressing error again.

If a bucket receives a message (query) concering a particular record, it must check whether it is the correct one to store the record in question. Every bucket has its own *bucket level* denoted $j$, which is either equal to $i$ or $i + 1$. Using the value $j$, a bucket $a$ verifies whether it is a correct recipient for key $C$, using Algorithm 2.

**Algorithm 2.** LH* bucket addressing (server side)

```
a⁰ ← hⱼ (C);
if  a′ ≠ a  then
   a′′ ← hⱼ₋₁(C);
   if  a < a′′ and a′′ < a′ then
     a′ ← a′′;
   end if
end if
```

If the resulting value $a$' is not equal to a bucket number, the bucket *forwards* the message to the bucket $a$'. After at most one additional addressing error [16], the message reaches its correct recipient.

If a bucket is overloaded, it sends *collision* message to the SC. The SC then orders the bucket number n to split, if there are available nodes for storing a new bucket. The bucket number n, using its level j, creates a new bucket, which number is computed from the following equation:

$$m = n + 2^j$$

The bucket number m is the new one to be created. Afterwards, the splitting bucket increases its level $j$ and sends all records not matching the new hashing function to the newly created bucket. After the process is finished, the splitting bucket sends confirmation message to the SC, which updates parameters of the file ($i$ and $n$) using Algorithm 3.

Figure 1 shows an evolution of sample SDDS LH* file, starting from a single bucket. The bucket capacity is set to 4 records each. Initially, the lonely bucket with its level $j = 0$ accepts all incoming records. The bucket becomes overloaded and another record with key 65 is to be inserted into the file (Fig. 1a). The overloaded bucket is split and a new one is created (Fig. 1b), moving about half of its capacity into the new bucket. Both buckets' level is set to $j = 1$. After a few more insertions, the file consists of three buckets and the bucket number 1 is overloaded (Fig. 1c). Because the split pointer denoted ($n$) points to the overloaded bucket, it splits and a new one is created

(Fig. 1d). All buckets reached level $j = 2$ and the split pointer returns to bucket number 0. The process continues.

**Algorithm 3.** Updating SC file parameters

```
n = n + 1;
if  n > 2^i   then
   n = 0; i = i + 1;
end if
```

On of the main disadvantages of SDDS LH* is that a large amounts of data are transferred between bucket during splits. The process wastes about 33 % of the total throughput of the network connecting the multicomputer nodes [16]. However, our experiments show that SDDS LH* is very efficient for management of the first layer of SD2DS, where headers are used as records. Headers are very small (e.g. tens of bytes) pieces of data, thus there is no need to transfer much data during expansion of the first layer. An example SD2DS file presented on Fig. 2 uses LH* for the first layer.

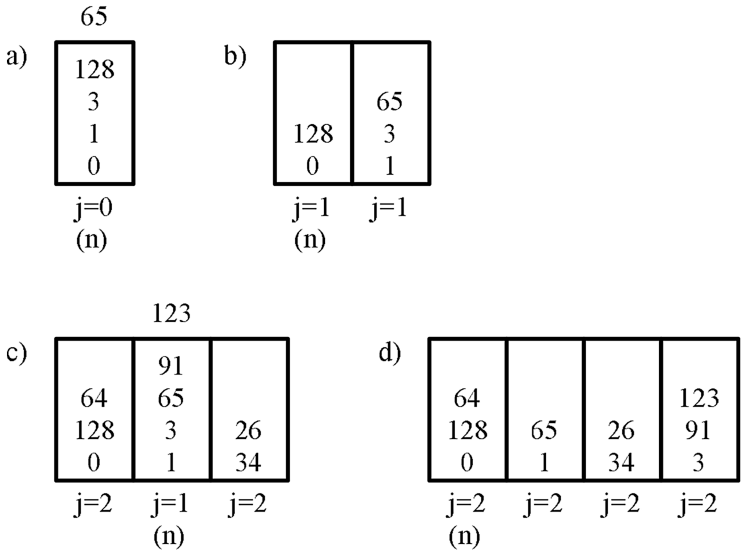



**Fig. 1.** Example SDDS LH* file evolution

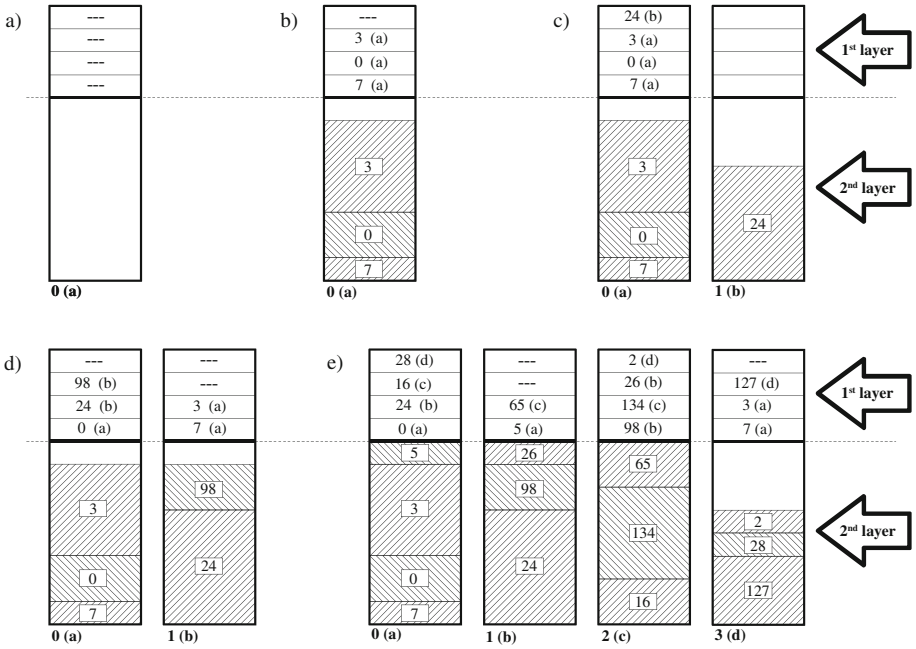



**Fig. 2.** Sample SD2DS file evolution

### 4.3 Scalability of the SD2DS

An empty SD2DS consists of one bucket. As more and more components are inserted into SD2DS, the structure expands by attaching new buckets. If SDDS scheme is applied for the first layer, it expands by performing splits [16], thus there is a need to move data between buckets.

In case of the second layer, once inserted, component bodies stay in the same bucket during the whole lifetime of SD2DS. This is a big advantage, as there is no need to transfer large amounts of data during evolution of the structure. Because both, the first and the second layer, may use the same set of servers, a whole component may be stored on a single server, what is the most optimistic case.

Figure 2 presents a sample SD2DS file, starting from single, empty bucket with address (a). All inserted components are stored in the single bucket and locators in their headers are set to (a) (Fig. 2b). As more and more components are inserted, more buckets are attached to the file (Fig. 2d–e). Headers are stored with respect to the LH* rules, while bodies are inserted into buckets having available space.

Every bucket stores so called Redirection Bucket Address (RBA), which is used to store component bodies. Initially, the RBA is set to NULL, thus all bodies are inserted locally. If a bucket overloads, its RBA is adjusted to point a bucket with available space. The SC is responsible for keeping a *pool* of buckets with available space and for adjusting RBA of every bucket. The *pool* of buckets, maintained by the SC, consists of one bucket in the simplest case. If the pool size is larger, the SC chooses a bucket from

the pool using a simple strategy (e.g. randomly or sequentially). This way there are many buckets with available space for storing bodies simultaneously, what otherwise may become a bottleneck. The size of the pool may be set to fixed size or may vary along with increasing number of buckets. The problem is especially important in case of large files (consisting of hundreds of buckets).

Two–layer design implies an indirect addressing, thus before reaching the data, a client must contact the first layer for obtaining the corresponding locator. However, our experiments showed that the need of indirect addressing doesn't affect much the total efficiency of SD2DS, especially for relatively large portions of data (starting from about 1 MiB). Moreover, as each component body stays in the same bucket as it was firstly inserted, locators may be stored and reused by clients (depending on the particular implementation of SD2DS).

### 4.4 Implementation

Our previous experiments showed that SDDS LH* is especially useful for management of the first layer of SD2DS. Moreover, both layers may be supplemented with additional elements like checksums (for fault tolerance), reference counters (for load balancing) and so on. Thus, the general idea of SD2DS can be tailored to specific needs. The datastore presented in the paper uses basic SD2DS architecture, where LH* is used for management of the first layer and a simple expansion algorithm (buckets are filled with component bodies one by another) for the second layer.

We utilize SD2DS conception to develop efficient data store mostly for storing large multimedia files like photos, videos etc. We choose C ++ language for developing the core structure of our system, mostly because it allows efficient memory management which is crucial in our project.

The original concept of SD2DS assumes that the data set is located entirely in the main memory of the cluster nodes. In the real world environment it is often unacceptable because of the risk of data loss, that may occur even after slight failure like memory bit flips. To preserve the data from unexpected loss, we decided to store data on hard disks of the cluster nodes. To preserve the high efficiency we prepared a simple caching mechanism for storing the subset of the data also in the main memory. We used basic Last Recently Used algorithm to perform this task. In our future work we plan to develop more advanced algorithms which will be more suitable for our needs.

For the performance reasons we decided to not persist the metadata from the first layer into hard disks. This allows us to achieve greater performance but seriously affect the risks of data loss. The appropriate body cannot be accessed if the header does not exist or it is corrupted. To resolve this problem we introduced feature that allows us to restore all of the header information by pooling the second layer. So in case of a first layer bucket failure it can be fully restored. Most of this information, like locators, sizes, can be read from the second layer. Others, mostly checksums, must be recalculated from the original data stored in the second layer bucket.

The header was declared as follows:

```
struct ObjectHeader{
   unsigned long key ;
   unsigned long size;
   char url[512] ;
   char checksum [DIGESTSIZE];
} ;
```

Our priority was to minimize the size of the header to reduce the costs of accessing it. Despite from the most basic metadata (like key, size and url of the locator) we added only checksum, based on the SHA algorithm, to increase the reliability of the data cached in the main memory. In case that the checksum calculated from the retrieved data and the checksum stored in the first layer are not correct we force store to acquire the data from the reliable persistent memory. We understand that this will give us only partial protection from the possible faults. We are planning to investigate the problems connected with faults deeply in our next papers.

The component body was declared as:

```
struct ObjectBody{
   unsigned int size;
   char* body ;
   unsigned long key;
};
```

The essential part of the body are the pointer to the actual data and its size. The additional key stored in the second layer is responsible for restoring process which is essential in process of recreating the headers.

## 5 Experimental Results

The performance of our system was measured in comparison with two well-known solutions: MongoDB and Mem- cached. We choose these two products because they all have similar features. First of all, they were developed in similar programming languages (C ++ and C). Furthermore, they all allows to distribute data on many nodes in the cluster by utilizing hashing or partition ranges. All three of them use the same data model. But the most important fact is that they all try to utilize main memory to achieve great performance. They all associate the portions of data with the unique key. Additionally, they all can be accessed by the Java clients which allows us to perform performance comparison tests.

In addition, the similarities of MongoDB with our solution concerns the same data organization. They both stores the metadata separately from the actual data, in different containers that can even be located on separate locations. Moreover, the architectures of MongoDB and SD2DS are very similar. Both datastores use special element for managing the expansion and shrinking of the structure (Split Coordinator in case of SD2DS and Config Server in case of MongoDB).

The original conception of the SD2DS assumes that all of the data are stored in the main memory. This makes it very similar to Memcached. Both systems are focused on allowing fast access to the data.

On the other hand, MongoDB and Memcached are very different systems with different applications. MongoDB are focused on storing the data permanently even at the expense of performance. In case of the Memcached there is an opposite approach. Managing of the cluster also differ one from another. In case of Memcached the cluster is created and managed from the client perspective. Adding or removing nodes requires changing the clients behaviour. MongoDB, however, manages the cluster from the system so it allows to scale the system transparently to the clients.

Just like in our previous work we use our own utility to perform performance tests. We are intensively developing our benchmarking utility inspired by Yahoo! Cloud Serving Benchmark (YCSB) [21]. We could not use YCSB because it is focused mostly on small pieces of data which was unaccept- able in our case. The utility was developed in Java because all three systems provide their clients for this language. We were able to integrate Memcached to our benchmark tool and expand the performance test cases.

We conduct our real world experiments on a cluster, consisting of 16 machines, on which the data were stored. We also used additional machine for running SD2DS Split Coordinator and MongoDB Config Servers. We used 10 machines for running clients. For testing MongoDB we additionally used up to 5 machines to run mongos instances which are crucial for running clients. A single node in the cluster consisted of two 8 cores processors with 8 GiB of RAM. The servers were connected using 1 GiB Ethernet.

In this paper we focused mostly on performance tests in multi-clients environment. We are aware of the fact that the web-based applications needs to process massive number of clients simultaneously. We perform our first test by retrieving data of different but relatively small sizes. The smallest data was 0.5 MiB and the biggest was 1.5 MiB. The results are presented on Fig. 3. In case of such small portions of data the best performance was attained for the Memcached system but our SD2DS drastically outperforms MongoDB even using up to 5 mongoses instances. Those results were expected because average size of the data portion is still very well supported by Memcached.

We performed our second test on real world set of 1085 photos. The existing sets of photos [22] or [23] are mostly focused on computer vision testing and have small sizes. We use high-resolution photos which sizes vary from 0.2 MiB to 9.5 MiB while the average size of one photos was 6 MiB. The results are presented on Fig. 4. In this case our SD2DS system still outperforms MongoDB. Better performance of the Memcached is the cost of not retrieving the all parts of the data due to timeouts. The number of timeouts grows with the number of clients as it can be seen of Fig. 5. Because we can not predict the size of the data that was not retrieved the speed of Memcached on Fig. 4 only concerns the fastest readings.

To estimate the real performance of the Memcached we performed the test for the portions of data of fixed size. We calculated the most optimistic average processing time by assuming that each operation that was not completed takes exactly 2.5 s (the time of timeout). The real time could be greater but we could not estimate it more. We have repeated the test four times with different sizes of the data portions. Figure 6 presents the results of the retrieving data of 1 MiB. Figure 7 presents the results of retrieving data of 2 MiB. Figures 8 and 9 presents the results of retrieving data of 5 MiB and 10 MiB

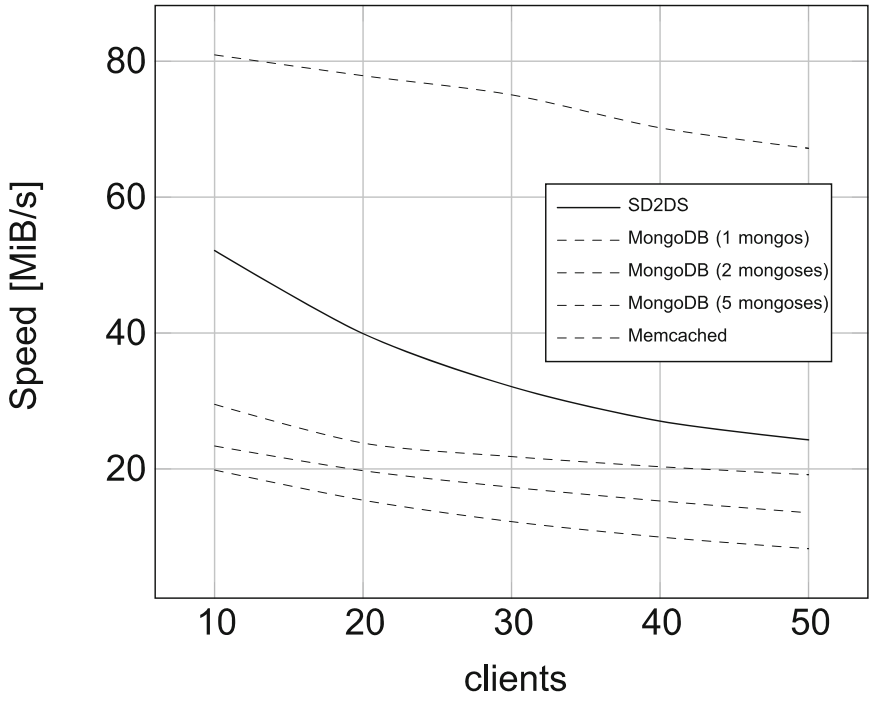



**Fig. 3.** Speed comparison of getting components of different sizes (0.5 MiB – 1.5 MiB)

respectively. In that cases even for the sizes of 1 MiB the SD2DS outperformed Memcached. The real processing for the Memcached will be slightly slower due to the timeouts. The number of timeouts in those cases are presented on Fig. 10.

As indicates from the Figs. 5 and 10 the number of timeouts are strictly related with the number of clients and the data sizes. Because of the nature of Memcached, it was developed as a caching mechanism, it does not works well in a multi- clients environments.

We also performed the tests to measure the time needed to restore the content of the first layer bucket. This very important feature determines the time that services based on our store are unavailable. We performed this test for fixed number of records of different sizes from 1 MiB to 20 MiB. We repeated those tests for different number of the second layer buckets from 2 to 16. The results we given are presented on Fig. 11. The obtained results are strictly correlated with the sizes of records. It is caused by the fact that during restoration the calculation of checksums must be performed. This operation is strictly dependent of the size of the data. The interesting fact is that the results are not correlated with the number of bucket so our solution can easily scale over different number of buckets.

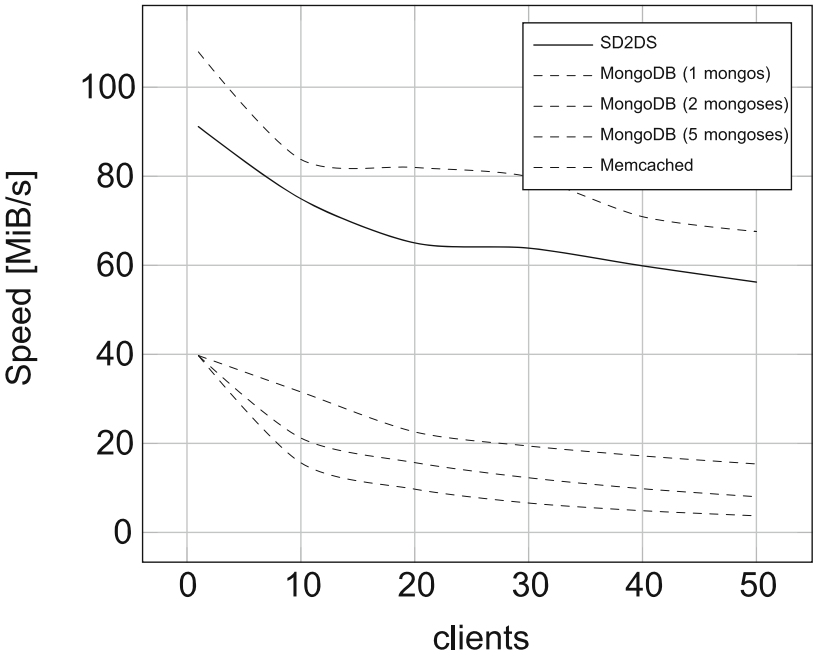



**Fig. 4.** Speed comparison of getting big set of photos

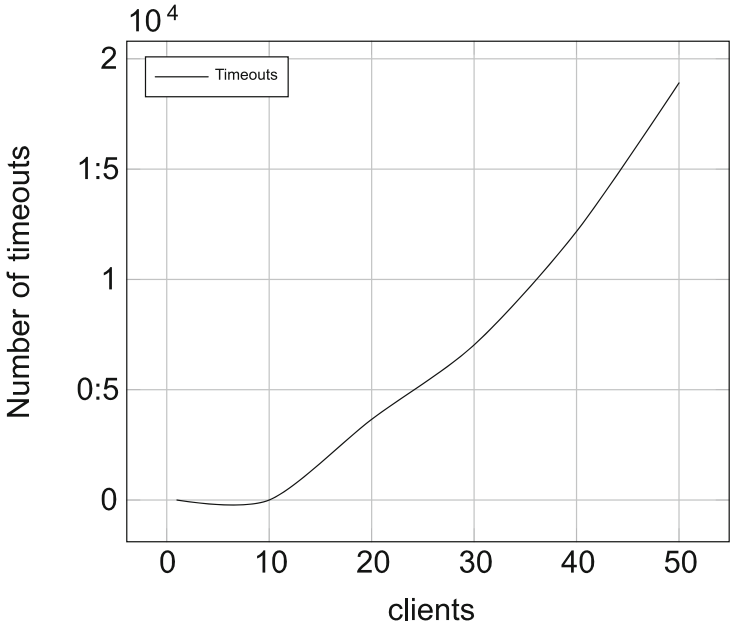



**Fig. 5.** The number of timeouts while retrieving data from Memcached

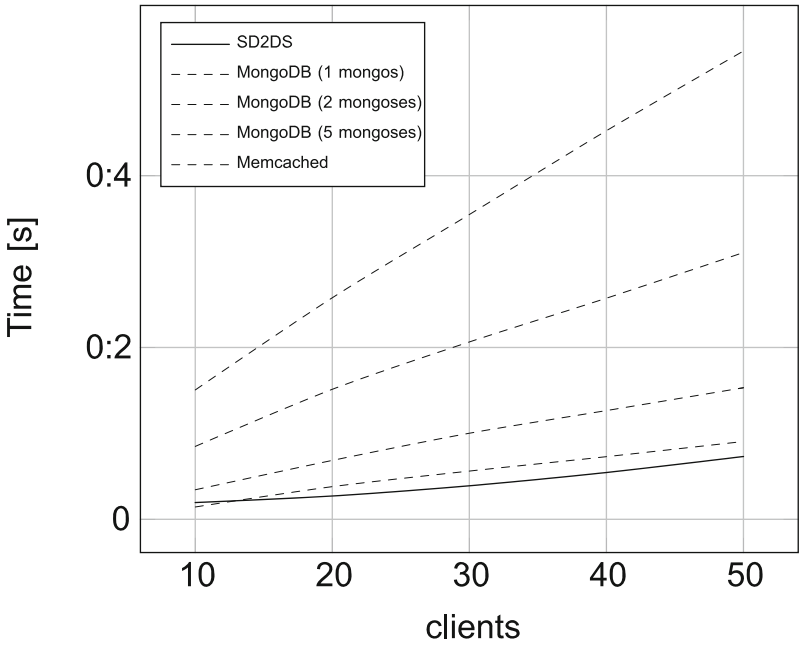



**Fig. 6.** Time comparison of getting components of fixed (1 MiB) size

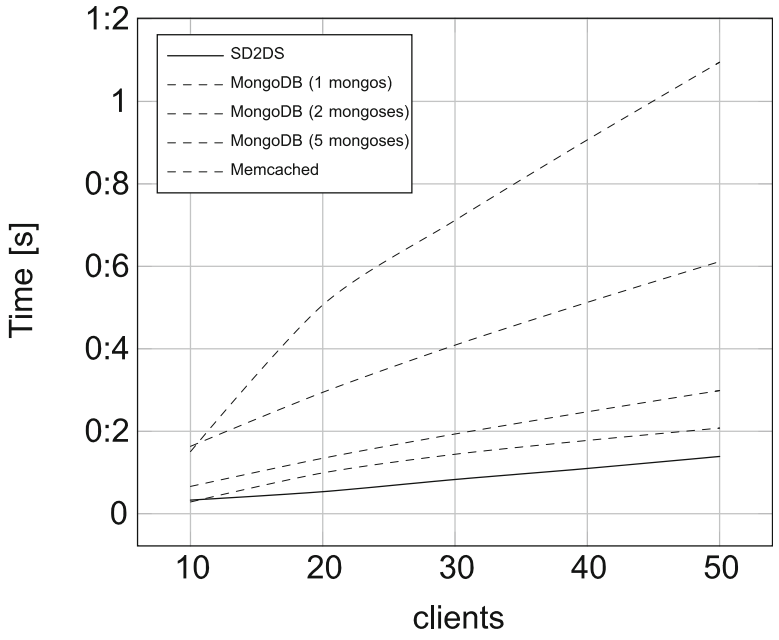



**Fig. 7.** Time comparison of getting components of fixed (2 MiB) size

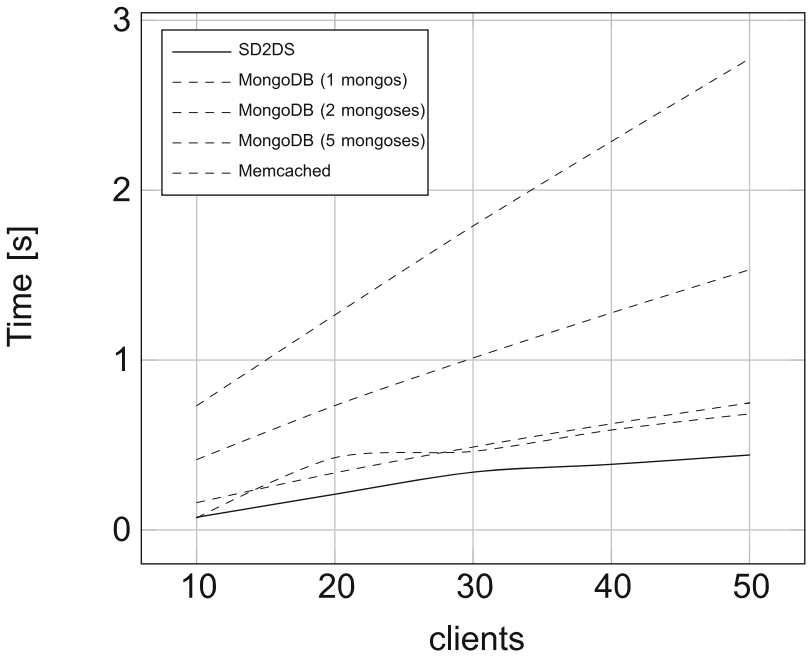



**Fig. 8.** Time comparison of getting components of fixed (5 MiB) size

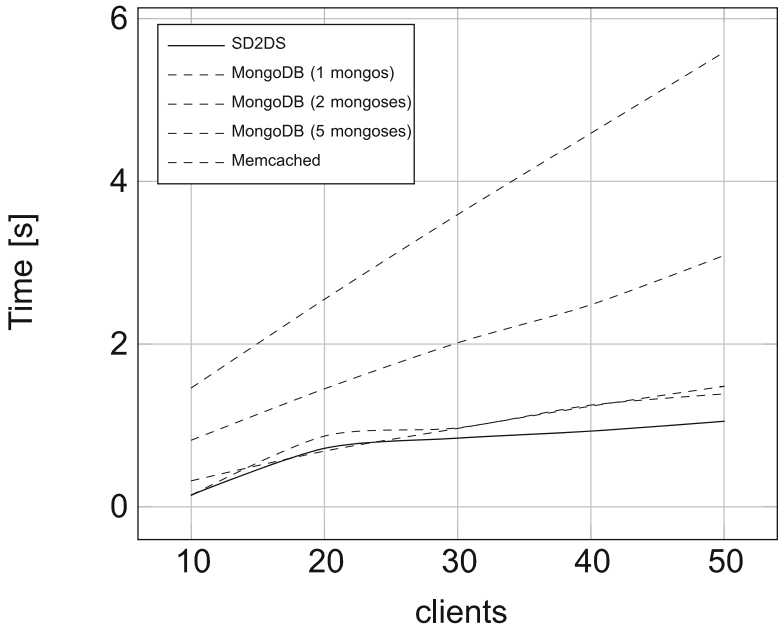



**Fig. 9.** Time comparison of getting components of fixed (10 MiB) size

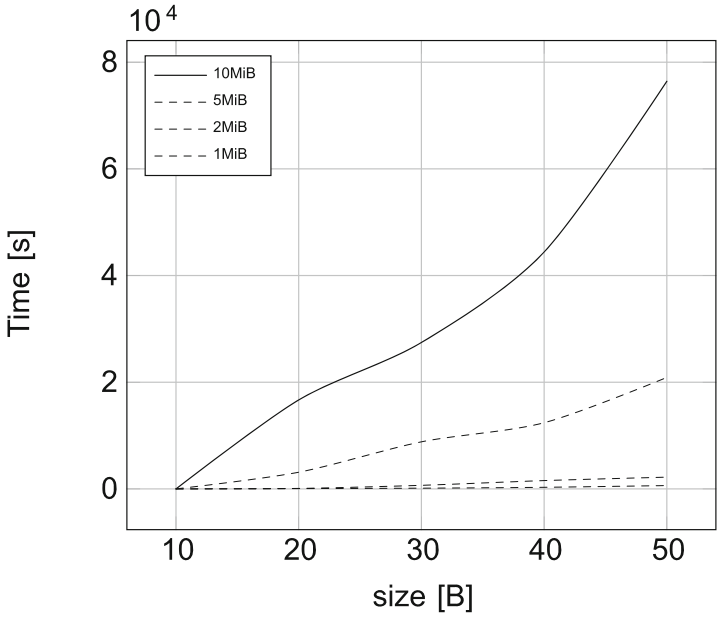



**Fig. 10.** The number of timeouts for tests with fixed data sizes in Memcached

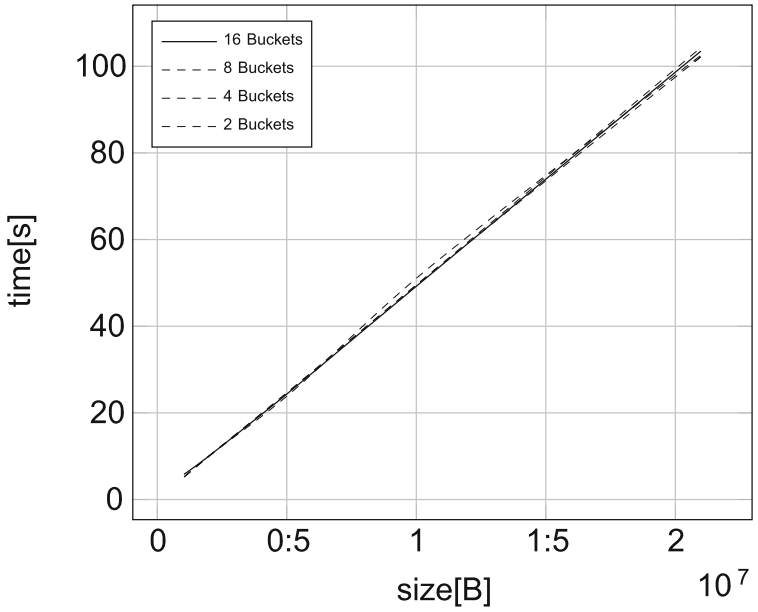



**Fig. 11.** The time of restoring first layer depending on the number of second layer buckets

## 6 Conclusions

In this paper we continue our work started in [5] of developing efficient, scalable datastore that is capable of storing relatively big multimedia files, especially high-resolution photos. In our opinion the research in this area is very crucial because the existing NoSQL solutions are mostly still not ready to store and process such large data objects.

The most important part of this paper focuses on evaluation of our datastore within the context of storing large data objects. Compared to our previous work [5] we have extended the comparison with the MongoDB on the new data sets and we performed the tests for different fixed size data. We have also introduced the comparison results with Memcached.

The results indicates that our Scalable Distributed Two- Layer Data Store is an efficient product that can seriously compete with ones of the most recognizable representatives of the NoSQL world. We achieved better performance speed both with MongoDB and Memcached in case of using big data objects (bigger than 1 MiB). We also proved that our SD2DS is more efficient than compared systems in case of the situation where the system is stressed under relatively big number of clients.

We are also aware of some of the problems concerning our product. An additional research addressing performance issues for the smaller objects (less than 1 MiB) is required. We understand that in many applications there is still a lot of small data objects that need to be processed in an efficient way. Our current effort is focused on fixing this issue. We are also planning to improve the dependability of our store. We will equip and enhance the store with fault tolerant methods [24]. Additionally, we are planning farther improvements to our data store like fast searching based on the data contents or advanced efficient and scalable processing of data objects. Developing such massive distributed system is always very complicated task at hand. We believe that despite of some minor works to do we were able to introduce solid solution. We hope that it will appeal the developers who are working on efficient web-scale applications. We believe that our storage will find application in systems that uses fast image recognition, like [25], or even in real-time cloud systems [26] where good scalability is needed. This is very important issue because our goal is to prepare the store that allows to process photos in real-time.

To support the assumption that SD2DS datastore is suitable for storing relatively big multimedia files we have developed, as a proof of a concept, a simple image browser. The application was written entirely in Java. The Fig. 12 shows a screen shot of its GUI. The left panel is a list of images stored on a hard disk or other typical data storage. If a user chooses a file from the list, the image is displayed in the middle of the window and the file is copied to SD2DS. The name of the file is also its key in the datastore. The right panel allows the user to browser the images directly from the SD2DS datastore.

The application will evolve in the direction of a public database of images, where will be possible to store private photos or to share photos with other users. We expect that in this way we obtain a system with a high degree of scalability, that exceeds the capacity of existing solutions.

## References


1. Sadalage, P.J., Fowler, M.: NoSQL distilled: a brief guide to the emerging world of polyglot persistence. Pearson Education, London (2012)
2. DataStax, "DataStax Documentation Apache CassandraTM 2.1." http://docs.datastax.com/en/cassandra/2.1/. Accessed 14 Apr 2015
3. Quora, "Is HBase appropriate for indexed blob storage in HDFS?" https://www.quora.com/Is-HBase-appropriate-for-indexed-blob-storage-in-HDFS. Accessed 7 Oct 2015
4. Sapiecha, K., Łukawski, G.: Scalable distributed two–layer data structures (SD2DS). IJDST **4**, 15–30 (2013)
5. Krechowicz, A., Deniziak, S., Bedla, M., Chrobot, A., Łukawski, G.: Scalable distributed two-layer block based datastore. In: Wyrzykowski, R., Deelman, E., Dongarra, J., Karczewski, K., Kitowski, J., Wiatr, K. (eds.) PPAM 2015. LNCS, vol. 9573, pp. 302–311. Springer, Heidelberg (2016). doi:10.1007/978-3-319-32149-3_29
6. Chang, F., Dean, J., Ghemawat, S., Hsieh, W.C., Wallach, D.A., Burrows, M., Chandra, T., Fikes, A., Gruber, R.E.: Bigtable: a distributed storage system for structured data. ACM Trans. Comput. Syst. (TOCS) **26**(2), 4 (2008)
7. DeCandia, G., Hastorun, D., Jampani, M., Kakulapati, G., Lakshman, A., Pilchin, A., Sivasubramanian, S., Vosshall, P., Vogels, W.: Dynamo: amazon's highly available key-value store. In: ACM SIGOPS Operating Systems Review, vol. 41, no. 6, pp. 205–220. ACM (2007)
8. MongoDB, "The MongoDB 3.0 Manual." http://docs.mongodb.org/manual/. Accessed 14 Apr 2015
9. Bronson, N., Amsden, Z., Cabrera, G., Chakka, P., Dimov, P., Ding, H., Ferris, J., Giardullo, A., Kulkarni, S., Li, H.C., et al.: Tao: facebook's distributed data store for the social graph. In: USENIX Annual Technical Conference, pp. 49–60 (2013)
10. Memcached, "Memcached – A Distributed Memory Object Caching System." http://memcached.org. Accessed 13 Apr 2015
11. Chidambaram, V., Ramamurthi, D.: "Performance analysis of mem- cached." http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.409.411&rep=rep1&type=pdf. Accessed 13 Apr 2015
12. Carra, D., Michiardi, P.: Memory partitioning in memcached: an experimental performance analysis. In: ICC 2014, IEEE International Conference on Communications, 10–14 June 2014, Sydney, Australia, June 2014. http://wwweurecom.fr/publication/4320
13. Memcached, "Timeouts." https://code.google.com/p/memcached/wiki/Timeouts. Accessed 5 Oct 2015
14. Chu, S.: "Memcachedb: The Complete Guide." http://memcachedb.org/memcachedb-guide-1.0.pdf. Accessed 13 Apr 2015
15. Krechowicz, A., Deniziak, S., Łukawski, G., Bedla, M.: Preserving data consistency in scalable distributed two layer data structures. Beyond Databases, Architectures and Structures. Communications in Computer and Information Science, vol. 521, pp. 126–135. Springer, Heidelberg (2015). doi:10.1007/978-3-319-18422-7_11
16. Litwin, W., Neimat, M.-A., Schneider, D.A.: LH* — a scalable, distributed data structure. ACM Trans. Database Syst. **21**(4), 480–525 (1996). http://citeseer.ist.psu.edu/litwin96lh.html
17. Litwin, W., Neimat, M.-A., Schneider, D.: RP*: a family of order preserving scalable distributed data structures. In: Proceedings of the Twentieth International Conference on Very Large Databases, Santiago, Chile, pp. 342–353 (1994). citeseer.ist.psu.edu/736278.html

18. Litwin, W., Neimat, M.-A., Schneider, D.: Rp*: a family of order preserving scalable distributed data structures. In: VLDB, vol. 94, pp. 12–15 (1994)
19. Litwin, W.: Linear hashing: a new tool for file and table addressing. In: VLDB 1980: Proceedings of the Sixth International Conference on Very Large Data Bases. VLDB Endowment, pp. 212–223 (1980)
20. Litwin, W.: Trie hashing. In: Proceedings of the 1981 ACM SIGMOD International Conference on Management of Data, pp. 19–29. ACM (1981)
21. Cooper, B.F., Silberstein, A., Tam, E., Ramakrishnan, R., Sears, R.: Benchmarking cloud serving systems with YCSB. In: Proceedings of the 1st ACM Symposium on Cloud Computing, pp. 143–154. ACM (2010)
22. Fei-Fei, R.F.L., Perona, P.: Learning generative visual models from few training examples: an incremental bayesian approach tested on 101 object categories. In: CVPR 2004, IEEE Workshop on Generative-Model Based Vision (2004)
23. Griffin, G., Holub, A., Perona, P.: "Caltech-256 Object Category Dataset," California Institute of Technology, Technical report CNS-TR-2007- 001 (2007). http://authors.library.caltech.edu/7694
24. Łukawski, G., Sapiecha, K.: Fault tolerant record placement for decentralized SDDS LH*. In: Wyrzykowski, R., Dongarra, J., Karczewski, K., Wasniewski, J. (eds.) PPAM 2007. LNCS, vol. 4967, pp. 312–320. Springer, Heidelberg (2008). doi:10.1007/978-3-540-68111-3_33
25. Janowski, L., Kozowski, P., Baran, R., Romaniak, P., Gowacz, A., Rusc, T.: Quality assessment for a visual and automatic license plate recognition. Multimedia Tools Appl. **68** (1), 23–40 (2014)
26. Deniziak, R., Bak, S., Czarnecki, R.: Synthesis of real-time cloud applications for internet of things. Turk. J. Electr. Eng. Comput. Sci. **7719**, 35–49 (2013)

# Temporal Context Manager

Michal Kvet(✉) and Karol Matiaško

Faculty of Management Science and Informatics,
University of Žilina, Žilina, Slovakia
{michal.kvet,karol.matiasko}@fri.uniza.sk

**Abstract.** Temporal database approach is one of the most significant sphere of data processing, the basic part is exact time management, changes and progress monitoring over the time. Conventional database does not support time management, historical data can be processed only partially by difficult and time consuming transformations. Object level principle uses state granularity, thus the whole state is updated, if any attribute value is changed. This approach can be inappropriate, if the frequency and granularity of the changes is not the same and even time varying. This paper deals with object level and attribute level temporal architecture. Usually, data are shared, however, not all data should be accessible to all users, and therefore concept of index definition using contexts is proposed. In terms of efficiency, indexes are restructured using pointer mapping, hybrid context trees are defined.

## 1 Introduction

Database systems are one of the most important part of the information technology. It creates the core of the data management in the information systems, offers effective access to stored data. Nowadays, the development requires management of the extensive data amount, therefore the relevance and necessity is more significant. Moreover, if the data should be processed and monitored over the time, the requirement for efficiency is more important. However, how to store data over the time? How are the requirements of temporal approach defined? What are the aspects, which form the core?

This paper deals with the conventional approach, defines principles of temporal access based on the temporal requirements. In the Sect. 10, context manager is proposed to share data with regards on defined conditions. It influences the performance, because indexed values are determined by the access layer, therefore hybrid trees based on managed context must be defined. The Experiment section compares the performance of the developed system.

## 2 Conventional Approach

Tracing the history and development of the database systems, we come to the conclusion, that they were created and highlighted especially current valid data. This approach is significantly true also today. On the other hand, temporal access and state


J. Janech et al. (eds.), *Proceedings of the 2015 Federated Conference on Software Development and Object Technologies*, Advances in Intelligent Systems and Computing 511,
DOI 10.1007/978-3-319-46535-7_14

management over the time is really important, which was even reflected during the soon development of the database management systems. During the beginning phase, time was treated using the log files and backups. However, that approach is ineffective due to processing raw material, which contains a lot of non-temporal information. Thus, the size optimization is not performed at all. Moreover, some attributes do not evolve over the time, therefore it is not necessary and even useful to store values of them multiple times [5, 16, 19].

Later, the developers realized, that this method cannot be used for large amount of data. This fact even promotes the time processing extent. The amount of the stored and processed data is still growing, frequency of the changes is also growing. Enormous power of the system requires sensor network data processing, storing data from thousands of sensors in industrial environment that produce data on the order of milli, micro and nano-seconds. Even more, future shows, that the granularity will be shortened and amount of the data will grow.

At this point, it should be emphasized, that time bordered data can be processed also in conventional (non-temporal) approach. How it works? And what are the disadvantages? First of all, it is necessary to mention the requirement of primary key. Primary key (PK) of table provides unambiguous identification of the object. Originally, if the primary key contains object identifier as one attribute or the set of attributes providing uniqueness and it does not contain time delimitation, used type of database approach is conventional regardless the other attributes containing time information, e.g. validity [23, 27]. However, nowadays, this rule is not so strict and was transformed. Important side of this issue is the fact, whether it is possible to monitor individual changes of one object over the time or not. If so, the system can be considered as temporal. However, if primary key is not temporal, but the table is covered as temporal, two unique indexes must be defined for each thus defined table. The first originates based on primary key definition, the second one provides consistency – each object can be defined at any time point by no more than one (or exactly one) valid state depending on the implementation. Moreover, fully temporal system can be called only system, which can manage also future valid data with regards on automatic change in the time of changing validity [26, 28].

## 3 Object Level Architecture

As it was partially mentioned, object level approach has been introduced based on conventional approach, it extends the definition of the primary key by the time borders. Usually, time attributes delimit the validity (uni-temporal table) or extends the definition also by the other time attributes expressing transaction validity (bi-temporal table) – Fig. 1. These attributes can characterize interval by two attributes (begin (BD) and end (ED) point of the validity. Another solution is based on only one attribute, which reflects the begin date of the validity. Thus, each newer object state delimits the duration of the validity of the previous one [3, 4, 9]. In that case, management for undefined states processing and managing must be provided. Thus, some flag expressing invalid or deleted object must be done – Fig. 2.

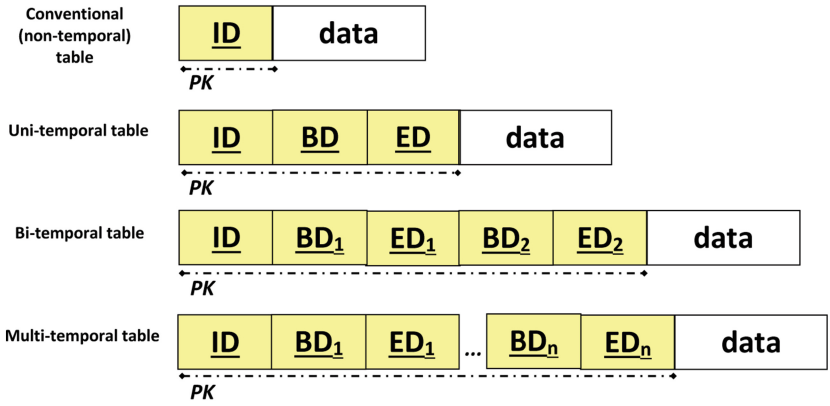



**Fig. 1.** Conventional and temporal table defined by time interval

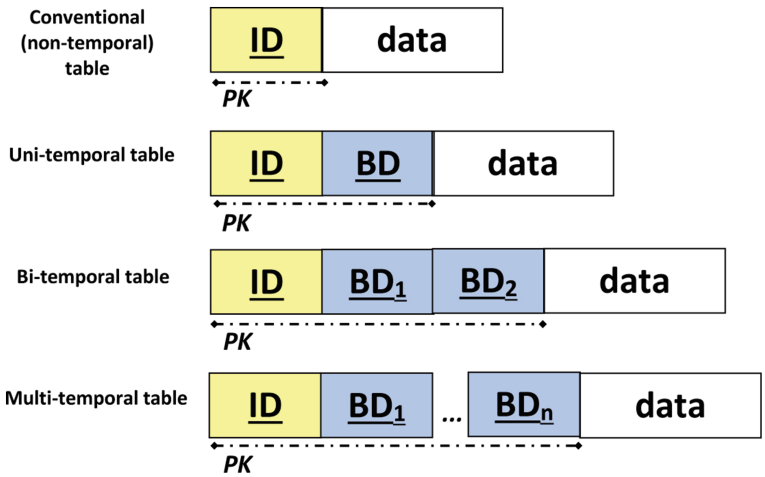



**Fig. 2.** Conventional and temporal table defined by the first timepoint

Bi-temporal approach is also important part of the temporal system definition. For the purpose of audits and global information about the changes and error correction, it is necessary to store also states, which were valid in the past, but later determined, that those data were incorrect. If incorrectly defined data were used (were considered as true (valid) during some time interval), that change cannot be performed directly only by using the *Update* statement itself without preserving original value [15]. On the other hand, transaction definition as the basic unit of the database server processing stores automatically the time of definition and execution, thus it is essential to pair the executed command to the transaction. More about transaction definition and access rules can be found in [21, 22].

Generally, we can speak about multi-temporal approach managing multiple time aspect spheres (e.g. time locality).

## 4 Temporal Definition Aspects

Object level architecture is based on the whole state of the object processing as the main part of the granularity. This approach reflects the requirements for temporal system defined in [11, 16]. Authors of his book has brought the manual for temporal design in relational databases with regards on easy data manipulation. Development aspects are based on two types. The first one is based on the usability. The aim is to provide easy manageable access to historical data in comparison to actual data. In our opinion, this rule should also highlight future valid data processing as one of the keystone of the temporal structures. The second rule is based on performance (speed of the results). Surely, this rule is based also on providing correct data, e.g. undefined states must be separated. However, how to get good performance, if the core of the system operates bad and provides insufficient efficiency and robustness? In my sooner papers, I have presented also two other aspects – requirement of data structure, which basis is to rebuild the core of the system with regards on system application area. It is therefore not possible to get good performance results, if the system in the fundamental level is not set correctly, if the data structure is not optimized for the particular application. We can get performance improvements, but minimally. Sensor network data processing is enormously important to optimize system using all levels – size and performance of the system expressed by the time and total costs. This extended aspect was also the reason for attribute level architecture development (Sect. 5).

The last criterion is based on transaction processing and collisions solving [21, 22, 24, 25].

In this paper, however, temporal access rules are shifted to sharing data environment, therefore the fifth aspect – temporal data processing requirement – is defined. Nevertheless, this sphere is partially covered by the second aspect, however, complex definition may be defined and added to the definition as the individual aspect.

Data sharing is one of the important areas of intelligent data and information technology. It is also covered by the time bordered validity, where also time interval validity can influence the accessibility. It is necessary to ensure correct data processing given by the security and access layer. Thus, each application, user of session can receive only data to which access has been granted by the authority. Again, it can be granted at the object level – the whole row is either accessible or not at all (e.g. manager can get data about the employees from his department). Granted rules can be also used on attribute level (e.g. salary). In general, combination of both rules are used. More specifically, the problem and the solution will be discussed in Sect. 10.

## 5 Attribute Oriented Granularity

Attribute oriented architecture covers the problematics of granularity and frequency of changes diversity. If the attributes are updated asynchronously, object level system would provide significant inefficiency based mostly on size of the structure. In sensor data processing (Fig. 3), the problem is sharper, thousands of input data stream from sensors are loaded, some of them provide data each micro or nano-seconds, some of

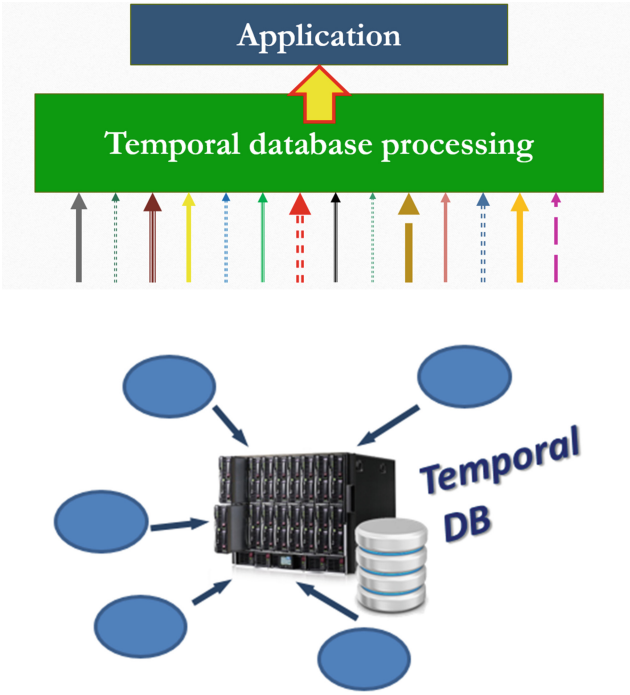



**Fig. 3.** Sensor data processing

them, however, provide these data less commonly. Input stream main factors are velocity (speed, frequency), veracity (credibility) and variety.

Attribute level approach itself is not one table, but the system consists of several tables managed using procedures, functions and triggers. In common, we can speak about three layer architecture (Fig. 4). First layer contains only actual data and is directly connected to the applications. Thus, existing applications can be used without any necessity for code modification and debugging. The core of the system provides the second layer, which stores global information about the progress, changes over the time. The third layer consists of non-actual data states – historical values, future valid data, but also data to be deleted. Actual data cannot directly query historical and future valid data (without the access to the temporal layer), because these data are descriptive and do not get relevant information without temporal definition (second layer).

Second temporal layer uses temporal table (Fig. 5) and consists of these attributes [21–23]:

- *ID_change* – got using sequence and trigger – primary key of the table.
- *ID_previous_change* – references the last change of an object identified by *ID*. This attribute can also have *NULL* value that means, the data have not been updated yet, so the data were inserted for the first time in past and are still actual.
- *ID_tab* – references the table, record of which has been processed by DML statement *(Insert, Delete, Update, Restore)*.

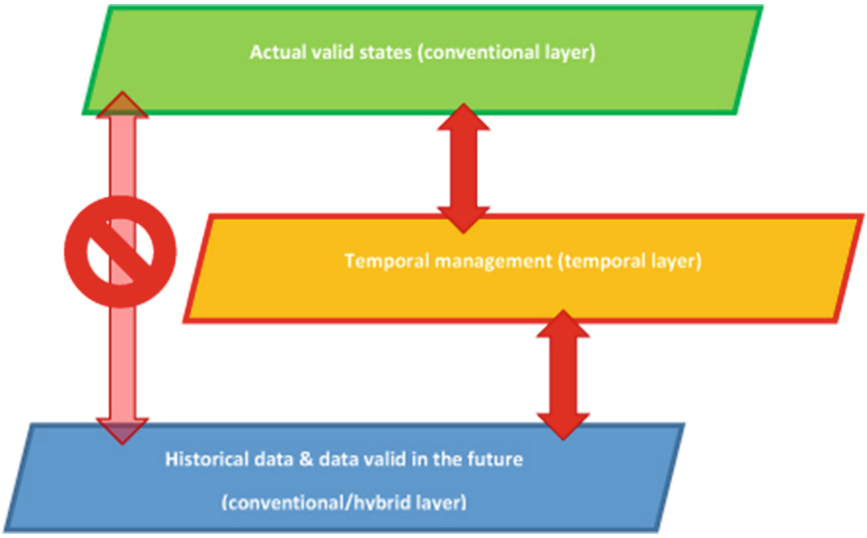



**Fig. 4.** Attribute temporal architecture

| Temporal_table | | | |
|---|---|---|---|
| id_change | Integer | NN | (PK) |
| id_previous_change | Integer | | |
| operation | operation_domain | NN | |
| id_tab | Integer | NN | |
| id_orig | Integer | NN | |
| id_column | Integer | | |
| id_row | Integer | | |
| bd | Date | NN | |
| data_type | data_type_domain | | |

**Fig. 5.** Temporal table definition

- *ID_orig* - carries the information about the identifier of the row that has been changed.
- *ID_column* – holds the information about the changed attribute (each temporal attribute has defined value for the referencing).
- *Data_type* – defines the data type of the changed attribute:
  - *C = char/varchar*
  - *N = numeric values (real, integer, ...)*
  - *D = date*
  - *T = timestamp*

This model can be also extended by the definition of the other data types like binary objects.

- *ID_row* – references to the old value of attribute (if the DML statement was *Update*). Only update statement of temporal column sets not *NULL* value.
- *Operation* – determines the provided operation:
  - *I = Insert*
  - *D = Delete*
  - *U = Update*
  - *R = Restore* (renew validity)

  The principles and usage of proposed operations are defined the in the part of this paper.
- *BD* – the begin date of the new state validity of an object.

As we can see, individual changes and progress can be easily accessible and managed using *id_change* and *id_previous_change* (Fig. 6, part. 2). If the future valid

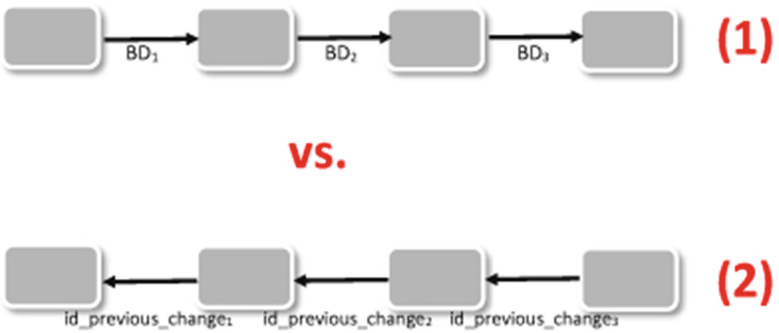



**Fig. 6.** Ordered list of changes

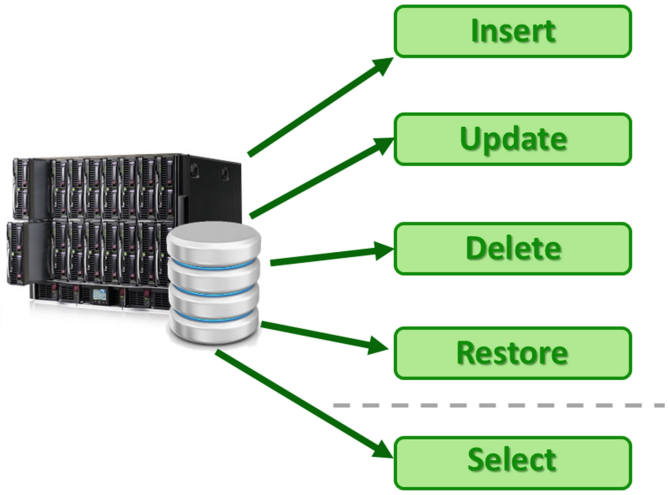



**Fig. 7.** Statement consideration in temporal system

data are not planned at all, we can sort data also by using time identifier (*BD*). In that case, time of validity is the same as time of the *Insert* statement. However, this requirement is often infeasible (Fig. 6, part 1).

Temporal system is based on several main operations expressed by the DML statements (*Insert, Update, Delete, Select* and *Restore* statement) – Fig. 7. *Insert* statement expresses adding new temporal attribute to the system, individual changes are expressed exclusively using *Update* statement. *Delete* operation can be performed, if the data should be invalidated – deleted or we cannot create consistent valid state at all. Vice versa, *Restore* operation provides renewing the validity of the object. Principles of these methods (*Insert, Update, Delete and Restore*) are described in [19]. However, the most significant factor influencing the global performance is based on *Select* statement, therefore the circumstances, conditions and evaluations must be performed very emphatically. In this step, we must distinguish two types – selecting the snapshot of the database at defined timepoint or progress monitoring of the individual object (or the set of objects).

## 6 Data Selecting

Selecting data from the structure is one of the most significant factor influencing performance of the whole structure. It is inevitable to accent the effectivity. In temporal database, time is processed as the main layer. Time frame defined in the Select statement can represent time point (time interval without the duration itself, thus the first and last moment of the interval is the same or one granule later depending on defined time interval model). Another more common situation is characterized by the time interval, which is usually modelled using closed-closed or closed-open representation. Sure, this parameter can be set on database, respectively session level. However, it can be used directly defined by the select statement to ensure required type, which would not influence the whole database or session settings.

## 7 Performance Evaluation

The importance of the individual commands and their effect on the global performance of the temporal system is not the same as well as the quantity of each command may differ. *Insert* statement, for example, only performs adding temporal column to the system, thus the impact on the system is really low (e.g. can be done at night). That was the reason to define criterion (1), which is characterized by weight of particular statements (*Insert, Update, Delete, Restore* and *Select*) (Fig. 7):

$$T_{\text{krit8}} = \sum_{i \in I} w_i * T_i \tag{1}$$

$$I = \{\text{Insert}, \text{Update}, \text{Delete}, \text{Restore}, \text{Select}\}$$

Size of the structure as the second criterion can be monitored too. However, the size of the index structure is very often the same in comparison with other index approaches.

Table 1 shows the weights of the statement types for concrete solutions, as were used. These values are set by application domain expert or by simulation. Element *A* expresses standard temporal system, element *B* reflects the system with frequent *Restore* operation (renew validity), e.g. when using not reliable communication network – wireless internet connection. Element *C* delimits the weights of the medical information system, as were used [8, 10]. Element *D* is based on intelligent transport system solution.

**Table 1.** Weights

| Statement | Notation | A | B | C | D |
|---|---|---|---|---|---|
| | | Weight value | Weight value | Weight value | Weight value |
| Insert | $W_{Insert}$ | 0,1 | 0,02 | 0,01 | 0,02 |
| Update | $W_{Update}$ | 0,3 | 0,23 | 0,3 | 0,5 |
| Delete | $W_{Delete}$ | 0,1 | 0,17 | 0,05 | 0,06 |
| Restore | $W_{Restore}$ | 0,1 | 0,18 | 0,04 | 0,1 |
| Select | $W_{Select}$ | 0,4 | 0,4 | 0,6 | 0,32 |

Principles of data management in distributed systems based on mathematical models can be found in [7, 12–14, 20].

## 8 B-Trees, B+trees

The index structure of the B+tree is mostly used because it maintains the efficiency despite frequent changes of records (*Insert, Delete, Update*). B+tree index consists of a balanced tree in which each path from the root to the leaf has the same length.

In this structure, we distinguish three types of nodes - root, internal node and leaf node. Root and internal node contains pointers $S_i$ and values $K_i$, the pointer $S_i$ refers to nodes with lower values the corresponding value ($K_i$), pointer $S_{i+1}$ references higher (or equal) values. Leaf nodes are directly connected to the file data (using pointers).

B+tree (Fig. 8) extends the concept of B-tree (Fig. 9) by chaining nodes at leaf level, which allows faster data sorting. DBS Oracle uses the model of two-way linked list, which makes it possible to sort ascending and descending, too.

The tree height (the distance from the root to leaf nodes) and also tree degree affect the number of blocks needed to search for items in the index structure, as well as the block size. For each B-tree limited by $n$ degree, these facts applies:

- Each node contains a maximum of $n$ nodes and n + 1 pointers.
- Each leaf node consists of at least (n + 1)/2 keys.
- Each internal node consists of at least (n + 1)/2 pointers.
- Root node has at least two used pointers.
- All of the leaf nodes are on the same layer.
- Each leaf consists of the key and pointer to direct data. Keys are sorted (ascending) in the leaves.

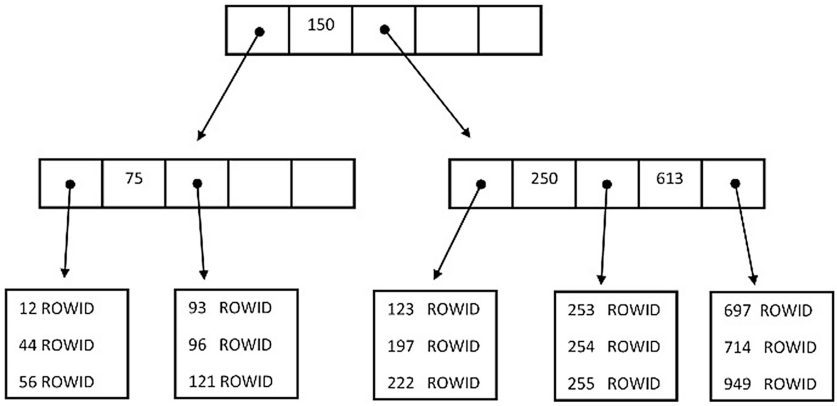



**Fig. 8.** B-tree

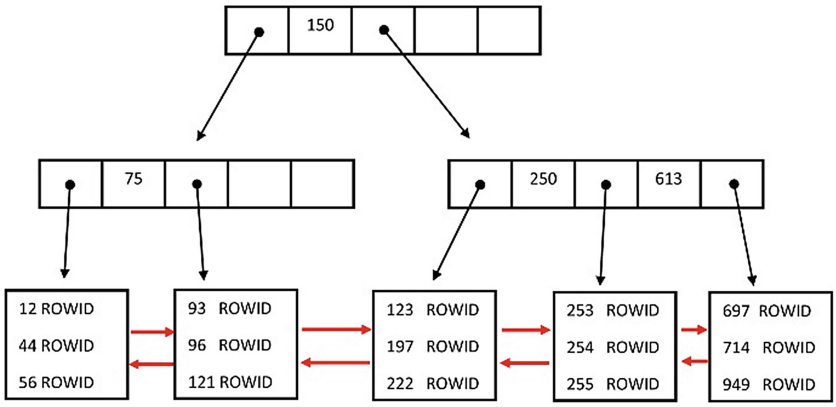



**Fig. 9.** B+tree

Limitation of this approach is a small number of records (low cardinality). In that case, using index does not produce the desired effect in terms of performance (acceleration). However, temporal data are large-scale by the definition. Special category uses inverted value for the key to prevent often tree balancing, if the indexed values are set using sequences and triggers. Moreover, the index definition is replaced by the database system manager by the *alter table* statement and cannot influence the primary key index, which is created automatically with the unique flag (Fig. 10).

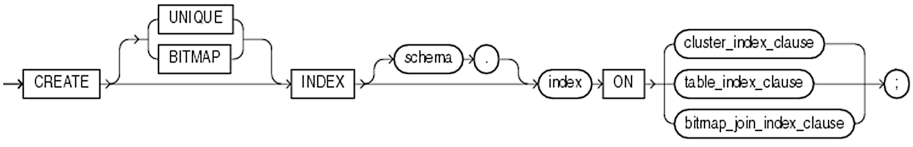



**Fig. 10.** Create index syntax [19]

## 9 Security and Access Layer

As mentioned above, variety of transactions may require different data with emphasis on their accuracy. Temporal system can, however, also store sensitive data. It is necessary to ensure their safety, so that each user has only access to the granted data. Imagine a company with different departments and employees. Normally, the chief executive has access to data about his employees. In temporal system, this requirement is even stricter and must take emphasis on the time processing represented by the time frames and timepoints. Moreover, these rules are usually changed over the time.

## 10 Context Definition

Context definition allows user to define conditions and policies on the tables. This feature has been introduced for conventional databases in Oracle 8i version and allows to hide rows, which do not meet the conditions to users [6]. Thus, one table data can be shared by various users reflecting the restrictions. In the past, the context could be defined only on the object level (e.g. using views and instead of triggers), nowadays, this approach can be used also for individual columns. This technique can be extended by time delimitation to restrict data processing not only based on standard conditions, but also based on time spectrum. Thus, using security and access layer, users can be granted or revoked privileges to access the structure. The advantage of this approach is, that user does not know about these restrictions at all – if he selects all data from the table, processing automatically ensures the use of restrictive conditions and the user receives accessible data without the need to change the statement syntax by adding new conditions in the *Where* clause. Typical examples are personal and confidential data that are accessible only to specific users (column level policy). If we want to make unavailable the entire record, object (row) level policy is used.

Let have the table consisting of three objects with the state evolution over the time (table *temp_tab*). Closed-open representation is used. For the simplicity, the concept

```
SQL> desc temp_tab
 Name  Null?    Type
 ----- -------- -------------
 ID             NUMBER(38)
 BD             DATE
 ED             DATE
 X              NUMBER(38)
 Y              NUMBER(38)
```

**Fig. 11.** Temporal table (*temp_tab*) structure

| ID | BD | ED | X | Y |
|---|---|---|---|---|
| 1 | 01-JAN-12 | 01-JAN-13 | 1 | 1 |
| 1 | 01-JAN-13 | 01-JAN-14 | 11 | 11 |
| 1 | 01-JAN-14 | 01-JAN-15 | 111 | 111 |
| 1 | 01-JAN-15 | 01-JAN-16 | 1111 | 1111 |
| 1 | 01-JAN-16 | 01-JAN-17 | 11111 | 11111 |
| 2 | 01-JAN-13 | 01-JAN-14 | 2 | 2 |
| 2 | 01-JAN-14 | 01-JAN-15 | 22 | 22 |
| 2 | 01-JAN-15 | 01-JAN-16 | 222 | 222 |
| 3 | 01-JAN-11 | 01-JAN-12 | 3 | 3 |
| 3 | 01-JAN-12 | 01-JAN-13 | 33 | 33 |

**Fig. 12.** Data stored in temporal table (*temp_tab*)

will be described for object level temporal architecture, but can be used for any temporal approach (Fig. 11). Data stored in this table are shown in Fig. 12.

User *User1* has access only to data identified by the *ID = 1* or *ID = 2*. User *User2* has access to the actual data and data valid during the last year. The user *User3* should have access only to the attribute "*x*", not the attribute "*y*" (Fig. 13). Solution can be done using views only partially, however in temporal access, the conditions and policies evolve over the time, therefore views as the root for the processing is completely inappropriate. Moreover, even the owner of the data should not have complete data access. If necessary, we can also apply this policy to the *Update* command.

New policy definition is provided using the package *DBMS_RLS*. Object schema plus object itself must be defined and statements, which are linked to the policy, are set using *statement_types* parameter (*Insert, Update, Delete, Restore, Select*). Moreover, each policy definition must have name and defined function and procedure, through which conditions are defined and evaluated.

Let have the previous example table. Policy definition requires to perform three steps. First of all, context must be created (Fig. 14). There must be also defined procedure, using which conditions are loaded into the context. Be aware, these conditions can be set only using the defined procedure (*set_temp_context*), it cannot be done outside this block (Fig. 15). The body (conditions) of the context policy can be found by querying using function *sys_context* on *dual* table. These values are evaluated and processed using function. That function is determined using the *policy_function* parameter, when defining policy - *add_policy* method.

Parameter *set_temp_context* reflects the name of the procedure, which can set the conditions of the context. In our case, conditions are stored separately in the table *Policy_tab* (Fig. 16). First of all, let´s solve the conditions for the user *User1* and *User2* (Fig. 17).

The defined procedure set_temp_context must create the conditions, which will be evaluated automatically (coded procedure principle shown in Fig. 18). For our example, these conditions must look like this (Fig. 19) for particular session settings – nls_date_format:

USER1> select * from temp_tab;

| ID | BD | ED | X | Y |
|---|---|---|---|---|
| 1 | 01-JAN-12 | 01-JAN-13 | 1 | 1 |
| 1 | 01-JAN-13 | 01-JAN-14 | 11 | 11 |
| 1 | 01-JAN-14 | 01-JAN-15 | 111 | 111 |
| 1 | 01-JAN-15 | 01-JAN-16 | 1111 | 1111 |
| 1 | 01-JAN-16 | 01-JAN-17 | 11111 | 11111 |
| 2 | 01-JAN-13 | 01-JAN-14 | 2 | 2 |
| 2 | 01-JAN-14 | 01-JAN-15 | 22 | 22 |
| 2 | 01-JAN-15 | 01-JAN-16 | 222 | 222 |

USER2> select * from temp_tab;

| ID | BD | ED | X | Y |
|---|---|---|---|---|
| 1 | 01-JAN-14 | 01-JAN-15 | 111 | 111 |
| 1 | 01-JAN-15 | 01-JAN-16 | 1111 | 1111 |
| 2 | 01-JAN-14 | 01-JAN-15 | 22 | 22 |
| 2 | 01-JAN-15 | 01-JAN-16 | 222 | 222 |

USER3> select * from temp_tab;

| ID | BD | ED | X |
|---|---|---|---|
| 1 | 01-JAN-12 | 01-JAN-13 | 1 |
| 1 | 01-JAN-13 | 01-JAN-14 | 11 |
| 1 | 01-JAN-14 | 01-JAN-15 | 111 |
| 1 | 01-JAN-15 | 01-JAN-16 | 1111 |
| 1 | 01-JAN-16 | 01-JAN-17 | 11111 |
| 2 | 01-JAN-13 | 01-JAN-14 | 2 |
| 2 | 01-JAN-14 | 01-JAN-15 | 22 |
| 2 | 01-JAN-15 | 01-JAN-16 | 222 |
| 3 | 01-JAN-11 | 01-JAN-12 | 3 |
| 3 | 01-JAN-12 | 01-JAN-13 | 33 |

**Fig. 13.** User access

```
SQL> Create context temp_context using set_temp_context;
```

**Fig. 14.** Create context

At the step, we have defined the temporal rule and conditions, however, it is necessary to register it in the system for specific table. For this purpose, we have used the functionality *add_policy* of the package *DBMS_RLS* (Fig. 20). Function for selecting the conditions have to be defined. In the body of this function, defined context using *sys_context* function is returned (Fig. 21).

As it was already mentioned, policy can be defined also for individual columns or as combination of the object and column level security layer. The principles are the same, but the monitored column list covered by the policy is defined in the *audit_column*

```
File afiedt.buf saved

  1  begin
  2  DBMS_SESSION.set_context('temp_context',
  3                           'conditions',
  4                           'xxx');
  5* end;
SQL> /
begin
*
ERROR at line 1:
ORA-01031: insufficient privileges
ORA-06512: at "SYS.DBMS_SESSION", line 101
ORA-06512: at line 2
```

**Fig. 15.** Exception – calling set_context procedure outside the defined method.

```
SQL> desc policy_tab
 Name      Null?    Type
 ----------------- -------------
 USERNAME           VARCHAR2(50)
 ID                 NUMBER(38)
 BD                 DATE
 ED                 DATE
```

**Fig. 16.** Policy_tab table structure

```
select * from policy_tab
USERNAME    ID        BD          ED
---------------------------------------
USER1        1
USER1        2
USER2            01-JAN-14   01-JAN-16
```

**Fig. 17.** Policy_tab table data

parameter of the *add_policy* function. In the parameter *audit_columns_opts* is stored the flag, whether the *Select* statement must contain all of the attributes defined in the parameter *audit_column* (flag *DBMS_FGA.all_columns*) or the policy will be used, if any of them is used in Select statement (flag *DBMS_FGA.any_columns*).

```
Create or replace procedure set_temp_context
 is
   l_str varchar2(2000);
begin
  --get the conditions into the l_str variable
  --from the policy_tab reflecting
  --the rectrictions for particular users.
  --then, the context is set
DBMS_SESSION.set_context('temp_context', 'conditions', l_ret);
end;
/
```

**Fig. 18.** The procedure to set the context

```
User1 : ID in (1,2)
User2 : BD>='01-JAN-14' AND ED<='01-JAN-16'
```

**Fig. 19.** Regular conditions

```
BEGIN
 DBMS_RLS.add_policy (
  object_schema => 'KVET1',
  object_name => 'TEMP_TAB',
  policy_name => 'TEMP_POLICY',
  function_schema => 'KVET1',
  policy_function => 'AUTHORIZED_TEMP_COND',
  statement_types => 'INSERT, UPDATE, DELETE, SELECT'
                     );
END;
/
```

**Fig. 20.** Add_policy method

```
return sys_context('temp_context', 'conditions');
```

**Fig. 21.** Get context conditions principle

Be aware for using aggregate functions with the policy because of the *NULL* values processing and influencing the global values and results.

If you do not want to control the security conditions for a specific user, use the following command (Fig. 22):

Policy is dropped automatically during the process of table dropping. However, it can be also done explicitly using the method *drop_policy* of the same package *DBMS_RLS* with three parameters – owner of the policy, accessed table and name of the policy.

Reliability and importance analysis of multi-state systems are described in [17–20].

```
GRANT EXEMPT ACCESS POLICY TO KVET;
```

**Fig. 22.** Not to use policy definition statement [1, 2]

## 11 Hybrid Context Trees

The solution proposed in the previous section can manage and share data reflecting the access restrictions. Temporal database approach is, however, based on large data amount of data stored over the time with regards on undefined states and future valid data. Adding new conditions (although invisible for the particular user) can cause the performance degradation because of the indexes. System can evaluate not to use index at all and prefers full table scan. Extension of the system by more and more index structures is not the right solution. Forgetting the fact that new index can even improve the system performance in the terms of *Select* statements, on the other hand, *Update* operation can be (rapidly) performance decreased, although it consists also of the core part of the temporal system. As we can see in the Experiment section, B+trees provides significant improvement in comparison with the solution without explicit definition of the index structures mostly because of the effectiveness of structure reconstruction after *Update* operation. It consists of two elements – nodes and the pointers. Pointers, however, can be shifted to reflect the policy. Thus, it is the basis of the development of hybrid context tree.

The nodes and links in the index structure can be trimmed to improve the traversing method in that structure to reduce the amount of processed data, to reduce the tree height. Therefore, process of tree branches and nodes trimming is proposed. Using the policy, we can create links across the main index structure for each table and each user defined in some policy. Moreover, it can be done dynamically, as shown in the previous section, where the table containing policy rules are defined and in case of change, restrictions must be reconstructed automatically. Therefore, we can define triggers and procedures providing us the pointer mapping approach based on defined policy context. Main index tree in the table owner schema is reduced using pointer map. Figure 23 shows the principle of pointer mapping, simply, some nodes, which do not cover the conditions, are skipped processing. It can seem to be waste work, however, take care of the amount of the processed data in the temporal, these conditions can really significantly reduce the set of fundamental data, which should be processed. Therefore, this approach is really significant. If the node cannot be processed by the particular user, using the pointer map, new pointer from the direct ancestor to the child (or even hierarchically below) is created (Fig. 24 – example based on Fig. 23). Thus, there is only one root index structure and individual policies build the improving transitions. Because of the balance of the B+tree, new hybrid context approach cannot degrade to the linear linked list in the pure meaning. Some nodes can be trimmed, but the height of the original index ensures the efficiency – it is significant better solution in comparison with new index definition for each user. In case of policy changes (which can be performed often), only mapping layer is updated, not the entire index.

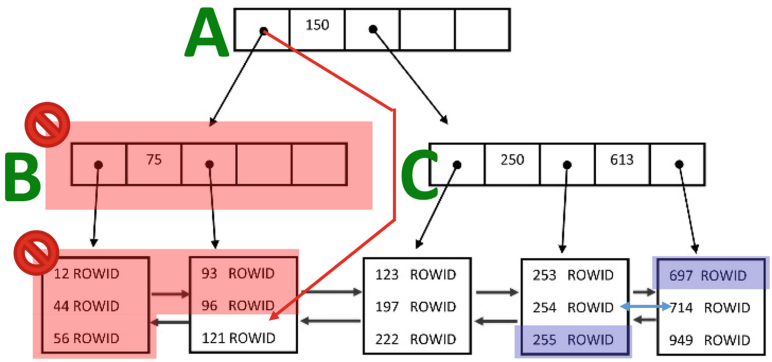



**Fig. 23.** Index structure trimming

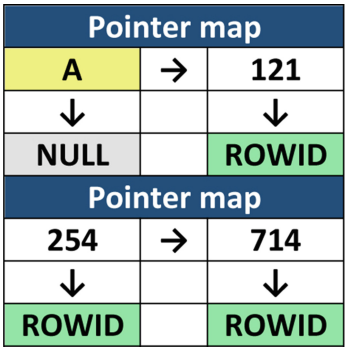



**Fig. 24.** Pointer map

## 12 Conditional Select Statements

Database system management allows you to extend the definition of the conditions in the *Where* clause of the *Select* statement. We distinguish tree values in the logic, that can be grouped and compared using negation (*NOT*) – Fig. 25, disjunction (*OR*) – Fig. 26 and conjunction (*AND*) – Fig. 27. In this case, we should avoid using *NULL* values, which are mostly evaluated as *NULL*. Although the processing will raise the *ELSE* clause of the condition, it cannot be directly evaluated as *FALSE*. Figures 25, 26 and 27 show individual operations. Symbol "*T*" delimits the *TRUE* value, "*F*" expresses incorrect value – *FALSE* and "*N*" expresses undefined value – *NULL* notation.

| Predicate (P) | NOT P |
|---|---|
| T | F |
| F | T |
| N | N |

**Fig. 25.** Negation [23]

| OR | T | N | F |
|---|---|---|---|
| T | T | T | T |
| N | T | N | N |
| F | T | N | F |

**Fig. 26.** Logical disjunction [23]

| AND | T | N | F |
|---|---|---|---|
| T | T | N | F |
| N | N | N | F |
| F | F | F | F |

**Fig. 27.** Logical conjunction [23]

In this context, we can therefore define conditional processing for evaluating conditions by using them or not, or by adding new ones. Unlike the definition of the context, the following conditions must be defined explicitly. The statement definition would be therefore more complex and conditional processing can be very complicated for manipulation.

```
select *
 from KVET.Temp_tab
  where ( (id = 1 or id = 2)
            or
           case
            when user='USER1' then 0
             else 1
           end = 1
        )
       and
        ( (BD>=trunc(add_months(sysdate, -12)))
            and
          (ED<=trunc(add_months,sysdate, 12)))
                          or
                 case
                  when user='USER1' then 0
                  else 1
                 end = 1
          )
```

**Fig. 28.** Conditional *Select* statement

Let have a table containing temporal data (previous example, for simplicity of example, the model is based on object level granularity). We want to have identical statement, but the processing and results should be influenced by the caller user based on conditions. Data are shown in Fig. 12 Once again, user *User1* can access data of the objects identified by *ID = 1* and *ID = 2*, user *User2* processes data of the actual and last year. Thus, we need to handle two different scenarios, but in the one statement. For this purpose, we need to have two conditions delimiting the identifiers, respectively the time to be processed. However, for conditional *Where* clause processing (only one condition is used) - it is necessary to determine the user, who launched that command. It is performed using other two extended conditions (Fig. 28 – yellow (second box) and blue signature (forth box)).

## 13 Experiments

Experiment results were provided using Oracle Database 11g Enterprise Edition Release 11.2.0.1.0 – 64 bit Production; PL/SQL Release 11.2.0.1.0 – Production. Parameters of used computer are:

- Processor: Intel Xeon E5620; 2,4 GHz (8 cores),
- Operation memory: 16 GB,
- HDD: 500 GB.

Complete number of each operation was 10 000 (*Insert, Update, Delete, Restore*). Number of updated temporal attributes has been generated, total number was 24 965. Minimal number of operations on the object was 3, maximal number was 26, the average value was 5,4965.

The Fig. 29 characterizes the structure of main table:

```
desc tab1
Name            Type
--------------- ------------
 ID              NUMBER(38)
 DATUM           DATE
 CISLO           NUMBER
 CISLO2          NUMBER(38)
 RETAZEC         VARCHAR2(100)
 VALIDITY        DATE
```

**Fig. 29.** Main table structure

The aim of this paper is to address the lack of the data management in shared data system based on attribute granularity. Comparison of the attribute level system and object level approach can be found in [19]. The performance results uses weighted

criterion (Sect. 7). First of all, index structures based on B+trees are compared to choose the best solution for application domain. Then, the hybrid context trees are managed to show the time improvements.

This section deals with the eight different B+tree indexes, which are compared to declare quality:

- no explicit index (*ind1*)
- *ID_orig (ind2),*
- *ID_orig, id_previous_change (ind3),*
- *ID_orig, BD (ind4),*
- *ID_orig, ID_previous_change, BD (ind5),*
- *Unique – ID_orig, ID_previous_change (ind6),*
- *BD, ID_orig (ind7),*
- *BD, ID_orig, ID_previous_change (ind8).*

Figure 30 shows the time consumption perspective of the *Update* operation, Fig. 31 shows the time consumption of the *Select* statement to monitor each object over the time.

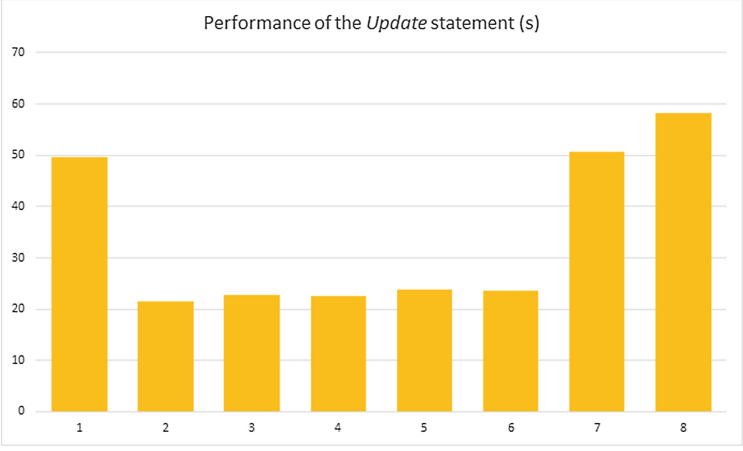



**Fig. 30.** Performance of the *Update* statement

The proposed results of the weighted criterion are then divided into four categories (Fig. 32, Table 2):

- A (standard temporal system) – the best solution provides the index *ind3*.
- B (system highlighting management of undefined states) – the best solution provides the index *ind3*.
- C (temporal system for medical data processing) – the best solution provides the index *ind3*.
- D (temporal system for intelligent transport system) – the best solution provides the index *ind3*.

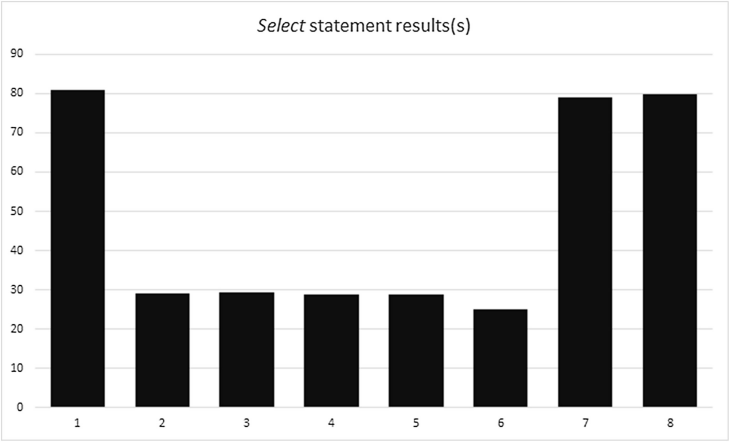



**Fig. 31.** Select statement results

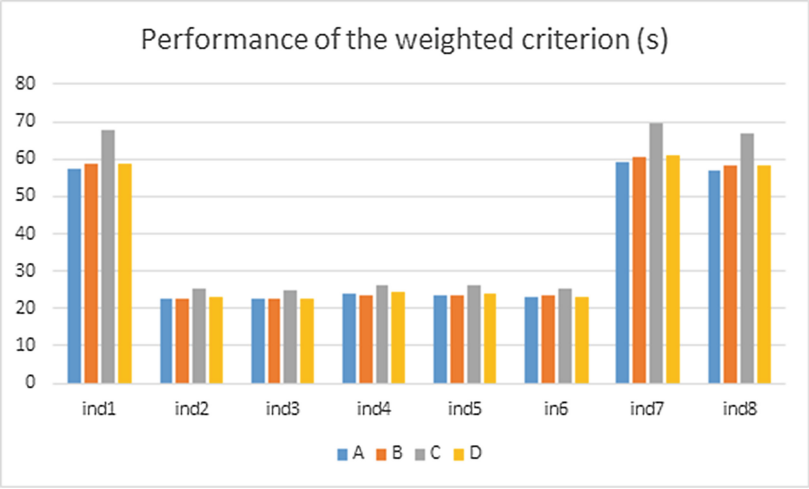



**Fig. 32.** Performance of the weighted criterion

When moving sight to the shared based data system, it is necessary to get the reduction factor, which expresses the ratio corresponding all of the data in the table (or structure) in comparison with the results considering the context conditions. In principle, the performance is affected by the reduction factor (90% rate), the rest part covers the problematics of pointer mapping. Thus, if the reduction factor is growing, also performance results are growing. The dependence is not, however, linear. Figure 33 shows the performance in comparison with reduction factor.

**Table 2.** Time processing (s) based of the weighted criterion

| | *ind1* | *ind2* | *ind3* | *ind4* | *ind5* | *ind6* | *ind7* | *ind8* |
|---|---|---|---|---|---|---|---|---|
| A | 57,40 | 22,69 | 22,63 | 23,86 | 23,74 | 23,20 | 59,22 | 56,84 |
| B | 58,97 | 22,56 | 22,55 | 23,70 | 23,67 | 23,37 | 60,68 | 58,24 |
| C | 67,69 | 25,19 | 25,01 | 26,34 | 26,20 | 25,52 | 69,43 | 66,91 |
| D | 58,76 | 22,88 | 22,54 | 24,51 | 23,97 | 23,23 | 60,91 | 58,16 |

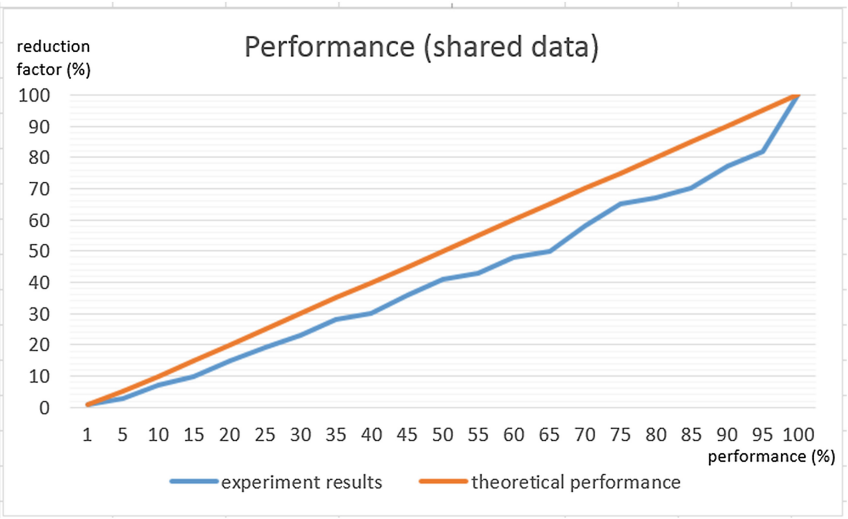



**Fig. 33.** Reduction factor

## 14 Conclusions

Conventional database approach characterizes each object by one row identified by the primary key. It can be said, that it represents only actual valid data, whereas temporal system uses various rows even in multiple tables to define object states over the time. In general, attribute oriented architectonical solution based on column granularity offers far more improved performance for industrial environment, where thousands sensor data each microsecond should be processed. Therefore, it is necessary to develop appropriate solution and evaluate the performance model. Weighted criterion has been proposed, each statement type is characterized by the weight representing the importance and significance in temporal system. It also proposes solution for security access layer, which is important for sharing data. It is based on contexts and policies, which automatically add conditions to individual statements to cover the requirements and restrictions. However, effectiveness of that system must be highlighted, therefore the solution is extended by the pointer mapping functionality to prevent nodes processing, which do not pass the defined policy.

Sensor data processing belongs the most difficult data processing area because of the frequency and amount of data. Therefore, we will focus of index, data and pointer

mapping distribution in the future. It is another sphere, which can improve the global performance reflecting mostly processing time. Dependencies detection between the states and the level of the state results interference is also one of the optimization method, which well be focused by us. Thank to this methodology, we will be able to define dependencies and extended rules to reduce the need for storing and processing data using patterns.

**Acknowledgment.** This publication is the result of the project implementation:

*Centre of excellence for systems and services of intelligent transport*, ITMS 26220120028 supported by the Research & Development Operational Programme funded by the ERDF and *Centre of excellence for systems and services of intelligent transport II.,* ITMS 26220120050 supported by the Research & Development Operational Programme funded by the ERDF.

This paper is also the result of the project implementation *Center of translational medicine*, ITMS 26220220021 supported by the Research & Development Operational Programme funded by the ERDF.


*"Podporujeme výskumné aktivity na Slovensku*

*Projekt je spolufinancovaný zo zdrojov EÚ*

## References

1. Bryla, B.: Oracle Database 11g Administration II Exam Guide. McGraw-Hill Education, New York (2009)
2. Calero, C.: Measuring oracle database schemas. In: IMACS/IEEE CSCC 1999 Proceedings, pp. 7101–7107 (1999)
3. Date, C.J.: Date on Database. Apress, New York (2006)
4. Date, C.J.: Logic and Databases – The Roots of Relational Theory. Trafford Publishing (2007)
5. Date, C.J., Darwen, H., Lorentzos, N.A.: Temporal Data and the Relational Model. Morgan Kaufmann, San Francisco (2003)
6. Feuerstein, S.: Oracle PL/SQL Programming. O'Reilly, Sebastopol (2009)
7. Haan, L., et al.: Applied Mathematics for Database Professionals. Apress, New York (2007). ISBN: 1590597451
8. Hornak, J.: The Basics of MRI. Interactive Learning Software (2008)
9. Hubler, P.N., Edelweiss, N.: Implementing a temporal database on top of a conventional database. In: Conference SCCC 2000, pp. 58–67 (2000)
10. Jähne, B.: Digital Image Processing, pp. 156–294. Springer, Heidelberg (2002). ISBN: 3-540-67754-2
11. Jensen, C.S., Snodgrass, R.T.: Temporally Enhanced Database Design. MIT Press, Cambridge (2000)
12. Janáček, J., Kvet, M.: Min-max optimization of emergency service system by exposing constraints. In: Communications: Scientific Letters of the University of Žilina, vol. 2/2015, pp. 15–22 (2015)

13. Janáček, J., Kvet, M.: Public service system design by radial formulation with dividing points. Procedia Comput. Sci. **51**, 2277–2286 (2015)
14. Janáček, J., Kvet, M.: Sequential approximate approach to the p-median problem. Comput. Ind. Eng. **94**, 83–92 (2016). Elsevier. ISSN: 0360-8352
15. Johnston, T.: Bitemporal Data – Theory and Practice. Morgan Kaufmann, San Francisco (2014)
16. Johnston, T., Weis, R.: Managing Time in Relational Databases. Morgan Kaufmann, San Francisco (2010)
17. Kvassay, M., Zaitseva, E., Levashenko, V., Kostolny, J.: Minimal cut vectors and logical differential calculus. In: 2014 IEEE 44th International Symposium on Multiple-Valued Logic, pp. 167–172 (2014)
18. Kvassay, M., Zaitseva, E., Kostolny, J., Levashenko, V.: Importance analysis of multi-state systems based on integrated direct partial logic derivatives. In: 2015 International Conference on Information and Digital Technologies, pp. 183–195 (2015)
19. Kvet, M.: Temporal data approach performance. In: New developments in Circuits, Systems, Signal Processing, Communications and Computers: Proceedings of the International Conference Circuits, Systems, Signal Processing, Communications and Computers (CSSCC 2015), Vienna, Austria, pp. 75–83, 15–17 March 2015
20. Kvet, M., Janáček, J.: Relevant network distances for approximate approach to the p-median problem. In: Helber, S., et al. (eds.) Operations Research Proceedings 2012: Selected Papers of the International Conference of the German operations research society (GOR), Leibniz Univesität Hannover, Germany, pp. 123–128. Springer, Switzerland (2014). ISSN: 0721-5924, ISBN: 978-3-319-00794-6
21. Kvet, M., Matiaško, K.: Transaction management. In: CISTI, Barcelona, pp. 868–873 (2014)
22. Kvet, M., Vajsová, M.: Transaction management in fully temporal system. In: UkSim, Pisa, pp. 147–152 (2014)
23. Kvet, M., Matiaško, K., Kvet, M.: Complex time management in databases. Central Eur. J. Comput. Sci. **4**(4), 269–284 (2014)
24. Lewis, P., Bernstein, A., Kifer, M.: Databases and Transaction Processing. An Application Oriented Approach. Addison-Wesley, Reading (2002)
25. Maté, J.: Transformation of relational databases to transaction-time temporal databases. In: ECBS-EERC, pp. 27–34 (2011)
26. Matiaško, K., et al.: Database Systems. EDIS (2008)
27. Simsion, G.C., Witt, G.C.: Data Modeling Essentials. Elsevier, Amsterdam (2005)
28. Snodgrass, R.: Developing Time-Oriented Database Applications in SQL. Morgan Kaufmann Publishers, San Francisco (2000)

# Scalable Distributed Datastore for Real-Time Cloud Computing

Maciej Lasota(✉), Stanisław Deniziak, and Arkadiusz Chrobot

Department of Computer Science,
Kielce University of Technology, Kielce, Poland
{m.lasota, s.deniziak, a.chrobot}@tu.kielce.pl

**Abstract.** Recent prognoses about the future of Cloud Computing, Internet of Things and Internet Services show growing demand for an efficient processing of huge amounts of data within strict time limits. First of all, a real-time data store is necessary to fulfill that requirement. One of the most promising architecture that is able to efficiently store large volumes of data in distributed environment is SDDS (Scalable Distributed Data Structure). In this paper we present SDDS $LH_{RT}$, an architecture that is suitable for real-time cloud applications. We assume that deadlines, defining the data validity, are associated with real time requests. In the data store a real-time scheduling strategy is applied to determine the order of processing the requests. Experimental results shows that our approach significantly improves the storage Quality-of-service in a real-time cloud environment.

## 1 Introduction

Cloud computing becomes one of the most crucial paradigm of providing services on the Internet. Cloud providers offer computation, storage and application hosting services, according to different cloud models like Software as a Service (SaaS), Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and others [1]. The most challenging problem is to achieve consistent and reliable operation under peak loads. To guarantee the desired quality of service (QoS) and minimize the cost of the system, an efficient cloud organizations should be developed.

Besides supporting computing environment, clouds are mainly used for storing user data. Storage services were also defined as a new business model Storage as a Service (STaaS). Storing data in the cloud has many advantages. It gives the ability to store large amounts of data without having to purchase and maintain large-scale storage resources. It also allows the use of advanced storage infrastructure providing fast access to data.

The majority of contemporary data repositories is built with the purpose to store data for off-line processing. The Relational Database Management Systems (RDBMs) are in a common use, however a recent grow of interest in Big Data processing environments caused a significant development of other data store models like NoSQL or NewSQL. The increasing number of Internet of Things (IoT) devices and Internet services introduces a new category of data sets, the Fast Data [2, 3]. They are characterized not only by a high volume but also by a high velocity.


J. Janech et al. (eds.), *Proceedings of the 2015 Federated Conference on Software Development and Object Technologies*, Advances in Intelligent Systems and Computing 511,
DOI 10.1007/978-3-319-46535-7_15

In Fast Data every data set and every request concerning the data have attributed validity that starts to decrease after a given deadline. Proper storing and processing of such data may thus require a real-time approach. The consequences of missing a deadline depend on the type of a real-time policy [4–6]:

- in a hard real-time policy a negative value of validity is assigned to tardy requests and data,
- in a firm real-time policy tardy requests and data have no validity,
- in a soft real-time policy the validity of tardy requests and data is still positive, but it diminishes over time.

The problem of Fast Data storing and processing was addressed in several data store implementations, however most of them do not enforce any time restrictions on that operations. They usually define the real-time as "as fast as possible". Since, such approach does not conform to any real-time policy used in real-time Internet services, it is not suitable for real-time cloud applications.

In this paper we present the datastore which is based on SDDS $LH_{RT}$ architecture presented in [7]. SDDS $LH_{RT}$ is a Scalable Distributed Data Structure (SDDS)-based architecture for data store that adheres to the firm real-time policy. The data validity is specified by the deadline associated with the request. The SDDS server schedules requests using realtime scheduling method. Experimental results showed that in our approach significantly more requests may be served in the required time, than in existing data stores, where requests are processed in the FIFO order. In this way a significantly better storage QoS may be achieved.

The paper is organized as follows. Section 2 contains a short summary of related work. Section 3 describes the concept of SDDS and in Sect. 4 a motivation for the research is given. An implementation of real-time policy for SDDS is described in Sect. 5. Results of testing the prototype implementation of SDDS $LH_{RT}$ are presented in Sect. 6. The paper is concluded in Sect. 7.

## 2 Related Work

An RDBM that supports a real-time request processing is called a Real-Time Database System (RTDBS). Many such systems were built for both research and commercial purposes [4–6]. Some of them offer only partial real-time functionality while others provide a full support [6].

The real-time data stores are a less explored subject. The majority of such systems is built with the purpose of storing as quickly as possible huge amounts of real-time data. In that context real-time means "happening recently".

VoltDB [3] is an in-memory scalable relational database designed for handling Fast Data. Some large companies like Ericsson and HP take advantage of VoltDB. Similarly to other RDBMs, VoltDB stores data in tables. However, these tables are partitioned column-wise and distributed together with stored procedures over nodes of a cluster. Every stored procedure is executed in a single thread, hence it does not require locking or synchronization. That allows VoltDB to process many queries in parallel, which greatly contributes to its efficiency. To further improve its performance VoltDB

replicates some of the most frequently used tables. Despite all of these features, the VoltDB does not guarantee that requests from clients will be processed within predictable time periods. The Mahanaxar [8] is an ongoing research project, which goal is to build a data storage for intercepting, evaluating and storing real-time data for latter processing. Unfortunately many details about this effort are not yet known.

Druid [9] is an in-memory real-time data store used by such companies like Netflix or eBay. Its architecture consists of four main elements:

- Real-Time Nodes are responsible for ingesting data about events that happened in a given period of time; these data are immediately available for clients,
- Historical Nodes store previously acquired data about past events,
- Broker Nodes direct requests to Historical or Real-Time Nodes,
- Coordinator Nodes distribute and manage Historical Nodes.

Aside from these elements Druid also utilizes external components like databases and file systems. It is built to be high available and fault-tolerant. However, its main drawback, as far as in real-time applications are considered, is significant variability in request processing time (latency).

The Chord [10] is an algorithm and protocol used in peer-to-peer environments to locate data across a peer-to-peer network. It is designed for a network that changes dynamically. The Chord uses Distributed Hash Table (DHT) [11] to look up data in distributed environment. It organizes nodes of such network into a ring. Every node in a ring is given a unique identifier, which is used for addressing and assigning data to the node. Nodes may join or depart from the network at any time. The Chord protocol is scalable and efficient for P2P purposes. It is available for user applications via a special library.

An interesting real-time distributed storage was developed for gathering and processing data from Phasor Measurement Units used in Wide-Area Measurement System (WAMS) for power grids [12]. The data store is based on Chord protocol which basic version is well-suited for the needs of WAMS. The real-time requirements are met by recognizing a pattern in which periodic and aperiodic tasks appear and by applying a cyclic executive which serves those tasks within given time intervals called time frames. Periodic tasks are given precedence before aperiodic ones. Failing to serve a task of any kind in a given time frame results in postponing its execution to the next time frame. That means that the system adheres to soft real-time requirements.

## 3 Scalable Distributed Data Structures (SDDSs)

Multicomputer systems are often used in applications that need huge data storage with a short access time. Local hard disks are not sufficient in such situations. Scalable Distributed Data Structures (SDDSs) are a family of data structures designed for efficient data management in a multicomputer [13, 14]. The basic data unit in SDDS may be either a record or an object with a unique key. Those data are organized in larger structures called servers/buckets and stored usually in RAM of multicomputer nodes that run an SDDS server software and connected each other with the Fast Local Area Network (Gigabit or Infiniband). Data from a bucket are saved on a hard disk only

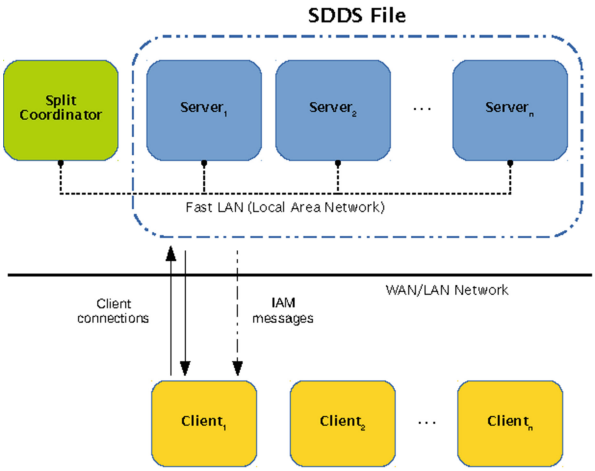



**Fig. 1.** SDDS architecture

when necessary e.g. when the server is shutting down. All of the buckets form an SDDS file (Fig. 1).

Initially, SDDS file consists of only one bucket which level is equal to zero. When the bucket reaches its capacity limit a collision occurs. SDDS adjusts its capacity to current needs by splitting buckets. When a bucket is overloaded it sends a message to the Split Coordinator (SC) [13–15]. The SC takes a decision which bucket should split. During a split operation a new bucket is created with the level higher by one than the splitting bucket. Next, about half of the data stored in the splitting bucket is moved to the new bucket. When the transfer is completed the splitting bucket increments its level by one and acknowledges the SC. After the split, both buckets have the same level (Fig. 2).

The data in the SDDS file are accessed by an SDDS client software executing on multicomputer nodes, other than servers. The client does not have the whole information about the file. Instead it has its own file image, which may differ from the actual image of the file. The client uses a special function to address data item (a record or an object) in buckets. SDDS may be classified according to addressing functions they use, for example:

- LH* - the client uses linear hashing [13].
- RP* - the client uses range partitioning [16].

Every client has its private image of SDDS file. After the split this image becomes outdated and the client may send request to an incorrect bucket. In such situations the message is forwarded to the correct bucket and the client receives Image Adjustment Message (IAM). The IAM contains new parameters for client's addressing function that updates its SDDS image.

There is no single point of failure in SDDS, except for the SC, because no central directory is used for addressing. However, there are SDDS architectures (like RP) that do not require the SC. All SDDS implementations follow the same designs rules [13–16]:

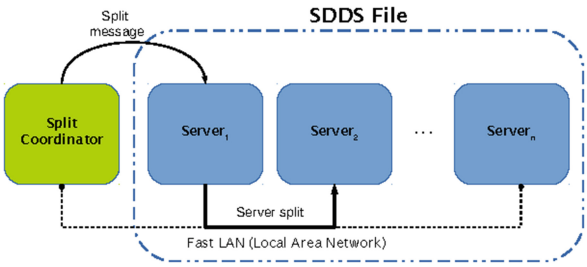



**Fig. 2.** SDDS bucket split operation

1. For performance reasons, no central directory is used by the clients in the process of data addressing.
2. The SDDS file adjusts its size to the clients needs by splitting buckets, i.e. moving about half of the content of buckets that reached their capacity to the newly created buckets.
3. Due to the split operations the client's image may become outdated, but it is corrected only when the client makes an addressing error.
4. None of the basic operations on data requires immediate, atomic updating of client's image.
5. If the client incorrectly addresses data, then an Image Adjusting Message (IAM) that allows it to update its image is sent, and the data are forwarded to the correct server.

The SDDSs constitute an example of NoSQL data store design. The data items are stored in-memory in a key-value form. However, the original concept of SDDS is not entirely suitable for Fast Data processing. It lacks the realtime capabilities.

## 4 Motivation

Aside of Fast Data processing there are many applications that could benefit from using a real-time data store [17, 18]. For example, cloud computing environments offer services that typically follow the soft real-time policy by monitoring Quality of Service (QoS) and dynamically allocating resources to applications that are critical for clients [19–21]. Using a real-time data storage could simplify this task. Also the automated trading [22], financial [23] and surveillance [24] systems require an access to relevant data in a limited time [4].

The architecture of SDDS is relatively simple and easy to modify, when compared to such data stores like Druid [9]. In our research we have addressed the issue of enabling firm real-time requests of data fetching from SDDS LH*. Such operations are more critical for real-time applications than data write requests.

We have assumed that the data store is used by many clients with different time requirements. This is a common case in many systems. For example, in a large scale medical information system access to the data about intensive care patients should be of higher priority than the access to information about patients who undergo a long term

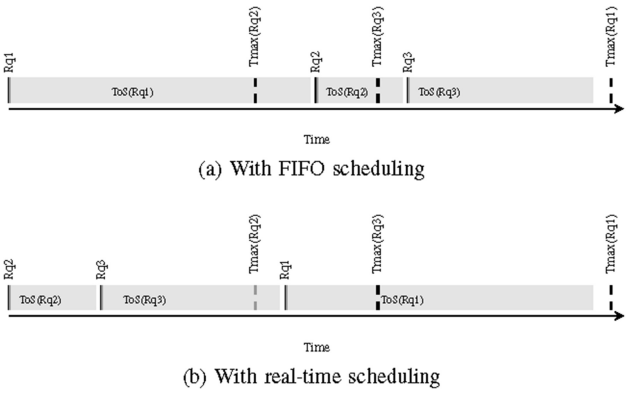



**Fig. 3.** Handling of requests

diagnostic procedure. Here, the priority is based on combination of a deadline and service time of a request. Without a real-time scheduling the requests with a longer deadline and time of service (ToS) would postpone the more time constrained ones. As a result they would fail to complete before their deadlines. Figure 3 illustrates such a situation. When requests will be serviced according to the first-in first-out rule (Fig. 3(a)) then only the deadline for request Rq1 will be satisfied. If all requests will be reordered according to their deadlines then all-time requirements will be met (Fig. 3(b)).

We have also assumed that the time of transferring request through network may be properly managed or, at least to some extent, predicted [25].

In our approach we have chosen SDDS LH* [13]. This type of SDDS uses distributed linear hashing for addressing buckets. This algorithm requires an additional element in SDDS architecture called Split Coordinator for overseeing the order in which buckets split. The split operation is performed only after a write requests. Hence providing firm real-time writes for such an SDDS would be more challenging than in the case of any other version of Scalable Distributed Data Structure. On the other hand, it would also give a more general solution to the problem. We also do not discuss in the paper the read requests that result in IAM messages. Those may be prevented by allowing the clients to use filled-up read-only SDDS LH* or by permitting only these writes that modify data instead of adding them. However, we want to address both of the aforementioned issues in our future works.

## 5 Implementation of Real-Time Policy in SDDS LH*

The original SDDS LH* architecture consists of three basic software components:

1. a client that accesses data items in buckets,
2. a server which manages a single bucket located in RAM,
3. a split coordinator overseeing split operations of all buckets.

The data in the SDDS file are accessed by clients performing four basic operations: read, write, update and delete. Only the read requests are real-time and they require the shortest possible time of handling to be ensured. Other operations are nonreal-time, so they do not have assigned deadlines. Providing real-time scheduling policy for read requests requires a proper internal structure of the server. We propose an organization that is shown in Fig. 4. The incoming requests from clients are received by a pool with a fixed number of threads that operate on a connection queue. If the priority of request is less than 0 the request is rejected. The threads insert the requests into a request queue which is handled by a single thread operating on a bucket. If a read request taken form the head of the queue meets its deadline, the thread fetches a corresponding data item from the bucket and sends it to the client. Otherwise the request is rejected. Other types of requests are handled in request queue in the FIFO order. Requests that need to be forwarded to another servers are put into forward queue that is processed by a dedicated thread pool also with a fixed number of threads. In non-real-time SDDS LH- servers the

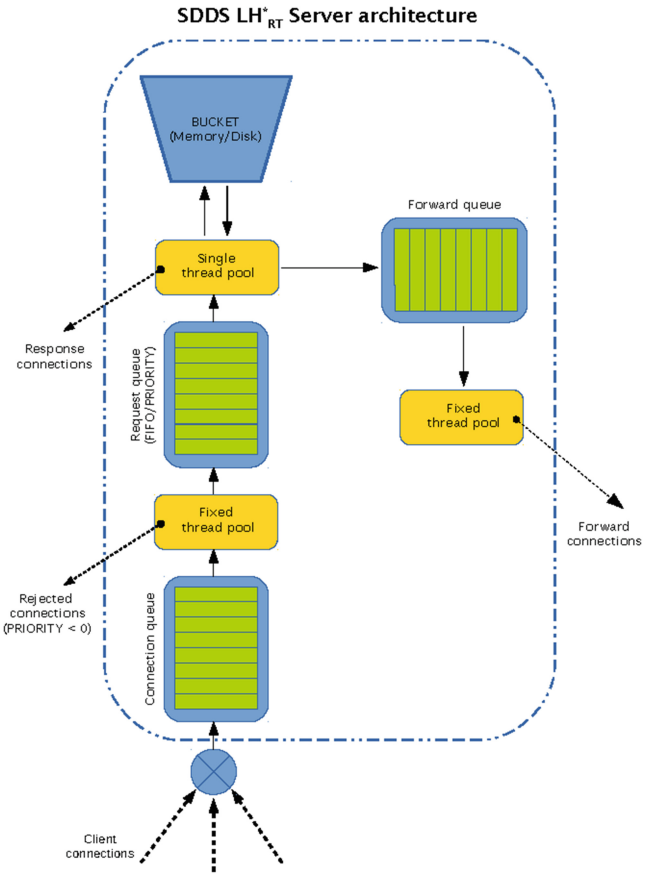



**Fig. 4.** Server organization

request queue is a regular FIFO. In real-time servers a priority queue is used instead. Requests are ordered using Least Laxity First (LLF) scheduling method [26].

The priority of the request is given by the Formula 1:

$$Priority = \begin{cases} \frac{1}{D - T_{rp}} & if\ D > T_{rp} \\ 0 & otherwise \end{cases} \quad (1)$$

where $D$ is a deadline and $T_{\text{rp}}$ is time of request processing. Requests are processed in descending order of their priorities. Requests with priority equal to 0 are rejected.

Since the time of accessing a data item in a bucket is negligible comparing to the average time of transmitting the item through a network ($T_{\text{nm}}$), we define the $T_{\text{rp}}$ as follows (Formula 2):

$$T_{\text{rp}} = n * T_{\text{nm}} \quad (2)$$

where the n factor is equal 4 and it is introduced to provide for potential delays in network transmission. The Split Coordinator (SC) is a special element that oversees all bucket splits. SC takes a decision which bucket should split. During a split operation a new bucket is created and about half of the data stored in the splitting bucket are moved to the new bucket. In order to support real-time policy during the splits, transfer between buckets occurs in the background. The read requests operate on a copy of data until the split ends, while other requests are blocked. In the rest of the paper we refer to the real-time enabled SDDS LH* by the name SDDS $\text{LH}_{\text{RT}}$.

## 6 Experimental Results

Since there is no similar solutions that deal with real time requests, our architecture were evaluated by comparing the QoS obtained in traditional SDDS LH* and the QoS supported by the SDDS $\text{LH}_{\text{RT}}$. We performed also some other experiments, showing the advantages of our approach. Both SDDS architectures (SDDS LH* and SDDS $\text{LH}_{\text{RT}}$) were implemented as prototype distributed data stores, then experiments comparing the effectiveness of both solutions were performed. In the experiments we have evaluated

**Table 1.** Characteristics of requests

| Size [KiB] | Deadline range [ms] | Average time of transmission [ms] |
|---|---|---|
| 4 | 20–40 | 1 |
| 8 | 25–50 | 2 |
| 16 | 35–70 | 3 |
| 32 | 50–100 | 4 |
| 64 | 60–120 | 5 |
| 128 | 75–150 | 8 |
| 256 | 100–200 | 16 |
| 512 | 750–1000 | 64 |
| 1024 | 1000–2000 | 192 |

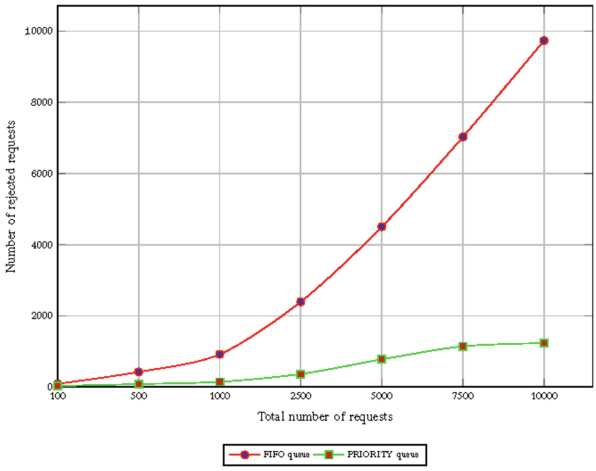



**Fig. 5.** Rejected read requests with the regard to the scheduling policy, records size 4 KiB–1024 KiB, deadlines 20 ms–2000 ms

the amount of rejected time-constrained read and update requests and the maximal queue length for the real-time and the nonreal-time servers.

As an environment for tests we have used tree nodes of a cluster computer. The client software was executed on a nodes with Intel Xeon E5410@2.33 GHz processor (4 cores) and 16 GiB RAM. Server software was run on a node with Intel Xeon E5205@1.86 GHz processor (2 cores) and 6 GiB RAM. All nodes were connected through Gigabit Ethernet network. In a single test scenario the client was sending a number of requests ranging from 100 to 10000. The deadlines for the request was

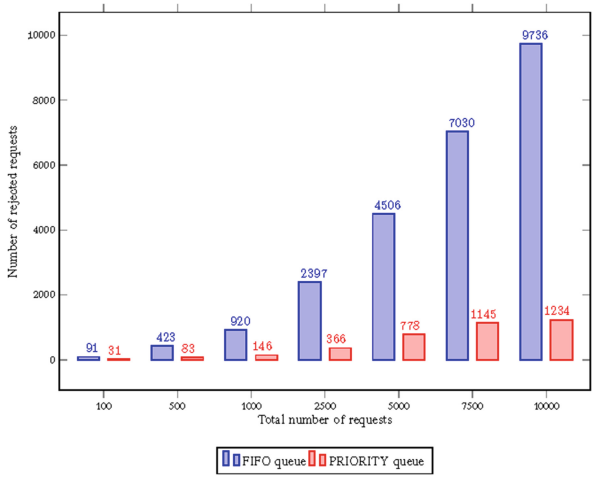



**Fig. 6.** Rejected read requests with the regard to the scheduling policy, records size 4 KiB–1024 KiB, deadlines 20 ms–2000 ms

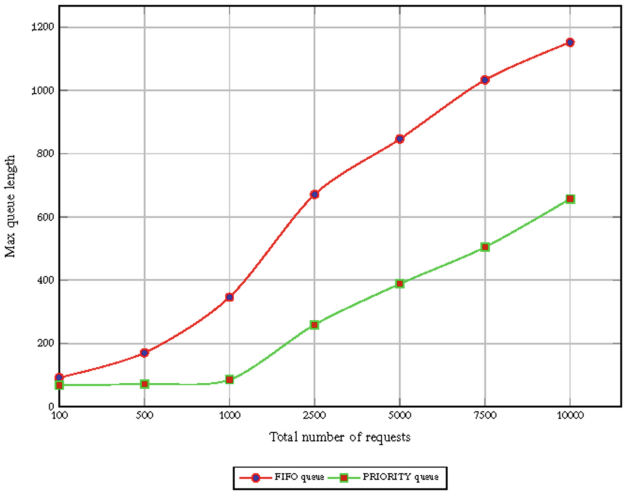



**Fig. 7.** Maximal length of the requests queue, records size 4 KiB–1024 KiB, deadlines 20 ms–2000 ms

chosen from 20 ms to 2000 ms accordingly to an estimated size of the request's response which ranged from 4 KiB to 1024 KiB. The stream of requests in each test was unordered i.e. the requests formed a random pattern with the respect to the value of deadlines. More detailed information about requests is given in Table 1. For each record size the appropriate average time of transmission and the deadline range are assigned. The time of transmission is an average time of processing the single request, which was measured in our testing environment. Deadline ranges were chosen

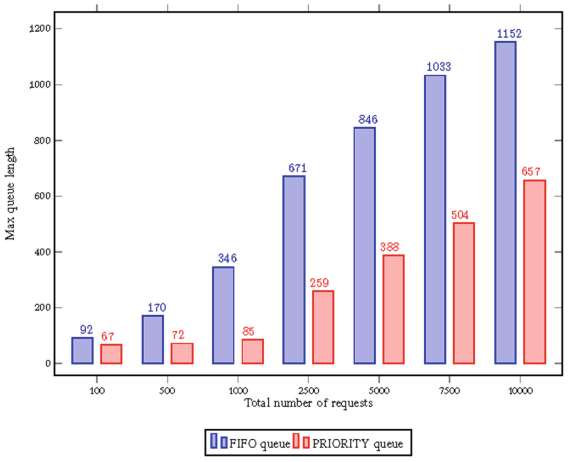



**Fig. 8.** Maximal length of the requests queue, records size 4 KiB–1024 KiB, deadlines 20 ms–2000 ms

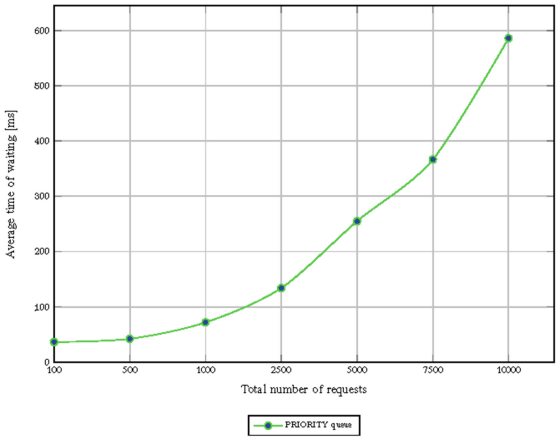



**Fig. 9.** Average waiting time for the execution of GET operations

experimentally to test the wide range of QoS obtained in the SDDS LH*, i.e. QoS < 10% for minimal deadlines and QoS > 90% for maximal deadlines.

Figures 5 and 6 show that with the growing load, the SDDS LH* server (FIFO queue) may reject up to 97% of all requests, while the rejection rate of the SDDS $LH_{RT}$ server (PRIORITY queue) stays below 13%.

Figures 7 and 8 show the maximum length of requests queue for a given scheduling policy. With the increasing total number of request the SDDS $LH_{RT}$ server unloads its request queue almost twice more efficiently than the SDDS LH* server.

Figure 9 presents average delay of tasks waiting in the priority queue. We may observe that this delay linearly increases according to the number of records that are read i.e. according to the processing time. Since deadlines are also proportional to the size of transmission (Table 1), it is conformable with the LLF strategy. In the FIFO

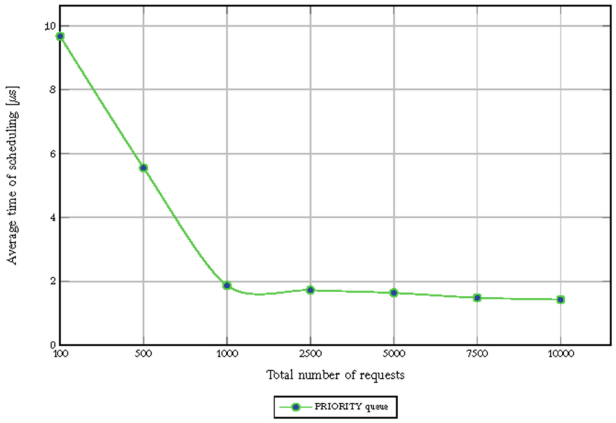



**Fig. 10.** Average time of scheduling GET operations

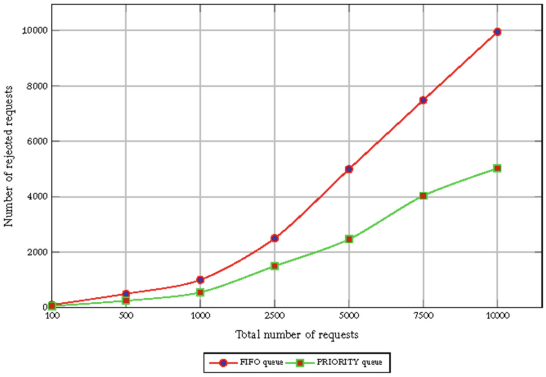



**Fig. 11.** Rejected GET and UPDATE operations, records size 4 KB–1024 KB, deadlines 20 ms–5000 ms

scheduling, the average waiting time will be the same regardless of the size of transmitted data items.

Scheduling of requests introduces a time overhead associated with the calculation of priorities and queue management. Figure 10 shows that for more than 1000 requests this overhead is below 2 µs and slightly decreases with increasing number of requests.

Similar experiments were performed in the environment where one client was sending GET requests while other was sending UPDATE ones. Figure 11 presents results. We may observe that in this case significantly more requests were rejected in the PRIORITY queue while number of rejected requests processed in the FIFO order is unchanged. But still our approach twice improves the results.

UPDATE requests require more time for processing, thus such requests significantly decrease the performance of the data store. This results in higher number of rejected requests as well as a longer queues waiting for the processing (Fig. 12). But still the queue length is twice shorter in the PRIORITY queue.

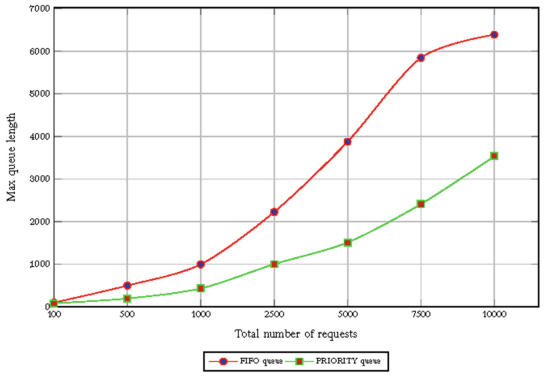



**Fig. 12.** Maximum request queue length for GET and UP- DATE operations, records size 4 KB–1024 KB, deadlines 20 ms–5000 ms

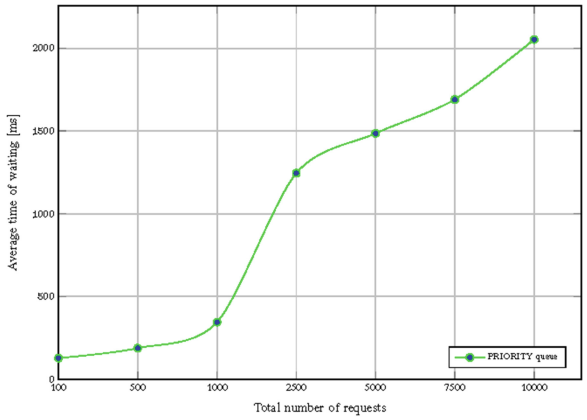



**Fig. 13.** Average waiting time for the execution of GET and UPDATE operations

Figure 13 presents average delay of tasks waiting in the priority queue when one client was sending GET requests while the other was sending UPDATE ones. We may observe decrease in performance, the delay linearly increases according to the number of records. This is caused by increasing number of requests, that causes lengthening of the queue. Since UPDATE requires more time to process, delays are significantly longer then in case of GET requests (Fig. 9).

The similar situation occurs in the case of scheduling of requests shown in Fig. 14. With smaller number of the requests a priority queue behaves like a FIFO queue, increasing number of requests improves performance and reduces scheduling time.

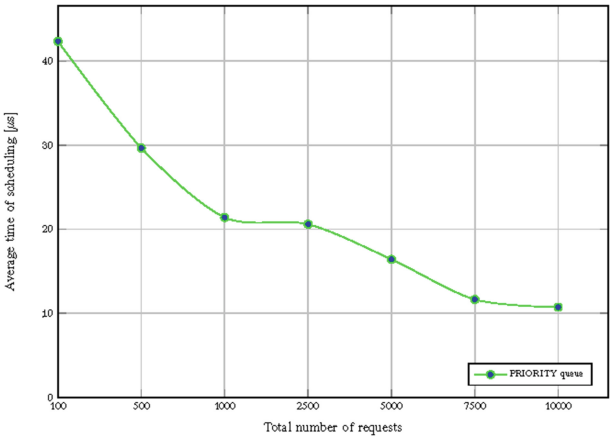



**Fig. 14.** Average time of scheduling GET and UPDATE operations

## 7 Conclusions

In this paper we have presented the architecture of real-time data store that conforms to the real-time policy used in the real-time cloud computing. Our approach is based on scalable distributed data structures SDDS LH*. We have developed a real-time variant of SDDS LH* called SDDS $LH_{RT}$, where requests are scheduled according to the LLF method. The experimental results indicate that when a server is overloaded with incoming requests, even a relatively simple real-time scheduling policy allows it to process significantly more request, before their deadlines expire. Therefore, the number of rejected requests is greatly reduced. In this way SDDS LHRT assures significantly higher QoS in the cloud environment.

We expect that the results could be further improved by applying aging of queued requests or using a dedicated implementation of the priority queue. Prototype implementation were not optimized, thus still there is a room for further improvements.

In our future work we plan to address the problem of ensuring the firm real-time policy for data insertion/update requests and bucket split operations. Solving those problems will also require providing a real-time forwarding of misaddressed requests. We will also consider other methods (e.g. hybrid MP/CLP [27]) for optimization of SDDS LH*-based datastores.

## References

1. Buyya, R., Broberg, J., Goscinski, A.: Cloud Computing: Principles and Paradigms. Wiley, Hoboken (2011)
2. Hui, P., Chikkagoudar, S., Chavarría-Miranda, D., Johnston, M.: Towards a real-time cluster computing infrastructure. In: The 32nd IEEE Real-Time Systems Symposium (RTSS 2011), pp. 17–20. IEEE, Piscataway (2011)
3. VoltDB: Fast data – fast, smart, scale|voltdb. www.voltdb.com. Accessed 14 Apr 2015
4. Kao, B., Garcia-Molina, H.: An overview of real-time database systems. In: Halang, W.A., Stoyenko, A.D. (eds.) Advances in Real-Time Systems, pp. 463–486. Springer, Heidelberg (1994)
5. Aldarmi, S.A.: Real-time database systems: concepts and design (1998)
6. Lindström, J.: Real Time Database Systems. Wiley Encyclopedia of Computer Science and Engineering (2008). http://dx.doi.org/10.1002/9780470050118.ecse575
7. Lasota, M., Deniziak, S., Chrobot, A.: An SDDS-based architecture for a real-time data store. Int. J. Inf. Eng. Electron. Bus, November 2015. MECS Publisher
8. Bigelow, D., Brandt, S., Bent, J., Chen, H., Nunez, J., Wingate, M.: Mahanaxar: managing high-bandwidth real-time data storage. https://systems.soe.ucsc.edu/node/389. Accessed 14 Apr 2015
9. Yang, F., Tschetter, E., Léauté, X., Ray, N., Merlino, G., Ganguli, D.: Druid: a real-time analytical data store. In: Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data, SIGMOD 2014, pp. 157–168. ACM, New York (2014). http://doi.acm.org/10.1145/2588555.2595631
10. Stoica, I., Morris, R., Karger, D., Kaashoek, F., Balakrishnan, H.: Chord: a scalable peer-to-peer lookup service for internet applications. In: Proceedings of the 2001 ACM SIGCOMM Conference, pp. 149–160 (2001). http://pdos.csail.mit.edu/papers/chord:sigcomm01/chord_sigcomm.pdf

11. Steinmetz, R., Wehrle, K.: Peer-to-Peer Systems and Applications. LNCS, vol. 3485. Springer-Verlag, Heidelberg (2005)
12. Qian, T., Chakrabortty, A., Mueller, F., Xin, Y.: A real-time distributed storage system for multi-resolution virtual synchrophasor. In: PES General Meeting on Conference Exposition, July 2014, pp. 1–5. IEEE (2014)
13. Litwin, W., Neimat, M.-A., Schneider, D.A.: LH* — a scalable, distributed data structure. ACM Trans. Database Syst. **21**(4), 480–525 (1996). citeseer.ist.psu.edu/litwin96lh.html
14. Ndiaye, Y., Diene, A., Litwin, W., Risch, T.: AMOS-SDDS: a scalable distributed data manager for windows multicomputers. In: 14th International Conference on Parallel and Distributed Computing Systems, PDCS (2001). citeseer.ist.psu.edu/ndiaye01amossdds.html
15. Sapiecha, K., Łukawski, G.: Scalable Distributed Two-Layer Data Structures (SD2DS). IJDST **4**, 15–30 (2013)
16. Litwin, W., Neimat, M.-A., Schneider, D.: RP*: a family of order preserving scalable distributed data structures. In: Proceedings of the Twentieth International Conference on Very Large Databases, Santiago, Chile, pp. 342–353 (1994). citeseer.ist.psu.edu/736278.html
17. Bak, S., Czarnecki, R., Deniziak, S.: Synthesis of real-time cloud applications for Internet of Things. Turkish J. Electr. Eng. Comput. Sci. **37**(3), 913–929 (2015)
18. McGregor, C.: A cloud computing framework for real-time rural and remote service of critical care. In: 2011 24th International Symposium on Computer-Based Medical Systems (CBMS), pp. 1–6, June 2011
19. Tsai, W., Shao, Q., Sun, X., Elston, J.: Real-time service-oriented cloud computing. In: 6th World Congress on Services, SERVICES 2010, Miami, Florida, USA, 5–10 July 2010, pp. 473–478 (2010). http://dx.doi.org/10.1109/SERVICES.2010.127
20. Liu, S., Quan, G., Ren, S.: On-line scheduling of real-time services for cloud computing. In: 6th World Congress on Services, SERVICES 2010, Miami, Florida, USA, 5–10 July 2010, pp. 459–464 (2010). http://dx.doi.org/10.1109/SERVICES.2010.109
21. Kyriazis, D., Menychtas, A., Oberle, K., Voith, T., Lucent, A., Boniface, M., Oliveros, E., Cucinotta, T., Berger, S.: A real-time service oriented infrastructure. In: Proceedings of Annual International Conference on Real-Time and Embedded Systems (RTES 2010), pp. 39–44 (2010)
22. Freeny, C.: Automatic Stock Trading System, uS Patent 6,594,643 (2003). http://www.google.com/
23. Fenu, G., Surcis, S.: A cloud computing based real time financial system. In: Bestak, R., George, L., Zaborovsky, V.S., Dini, C. (eds.) ICN 2009, pp. 374–379. IEEE Computer Society (2009). http://dblp.uni-trier.de/db/conf/icn/icn2009.html#FenuS09
24. Javed, O., Rasheed, Z., Alatas, O., Shah, M.: KNIGHTTM: a real time surveillance system for multiple and non-overlapping cameras. In: Proceedings of the 2003 IEEE International Conference on Multimedia and Expo, ICME 2003, Baltimore, MD, USA, 6–9 July 2003, pp. 649–652 (2003). http://dx.doi.org/10.1109/ICME.2003.1221001
25. Lu, F., Wang, J., Cheng, L., Xu, M., Zhu, M., Chang, G.-K.: Millimeter-wave radio-over-fiber access architecture for implementing real-time cloud computing service. In: CLEO 2014, p. STu1J.1. Optical Society of America (2014). http://www.opticsinfobase.org/abstract.cfm?URI=CLEO_SI-2014-STu1J.1
26. Han, S., Park, M.: Predictability of least laxity first scheduling algorithm on multiprocessor real-time systems. In: Zhou, X., et al. (eds.) EUC Workshops 2006. LNCS, vol. 4097, pp. 755–764. Springer, Heidelberg (2006)
27. Sitek, P., Wikarek, J.: A hybrid approach to the optimization of multiechelon systems. Math. Prob. Eng. **2015**(12) (2015)

# Application of Statistical Classifiers on Java Source Code

Matej Mojzes, Michal Rost, Josef Smolka(✉), and Miroslav Virius

Department of Software Engineering, Faculty of Nuclear Sciences and Physical Engineering, CTU in Prague, Prague, Czech Republic
smolkjos@fjfi.cvut.cz

**Abstract.** The paper deals with detection of structures in source codes employing statistical classification. To enhance source code perception by development tools like code editors, modeling tools and source code repositories, various methods of patterns classification are proposed and tested. To be able to apply classification algorithms, well-defined feature space is required. Thus, such a feature space is presented and tested. Sub-models search is carried out by a genetic algorithm to select the optimal feature space subset without deterioration of a classification system. The results show that with standard classification algorithms like k-NN or Perceptron, accuracy of 0.8 can be achieved.

## 1 Introduction

Detection and classification of patterns that surround man are not an invention of this or previous century. The urge to bring an organization into apparent chaos of the natural world is old as Plato, the classical Greek philosopher, mathematician and pedagogue, whose work on forms and properties was further extended by his most famous student Aristotle [3]. He divided properties of things into two categories, ones that are the same for each member of a class and ones that differ between classes. Definition of feature space for statistical classification is exactly the same concept, to find properties that distinguish members of different classification classes [9]. This paper deals with an uncommon data type for a statistical classification, on contrary to the image or sound processing, for source code of an object-oriented strongly typed languages, like Java, there are no best practices concerning definition of feature space for various tasks. Thus, to be able to employ various classical classifiers as Nearest Neighbour (NN, and k- NN extension), logistic regression, Perceptron (linear as well as modern kernel variants), Artificial Neural Network (ANN) and Support Vector Machine (SVM) for pattern recognition, means of feature space definitions have to be examined thoroughly.

Recognition and classification of defined structures in source code can be regarded as a required prerequisite of complex automatic refactoring, advanced measurement of software and design quality or source code verification against a specification. In sum, successful application of statistical principles traditionally used in machine perception can enhance the source code perception by standard software development tools.


J. Janech et al. (eds.), *Proceedings of the 2015 Federated Conference on Software Development and Object Technologies*, Advances in Intelligent Systems and Computing 511,
DOI 10.1007/978-3-319-46535-7_16

### 1.1 Previous Research

The idea of complex patterns recognition in source code emerged in the late 90 s with the expansion of object-oriented languages. The principles of recognition differ with authors of particular papers. In 1998, Antoniol et al. published a work on pattern detection in C++ source codes [2]. The important idea is to use source code metrics for design patterns detection. In their second paper, Antoniol et al. presented a method of design pattern recovery built on feature vector [1]. The main concept is to represent a source code in the simplified form of Abstract Object Language (AOL), which is then used for feature extraction. Classification is performed by rules extraction method. In 2004, Gueheneuc et al. issued a paper, in which much more sophisticated feature space is presented [15]. In contrary to Antoniol, Gueheneuc stated that even single class can be classified as a member of more complex structure [15]. Same as Antoniol, Gueheneuc also used rules extraction method of classification. In contrary to Antoniol and Gueheneuc, Ferenc et al. are using various methods of machine learning, decision trees and artificial neural networks in particular. Another approach to classification on source code is deciding whether the presented code is right or bad according to some coding standards. Lerthathairat and Prompoon used a fuzzy rules extraction method for this purpose [16]. Maiga et al. further extend this idea by detecting anti-patterns in a code, which indicate possible problems. Detection is carried out by Support Vector Machines [18].

### 1.2 Basis of the Paper

This paper honors the idea, that every class (as a data type) can be classified as single-class pattern or member of some more complex pattern and further develops the application of statistical classifiers for patterns detection and classification. For this purpose, new feature space is proposed as well as a method of features extraction from Java source code.

## 2 Methods of Classification

The paper deals with classification in Java source code, but principles described here can be used in any other objectoriented strongly typed languages like C++ or C#. The proposed method cannot be applied to dynamic languages (like Ruby, Groovy or Python) as some of the developed features depend on variable data type matching during static analysis.

### 2.1 Detected Structures

To test the proposed approach some simple single-class design patterns [14] as well as structures that correspond to several UML 2 class stereotypes [21] were chosen. In sum, eleven classes were defined:

- **Utility** is a class that contains mainly static methods that are mostly used by other classes. In UML, such a class is usually denoted by Utility stereotype. Utility class can be identified by the ratio of static methods and ratio of self-method invocation.

- **Factory** is a class that contains factory methods. Factory methods are responsible for object instances creation, abstracting the process of object creation so that the actual type can be resolved at run-time. Factory is a typical design pattern. Factory pattern can be identified by the ratio of methods that return a result of class constructor invocation (directly or through another factory method).
- **Builder** is a support class for setting up a state of complex objects; it is a typical design pattern. Methods of builder usually have one or more arguments, return self as a result and work with class properties.
- **Adapter** is a class that addresses a problem of incompatible interfaces by providing an interlink. Adapter is another common design pattern. Adapter usually implements one or more interfaces and contains a property of the adapted type.
- **Proxy** is a class that serves as a placeholder for an actual object. Usually, it is used to introduce a caching or a security feature not present in the actual object. Proxy is derived from the same base class as the actual object and contains a property of the actual object.
- **Decorator** is a class that usually alter the behavior of decorated object or add some new functionality. Decorator usually has a constructor with decorated object as an argument.
- **Bean** is a primitive class holding data. This is typical for UML Type stereotype or Crate design pattern. Bean can be easily identified by a high ratio of setters and getters.
- **Data Access Object (DAO)** is a class serving as an accessor to some (typical remote) collection of data. For DAO, a high ratio of methods with a relatively small number of arguments and excessive usage of try/catch are the typical symptoms. In UML, DAO is usually denoted by Entity stereotype.
- **Worker** is a class usually implementing some data transformation that is a core process for an application. Worker class usually uses or combine other objects to achieve a principal goal of the application. Such a class can be denoted in UML by a Focus stereotype. Identification of worker is not an easy task as it is a very dynamic structure with little or no rules of implementation.
- **Composite** represents some hierarchical structure, usually a recursive tree. Composite typical holds a reference to children or parent of the same type.
- **Constant** is similar to Utility class, but for the Constant pattern, the ratio of static properties instead of methods is typical.

Design patterns Adapter, Proxy and Decorator are quite similar in an implementation and their successful classification requires to grasp the little differences and project them into a feature space. It is not an easy task to collect sufficient amount of samples for every proposed class. For this, public source code repositories were manually searched, and representative samples were hand-picked.

## 2.2 Feature Space

In machine learning, a feature is a measurable property that describes a certain quality of observed phenomena and typically has a numerical form. Features are used as a whole set called the feature vector. Features in the feature vector should be independent

and should distinguish patterns or classes from each other. The presented feature space is based on a study of selected classes, application of software engineering and object-oriented programming knowledge. Feature $F_i$ from feature space $S = \{F_1, F_2, \ldots, F_p\}$ is understood as function $F_i : C \rightarrow \mathbf{R}$, which transforms declaration of data type $c$ in $C$ to a real number. The $F_i$ is from an interval $< 0, 1 >$ and represents a ratio of occurrence of some source code artifact in particular data type declaration. The feature space consists of 40 features divided into several groups: expressions, statements, members and relations. The first two feature sets are connected with expressions and statements in the project code. For instance, a typical expression feature is a number of instantiations within a definition of a data type weighted by the total number of expressions of the same data type. A typical member feature is, for example, a number of public, non-static setters and getters in a selected data type weighted by the total number of methods of the same data type. Relation features depict the relationship of a data type with its surroundings; this kind of feature is, for instance, a logical value which is set to true if the data type uses its direct parent type as an attribute. Features used for the classification are more thoroughly described in [20].

### 2.3 Feature Space Optimization

A relevant question, in accordance with the curse of dimensionality, is whether 40 features are not too many and whether the number of features could be reduced without statistically significant deterioration of the classification system. To improve the classification performance in the first place, search for the optimal feature sub-model was performed. Since there is: $2^{40}$ possible sub-models, it is hardly possible to search systematically for the optimal one, therefore a heuristic search was employed. Genetic algorithm (GA) with tournament selection and a combination of crossover and mutation genetic operators was utilized for this purpose.

### 2.4 Method of Features Extraction

Classification data comprise source codes of Java programs. Each text file represents a compilation unit, which can hold zero or more public type declarations and zero or more package private and inner type declarations. Structure of compilation unit is depicted by Fig. 2. The unit of classification is a declaration of data type: class, interface, or enum.

A process of features extraction can be simply described as follows. For all $f \in D$, where $f$ is a source file, and $D$ is a set of input files (like directory), perform following steps:

1. read all content of $f$ and store it into memory into string variable $A$,
2. extract set of data types declarations $C(f)$ from $A$,
3. $\forall c_i \in C(f) \overline{x_1} = (F_1(c_i), F_2(c_i), F_3(c_i), \ldots, F_p(c_i))$, where $F_i$ is a feature.

The question was how to extract the data types declarations and implement the presented transformation Fj (ci). Inspired by XQuery, SQL (an etalon of query languages), and LINQ, a new internal homogeneous domain-specific language for the

definition of predictors, hosted in Groovy, was designed and implemented. The core of the language consists of a query statement similar to SQL:

**Listing 1**: An example of the query statement

```
select method, m: {m.name}, from: T, where:
      [modifiers: PUBLIC & not(STATIC)]
```

Apart from the selection of methods as in the example, types, attributes, expressions and statements can be selected by specifying an appropriate keyword after the select. Selection can be executed on a current node of AST, or the AST can be searched in a depth-first manner. When restricting string properties of nodes, regular expressions can be used. As Groovy is very flexible language, the presented example query expression is, in fact, a valid Groovy method invocation statement. The language does not have support for graph matching, so there is no statement for searching whole structures. On the other hand, the language supports logic statements such as exactly one exists, at least one exists, and one or more exist.

**Listing 2**: An example of the exists statement.

```
E one, method, in: T, where: [modifiers: PUBLIC &
    not(STATIC) & not(NATIVE), constructor: false]
```

To be able to reuse predefined queries across feature definitions, the language contains a possibility to define library functions.

**Listing 3**: An example of an user-defined function.

```
function 'isPureGetter', { m ->
  a = E one, method, in: m, where: [modifiers:
      PUBLIC & not(STATIC) & not(NATIVE),
      constructor: false]
  b = E two, statement, in: m /* block, return */
  c = E one, statement, in: m, where: [nodeType:
      RETURN_STATEMENT]
  st = select statement, from: m, where: [nodeType:
       RETURN_STATEMENT]
  d1 = E one, expression, in: st, where: [nodeType:
       FIELD_ACCESS | SUPER_FIELD_ACCESS], deep: true
  d2 = false
  d0 = E one, expression, in: st, where: [noteType:
       SIMPLE_NAME], deep: true

  /* ... */

  return a && b && c && (d1 || d2)
}
```

### 2.5 Used Classifiers

Processing of big data set is mainly the domain of statistics, thus it is not a surprise that many classification methods have roots in statistical principles. In this paper, a subset of classifiers with supervised learning is used. If classifier does not support natively multi-class problem, it is adapted using one of the following technique:

- One to All reduction (OtA) Single classifier per class is trained with samples of that class as positive samples and all other as negatives.
- One to One reduction (OtO) $K(K-1)/2$ binary classifiers for K-class problem are trained. Each classifier instance is trained using samples of a pair of classes from the original training set to distinguish these two classes.

At prediction time, a voting mechanism using balance criterion is applied to determine the sample resulting label. Classifiers were implemented using object-oriented programming principles. Simplified view on objects structure offers Fig. 1. Following classifiers were implemented and tested.

*k-Nearest Neighbour:* In the field of classification, k-Nearest Neighbour (*k*-NN) is one of the simplest, yet efficient and widely employed non-parametric algorithm [5, 10]. A point in a feature space is classified by the nearest neighbours in the training set. Algorithm can be modified by the selection of the k and a distance measure. Algorithm weakness is a case with a non-uniform probability distribution, in which samples of most present class bias the classification result. This can be, for example, solved by weighting the point by its distance to examined point [10]. Algorithm efficiency is decreasing with increasing problem dimension [5]. An advantage of *k*-NN is its computational complexity, which is very low. The algorithm can be further optimized by simplifying the distance computation (by data set compression or search tree application). Weighted and non-weighted variants were implemented and tested.

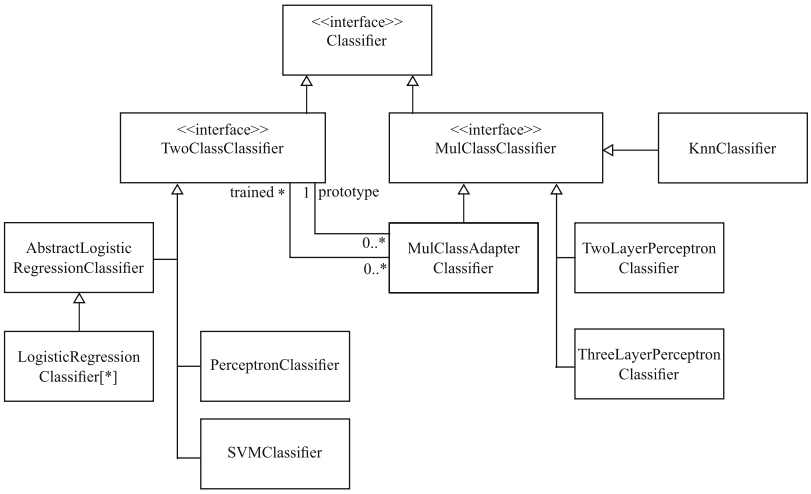



**Fig. 1.** Simplified diagram of classifiers implementation.

*Logistic Regression:* Logistic regression is linear probabilistic classifier. A probability that presented sample belongs to a particular class is modeled with the help of logistic function. Application of logistic regression is usually based on maximum like-hood optimization by various numeric methods [6]. Many variants of logistic regression with different methods of training were implemented and tested, including:

- Iterative Reweighted Least Squares (IRLS),
- Broyden-Fletcher-Goldfarb-Shanno (BFGS) [12, 27],
- Davidon-Fletcher-Powell (DFP) [12],
- Fletcher-Reeves Conjugate Gradient (FRCG) [12],
- Liu-Storey Conjugate Gradient (LSCG) [17], Polack-Ribiere Conjugate Gradient (PRCG) [12],
- Gradient Descent (GD).

*Perceptron:* Perceptron is the first and one of the simplest models of artificial neural networks introduced by F. Rosenblatt in 1957 [23]. In essence, it is a weighted linear network. Network weights, which can be initialized randomly in the beginning, are gradually modified during the training according to Perceptron algorithm [19]. Various linear and kernel derivatives were implemented:

- Classic Perceptron with Perceptron algorithm and STEP activation function [23],
- Voted Perceptron [13],
- Ballseptron (linear and kernel) [26],
- Passive Aggressive Perceptron [7],
- Kernel Perceptron,
- Projectron [22],
- Forgetron [8].

*Neural Network:* Multi-layered perceptron overcomes the condition of linear separability, which limits classic perceptron, by inserting one or more disjunctive layers with non- linear activation function between input and output layer. An output of one layer servers as an input of another layer. All neurons from one layer are usually connected with all neurons of the following layer. Well-known Neuroph library was used for implementation of ANN classifiers [25].

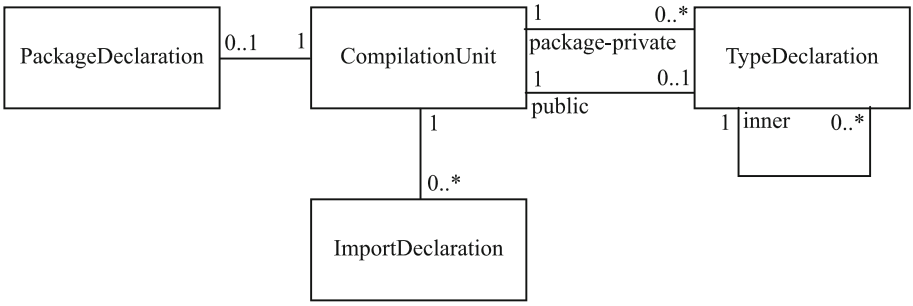



**Fig. 2.** Structure of compilation unit of Java platform languages

*Support Vector Machine:* The method of support vectors deals with non-linear separability by transferring the classification problem into a higher level dimension. The fundamental principle is to find such a hyperplane, which divides the problem space into two subspaces, where every subspace contains mainly samples from one class and maximizes distance of the closest samples (support vectors) from the border formed by this hyperplane at the same time [4]. For the implementation of SVM classifier, the JLibSVM library was used [28].

## 3 Results

Each classifier variant was tested using a 5-folded cross- validation. In this variant of cross-validation, the data set is divided into five folds with a uniform distribution of classes. In each iteration, four of the five folds are used for training and one for verification. This method simulates a behavior of classifiers on an independent data set and is more close to real-life classifiers deployment then common fitting. One thousand evaluations of cross-validation were carried out, and average error, and standard deviation were computed. The cross-validation was implemented with multi-thread support in order to be able to utilize nowadays multi-core CPUs. All computations were performed on a server with following configuration: 4 × CPU AMD Opteron 2200 MHz (4 cores), 16 GB RAM, CentOS 7, Java 1.7. Table 1 contains achieved results for best performing classifier variants.

As shows the Table 1, the best performing classifier was distance weighted *k*-NN ($k = 2$) with Manhattan (grid) distance measure, which achieved average error of 0.190 on full model and 0.167 on sub-model with 33 features. This simple multiclass classifier performed better than all the more binary classifiers with different adaptations on the many-class problem. The only other classifier that achieved accuracy higher than 0.8 was kernel perceptron, Ballseptron with RBF kernel and OtO adaptation in particular [26], with an average error of 0.196 on sub-space with 30 features.

The classification error is partially influenced by Worker and DAO classes, which are very variable in implementation and thus hard to identify. The majority of mistakes consists of misclassification of Worker and DAO classes. In general, the prediction error depends on ability of feature space to grasp the subtleties of particular patterns.

## 4 Discussion

In sum, over fifty classifier variants were tested during the experiment. For k-NN, various distance measures were tested: normal (1/2, 2, 3), Chebyshev, cosine, as well as various values of k. All k-NN classifiers preferred smaller values of k (<10). The important feature of implemented k-NN classifier is the weighting mechanism. Classifier performance without weighting according to distance was unsatisfactory. For logistic regression, some precautions have to be implemented, in order to achieve numerical stability. To penalize logistic regression classifier for too high weights values, L2 regularization was implemented and utilized during training. Search for weights optimizing the like-hood criterion was carried out using various iterative methods: Iterative

**Table 1.** Overall classifiers results on full feature space as well as on sub-model. Error of prediction e, standard deviation σ and number of utilized features #F are listed

| Classifier | Variant | Full model | | Sub-model | | |
|---|---|---|---|---|---|---|
| | | e | σ | e | σ | #F |
| k-NN | k = 2, Manhattan measure | 0.190 | 0.062 | 0.167 | 0.047 | 33 |
| Logistic Regression | FRCG/OtA | 0.241 | 0.048 | 0.218 | 0.066 | 33 |
| Logistic Regression | IRLS/OtO | 0.250 | 0.033 | 0.208 | 0.055 | 33 |
| Linear Perceptron | PRC/OtA | 0.336 | 0.052 | 0.302 | 0.037 | 32 |
| Kernel Perceptron | BALRBF/OtO | 0.239 | 0.070 | 0.196 | 0.046 | 30 |
| Neural Network | TANH/OtO, h = 10 | 0.218 | 0.040 | 0.214 | 0.040 | 35 |
| Neural Network | GAUS/OtO, h = 2 | 0.227 | 0.047 | 0.202 | 0.048 | 30 |
| Support Vector Machine | RBF/OtO | 0.227 | 0.060 | 0.213 | 0.045 | 35 |

Reweighted Least Squares (IRLS), Broyden-Fletcher-Goldfarb-Shanno (BFGS) [12, 27], Davidon-Fletcher-Powell (DFP) [12], Fletcher- Reeves Conjugate Gradient (FRCG) [12], Liu-Storey Conjugate Gradient (LSCG) [17], Polack-Ribiere Conjugate Gradient (PRCG) [12], and Gradient Descent (GD). Those methods were implemented using the Cognitive Foundry library [24]. Despite the classic perceptron from the 70s, more modern linear and kernel variants with various kernels (sigmoid, polynomial, RBF) were tested. As for ANN, two-layer (without a hidden layer, but with non-linear activation function) and three-layer perceptrons were tested with various activation functions. Support Vector Machine was tested with various kernels (linear, RBF, sigmoid, polynomial). Overall, experiments did not show that one method of adaptation is better than the other. Classifiers, in general, preferred settings that enabled them to generalize more. As to feature space reduction, all examined classifiers performed better on optimized sub-models, which leads to a discussion how should be the feature space constructed. In average, classifiers utilized 80% of original feature space.

As there are no reference data sets for pattern matching in source code, that could be used for results verification, the results conclusiveness is endangered. The authors are aware that without proper comparison with other methods of source code patterns recognition proposed method validity cannot be evaluated and authors are working towards such comparison.

## 5 Conclusion

The paper experimentally proved that classic statistical classifiers can be utilized in pattern recognition and classification in Java source code with success rate on presented patterns subset slightly exceeding 83% in cross-validation. In order to be able to apply such classifiers on source code, new feature space and method of features extraction was proposed and implemented. During the feature extraction process, the source code was translated into AST, which was further regarded as an object database. A new query language for the database was designed and implemented. Features were defined as queries to this object database.

The vision is, that with enhanced feature space and refined methods of classification, the source code perception can be substantially enhanced. This could be used by integrated development environments to assist further with coding, by analytical modeling tools and tools supporting the model- driven architecture to verify the implementation against a specification. Such methods can be also used in security to detect possible dangerous anomalies in source code repositories of open source projects.

Only supervised classifiers were tested thoroughly so far, but for other listed applications, unsupervised learning (clustering) can be more appropriate. Clustering methods like SOM (Kohonen Map), k-Means Clustering and Fuzzy k-Means clustering are scheduled for implementation and study. Moreover, a comparison with other methods of source code patterns recognition, as sub-graphs matching, is going to be carried out in order to evaluate viability of the proposed method.

**Acknowledgment.** This paper was supported by the grant SGS14/210/OHK4/3T/14.

## References

1. Antoniol, G., Fiutem, R., Cristoforetti, L.: Design pattern recovery in object-oriented software. Proc. IWPC **98**, 153–160 (1998). doi:10.1109/WPC.1998.693342
2. Antoniol, G., Fiutem, R., Cristoforetti, L.: Using metrics to identify design patterns in object-oriented software. In: Proceedings of Fifth International Software Metrics Symposium, pp. 23–34 (1998). doi:10.1109/MET-RIC.1998.731224
3. Bloom, A.: The Republic of Plato, 2nd edn. Basic Books, New York (1991). ISBN 978-0465069347
4. Cortes, C., Vapnik, V.: Support-vector networks. Mach. Learn. **20**(3), 273–297 (1995)
5. Cover, T., Hart, P.: Nearest neighbor pattern classification. IEEE Trans. Inf. Theor. **13**(1), 21–27 (1967). doi:10.1109/TIT.1967.1053964
6. Cox, D.R.: The regression analysis of binary sequences. J. Royal Stat. Soc. Ser. B (Methodological) **20**, 215–242 (1958)
7. Crammer, K., Dekel, O., Keshet, J., Shalev-Shwartz, S., Singer, Y.: Online passive-aggressive algorithms. J. Mach. Learn. Res. **7**, 551–585 (2006)
8. Dekel, O., Shalev-Shwartz, S., Singer, Y.: The forgetron: a kernel- based perceptron on a budget. SIAM J. Comput. **37**(5), 1342–1372 (2008)
9. Duda, R.O., Hart, P.E., Stork, D.G.: Pattern Classification. Wiley, New York (2000). ISBN 978-0471056690
10. Dudani, S.A.: The distance-weighted k-nearest-neighbor rule. IEEE Trans. Syst. Man Cybern. **SMC-6**(4), 325–327 (1976). doi:10.1109/TSMC.1976.5408784
11. Ferenc, R., Beszedes, A., Fueloep, L., Lele, J.: Design pattern mining enhanced by machine learning. In: Proceedings of International Conference on Software Maintenance (ICSM 2005), pp. 295–304. IEEE Computer Society (2005). doi:10.1109/ICSM.2005.40
12. Fletcher, R.: Practical Methods of Optimization. Wiley, New York (2000). ISBN 978-0471494638
13. Freund, Y., Schapire, R.E.: Large margin classification using the perceptron algorithm. Mach. Learn. **37**, 277–296 (1999). doi:10.1.1.48.8200

14. Gamma, E., Helm, R., Johnson, R., Vlissides, J.: Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley Professional, Boston (1994). ISBN 978-0201633610
15. Gueheneuc, Y.G., Sahraoui, H., Zaidi, F.: Fingerprinting design patterns. In: Proceedings of 11th Working Conference on Reverse Engineering (WCRE 04), pp. 172–181. IEEE Computer Society (2004). doi:10.1109/WCRE.2004.21
16. Lerthathairat, P., Prompoon, N.: An approach for source code classification to enhance maintainability. In: Eighth International Joint Conference on Computer Science and Software Engineering (JCSSE), pp. 319–324 (2011). doi:10.1109/JCSSE.2011.5930141
17. Liu, Y., Storey, C.: Efficient generalized conjugate gradient algorithms, part 1: theory. J. Optim. Theory Appl. **69**(1), 129–137 (1991)
18. Maiga, A., Ali, N., Bhattacharya, N., Sabane, A., Gueheneuc, Y., Antoniol, G., Aimeur, E.: Support vector machines for anti-pattern detection. In: Proceedings of the 27th IEEE/ACM International Conference on Automated Software Engineering (ASE), pp. 278–281 (2012). doi:10.1145/2351676.2351723
19. Minsky, M.L., Papert, S.A.: Perceptrons: An Introduction to Computational Geometry. The MIT Press, Cambridge (1987). ISBN 978- 0262631112
20. Mojzes, M., Rost, M., Smolka, J., Virius, M.: Feature space for statistical classification of Java source code patterns. In: 15th International Carpathian Control Conference (ICCC), pp. 357–361 (2014). doi:10.1109/CarpathianCC.2014.6843627
21. Object Management Group, OMG Unified Modeling Language (OMG UML) Superstructure, version 2.4.1 (2011)
22. Orabona, F., Keshet, J., Caputo, B.: Bounded kernel-based online learning. J. Mach. Learn. Res. **10**, 2643–2666 (2009). Rosenblatt, F.: The Perceptron: A Perceiving and Recognizing Automaton
23. Rosenblatt, F.: The perceptron: a perceiving and recognizing automaton, Technical report 85-460-1. Aeronautical Lab., Cornell Univ. (1957)
24. Sandia National Laboratories. Cognitive Foundry. http://foundry.sandia.gov/
25. Sevarac, Z., et al.: Neuroph – Java Neural Network Framework. http://neuroph.sourceforge.net
26. Shalev-Shwartz, S., Singer, Y.: A new perspective on an old perceptron algorithm. In: Auer, P., Meir, R. (eds.) COLT 2005. LNCS (LNAI), vol. 3559, pp. 264–278. Springer, Heidelberg (2005)
27. Shanno, D.F.: Conditioning of quasi-Newton methods for function minimization. Math. Comput. **24**(111), 647–656. doi:10.1090/S0025-5718-1970-0274029-X
28. Soergel, D.: jLibSVM – Efficient training of Sup- port Vector Machines in Java. https://github.com/davidsoergel/jlibsvm

# Contribution to Teaching Programming Based on "Object-First" Style at College of Polytechnics Jihlava

Marek Musil(✉) and Karel Richta

Department of Technical Studies, The College of Polytechnics Jihlava, Jihlava, Czech Republic
{marek.musil,karel.richta}@vspj.cz

**Abstract.** There are several different approaches to teaching programming, based on programming styles. A concept "first the object-oriented style, then the other one" known as "object-first" is currently being promoted by a number of technical colleges. The reasons originate from the practical area. Also the teaching of programming at the College of Polytechnics Jihlava (COPJ) is being switched from the "structural-then-object" style of programming to the "object-first" style. After the second run, the results achieved by students do not seem (very) good. This can be confirmed from the courses' feedback. It seems that the students are puzzled and their skills are poorer. Therefore, we decided to examine the results achieved by students and their opinion on the "object-first" style especially. This survey was carried out after the completion of the course with "object-first" teaching and at the beginning of the course with "structural" teaching. We are interested in skills in object-oriented programming and also in structural programming, but especially skills at the beginning of study at COPJ and the type of completed high school. We addressed our students attending the course of "structural programming". The third run of this teaching approach started. In this paper we introduce the first survey results. Even though the number of respondents is not big, the statistic results are significant within the College.

## 1 Introduction

There are several approaches to teaching programming, based on programming styles. The first choice is the established approach "first the structural style, then the object-oriented style" known as "structure-first" approach. The second offered teaching programming concept is the style "first the object-oriented style, then the other one" known as "object-first". This is currently being promoted for the programming teaching by a number of technical colleges.

The reasons mentioned come from the practical area. Current programming systems are typically developed in the object-oriented style. For this purpose, knowledge and skills mentioned hereafter are required. Students should understand the principle of objects including meaning of their properties and methods. In addition, they should be able to use existing objects (components) without profound structural principles knowledge (without profound knowledge of structural principles and structural programming). Deep structural programming is not required for these work positions. On


J. Janech et al. (eds.), *Proceedings of the 2015 Federated Conference on Software Development and Object Technologies*, Advances in Intelligent Systems and Computing 511,
DOI 10.1007/978-3-319-46535-7_17

the contrary, the opinion on the object-first is not unified. However, even the form of the course "object-fist" including exercises is in discussion.

We are interested in the student's opinion on "object-first" especially. Our next goal is understanding programming skills following in relation to the programming ability and skills gained at high school. A technical high school or a technical study programme at high-school as well as respondent's self- evaluation of programming skills are important for our intended inquiring.

## 2 Related Works

Empirical studies are presented in [2, 5, 7]. These reveal results for object-first and against structural programming as first. Bennedsen and Schulte [2] described three categories – using objects, creating classes and concepts. This option is justified in their article. The presented discussion on object-first leads to the new and innovative ways of teaching introductory programming. Johnson and Moses [7] performed an empirical study for the purpose of comparison of two students groups after a semester including programming teaching. While the first group studies objects and classes very early in the semester, the second group studies the basic programming structures. The exam result is clear. It indicates that students who take the object-first approach outperform those who take a structure-first approach [7]. It seems that the object-first approach outperforms the structure-first.

Briand et al. [3] proclaim a distinct lack of empirical evidence. They claim that object-oriented techniques are more popular as a result of opinion and anecdotal evidence only. Result of empirical evidence is omitted. A few of these not too convincing results are presented in their paper and mentioned below.

Firstly, Jones [6] confirms insufficiency of evidence. He identified several areas where a distinct lack of empirical evidence exists to support the assertions of gains in productivity and quality, reduction in defect potential and improvement in defect removal, and reuse of software components. [3] However, no empirical research proves the object-oriented development techniques as the option always providing the many benefits. Several are presented hereafter.

Basili et al. [1] introduces a positive result of their study of object-oriented techniques. This result provides significant benefits from reuse in terms of reduced defect density and rework as well as increased productivity [3]. For example, it could be shown in a development environment after introducing an object-oriented design method.

On the other hand, there are presented two negative results. Van Hillergersberg et al. [10] deliver the finding, that object-oriented concept is not easy to learn and use quickly. However, their evidence is not too convincing as regards the object-oriented design. Daly et al. [4] discovered the finding about inheritance depth and class hierarchies. Inheritance depth and conceptual entropy of class hierarchies can cause programmers difficulty maintaining object-oriented software.

Finally, Brian et al. [3] presents a controlled experiment performed with computer science students as subjects. They try to answer questions regarding the ease of understanding and modifying as two essential components of maintainability, including their impact on quality standards. Whereas object-oriented design documents

win/triumph quality standards based on Coad and Yourdon principles, they are more sensitive to poor design practices.

Their results are as follows:

1. Maintainers with little experience don't gain great benefit maintaining object-oriented designs over structured designs. (By using practical significance, statistical was not achieved.)
2. Adherence to 'good' object-oriented design principles will provide ease of understanding and modification for the resulting design when compared to an object-oriented design in which the principles have not been adhered to. (Used by statistical significance).
3. An object-oriented design which did not adhere to quality design principles is likely to cause more understanding and modification difficulties than an appropriate structured design. (The significant evidence).

Results mentioned above suggest the following. Switching developers proficient in structured techniques to object-oriented techniques may actually have negative effects on the design of the product until they become as proficient with the object-oriented techniques.

## 3 Materials and Methods

In conformity with re-accreditation of the study programs of Applied Computer Science and Computer Systems at the College of Polytechnics Jihlava (COPJ), which started from the academic year 2013/2014, the teaching of programming is being switched from the "structural, then object-oriented" style to the "object-first" style. The education concept, form of exercises and illustrated examples are presented in [8, 9]. The "object-first" course is marked as PRG1 - Programming 1 (the first semester) and the next course "structural programming" is marked as PRG2 - Programming 2 (the second semester). The second run has finished, and the moment for the evaluation of teaching using "object-first" style has come. The results achieved by students are not very convincing; their skills seem poorer. It seems that some students are puzzled in addressing the challenges during exercises. The reactions from further courses concerning programming may serve as feedback.

Our goal lies in investigation of programming ability and skills in students after finish PRG1 in different aspects. Especially, we would like to know the students' opinion on the object-first approach. The responses should be categorized into groups by programming skills before starting at the College, by completing a technical high school, algorithmization and programming-teaching at high school, self-evaluation of programming skills at the begin of the study at COPJ, rating in the course PRG1, etc. Therefore, these questions should be in our intended questionnaire.

We prepared a questionnaire witch questions categorized into 3 sections. The first section contains questions oriented on the student and their programming knowledge and skills at the beginning of the study at the College including questions oriented on the completed high school and programming exercise there. The second section deals with object-oriented programming and the last section deals with structural

programming. Based on that, we can carry out evaluation of some aspects mentioned above (such as completed technical high school, programming skills from the high school, self- evaluation of programming skills at the beginning of the study at COPJ, rating in the course of PRG1, etc.). The secondary goal is the level of programming knowledge before starting the study at COPJ and their changes after PRG1.

The questionnaire was anonymous and voluntary. We collected 37 questionnaire responses in total. This number is expected, because 37 students out of the 55 students registered in the course of PRG2 regularly visited exercises. All students filled in the questionnaire. We experienced a great success. The evaluation is presented in the next section.

## 4 Results and Discussion

### 4.1 Self-evaluation of Programming Skills at the Beginning of Study at COPJ

70.27% of students mention the completion of a technical high school. Roughly the same number of students (67.57%) acknowledge they have acquired programming ability and skills at the completed high school. Approximately half of the students confirm two aspects, algorithmization teaching (45.95%), and the programming teaching (56.76%) at high school. 62.16% of students mention a programming language that was used in the programming teaching at high school, namely the language C and C ++ at the most, then Pascal, Java, Visual Basic, but also JavaScript, PHP and CSS. Half of students admit, that they have gained programing skills by individual study.

The most students (72.9%) get rating worse than B in PRG1, rating C and D prevail. Every rating category is represented by 2 groups of students in ration 2:1 - students from the technical and non-technical high school, students with programming skills and without programming skills, etc. As for Q3a, Q4a, Q6a, Q7a and Q10a, the most frequent self- evaluating rating is 3, the average level of skills between the best and the worst. Only some students that have gained programming skills by individual study evaluate themselves by rating 1. However, some students get rating A in PRG1 (Fig. 1).

The questions for the chart 1 are as follows:

Q3 – Do you have programming ability, skills and knowledge from a technical high school?
Q4 – Did you study at a technical high school?
Q5 – Did you graduate from a course of ICT at high school?
Q6 – Did you learn algorithmization at high school?
Q7 – Did you learn programming at high school?
Q10 – Did you gain programming skills by yourself?

The questions for the chart 2 are as follows:

Q2 – What is your rating in PRG1?
Q9 – Self-evaluate your programming skills at the beginning of study at VSPJ.

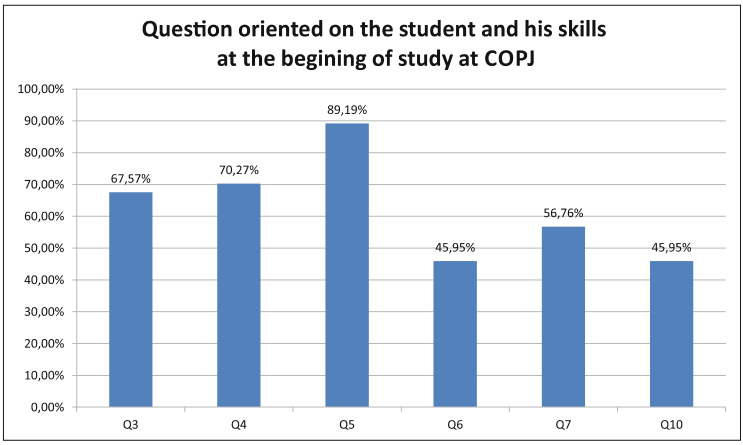



**Fig. 1.** Question oriented on the student and his skills in begin study at COPJ.

### 4.2 The Opinion on "Object-First"

As regards the opinion on "object-first", the most students (70.27%) prefer structural programming as first. Object-first is preferred only by some students having programming skills from the high school (13.51% of all students). 8% students have not decided, 8% students don't know. It includes the students having programming skills from the high school. The students that have no programming skills acquired at a high-school are definitely for "structure-first", several students of them are undecided.

While the most of students hold fast to the "structural-fist", only several students hold fast to the "object-first". While several students having programming skill from the high school don't know what to prefer, several students having no programming skills from the high school are undecided.

### 4.3 Ability and Skills of Object-Oriented Programming or Structural Programming

Object-oriented knowledge is not very clear in all students' categories. Nearly half of the responses (approximately 30%) are false. The responses to structural programming questions turn out a little better (Figs. 2, 3, 4, 5 and 6).

The questions for the chart 5 are as follows:

Q12 – What is a special method that is used for object creating?
Q13 – What is the name of a special method that is used when the object has been destroyed?
Q14 – Is their explicit definition necessary?
Q15 – Inheritance. What does it mean when object B is inherited from object A?
Q16 – Object B is inherited from object A. What are the names of these objects?
Q17 – Object B is inherited from object A. Who is the direct predecessor and the direct follower?

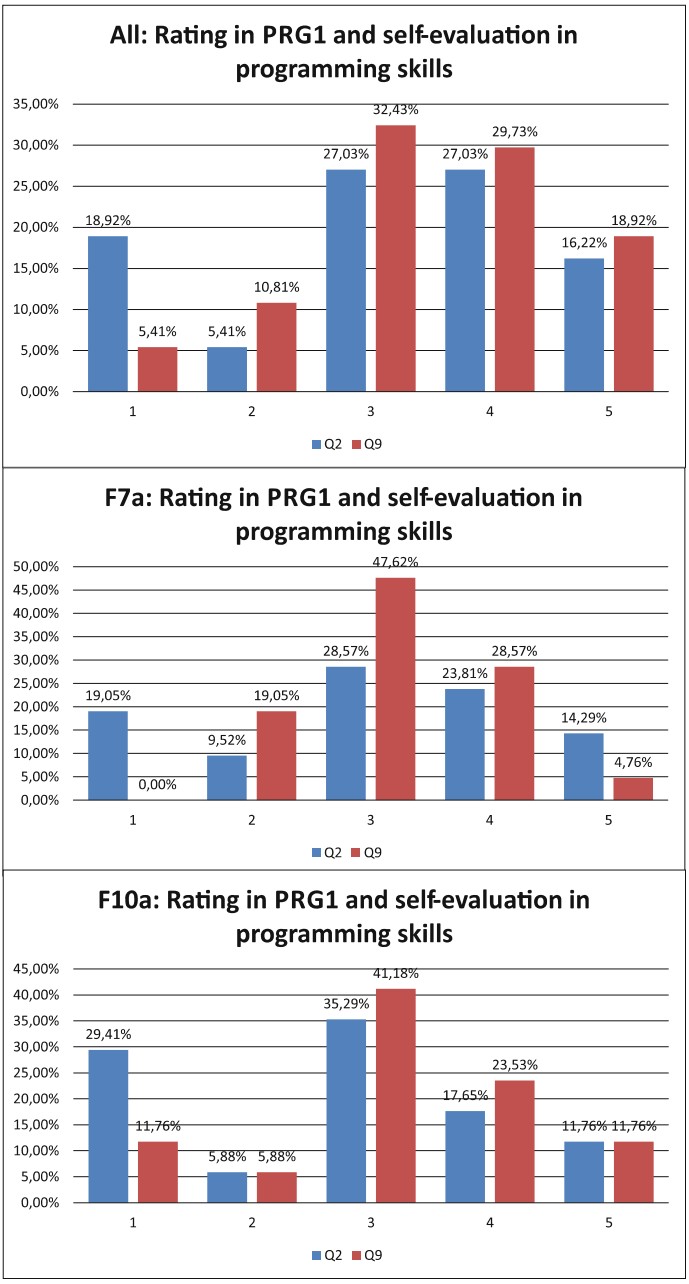



**Fig. 2.** Rating gained in PRG1 and self-evaluation filtered by some criterion. All - no filter (all respondents); F7a – title of programming in a course in the high-school; F10a – self-study of programming.

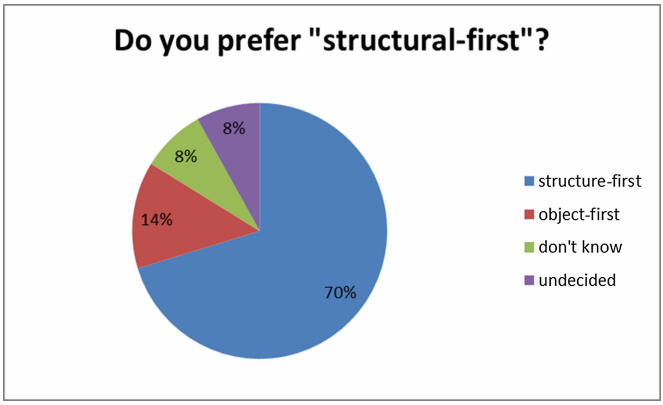



**Fig. 3.** The opinion on "object first".

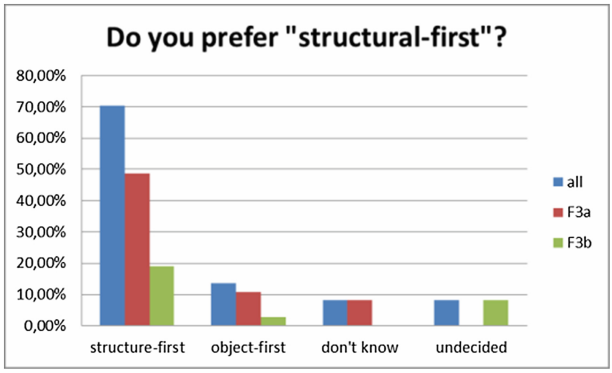



**Fig. 4.** The opinion on "object first" filtered by some criterion: all – no filter (all respondents); F3a – programming ability and –skills from the high-school; F3b – no programming ability or – skills from the high-school.

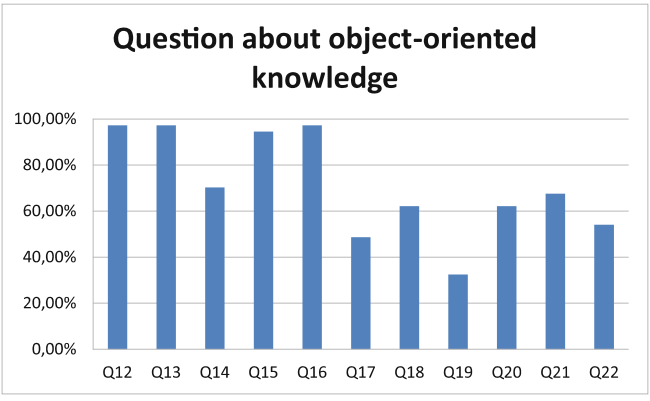



**Fig. 5.** Question about object-oriented knowledge.

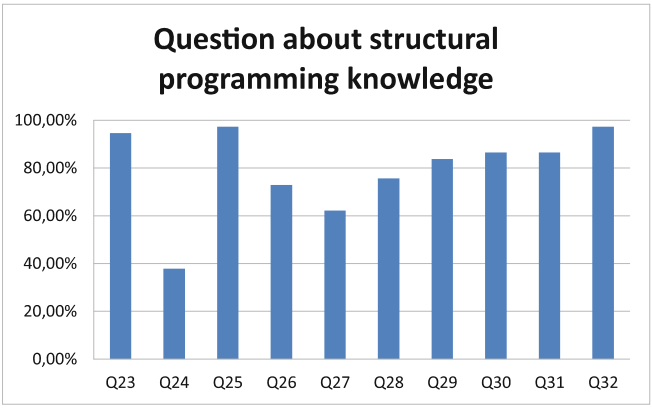



**Fig. 6.** Question about structural programming knowledge.

Q18 – What is the difference between "class" and "object"? Q19 – What does "state of an object" mean?
Q20 – An object owns a static variable/property. There are more objects of the same type. How many variables exist?
Q21 – What does "public interface of a class" mean?
Q22 – What does "a private item of an object" mean?

The questions for the chart 6 are as follows:

Q23 – There are commands. What is the value of x?

```
int x, y;
x = 5;
y = 4;
x = 2;
```

Q24 – There are commands. What is the value of x?

```
int x, *y;
x = 5;
y = &x;
*y = 2;
```

Q25 – There are commands. What is the value of x?

```
int x = 5, y;
if (x > 4)
        y = 4;
else
        y = -4;
```

Q26 – Is the meaning of commands sub-parts equivalent?

```
int x, y, z;

if (x > 5)
      if (y < 4)
         z = 1;
      else
         z = 2;
```

```
int x, y, z;

if (x > 5)
{
      if (y < 4)
            z = 1;
}
else
      z = 2;
```

Q27 – Thera are commands. What is the value of z?

```
int x, y, z;

if (x > y)
        z = x;
else
        z = y;
```

Q28 – Write commands for two values changing.

Q29 – There are commands. What is written in the standard output?

```
void tisk(int x)
{
        printf("%d", x);
}

int _tmain(int argc, _TCHAR* argv[])
{
        for (int i = 0; i < 3; i++)
                tisk(i);

        return 0;
}
```

Q30 – How many times is the cycle performed?

```
for (int i = 0; i < 3; i++)
        ;
```

Q31 – How many times is the cycle performed?

```
for (int i = 0; i <= 3; i++)
        ;
```

Q32 – How many times is the cycle performed?

```
for (int i = 3; i < 3; i++)
        ;
```

## 5 Conclusion

The paper describes our experience with the teaching of "object first" style. The result of the questionnaire survey was presented. We discussed students' opinion on "object first". The responses showed, that the students take a clearly negative stand to "object-first". They think that structural programming first is better. The students' categorization plays no significant role in terms of programming skills or knowledge and in terms of rating of PRG1.

Only individual study of programming added to a better or good self-evaluation. Probably, these students are sure of the programming.

Although the response number is small, the questionnaire brought up interesting questions. We would like to carry out the questionnaire survey at the begin of study at the College, then after the course of PRG1, and after the course of PRG2. This will provide the opportunity for a larger scale comparison and evaluation.

**Acknowledgment.** This paper was partially supported by the grant "Inovace předmětů Programování 1 a 2" of COPJ and also by the Avast Foundation.

## References

1. Basili, V.R., Briand, L.C., Melo, W.L.: How reuse influences productivity in object-oriented systems. Commun. ACM **39**(10), 104–116 (1996). doi:10.1145/236156.236184. Cited 3 Oct 2015
2. Bennedsen, J., Schulte, C.: What does "objects-first" mean?: an international study of teachers' perceptions of objects-first. In: Proceedings of the Seventh Baltic Sea Conference on Computing Education Research. Koli Calling, Finland (2007). http://crpit.com/abstracts/CRPITV88Bennedsen.html
3. Briand, L.C., Bunse, C., Daly, J.W., Differding, C.: Emp. Softw. Eng. **2**(3), 291–312 (1997). doi:10.1023/a:1009720117601. Cited 3 Oct 2015
. Daly, J., Brooks, A., Miller, J., Roper, M., Wood, M.: Evaluating inheritance depth on the maintainability of object-oriented software. Emp. Softw. Eng. **1**(2), 109–132 (1996). doi:10.1007/bf00368701. Cited 3 Oct 2015
5. Ehlert, A., Schulte, C.: Empirical comparison of objects-first and objects-later. In: Proceedings of the Fifth International Workshop on Computing Education Research Workshop, ICER 2009 (2009). doi:10.1145/1584322.1584326. Cited 28 Aug 2014
6. Jones, C.: Gaps in the object-oriented paradigm. IEEE Comput. **27**(6), 90–91 (1994). doi:10.1109/MC.1994.10064
7. Johnson, R.A., Moses, D.R.: Objects-first vs. structures-first approaches to OO programming education: an empirical study. In: Proceedings of the Allied Academies 2008, vyd. Reno, USA, pp. 244–248 (2008). http://www.researchgate.net/publication/242549890_objects-first_vs._structuresfirst_approaches_to_oo_programming_education_an_empirical_study. Cited 28 Aug 2014
8. Musil, M., Richta, K.: Approaches to teaching programming in the "Objects-first" style. Logos Polytechnikos **5**(4), 114–121 (2014). http://www.vspj.cz/soubory/download/id/3829. ISSN 1804-3682
9. Musil, M.: Přístupy k výuce programování "object-first". In: Informatika XXVII/2014: Sborník abstraktů z mezinárodní odborné pedagogicky zaměřené konference. MUSIL, Marek. 1 vydání, pp. 1–2. Ústav informatiky, Mendelova univerzita v Brně, Brno (2014). ISBN 978-80-7509-126-0
10. Van Hillegersberg, J., Kumar, K., Welke, R.: An empirical analysis of the performance and strategies of programmers new to object-oriented techniques. In: Psychology of Programming Interest Group: 7th Workshop (1995)

# The Survey of Current IPFRR Mechanisms

Jozef Papán(✉), Pavel Segeč, Peter Palúch, Ľudovít Mikuš, and Marek Moravčík

Faculty of Management Science and Informatics, University of Zilina, Zilina, Slovakia
{jozef.papan,pavel.segec,peter.paluch}@fri.uniza.sk

**Abstract.** The primary idea of the IP Fast Reroute (IPFRR) technology is to reduce the network recovery time after a link or router failure within an ISP network. The key feature of existing IPFRR mechanisms for reaching low recovery time is the usage of pre-computed alternative backup paths. These alternative backup paths have to be pre-calculated before a network failure will occur.

The calculation of the alternative backup path utilizes the specific information about destination networks, and thus most of existing IPFRR mechanisms are dependent on the distance-vector routing protocols (RIP, EIGRP). Other IPFRR mechanisms requires an additional information about the network topology, and therefore strongly depend on the usage of link-state routing protocols (OSPF, IS-IS). The paper is focusing on the analysis of existing IPFRR mechanisms and is identifying and presenting theirs primary problematic areas.

## 1 Introduction

Once a network link or a router failure occurs, a process of the network convergence begins. During the network convergence process affected networks are experiencing serious connectivity problems, where different destinations could become unreachable. This may cause many other difficulties, as for example for time critical services (such as Voice over IP).

To address these problems Fast Reroute (FRR) mechanisms has been developed. The first FRR mechanism was developed for the Multiprotocol Label Switching (MPLS) technology [1], and it uses explicit backup path. The research of IP Fast Reroute (IPFRR) mechanisms suitable for IP networks have begun immediately.

The strategic goal of all existing IPFRR mechanisms is to achieve a very short failure recovery time after the failure of a link or router is detected. Existing IPFRR mechanisms are actually able to reach the network recovery time up to 50 ms [2, 3].

The restoration of communication by means of a IPFRR mechanism is significantly faster as the process of the network convergence managed by means of a routing protocol itself (see Fig. 1) [4, 5].

Significant portion of the overall network recovery time is the time, which is required detect the failure itself. Therefore be able to achieve a fast network recovery time one of the main activities is fast failure detection and a need to minimize the time


J. Janech et al. (eds.), *Proceedings of the 2015 Federated Conference on Software Development and Object Technologies*, Advances in Intelligent Systems and Computing 511,
DOI 10.1007/978-3-319-46535-7_18

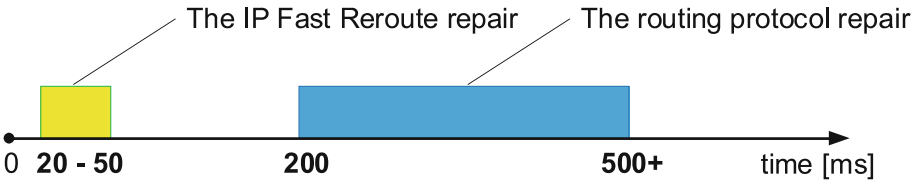



**Fig. 1.** Comparison of IPFRR and routing protocol repair time

of the main failure detection process [2, 3]. Actually we identify several types of failure detection mechanisms that could be classified as:

- Independent detection mechanisms, such as the Bidirectional Forwarding Detection protocol (BFD) [6].
- Physical detection mechanisms, such as loss of light, loss of carrier, increase in bit error rate, etc. [2].
- Routing protocol detection mechanisms, such as hello mechanisms [2].

IPFRR mechanisms may incorporate mentioned detection mechanisms, which reduce the failure detection time to few milliseconds (Fig. 2).

Installed alternative backup route, is valid only for a time, till the process of the network convergence will complete. During this period (IPFRR mechanism is still active), the routing protocol updates its necessary routing information, which was affected by the failure.

When the routing protocol converged and had updated its routing information, IPFRR mechanism is deactivated. This mean that the alternative IPFRR backup path is deleted from the routing table and the routing protocol overtakes control of the routing communication affected by the failure.

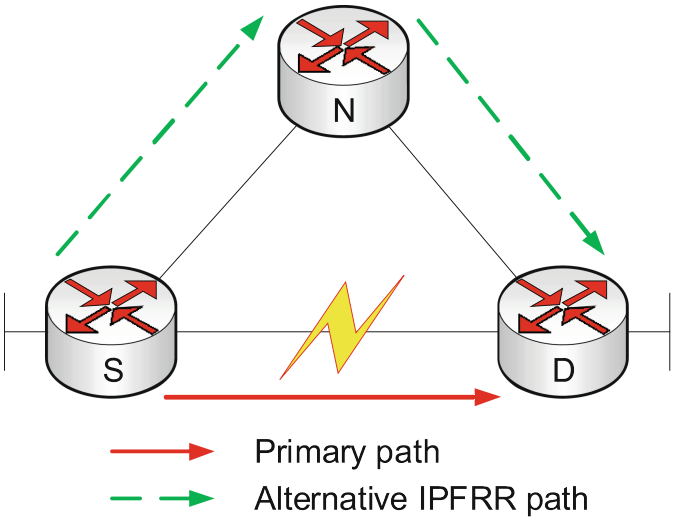



**Fig. 2.** The IP Fast Reroute mechanism

One possible solution, which limits duration of specific IPFRR mechanism and ensures a sufficient amount of time for correct completion of the process of the network convergence, is the utilization of special Hold Down timer [2].

The IPFRR terminology has specific terms for defining routers with a special purpose [2, 3]:

- Source router S is the router, which has detected a link or node failure and then started local IPFRR repair.
- Destination router D is the destination router of the original data flow.
- The Routers N, N1, N2 are routers, which are used as an alternative next-hops for alternative back-up path in the case of a link or router failure.

### 1.1 Types of Existing IPFRR Mechanisms

Each of existing IPFRR mechanisms have its own internal algorithm for calculating an alternative backup path. There actually exists several types of IPFRR mechanisms:

- Equal cost multi-paths (ECMP) mechanisms. ECMP alternative paths exists only when there are other paths through different links with equal total cost [7].
- Loop-free alternate paths (LFA) mechanisms. LFA uses an alternate directly connected router to send traffic around failed link. Alternative router has to be on a loop-free path [8, 9].
- Multi-hop repair path mechanism. These mechanisms use an alternate router that is more than one hop away from a router adjacent to the failure. From the remoter router packets will be send to their destination [2]. There are several multi hop mechanisms such as Remote LFA (RLFA) [10], Maximally Redundant Trees (MRT) [11], Not-Via Addresses [12], Multiple Routing Configurations (MRC) [13], etc.

### 1.2 The Bidirectional Forwarding Detection

Most often failure detection mechanism used in the IPFRR area is the BFD protocol. BFD for the fast link failure detection uses fast hello messages. Time needed by the BFD mechanism for the trustworthy failure detection begins at few milliseconds [6].

BFD has two operating modes, asynchronous and on demand mode. In both modes BFD provides an Echo function, in which one side may request its neighbor to loop back series of packets.

Main goals of the BFD mechanism are [2, 6]:

- Fast detection of failures.
- Protocol independent detection.
- Low-overhead efficient failure detection.
- Detection between interfaces, data links and forwarding engines themselves

The BFD mechanism is able to work with various IGPs (OSPF, IS-IS) and EGP (BGP) routing protocols, static routes and others (Fig. 3).

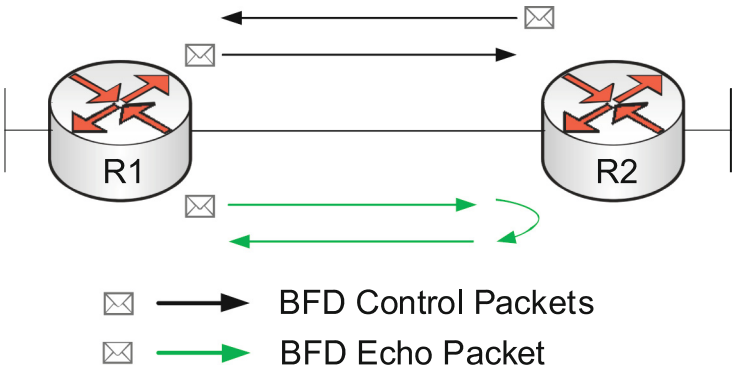



**Fig. 3.** The Bidirectional Forwarding Detection

### 1.3 The Repair Coverage

The Repair Coverage describes the efficiency of individual IPFRR mechanisms. If an IPFRR mechanism is able to repair all possible failures within a network, that specific IPFRR, mechanism provides 100% repair coverage. However Existing IPFRR mechanisms are usually tested against only one failure (link or router) in a time [2].

Currently, there does not exist exact method for measuring of the repair coverage. Therefore, testing of IPFRR mechanisms are made on randomly generated network topologies. The percentage of topologies in which the IPFRR mechanism was able to provide a workable backup path is taken as the mechanism repair coverage.

The repair coverage heavily depends on the network topology and the repair strategy of individual IPFRR mechanisms. If an IP FRR mechanism has a higher repair coverage, it is usually more complex.

In the present, there are several IPFRR mechanisms available, which differ in the principle of computation of alternative backup routes. Some of analyzed IPFRR mechanisms are described in Sect. 2.

## 2 Existing IPFRR Mechanisms

The section will present and describe functionalities of several important existing IPFRR mechanisms.

### 2.1 Remote LFA

Remote LFA is one of the most used IPFRR mechanism deployed in ISP networks. The Remote LFA mechanism [10] is an extended version of the basic LFA IPFRR mechanism [8, 9].

The basic LFA IPFRR mechanism provides good repair coverage in the topologies, which are “highly meshed”, i.e. in topologies with a good redundancy.

However, in some kind of topologies (for example ring), where there is no directly connected LFA router available, the Remote LFA can be used for providing the alternative backup path.

Elementary idea of the Remote LFA IPFRR mechanism is the utilization of tunnels to reach a remote router. The remote router is a loop free alternative for specific destination D. Note that Remote LFA router is not directly connected to the source router S.

Basic terminology of the Remote LFA IPFRR mechanism defines [10]:

- P-Space: The P-space of a router S, with respect to the protected link S-E, is the set of routers reachable from the router S using shortest paths without traversing via protected link S-E.
- Q-Space: The Q-space of a router E, with respect to the protected link S-E, is the set of routers, from which router E can be reached without traversing via protected link S-E.
- PQ Node: The PQ node (router), with respect to the protected link S-E, is a router, that is a member of both the P-space of the router S and the Q-space of the router E. The PQ router is the tunnel endpoint of an alternative path.
- Remote LFA: The use of the PQ node (router) rather than a directly connected neighbor of the repairing router S.

On the given topology (see Fig. 4) a link S-E is the protected link. The router S has detected the failure with its primary next-hop for the destination D and therefore starts local repair using the Remote LFA IPFRR mechanism.

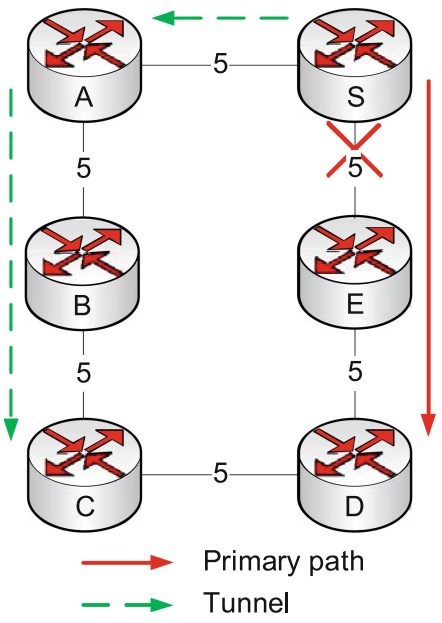



**Fig. 4.** The Remote LFA IPFRR mechanism

The shortest path from the router S to the destination D is via S-E-D. Shortest paths from the router S to other routers, which do not traverse via the protected link S-E, are S-A, S-A-B and S-A-B-C. Therefore, the P-Space of the router S are routers A, B and C.

Shortest paths from other routers to the router D, which do not traverse via the protected link S-E, are D-E and C-D-E. Therefore, the Q-Space of the router E are routers C and D.

The only router, which is in the P-Space of the router S and at the same time in the Q-Space of the router E is the router C.

Therefore, router C will be elected as a repair tunnel endpoint (PQ node). In other words, router C is Remote LFA of the router S for destination D.

According to the RFC7490 [10], Remote LFA is able to provide protection only for point-to-point links and only one failure within the network in the same time.

The Remote LFA IPFRR mechanism is primary used for a link-protection. When the router failure occurs, the Remote LFA IPFRR mechanism may create micro-loops. In the topology (see Fig. 4), when the router E fail, routing micro-loop may occur. In that case, the router D will send encapsulated communication to the router S and the router S will send encapsulated communication to the router D.

Classic LFA and Remote LFA depend on the metric of individual links. When there is not a router in the P-Space and the Q-Space, the repair with the Remote LFA IPFRR mechanism will not be possible. Remote LFA is an enhanced version of classic LFA, therefore Remote LFA can provide higher repair coverage than classic LFA [8].

### 2.2 Multiple Routing Configurations (MRC)

The MRC IPFRR mechanism utilizes the principle of multiple routing tables used by a router. Every router in the network calculates multiple alternative routing tables for specific failures, which can occur within the network [13].

On the following picture (see Fig. 5) there are two types of specific network failures. The first failure is the failure of the router A and the second is the router B failure.

In the first case (failure of the router A), router S will use the routing table, in which router A is ignored from routing calculation. In the second case (failure of the router B), the source router S will use the routing table, in which the router B is ignored from routing in the network.

Other routers inside of the network have to know which version of the routing table to use. Therefore router S puts the information on used version of routing table into a packet, i.e. it modifies an original packet. According specific modification of the original packet other routers will know which routing table they will use for the routing.

Router S may put the information in the Differentiated Services Code Point (DSCP) field of the IPv4 protocol and in a case of IPv6 – it use an extension header.

The MRC IPFRR mechanism is able to provide repair coverage reaching close to 100%. That means, that the MRC can repair all possible failures within the network. Multiple routing configurations allow efficient control over the alternative path, which means, network administrator can create custom routing scheme of the ISP network.

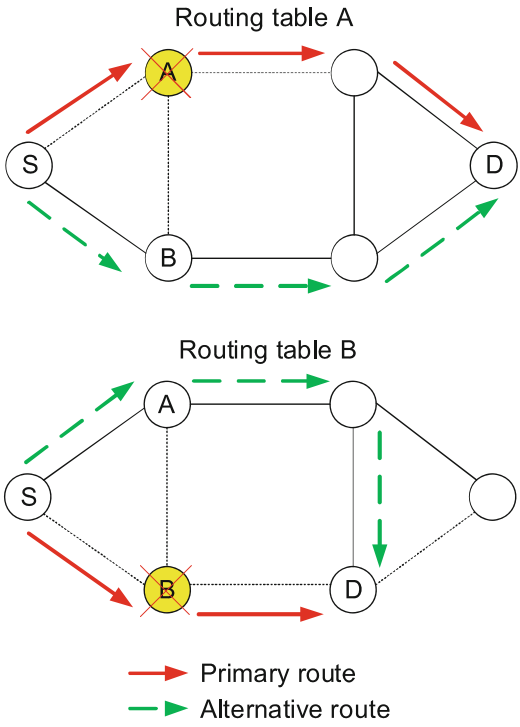



**Fig. 5.** The multiple routing configurations IPFRR mechanism

Main issue of the MRC IPFRR mechanism is the number of used alternative routing configurations. If we consider every routing table for every possible link or neighbor router failure, the number of alternative routing tables can be high. Actual research focuses how to reduce the number of routing configurations [13].

The MRC IPFRR mechanism is actually not implemented in routers of Cisco Systems neither Juniper Networks.

### 2.3 Not-via Addresses

The Not-Via Addresses IPFRR mechanism utilizes for detour of failed link/router the encapsulation of original packet into a packet with specific destination address, which is called Not-Via. On the following topology (see Fig. 6) the router S wants to send s packet to the specific destination D. The shortest path from the source router S to the destination D is via routers P and B [12].

When the router S detect the failure with the primary next-hop router P, it encapsulates packets to the address Bp. The Bp address carries the information, that the destination router for detour is the router B and the router P is failed router.

Other routers in the network will recognize specific Not-Via Address, and therefore router P will be avoided during the routing process. When the router B receives packet

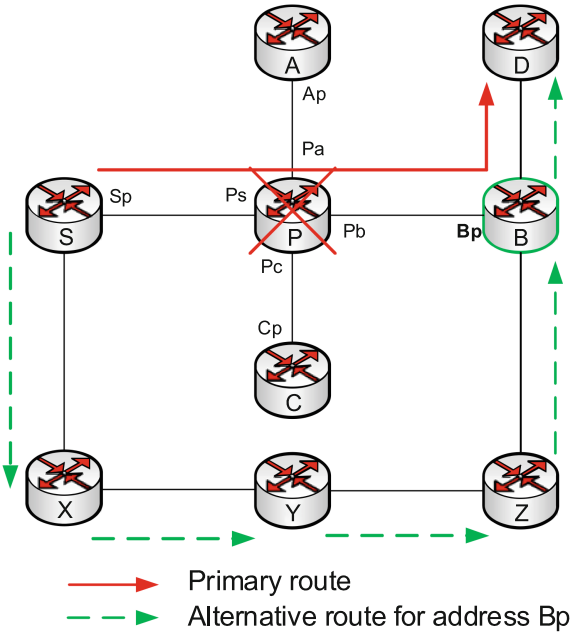



**Fig. 6.** The Not-Via Addresses IPFRR mechanism

with destination address Bp, decapsulates it and forwards packet to the destination router D.

The Not-Via Addresses IPFRR mechanism requires, that all routers on the alternative backup path must have calculated route for the Not-Via address Bp. This route can be calculated using a SPF algorithm by ignoring the router P from the routing scheme.

The Not-Via Address IPFRR mechanism provides 100% repair coverage. The problem of this IPFRR mechanism is the number of Not-Via addresses, which rapidly increases with number of routers in the network. Therefore, current research in this area focuses on the reduction of not-Via addresses quantity [12].

These addresses are computed in advance, after initialization of the Not-Via Address mechanism. Generally, Not-Via Address has high computational and implementation complexity. Currently, the IPFRR mechanism is not implemented in any devices of the Cisco Systems or Juniper Networks.

## 3 Analysis of Existing IPFRR Mechanisms

Following our research and the analysis of mentioned IPFRR solutions, some of mechanisms can provide the 100% repair coverage (MRT [11], MRC [13], Not-Via Addresses [12]), but at the cost of higher complexity of internal algorithm, which calculates alternative path.

**Table 1.** The overview of the existing IPFRR mechanisms

| IPFRR mechanism | Pre-computing | Dependence on routing protocols | The packet modification | 100% repair coverage |
|---|---|---|---|---|
| LFA | Yes | Yes | No | No |
| ECMP FRR | Yes | Yes | No | No |
| Remote LFA | Yes | Yes | Yes | No |
| MoFRR | Yes | Yes | No | No |
| Not-Via Addresses | Yes | Yes | Yes | Yes |
| MRC | Yes | Yes | Yes | Yes |
| MRT | Yes | Yes | Yes | Yes |

Therefore, only a few of existing IPFRR mechanisms can be easily implemented into existing architecture of present router OSs. Therefore most of them are implemented and tested inside of software simulators [2].

The most important information on analyzed IPFRR mechanisms [2–5, 7–14] are described in following Table 1.

LFA, Remote LFA and ECMP IPFRR mechanisms are most used solutions deployed in commercial sphere. These IPFRR mechanisms are implemented in routers of companies like Cisco Systems and Juniper Networks.

Following subsections will identify primary problematic areas of existing IPFRR mechanisms.

### 3.1 The Dependence on the Routing Protocols

During the analysis we identified, that all of IPFRR mechanisms are dependent on routing protocols used, which means, that for their correct operation they need an information provided by routing protocols [2].

Some of them for calculations of the alternative path requires topology based information. Therefore, these IPFRR mechanisms requires functionality of link-state routing protocols, and they are dependent on the link-state routing protocols (MRT, MRC, Remote LFA, Not-Via Addresses).

### 3.2 The Calculation of the Alternative Path

Other common feature of all analyzed IPFRR mechanisms is the fact, that the alternative backup path have to be calculated in advance, i.e. pre-prepared before the failure within the network occurs (pre-computing) [2, 3]. And then the calculation of the alternative paths is main issue of existing IPFRR mechanisms. With growing number of routers inside of an ISP network grows number of necessary calculations of alternative backup paths. These calculations must be repeated every network topology change.

Calculations of alternative backup paths are usually executed as a process with low resource priority and they are usually executed during idle CPU periods of a router.

According our research we have identified that the calculation of the alternative backup path (pre-computing) is the second issue of existing IPFRR mechanisms. Table 1 shows, that all of analyzed IPFRR mechanisms are based on the principle of pre-computation.

This problematic area brings a new research question: Is it possible to develop a new IPFRR mechanism, which will not explicitly calculates an alternative path without internal algorithm?

### 3.3 Modification of Traffic

Important characteristic of current IPFRR mechanisms is fast failure detection and the way how they inform other routers affected by the failure [2].

IPFRR mechanisms the failure information spread by modifying bites of the IP header, either encapsulating original packets into a new one with new IP header or by the income interface, with which the packet was received.

The modification of packets brings problems with compatibility within the ISP network (change of MTU) [2, 3].

### 3.4 Efficiency of IPFRR Mechanisms

Scientific papers demonstrates dissimilar results of the repair coverage of current IPFRR mechanisms ([2, 3]). According these results, the primary problem is, that there is no common technique exactly defined for measuring the repair coverage of existing IPFRR mechanisms. That means, the conditions under which each of IPFRR mechanisms were tested, had not been the same. Therefore, the repair coverage of individual IPFRR mechanisms cannot be directly compared.

Nowadays, there exists RFC 6894 [15], which defines the methodology for testing of the MPLS fast reroute mechanism.

The next important article dealing with testing of IPFRR mechanisms is RFC 6414 [16]. The document defines basic terminology for testing of MPLS-FRR mechanisms.

RFC 6894 [15] also defines testing methods and the reference topology for measuring time of the fast network recovery. However, the document does not include methodology for testing the repair coverage itself.

## 4 Conclusion and Future Work

Existing IPFRR mechanisms are very important for present IP networks. Customers, who use real-time services require reliable and protected network connections and is up to ISP providers meet these requirements.

The problem is, that most of existing IPFRR mechanisms have not been implemented in real ISP networks yet. The implementation of existing IPFRR mechanisms brings many problems with required modification of existing ISP networks. It is

certain, that in the close future will ISP providers think about the implementation of IPFRR solutions to their network architectures. Some of existing IPFRR mechanisms have found place in commercial companies such as Cisco Systems or Juniper Networks (ECMP FRR, LFA, Remote LFA).

According the analysis and research all of existing IPFRR mechanisms are dependent on pre-computation of the alternative backup path. In other words, IPFRR mechanisms pre-calculate alternative back path via an internal algorithm. This principle causes unwanted effects such as load of router CPU or the dependence on routing protocols.

Some of IPFRR mechanisms requires topology information, therefore they are dependent on link-state routing protocols. In such case, when the IPFRR mechanism cannot find the alternative backup path, packets are dropped.

The repair coverage of existing IPFRR mechanisms defines the efficiency of the individual IPFRR mechanism. Currently, there is no exact method for benchmarking neither for measuring of the repair coverage. The problem is, that tests of IPFRR mechanisms show different, often incomparable results. Therefore is necessary to develop exact methods suitable for objective comparing of IPFRR mechanisms.

The research shows, that existing IPFRR mechanisms is able to provide sufficient effectiveness for finding the alternative backup path (repair coverage). The problem is, that existing IPFRR mechanisms have complicated internal algorithm for calculating alternative backup path.

Therefore, in the close future we will focus on finding a solution for these negative characteristics. One of the possible way how to eliminate problematic areas of the current IPFRR solutions is multicast technology [17], which is not being used in IPFRR mechanisms yet. The most used multicast protocol in current ISP is Protocol Independent Multicast (PIM). This protocol can work in sparse mode (PIM-SM) [18] or dense mode (PIM-DM) [19].

The PIM protocol in the dense mode (PIM-DM) at the beginning of the multicast transmission sends multicast packets to all PIM enabled routers in the network. We used this specific behavior of the PIM-DM protocol in IPFRR for developing a new IPFRR mechanism called M-REP (Multicast Repair). In the future we will present this new M-REP IPFRR mechanism, which will solve the primary problematic areas of the current IPFRR mechanisms.

**Acknowledgments.** This paper is the outcome of the project "Quality education by supporting innovative forms, quality research and international cooperation – a successful graduate for practice", ITMS code 26110230090 supported by the Education Operational Program funded by the European Social Fund.

## References

1. Pan, A., Swallow, G., Atlas, A.: Fast Reroute Extensions to RSVP-TE for LSP Tunnels, RFC 4090, Network Working Group, pp 3–15 (2005)
2. Shand, M., Bryant, S.: IP Fast Reroute Framework, RFC 5714, Internet Engineering Task Force, Informational, pp. 5–7 (2010). ISSN: 2070-1721

3. Gjoka, M., Ram, V., Yang, X.: Evaluation of IP Fast Reroute proposals. In: 2nd International Conference, COMSWARE 2007, pp. 1–8 (2007)
4. Hassan, A.T.: Evaluation of fast reroute mechanisms in broadband networks, p. 1. Master of Electrical and Computer Engineering, University of Ottawa (2010)
5. Antonakopoulos, S., Bejerano, Y., Koppol, P.: A Simple IP Fast Reroute Scheme for Full Coverage, p. 1. BellLabs, Murray Hill (2012)
6. Katz, D., Ward, D.: Bidirectional Forwarding Detection (BFD). Request for Comments: 5880, Standards Track, IETF, pp. 1–50 (2010). ISSN: 2070-1721
7. Hopps, C.: Analysis of an Equal-Cost Multi-Path Algorithm. RFC 2992, Informational, Network Working Group, pp. 1–5 (2000)
8. Atlas, A., Zinin, A. (eds.): Basic Specification for IP Fast Reroute: Loop-Free Alternates, RFC 5286, Standards Track, Network Working Group, pp. 3–5 (2008)
9. Filsfils, C., Francois, P., Shand, M., Decraene, B., Uttaro, J., Leymann, N., Horneffer, M.: Loop-Free Alternate (LFA) Applicability in Service Provider (SP) Networks, RFC 6571, IETF, pp. 3–7 (2012). ISSN: 2070-1721
10. Bryant, S., Filsfils, C., Previdi, S., Shand, M., So, N.: Remote Loop-Free Alternate (LFA) Fast Re-Route (FRR). Network Working Group, RFC 7490, pp. 3–18 (2015)
11. Atlas, A., Kebler, R., Bowers, C., Enyedi, G., Csaszar, A., Tantsura, J., White, R.: An Architecture for IP/LDP Fast-Reroute Using Maximally Redundant Trees. Routing Area Working Group, Internet-Draft, pp. 3–33 (2015)
12. Bryant, S., Previdi, S., Shand, M.: A Framework for IP and MPLS Fast Reroute Using Not-Via Addresses, RFC 6981, Internet Engineering Task Force (IETF), pp. 4–25 (2013). ISSN: 2070-1721
13. Kvalbein, A., Hansen, A.F., Čičic, T., Gjessing, S., Lysne, O.: Multiple routing configurations for fast IP network recovery. IEEE/ACM Trans. Netw. **17**(2), 1–4 (2009). IEEE
14. Lor, S.S., Rio, M.: Enhancing Repair Coverage of Loop-Free Alternates, p. 3. University College London (2010)
15. Papneja, R., Vapiwala, S., Karthik, J., Poretsky, S., Rao, S., Le Roux, J.L.: Methodology for Benchmarking MPLS Traffic Engineered (MPLS-TE) Fast Reroute Protection, RFC 6894, IETF, pp. 3–27 (2013). ISSN: 2070-1721
16. Poretsky, S., Papneja, R., Karthik, J., Vapiwala, S.: Benchmarking Terminology for Protection Performance, RFC 6414, IETF, pp. 4–25 (2011). ISSN: 2070-1721
17. Deering, S.: Host Extensions for IP Multicasting, RFC 1112, Network Working Group, pp. 1–5 (1989)
18. Fenner, B., Handley, M., Kouvelas, I., Holbrook, H.: Protocol Independent Multicast - Sparse Mode (PIM-SM): Protocol Specification (Revised), RFC 4601, Standards Track, Network Working Group, pp. 1–146 (2006)
19. Adams, A., Nicholas, J., Siadak, W.: Protocol Independent Multicast - Dense Mode (PIM-DM): Protocol Specification (Revised), RFC 3973, Network Working Group, pp. 4–10 (2010)

# Synthesis of Low-Power Embedded Software Using Developmental Genetic Programming

Stanisław Deniziak, Leszek Ciopinski(✉), and Grzegorz Pawinski

Department of Computer Science,
Kielce University of Technology, Kielce, Poland
{s.deniziak, l.ciopinski, g.pawinski}@tu.kielce.pl

**Abstract.** A method of synthesis of software for low-power real-time embedded systems is presented in this paper. A function of the system is specified in the form of the task graph, then it is implemented using embedded processors with low-power and high-performance cores. The power consumption is minimized using the developmental genetic programming. The optimization is based on finding the makespan, satisfying all real-time constraints, for which the power consumption is as low as possible. We present experimental results, obtained for real-life examples and for some standard benchmarks. The results show that our method gives better solutions than makespans obtained using existing methods.

## 1 Introduction

Embedded systems are dedicated computer-based systems that are highly optimized for a given application. Optimization concerns hardware as well as software components. Since, finding the optimal hardware/software architecture implementing the target function is a very complex task, the design of embedded system should be assisted by efficient methods and tools. During the last two decades a lot of hardware/software co-design methods were developed. But the progress in the semiconductor technologies, more advanced embedded processors, increasing requirements, create new challenges in the optimization of embedded systems.

Techniques of minimizing the power consumption in computer systems, including embedded systems, are used for many years. Their goal is to reduce operating costs, protection of the environment, elimination of the problem of cooling, longer working hours without recharging the battery. Also, many methods of designing computer systems, in addition to minimizing costs and optimizing performance, minimize power consumption. This is consistent with the increasingly popular concept of green computing. It was observed that power demands are increasing rapidly, yet battery capacity cannot keep up [1].

Usually, embedded systems are real-time systems, in which time constraints are imposed on some tasks. Thus, during optimization of power consumption all time requirements should be satisfied. Performance and power consumption are related to each other. Generally, a high performance systems consume significantly more energy than low performance ones. Therefore, the optimization of real-time embedded systems should consider the trade-off between power consumption, performance, cost, etc.


J. Janech et al. (eds.), *Proceedings of the 2015 Federated Conference on Software Development and Object Technologies*, Advances in Intelligent Systems and Computing 511,
DOI 10.1007/978-3-319-46535-7_19

Advanced embedded systems are generally distributed systems. During designing such systems, a specification, presented as a set of communicating tasks, is mapped onto a dedicated architecture consisting of processors, memories and specialized hardware components. An alternative to heterogeneous multiprocessor architecture is a system based on multi-core processor. Then not only the design process reduces to a scheduling problem, but also can take advantage of advanced power management techniques such as DVFS (Dynamic Voltage and Frequency Scaling) or big.LITTLE [2], to minimize power consumption. Methods of optimizing embedded systems generally assume that some of the properties of tasks, such as execution time, power consumption, cost, are known. Only then it is possible to optimize these features of the system during the synthesis.

In this paper the novel method for synthesis of the power-aware software for real-time embedded systems is presented. Our method involves the specification of the system in the form of a graph of tasks. In addition the method is dedicated to the ARM big.LITTLE technology, but it can be generalized also to DVFS. The schedule is generated automatically and optimized using the developmental genetic programming (DGP) approach. Our goal of optimization is the minimization of power consumption by moving some tasks to low-power cores (LPCs), while critical tasks are assigned to high-performance ones (HPCs) to satisfy all time constraints. The advantages of using our methodology are shown on examples.

The paper is organized as follows. The related work is presented in the next section. The idea of the developmental genetic programming and a comparison to other genetic approaches are described in the Sect. 3. In Sect. 4 the method of software synthesis is presented. In Sect. 5 an example is shown and experimental results are described. At the end, a summary of the main conclusions and directions of further research are given.

## 2 Previous Works

The problem of design of low-power embedded systems has attracted researchers for many years. One direction of these research is finding the low-power architecture by optimizing the allocation of resources and task assignment according to the power consumption. COSYN-LP [3] is one of the first method which synthesizes the low-power hardware-software architectures of real-time embedded systems. SLOPES [4] minimizes the power consumption in systems implemented using dynamically reconfigurable FPGAs. Optimization is performed using genetic algorithm. The overview of some power aware codesign methods is also presented in [5]. Another direction of research concerning the design for low-power is to develop methodologies that takes into consideration dynamic reduction of the power consumption during runtime. Dynamic Power Management [3] tries to assign optimal power saving states. Other methods reduces power consumption by efficiently using voltage scale processors [6]. All above methods are based on power-aware scheduling, like YDS [7].

When using multi-core processors, the synthesis of embedded system specified in the form of a task graph resolves to the problem of optimal scheduling. This problem is known in literature as RCPSP (Resource-Constrained Project Scheduling Problem) [8].

It has been shown that RCPSP problem is NP-complete. Therefore, only heuristic methods may be used in the practice to optimize real-life systems. Very good results are obtained by genetic methods [9–11].

In the 2011 the big.LITTLE architecture for minimizing the power consumption in embedded systems was proposed by ARM Holdings. The goal of this technique is to enable building multi-core systems that will better adjust to dynamic computing needs. Depending on the scheduler, there are three different methods of applying big.LITTLE in multicore systems [12]:

- cluster switching: in this approach low-power cores are arranged into "little cluster", while high performance cores form "big cluster". The scheduler can see only one cluster at a time. When the load on the system changes from low to high then the system switches to the "big cluster", otherwise the "little cluster" is used. During cluster switching all relevant data are passes through the common L2 cache. Unused cluster is powered off.
- CPU migration (in-kernel switcher): in this approach, low-power and high-performance cores are paired, forming virtual cores (VC). At a time only one core inside VC is used while the other is switched off. When the load on the virtual core changes, the incoming core is powered up, running state is transferred, the outgoing is shut down, and processing continues on the new core.
- global task scheduling (heterogeneous multi-processing – HMP): it is the most powerful model of big.LITTLE architecture. In this model all cores are available at the same time. Tasks with high computational intensity can be allocated to the high performance cores while tasks with less computational intensity, can be executed by the low power cores.

The big.LITTLE technology can be used additionally to the DVFS technique and requires efficient scheduling policy. In [13] we presented the method of synthesis of adaptive schedulers for real-time embedded systems based on big.LITTLE architecture. The goal of the optimization was to find the makespan which will have the best self-adaptivity capabilities. According to our best knowledge there is no static scheduling methods that minimizes the power consumption in big.LITTLE-based real-time embedded systems.

## 3 Developmental Genetic Programming

A lot of real-life optimization problems cannot be efficiently solved using exact methods. In such cases heuristic methods, usually giving sub-optimal solutions, are more applicable. One of the most efficient method used for global optimization of complex problems is Genetic Algorithm (GA) [14]. Methods based on GA are resistant to getting stuck in a local extreme of optimized functions. Therefore, they often find the optimal solution or solution better than found by other methods. Efficient GA-based methods were developed for solving many RCPSP problems as well as for multi-objective optimization of distributed real-time systems [15].

However, in case of hard constrained problems, the effectiveness of GAs is not so high. Example showing the reason of such situation is presented on Fig. 1a. This figure

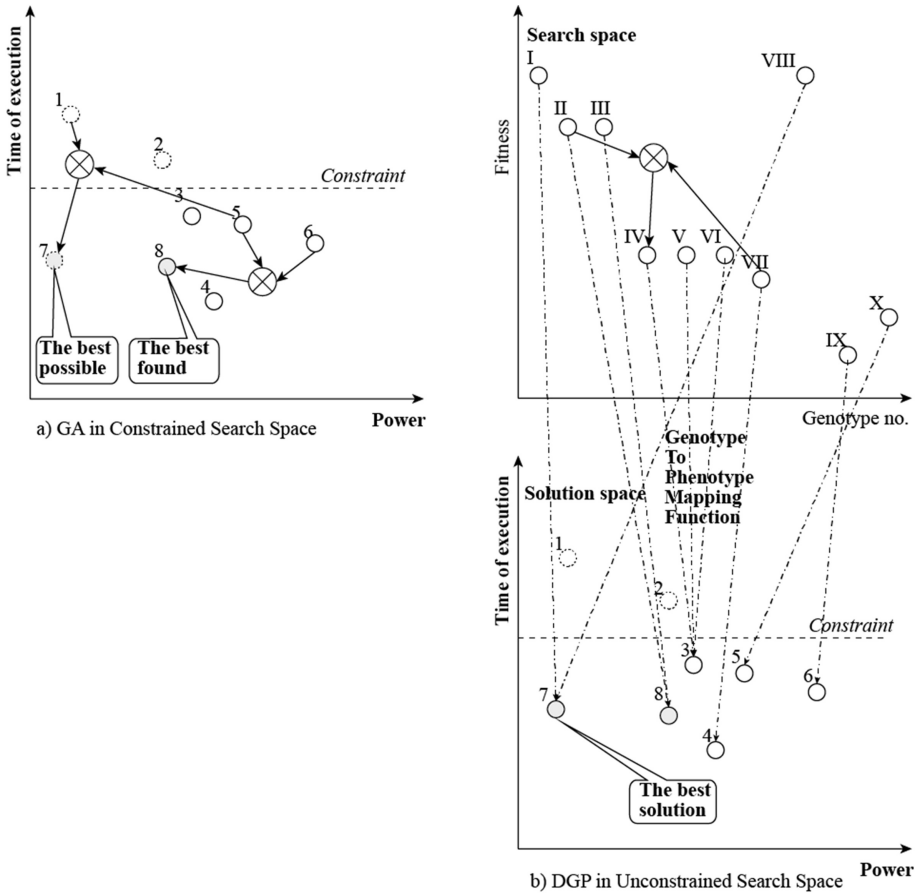



**Fig. 1.** Different search spaces in GA and in DGP

presents the solution space of real-time system, for which the power consumption is minimized. Only solutions satisfying time constraints i.e. individuals below the dashed line, are valid. In the GA approach invalid solutions are eliminated by using constrained genetic operators. It is necessary to avoid the situation where the GA will find the optimal but invalid solution. For example, genes 1 and 2 will not be considered during evolution. But sometimes, the crossover or mutation of invalid solutions may produce good results. For example, the crossover of individuals 1 and 5 may produce the optimal solution i.e. gene 7. But, due to the above limitation, the GA will be only able to find the solution 8, as a result of crossover of valid individuals 5 and 6.

Above problem does not exists in the Developmental Genetic Programming (DGP) [16]. DGP is an extension of the GA, where the solution space (phenotypes) and the search space (genotypes) are separated. Genotype describes the method of building the solution. Genotypes are evolved without any restrictions, then during the developmental stage genotypes are mapped onto valid phenotypes, i.e. solutions. Crossover,

mutation and reproduction are not restricted, i.e. there are no genotypes eliminated from the evolution. This is presented on Fig. 1b, since there is no limitations in the evolution, it is possible to obtain genotype I or VIII, that construct the optimal solution. It should be noticed that there is no genotype which is mapped onto invalid phenotype.

In the DGP the quality of each genotype is defined by the fitness of the corresponding phenotype. Since the genotype-to-phenotype mapping function should be restricted, to construct only valid solutions, the main problem in the DGP approach is to define genotypes and the mapping function that will be able to produce any valid solution. The main difference between GA and DGP is that in the DGP the methods constructing the solutions, instead solutions itself, evolve.

The inspiration for the DGP is the evolution process known from biology. Information describing building of proteins is stored in DNA (genotype). Evolution of the DNA has effect on proteins (phenotypes). The DGP was successfully applied for optimization in many domains [16, 17], where human-competitive results were obtained. High efficiency of the DGP-based optimization was also showed for hardware-software codesign [18] and cost minimization in real-time cloud computing [19].

## 4 DGP-Based Scheduling for Low-Power Systems

We assume that the system is specified as a task graph. According to the task graph semantics, the execution of tasks cannot be interrupted and there are no I/O operations. After completing all tasks the processing starts over again (periodically), with a specified frequency. We assume that the frequency is not greater than 1/T (where T is the deadline for the whole graph), so the problem of cycles may be omitted. Also, we assume that the execution time and the average power consumption of each of the tasks is known for each available processor core. These parameters may be estimated with the help of estimation methods.

In the paper, we consider ARM multicore processors supporting big.LITTLE architecture for implementation of the target system. Our method applies the HMP model of big.LITTLE. In such system there are two types of processors, usually quad-core. The first has higher performance (about 40% for Cortex A57), but is not power-efficient. On the other hand, the second processor is optimized to use less energy (about 75% for Cortex A53) to execute the same task but it is slower. Our goal is to find the makespan with the lowest power consumption, for which all time constraints will be met.

### 4.1 System Specification

Functions of embedded systems are specified as sets of communicating processes. A common method for describing relationship between processes is UML Activity Diagram [20]. For an embedded system software, the Activity Diagram can be simplified to a task graph (TG) [21]. TG is an acyclic directed graph where nodes correspond to tasks/processes while edges specifies the order of their execution. Each task starts as soon as all of its predecessors are completed. There are no additional synchronization mechanisms for tasks. An example of a task graph is given on Fig. 2.

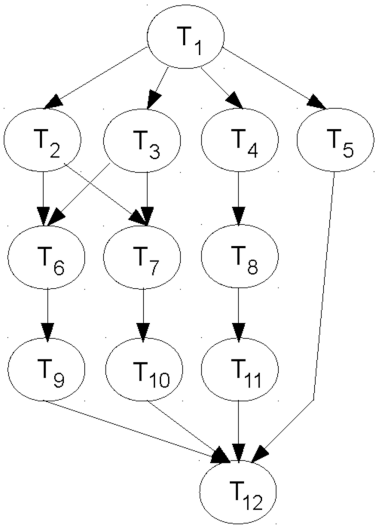



**Fig. 2.** An example of a task graph

### 4.2 Database of Resources

A database of available resources contains estimated execution parameters for each task. A resource is a single processor core. For each available core the execution time and the power consumption are given. Table 1 presents a part of a sample ARM Cortex-A15/Cortex-A7 database defined for task graph from Fig. 2.

### 4.3 Scheduling Options

We use similar DGP representation as in our previous works concerning the RCPSP problems [22–24]. Groups of tasks are assigned to the corresponding genes, then all tasks in the same group are scheduled according to the same strategy. A makespan is created by the scheduler in two steps:

1. task assignment: according to strategy specified for the corresponding gene, all tasks in the group are assigned to first available core which fulfills the preferences (row step 1 in Table 2),
2. task scheduling: corresponding group of tasks is scheduled using the scheduling strategy specified for the gene (row step 2 in Table 2). It is performed only when the core has more than one task assigned to it.

The evolution starts from the initial population consisting of randomly generated genotypes. Each gene has assigned preference, defining the decision rule for the scheduler. Table 2 contains the set of all available preferences that may be chosen by the scheduler. In the last column of Table 2 a probability of the selection is shown.

The first option prefers the HPC core. Second one prefers the LPC cores. Third option prefers a core with both the lowest power consumption and the shortest time of

**Table 1.** Database of resources

| Task # | Core # | Execution time [ns] | Power consumption [mJ] |
|---|---|---|---|
| 1 | 0 | 50 | 500 |
| 1 | 1 | 50 | 500 |
| 1 | 2 | 63 | 250 |
| 1 | 3 | 63 | 250 |
| 2 | 0 | 40 | 400 |
| 2 | 1 | 40 | 400 |
| 2 | 2 | 50 | 200 |
| 2 | 3 | 50 | 200 |
| 3 | 0 | 70 | 700 |
| … | … | … | … |
| 12 | 3 | 4 | 15 |

**Table 2.** Scheduling strategies

| Step | Preference | P |
|---|---|---|
| 1 | a. The HPC core | 0.143 |
| | b. The LPC core | 0.143 |
| | c. The lowest time * cost | 0.143 |
| | d. Determined by the alternative gene | 0.143 |
| | e. The fastest starting core | 0.143 |
| | f. The fastest ending core | 0.143 |
| | g. The cheapest from the fastest starting core | 0.143 |
| 2 | List scheduling | 1 |

execution. The next option allows using a core that cannot be obtained as a result of the remaining options. The fifth and sixth option prefer a core that may start and finish the task as soon as possible, respectively. And the last option similarly to option ‘e’ looks for the fastest starting core, but in opposite to ‘e’, this strategy always chooses the least energy consuming core. In the second step the preferred scheduling method is chosen. In this work only list scheduling method is considered.

### 4.4 Genotype and the Evolution

Genotype has a form of a tree where the nodes correspond to genes while the edges describe the subsequent stages of constructing the solution [22]. Each gene has the structure presented on Fig. 3.

Two types of genes are distinguished: leaves (*isLeaf* = *true*) and internal nodes (*isLeaf* = *false*). With each gene a subset of tasks is associated in the following way: the root of the genotype corresponds to the whole task graph, next the task graph is partitioned and the first subset corresponds to the left successor while the second subset is associated with the right one. Similar partitioning of subsets of tasks are performed

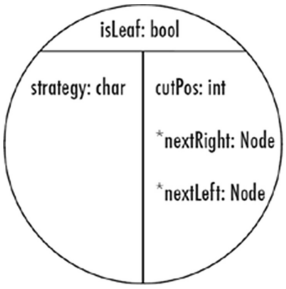



**Fig. 3.** A structure of the gene

for all internal genes. In this way the genotype specifies the hierarchical grouping of tasks, that corresponds to hierarchical construction of the solution. When the node is the internal node, *cutPos* specifies the position of cutting the group of assigned tasks into two subsets, the first subset will be assigned to the left node and the second subset to the right one (details are given in p. 4.5). Thus, for internal nodes fields *nextLeft* and *nextRight* must not be null pointers and the field *strategy* is omitted. When the node is a leaf the field *strategy* defines the strategy of scheduling of tasks corresponding to this node. Strategies considered in our approach are given in Table 2. For leaves, information from the other fields is neglected. Hence, the simplest genotype consists of only root node, which is also a leaf. To prevent unlimited growth of the genotypes during the evolution, size of the genotype is limited and it is proportional to the size of the graph task. During the evolution the edges that exceed this size are truncated. Figure 4 presents a sample genotype and the corresponding phenotype for system specified on Fig. 2 and resources from Table 1.

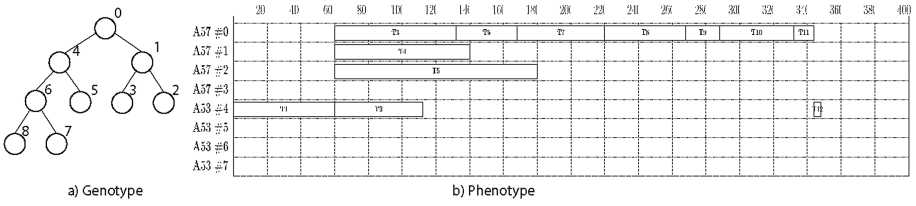



**Fig. 4.** Sample genotype (a) and the corresponding phenotype (b)

During the evolution new populations of genotypes are generated using genetic operators: mutation, crossover and reproduction. The algorithm of mutation is presented on Listing 1 [24]. The mutation modifies the randomly selected node. For leaves the gene is changed to the internal node or the strategy associated with it is randomly changed. For internal nodes the gene is changed to leaf or values of other fields are randomly changed. In this way a new genotype is created.

**Listing 1** Algorithm of mutation

```
1: if node is a leaf then
2:   Draw p ∈ [0,1];
3:   if p < 0.5 then
4:      isLeaf <= FALSE;
5:      if nextLeft OR nextRight == NULL then
6:        create a new leaf for it;
7:      end if
8:   else
9:    Draw new strategy;
10:  end if
11: else
12:   Draw p ∈ [0, 1];
13:   if p < 0.5 then
14:       isLeaf <= TRUE;
15:   else
16:       Change value of a randomly chosen
          field: cutPos, nextLeft or nextRight;
17:    end if
18: end if
```

In our approach, the traditional 1-point crossover is used. First, genotypes corresponding to parents are cut at randomly selected locations. Then cut off subtrees are swapped. In this way the two new individuals are created. Sample crossover of genotypes is illustrated on Fig. 5.

### 4.5 Genotype to Phenotype Mapping

The genotype-to-phenotype mapping is performed in three steps. During the first step, scheduling strategies (Table 2) are assigned to all tasks. For the task graph from Fig. 2, this step is illustrated on Fig. 6. With node 0 two sets of tasks are associated: {1, 2, 3, 4, 5} and {6, 7, 8, 9, 10, 11, 12}. The first set is partitioned by node 1 into next two groups. In the first one, there are only tasks 1 and 2. They strategy is defined to ‘b’ by the second node. Tasks 3, 4 and 5 belong to the second group, for which strategy ’f’ is defined by the node 3. The group of tasks from 6 to 12 is partitioned by the node 4. Its *cutPos* parameter equals 6, thus the first 6 tasks from the group are assigned to the first subgroup. The node 5 determines strategy ‘a’ for this subgroup. In the second subgroup we have only one task. Thus, although the node 6 divides this subgroup into next two parts, the cutPos of the node 6 is out of range and in effect, strategy for the task 12 is defined only by the node 7.

During the second step, all tasks are assigned to cores and scheduled. First, tasks without any predecessor in the task graph, next tasks with predecessors having already assigned core, are assigned to cores and scheduled according to preferences chosen during the first step.

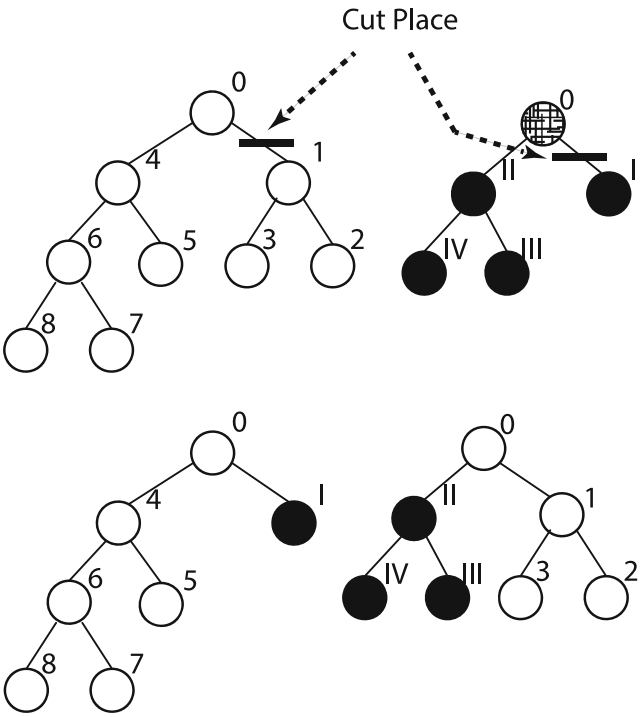



**Fig. 5.** Sample crossover

During the third step, the power consumption of the solution is calculated. For this purpose, the resource database (Table 1) is used.

### 4.6 Controlling the Evolution

Like in classical GA also in DGP the evolution is performed by generating new populations of solutions (genotypes) using the following genetic operators: crossover (re-combination), mutation, reproduction. Each new population replaces the current one. The efficiency of the genetic approach strongly depends on the number of individuals generated using the above operators as well as on the number of individuals in the population. Therefore, the evolution should be controlled by the following parameters:

- *population size*: in our approach, the number of individuals in each population is always the same and is equal to "number of tasks" * "number of cores",
- *reproduction size*: the value of this parameter is determined experimentally,
- *crossover size*: the value of this parameter is determined experimentally,
- *mutation size*: the value of this parameter is determined experimentally,
- *the number of generations*.

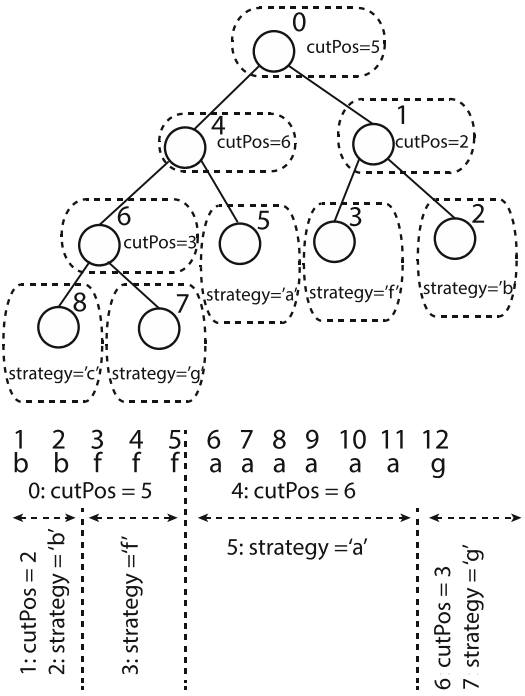



**Fig. 6.** Assignment of scheduling strategies to tasks

Genetic operator requires selection of individuals from the current generation. In our approach the tournament method [14] is applied. In this method, each time a fixed number of individuals is drawn, then the best one is chosen. Therefore, the size of the tournament is the additional parameter controlling the evolution.

### 4.7 Fitness

Fitness defines the quality of the genotype. The goal of the evolution is to find the individual with the best fitness. Therefore, the fitness determines the goal of optimization. Since, our method should find the schedule satisfying time requirements for which the power consumption is as low as possible, in our approach the fitness of the genotype is defined as a power consumption of the corresponding phenotype. The best fitness means the lowest power consumption. Genotype-to-phenotype mapping guarantees that all considered phenotypes satisfy time constraints.

## 5 Example and Experimental Results

The first step in our experiments was finding the best parameters of the evolution. Then, we have verified advantages of the presented method using example of the complex multimedia system, which was described in [25] and from e3s benchmark suite [26]. The result has been compared with the method based on Least-Laxity-First Scheduling Algorithm [27].

### 5.1 Algorithm Tuning

For presentation of the algorithm tuning, the task graph from Fig. 2 was used. The times and costs of executing each tasks on cores are presented in Table 3. Although processors working in big.LITTLE mode have four cores, in this experiment the number of cores in each processors was decreased to two. Otherwise, this problem would be reduced to too trivial and a more complex task graph would be needed.

**Table 3.** Execution parameters for TG from Fig. 2

| Task | Cores | | | |
|---|---|---|---|---|
| | A57 (high performance) | | A53 (energy efficient) | |
| | Energy | Time | Energy | Time |
| 1 | 500 | 50 | 250 | 63 |
| 2 | 400 | 40 | 200 | 50 |
| 3 | 700 | 70 | 350 | 88 |
| 4 | 800 | 80 | 400 | 100 |
| 5 | 1200 | 120 | 600 | 150 |
| 6 | 380 | 38 | 190 | 48 |
| 7 | 510 | 51 | 255 | 64 |
| 8 | 470 | 47 | 235 | 59 |
| 9 | 420 | 42 | 210 | 53 |
| 10 | 230 | 23 | 115 | 29 |
| 11 | 120 | 12 | 60 | 15 |
| 12 | 30 | 3 | 15 | 4 |

During the algorithm tuning experiment, the following values of genetics parameters were used:

- the evolution was stopped after 100 generations,
- the population size was equal 19,
- the tournament size was equal 10.

The values of the rest of parameters, were determined during experiments. The first step is to estimate the influence of *crossover size* and *mutation size* on finding results. Thus, the best results of evolution for different parameters combination was shown on

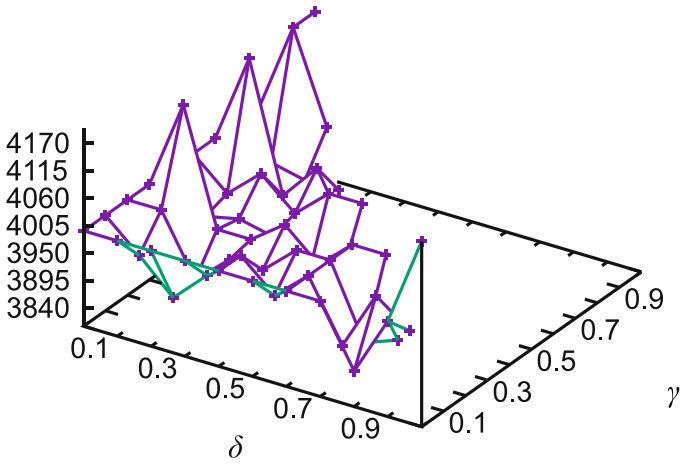



**Fig. 7.** Crossover size (γ) vs. mutation size (δ)

Fig. 7. Figure 7 presents, that the most influential parameter is *crossover size*, however an influence of *mutation size* is also visible. Thus investigation of influence of *crossover size* is necessary. The results are presented on Fig. 8.

An analysis of data from Fig. 8 presents clearly, that the best value for *crossover size* is 40%. For smaller values, the diversity in population is too small, to improve results. From the other hand, values higher than 40% cause too much disorder, thus the cost of a solution increases. For the next experiment, *crossover size* equals 40% was used.

Figure 9 presents the influence of *mutation size* on the best solution found. In this example, two minima are possible. Values smaller than 20% do not change the final result enough. The mutation makes too many changes in the genotype and therefore the value higher than 40% worse the evolution. Thus, *mutation size* = 40% was taken for generating a solution. An evolution progress is presented on Fig. 10.

Figure 10 shows, that the best solution was found very quickly and 100 generations are unnecessary but 15 generations are enough to solve this problem. To sum up, execution of the best schedule, obtained by using DGP method takes 280 ms using 3875 mJ. That is better results, than obtained by LPLLF algorithm (described in Subsect. 5.4) where execution is finished after 279 ms using 3975 mJ.

### 5.2 Sample System

Our method will be applied for implementation of the multimedia player described in [25]. This is the complex system consisting of 40 tasks, which will be implemented as a low-power real-time system. The task graph describing the sample system is given on Fig. 12. We assumed that the deadline for the system equals 100000 ns. An architecture with shared memory is assumed, thus the communication between tasks will be neglected. Table 4 presents run-time parameters for Cortex A57 and Cortex A53 cores.

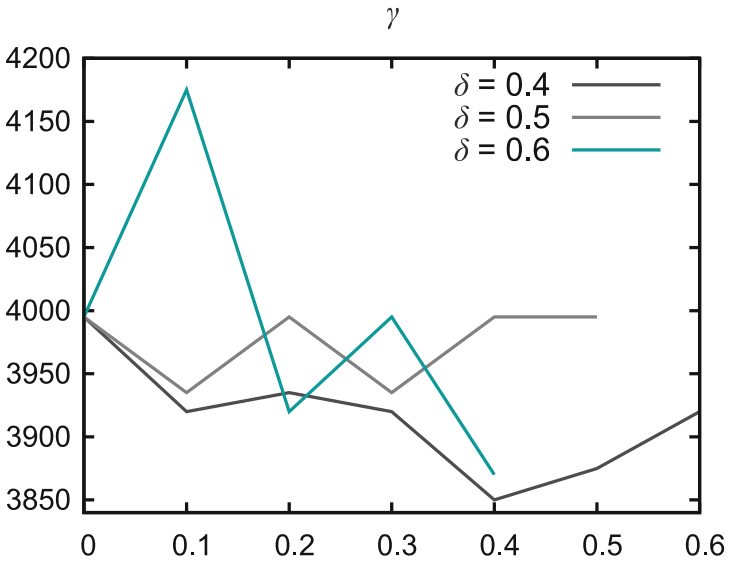



**Fig. 8.** The influence of crossover size

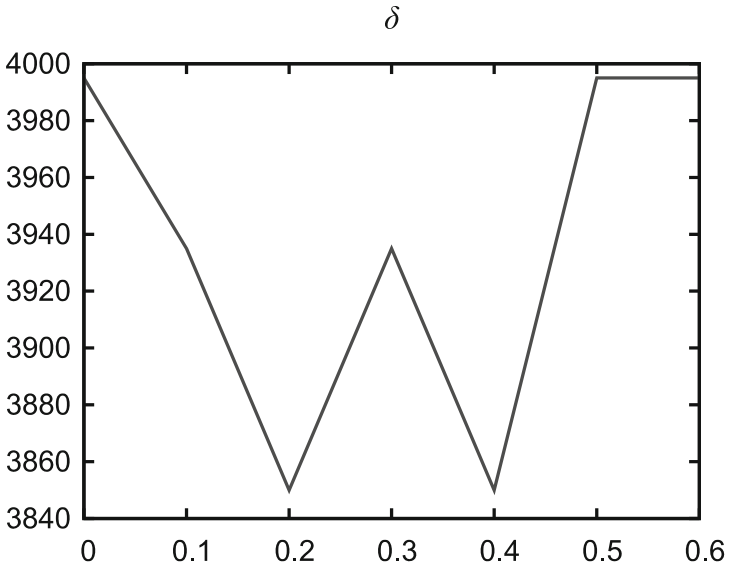



**Fig. 9.** The influence of mutation size

We assume that the target architecture will consist of 4-core A57 and 4-core A53 processors supporting big.LITTLE. The goal of the optimization will be the minimization of the power usage.

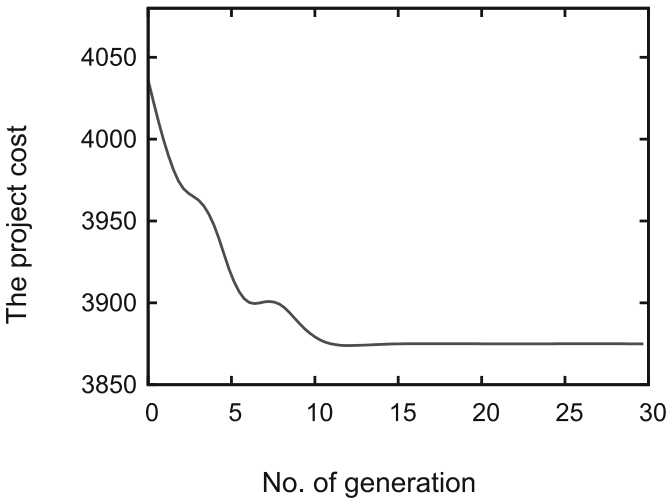



**Fig. 10.** The evolution progress

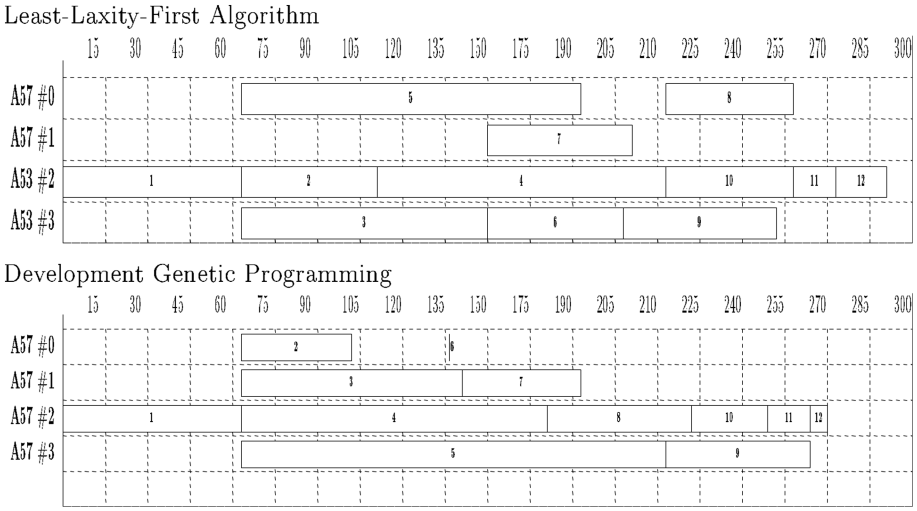



**Fig. 11.** Makespans obtained using LPLLF and DGP for algorithm tuning

### 5.3 Genetic Optimization

The makespan for the system described in p. 5.2 was generated using our method. For optimization, the following values of genetic parameters were defined:

- the evolution was stopped after 100 generations,
- each experiment was repeated 7 times,
- the population size was equal to 128,

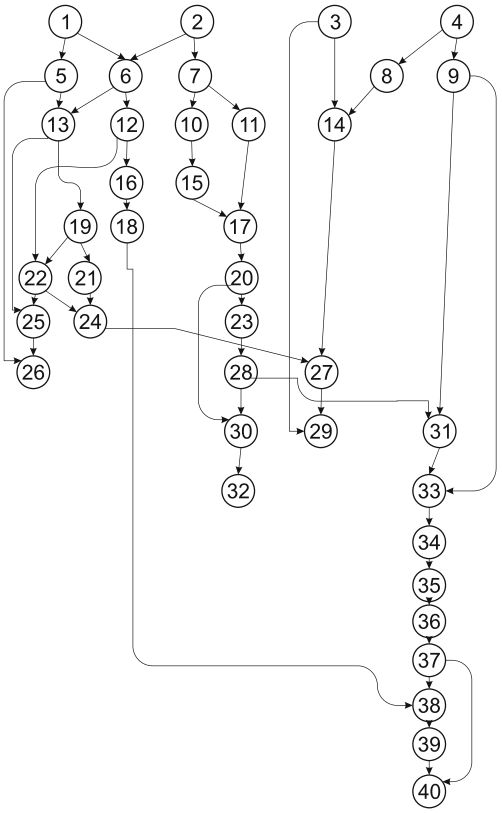



**Fig. 12.** Task graph of the multimedia system

- tournament size was equal to 10,
- the number genotypes created by mutation, was equal to 25%,
- 25% of individuals were created using reproduction.

The values of the above parameters were determined according to the method described in our previous work [21].

### 5.4 Least-Laxity-First Algorithm

One of the most known algorithms for scheduling tasks in real-time embedded systems is the Least-Laxity-First Algorithm (LLF) [27]. A description of the algorithm in pseudocode is shown on Listing 2. Basic LLF method schedules task according to the least laxity (slack time). The laxity is defined as a difference between an execution time and a task deadline. The goal of LLF is to find the schedule that satisfies all deadlines. It does not take into account power or cost optimization. Therefore, we modify this method by favoring energy-efficient cores. In other words, during scheduling, the method first tries assign a task to low-power core, only when it will violate the time

**Table 4.** Resource database for multimedia player

| Task | Processor cores | | | |
|---|---|---|---|---|
| | A57 (high performance) | | A53 (energy efficient) | |
| | Energy | Time | Energy | Time |
| 1 | 5 | 537 | 3 | 671 |
| 2 | 11 | 1072 | 6 | 1340 |
| 3 | 5 | 537 | 3 | 671 |
| 4 | 4 | 376 | 2 | 470 |
| 5 | 73 | 7337 | 37 | 9171 |
| 6 | 11 | 1072 | 6 | 1340 |
| 7 | 110 | 10958 | 55 | 13698 |
| 8 | 74 | 7358 | 37 | 9198 |
| 9 | 11 | 1051 | 6 | 1314 |
| 10 | 6 | 559 | 3 | 699 |
| 11 | 5 | 486 | 3 | 608 |
| 12 | 3 | 286 | 2 | 358 |
| 13 | 13 | 1298 | 7 | 1623 |
| 14 | 37 | 3679 | 19 | 4599 |
| 15 | 21 | 2065 | 11 | 2581 |
| 16 | 53 | 5253 | 27 | 6566 |
| 17 | 75 | 7523 | 38 | 9404 |
| 18 | 11 | 1076 | 6 | 1345 |
| 19 | 4 | 409 | 2 | 511 |
| 20 | 4 | 409 | 2 | 511 |
| 21 | 11 | 1076 | 6 | 1345 |
| 22 | 2 | 157 | 1 | 196 |
| 23 | 260 | 26018 | 130 | 32523 |
| 24 | 2 | 176 | 1 | 220 |
| 25 | 2 | 197 | 1 | 246 |
| 26 | 260 | 26018 | 130 | 32523 |
| 27 | 236 | 23607 | 118 | 29509 |
| 28 | 6 | 559 | 3 | 699 |
| 29 | 11 | 1072 | 6 | 1340 |
| 30 | 110 | 10958 | 55 | 13698 |
| 31 | 5 | 486 | 3 | 608 |
| 32 | 3 | 286 | 2 | 358 |
| 33 | 11 | 1072 | 6 | 1340 |
| 34 | 4 | 409 | 2 | 511 |
| 35 | 4 | 409 | 2 | 511 |
| 36 | 236 | 23607 | 118 | 29509 |
| 37 | 74 | 7414 | 37 | 9268 |
| 38 | 3 | 253 | 2 | 316 |
| 39 | 2 | 179 | 1 | 224 |
| 40 | 2 | 176 | 1 | 220 |

constraint, the task will be assigned to more efficient core. Our Low Power LLF (LPLLF) method is used only for reference, to verify that the DGP is efficient also for power optimization in real-time embedded systems.

### 5.5 Results of Power-Aware Scheduling

The makespans obtained using DGP and LPLLF methods are presented on Fig. 11. On Y-axis different cores are represented, while the time of execution is represented by the X-axis. Numbers correspond to the following tasks. The experimental results proved that the presented method is more efficient than LPLLF. Energy consumption for the system scheduled using DGP equals 990 mJ, while the same example scheduled using LPLLF requires 1018 mJ. To meet the deadline, the LPLLF method assigned the long task 36 to the most efficient core. But in the DGP, more energy-efficient solution was found by assigning some shorter tasks that in total consume less power than task 36, to the faster core.

Above experiment showed that the scheduler constructed using DGP is able to find highly optimized solutions. More experiments proving this remark are given in our previous work [19, 23, 24].

**Listing 2** Description of the LPLLF in pseudocode

1: Compute a slack time of all tasks:
$\underline{S}_{lk}(i) = L(i) - E(i)$
where L(i) is the maximal time by which the task i must be started, it is scheduled As Late As Possible (ALAP) on LPCs, and E(i) is the earliest start of the task, it is scheduled As Soon As Poosible (ASAP) on HPCs. The number of cores is unlimited;
2: Find a list of active tasks, which have no predecessors or have all predecessors already assigned to resources;
3: **if** the list is not empty **then**
4: go to 8;
5: **else**
6: go to 14;
7: **end if**
8: choose a task with the smallest slack time. Tasks with the same slack time are executed in a static order determined by tie-breaking rules. We assume that $i$ is preferred to $i$+1 if $\underline{S}_{lk}(i) = S_{lk}(i + 1)$;
9: assign the task ASAP to one of the LPCs;
10: **if** the start time of the task is greater than $L(i)$ **then**
11: assign the task ASAP to one of the HPCs;
12: **end if**
13: go to 2;
14: **end**;

### 5.6 Results of e3s Benchmark

The first task graph from e3s benchmark suite [26] is presented on Fig. 13. And its resource database is presented in Table 5. Evolution parameters were the same as in Subsect. 5.1. The results of presented method and their comparison with LLF algorithm were shown in Table 6. The DGP allows to consume significantly less energy than LLF when using 4 cores. Moreover, it is fully adaptive to the time requirements. The energy consumption decreases as the maximal allowed execution time increases. In case of 8 cores, the results are the same. However, the LLF is not able to find a shorter schedule than 23042 ms. Therefore, the DGP is better for high-constrained problems.

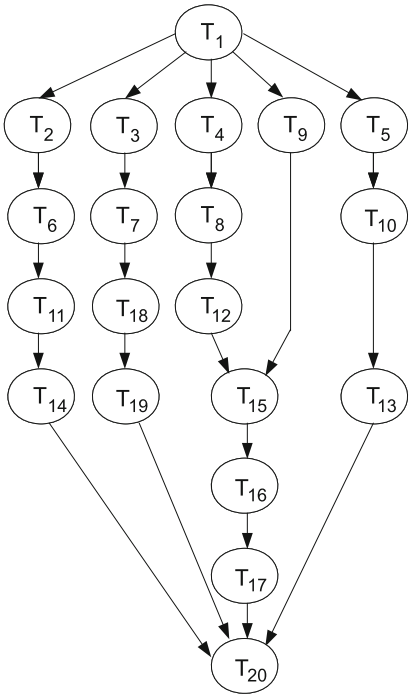



**Fig. 13.** Task Graph of auto-indust-mocsyn set.

**Table 5.** Resource database for auto-indust-mocsyn benchmark

| Task | Processor cores | | | |
|---|---|---|---|---|
| | A57 (high performance) | | A53 (energy efficient) | |
| | Energy [mJ] | Time [ns] | Energy [mJ] | Time [ns] |
| 1 | 100 | 10 | 50 | 13 |
| 2 | 90 | 9 | 45 | 11 |
| 3 | 800 | 80 | 400 | 100 |
| 4 | 140000 | 14000 | 70000 | 17500 |
| 5 | 3300 | 330 | 1650 | 413 |
| 6 | 230 | 23 | 115 | 29 |
| 7 | 9100 | 910 | 4550 | 1138 |
| 8 | 67000 | 6700 | 33500 | 8375 |
| 9 | 690 | 69 | 345 | 86 |
| 10 | 350 | 35 | 175 | 44 |
| 11 | 90 | 9 | 45 | 11 |
| 12 | 4900 | 490 | 2450 | 613 |
| 13 | 740 | 74 | 370 | 93 |
| 14 | 50 | 5 | 25 | 6 |
| 15 | 90 | 9 | 45 | 11 |
| 16 | 30 | 3 | 15 | 4 |
| 17 | 100 | 10 | 50 | 13 |
| 18 | 800 | 80 | 400 | 100 |
| 19 | 9100 | 910 | 4550 | 1138 |
| 20 | 100 | 10 | 50 | 13 |

**Table 6.** Power consumption and time execution of auto-indust-mocsyn program

| Deadline | DGP | | | | LLF | | | |
|---|---|---|---|---|---|---|---|---|
| | Cores | | | | Cores | | | |
| | 4 | | 8 | | 4 | | 8 | |
| | Energy | Time | Energy | Time | Energy | Time | Energy | Time |
| ms | mJ | ns | mJ | ns | mJ | ns | mJ | ns |
| 22 | 222330 | 21367 | 222330 | 21367 | 224940 | 21235 | – | – |
| 23 | 191280 | 22919 | 191280 | 22919 | 224940 | 21235 | – | – |
| 24 | 188830 | 23042 | 188830 | 23042 | 224940 | 21235 | 188830 | 23042 |
| 25 | 152330 | 24978 | 152330 | 24978 | 154940 | 24746 | 152330 | 24867 |

## 6 Conclusions

In this paper the method of synthesis of low-power embedded real-time software for multicore systems is presented. We use DGP-based approach to optimize the scheduling strategy that minimizes the power consumption. The method assumes the

ARM big.LITTLE architecture of the target system, but it can be easily adapted to other dynamic power management techniques like DVFS.

The computational experiments confirmed, that the schedulers, generated using DGP, are efficient and flexible. For the sample system our method gave better results than results obtained using LLF-based method.

Despite the above advantages of our method, there is still possible to improve the methodology. In the future work, we will consider adaptation of our method to other methods of system specification as well as apply other optimization methods e.g. based on constraint logic programming [28, 29].

## References


1. Ditzel, M., Serdijn, W., Otten, R.: Power-Aware Architecting: for Data-Dominated Applications. Springer, Netherlands (2007). http://dx.doi.org/10.1007/978-1-4020-6420-3
2. Greenhalgh, P.: Big. little processing with ARM CortexTM-A15 & ARM CortexTM-A7, ARM White paper, pp. 1–8 (2011). http://www.arm.com/files/downloads/big.LITTLE_Final.pdf
3. Benini, L., Bogliolo, A., De Micheli, G.: A survey of design techniques for system-level dynamic power management. IEEE Trans. Very Large Scale Integr. (VLSI) Syst. **8**(3), 299–316 (2000). http://dx.doi.org/10.1109/92.845896
4. Shang, L., Dick, R.P., Jha, N.K.: Slopes: hardware– software cosynthesis of low-power real-time distributed embedded systems with dynamically reconfigurable fpgas. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **26**(3), 508–526 (2007). http://dx.doi.org/10.1109/TCAD.2006.883909
5. Steger, C., Bachmann, C., Genser, A., Weiss, R., Haid, J.: Power-aware hardware/software codesign of mobile devices. e & i Elektrotech. Informationstechnik **127**(11), 327–334 (2010)
6. Luo, J., Jha, N.K.: Low power distributed embedded systems: dynamic voltage scaling and synthesis. In: Sahni, S., Prasanna, V.K., Shukla, U. (eds.) HiPC 2002. LNCS, vol. 2552, pp. 679–693. Springer, Heidelberg (2002). doi:10.1007/3-540-36265-7_63
7. Yao, F., Demers, A., Shenker, S.: A scheduling model for reduced cpu energy. In: Proceedings of the 36th Annual Symposium on Foundations of Computer Science, pp. 374–382. IEEE (1995). http://dx.doi.org/10.1016/j.ejor.2009.11.005
8. Hartmann, S., Briskorn, D.: A survey of variants and extensions of the resource-constrained project scheduling problem. Eur. J. Oper. Res. EJOR **207**(1), 1–15 (2010). http://dx.doi.org/10.1016/j.ejor.2009.11.005
9. Hartmann, S.: A competitive genetic algorithm for resource-constrained project scheduling. Naval Res. Logistics (NRL) **45**(7), 733–750 (1998). http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.45.359&rep=rep1&type=pdf
10. Li, X., Kang, L., Tan, W.: Optimized research of resource constrained project scheduling problem based on genetic algorithms. In: Kang, L., Liu, Y., Zeng, S. (eds.) ISICA 2007. LNCS, vol. 4683, pp. 177–186. Springer, Heidelberg (2007). doi:10.1007/978-3-540-74581-5_19
11. Zoulfaghari, H., Nematian, J., Mahmoudi, N., Khodabandeh, M.: A new genetic algorithm for the rcpsp in large scale. Int. J. Appl. Evol. Comput. (IJAEC) **4**(2), 29–40 (2013). http://dx.doi.org/10.4018/jaec.2013040103
12. Jeff, B.: Ten things to know about big.little, ARM Holdings (2013). http://community.arm.com/groups/processors/blog/2013/06/18/ten-things-to-know-about-biglittle

13. Deniziak, S., Ciopiński, L.: Synthesis of power aware adaptive embedded software using developmental genetic programming. In: Fidanova, S. (ed.) Recent Advances in Computational Optimization. SCI, vol. 655, pp. 97–121. Springer, Heidelberg (2016). doi:10.1007/978-3-319-40132-4_7
14. Michalewicz, Z.: Genetic Algorithms+Data Structures = Evolution Programs. Springer, Heidelberg (1996). http://dx.doi.org/10.1007/978-3-662-03315-9
15. Dick, R.P., Jha, N.K.: MOGAC: a multiobjective genetic algorithm for hardware-software cosynthesis of distributed embedded systems. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **17**(10), 920–935 (1998). http://dx.doi.org/10.1109/43.728914
16. Koza, J., Poli, R.: Genetic programming. In: Burke, E., Kendall, G. (eds.) Search Methodologies, pp. 127–164. Springer, US (2005). http://dx.doi.org/10.1007/0-387-28356-0_5
17. Koza, J.R., Bennett III, F.H., Andre, D., Keane, M.A.: Evolutionary design of analog electrical circuits using genetic programming. In: Parmee, I.C. (ed.) Adaptive Computing in Design and Manufacture, pp. 177–192. Springer, London (1998). http://dx.doi.org/10.1007/978-1-4471-1589-2_14
18. Deniziak, S., Gorski, A.: Hardware/Software co-synthesis of distributed embedded systems using genetic programming. In: Hornby, G.S., Sekanina, L., Haddow, P.C. (eds.) ICES 2008. LNCS, vol. 5216, pp. 83–93. Springer, Heidelberg (2008). doi:10.1007/978-3-540-85857-7_8
19. Deniziak, S., Ciopinski, L., Pawinski, G., Wieczorek, K., Bak, S.: Cost optimization of real-time cloud applications using developmental genetic programming. In: IEEE/ACM 7th International Conference on Utility and Cloud Computing, vol. 7269, pp. 182–189. IEEE Computer Society (2014). http://dx.doi.org/10.1109/UCC.2014.126
20. Briand, L., Labiche, Y.: A uml-based approach to system testing. Softw. Syst. Model. **1**(1), 10–42 (2002). http://dx.doi.org/10.1007/s10270-002-0004-8
21. Dick, R., Rhodes, D., Wolf, W.: TGFF: task graphs for free. In: Proceedings of the Sixth International Workshop on Hardware/Software Codesign (CODES/CASHE 1998), pp. 97–101, March 1998 . http://dx.doi.org/10.1109/HSC.1998.666245
22. Sapiecha, K., Ciopinski, L., Deniziak, S.: An application of developmental genetic programming for automatic creation of supervisors of multi-task real-time object-oriented systems. In: IEEE Federated Conference on Computer Science and Information Systems (FedCSIS), pp. 501–509 (2014). http://dx.doi.org/10.15439/2014F208
23. Sapiecha, K., Ciopiński, L., Deniziak, S.: Synthesis of self-adaptive supervisors of multi-task real-time object-oriented systems using developmental genetic programming. In: Fidanova, S. (ed.) Recent Advances in Computational Optimization. SCI, vol. 610, pp. 55–74. Springer, Heidelberg (2016). doi:10.1007/978-3-319-21133-6_4
24. Pawiński, G., Sapiecha, K.: An efficient solution of the resource constrained project scheduling problem based on an adaptation of the developmental genetic programming. In: Fidanova, S. (ed.) Recent Advances in Computational Optimization. SCI, vol. 610, pp. 205–223. Springer, Heidelberg (2016). doi:10.1007/978-3-319-21133-6_12
25. Hu, J., Marculescu, R.: Energy-and performance-aware mapping for regular noc architectures. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **24**(4), 551–562 (2005). http://dx.doi.org/10.1109/TCAD.2005.844106
26. E3s benchmark. http://ziyang.eecs.umich.edu/dickrp/e3s/
27. Han, S., Park, M.: Predictability of least laxity first scheduling algorithm on multiprocessor real-time systems. In: Zhou, X., Sokolsky, O., Yan, L., Jung, E.-S., Shao, Z., Mu, Y., Lee, D. C., Kim, D.Y., Jeong, Y.-S., Xu, C.-Z. (eds.) EUC 2006 Workshops. LNCS, vol. 4097, pp. 755–764. Springer, Heidelberg (2006). doi:10.1007/11807964_76

28. Sitek, P., Wikarek, J.: A hybrid framework for the modelling and optimisation of decision problems in sustainable supply chain management. Int. J. Prod. Res. **53**, 1–18 (2015). http://dx.doi.org/10.1080/00207543.2015.1005762
29. Sitek, P., Wikarek, J.: A hybrid programming framework for modeling and solving constraint satisfaction and optimization problems. Sci. Program. **2016**, 13 (2016). Article ID 5102616. http://dx.doi.org/10.1155/2016/5102616

# BlueJ as the NetBeans Plugin

Rudolf Pecinovský(✉)

Department of Information Technologies, University of Economics,
Prague, Prague, Czech Republic
rudolf@pecinovsky.cz

**Abstract.** One of the best IDE for introduction courses of object oriented programming is the BlueJ IDE. This IDE was developed with respect to needs of the absolute beginners. It offers almost everything what we need for teaching according the Architecture First methodology. However later, when the students go over to some professional IDE, they lose many of the BlueJ excellent features, especially the ability to design the program and its architecture in the interactive mode. The paper shows, how the BlueJ IDE was modified to work as fully functional plugin for NetBeans IDE and which new possibilities we obtain with it.

## 1 Introduction

### 1.1 *BlueJ* Integrated Development Environment

The *BlueJ* IDE ([1]) is at present probably the best development environment for using in introductory courses of programming. It has the following strengths:

- It uses a class diagram for primary depiction of the developed program and thus it pushes the students to think about the developed program at the architecture level, not at the code level.
- The above mentioned class diagram is interactive and enables to involve the required changes of architecture very simply without being distracted by certain graphic details that take the attention off when using other UML diagram editors.
- Editing the class diagram is closely connected with editing the source codes. All important changes in class diagram are immediately included into source codes of influenced data types (interfaces, classes, enums …) and, on the contrary, all changes in the source code are almost immediately shown in the class diagram.
- The *BlueJ* environment is equipped with a simple code generator so that within first few lessons it is possible to design functioning programs without presenting the created source code to students.
- The environment includes a simple and natural support for test class creating that makes teaching the students more effective, and enables that the unit test creating becomes a natural part of the program development for them.


J. Janech et al. (eds.), *Proceedings of the 2015 Federated Conference on Software Development and Object Technologies*, Advances in Intelligent Systems and Computing 511,
DOI 10.1007/978-3-319-46535-7_20

### 1.2 Architecture First Methodology

The abilities of development tools as well as used frameworks are further improved. The programmer's focus goes currently from creating the code to designing the architecture. Contemporary trends even show that the time when various code generators will take over coding of the architecture design is a near future.

Unfortunately the methodologies currently in use do not react to these trends and continue to prefer teaching of programming that, to certain extent, concentrates only how to code the explained programming constructions and that postpones the explanation of architectural design to follow-up courses.

This approach is, however, contrary to an important Early Bird pedagogical pattern, which says ([2, 3]): "Organize the course so that the most important topics are taught first. Teach the most important material, the "big ideas", first (and often). When this seems impossible, teach the most important material as early as possible."

The Architecture First methodology ([6–8]) respects the crucial importance of knowing the architecture design for future programmers and, according to the above stated pedagogical pattern, it lines up the teaching of the architecture design at the beginning of teaching process.

So that we could test the designed architecture, it is necessary to code the designed program. In case we would not like to distract the students with the explanation of syntactic rules and we would like to focus on the architecture design, we need certain code generator at our disposal.

At this time *BlueJ* with its code generator comes to help us. Using it, we can focus to the architecture design with students and coding of the relevant program will be provided by this generator.

Thus *BlueJ* with its code generator enables to divide the Architecture First methodology into required phases as follows:

1. Basics of object oriented programming are explained together with basics of object architecture design, whilst the coding of created programs is in charge of the program generator.
2. After exhausting all possibilities of the used code generator the programs designed in the first phase are assessed once again. But nowadays the source code created by the generator and the necessary syntactic rules are explained to students at these examples.
3. Next item involves an explanation of constructions and architectural principles, which are already behind the possibilities of the used code generator.

## 2 Where Are the Problems

*BlueJ* offers a number of properties which we can use in applying the Architecture First methodology (and we also use them), however, some of its properties need further improvement as follows:

### 2.1 Small Capability of the Code Generator

The current code generator has really only limited abilities. One of the activities, we are dealing with, is the significant extension of these abilities. Detailed information concerning this problem is described in e.g. [4].

### 2.2 No Graphical View of the Code

The *BlueJ* environment depicts graphically only the program architecture in the form of a class diagram. When teaching we would consider as very useful when it would be able to show also the used algorithms – for example through kopenograms. This topic is discussed in details in [5].

### 2.3 Bad Cooperation with Professional IDEs

*BlueJ* IDE is optimized for the introductory courses with the very beginners. Therefore its simplicity of using is especially important. However, this simplicity is redeemed by limitation of its functionality in certain areas.

On the other hand this limitation of functionality can be a good pretext for coming to the work with professional IDE in the moment when students reach certain level of their knowledge. However, the problem is in this case, that the possibility to work in the interactive mode as well as the close connection between the editor and the code generator is lost.

There is a plugin for *NetBeans* environment, which enables to open the project originally developed in *BlueJ* environment, and to open it in such way so that they could be open also in *BlueJ* environment again, but that is all what this plugin offers. In other areas the students have to fully submit to practice of this new environment.

We used the fact that *BlueJ* environment is distributed as an open source and we decided to modify the *BlueJ* environment in such way so that it itself becomes the plugin of *NetBeans* environment and, if need be, also the plugin of other development environment. This paper deals with this extension.

## 3 Requirements and Their Solution

### 3.1 Easy Installation

The first requirement for the created plugin was an easy installation available even for the beginners. The plugin is installed in a standard way in which only the nbm file containing full necessary information for installing the relevant plugin is selected. During its installation the plugin defines its own card in the Options dialogue. This card allows setting the *BlueJ's* location (see Fig. 1). If the user forgets it, plugin will points him.

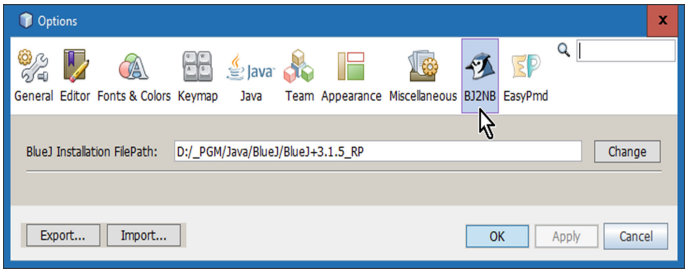



**Fig. 1.** The **Option** dialog for setting the *BlueJ* location

### 3.2 Simple Creation of New Project

Another requirement was the possibility to create easily the new project which would overtake the source codes of certain existing project created in *BlueJ*. Thus **Import BlueJ IDE Project** has been included into the menu of the project types. After its entering the dialogue from Fig. 2 opens. Besides the name and the location of the created project it requires also entering the location of source codes of the transferred original *BlueJ* project. The user can choose if these source codes will be copied into the created *NetBeans* project's folder or if he will work with original files.

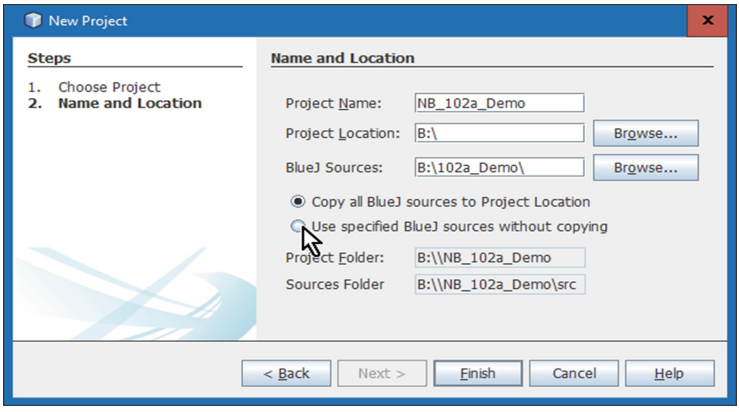



**Fig. 2.** Settings for the created project

This solution enables that the current *BlueJ* project could be easily converted into *NetBeans* project without losing the possibility to be used by independently operating *BlueJ* environment. With the other words, it enables that the project would be operating in both environments (and even simultaneously with certain limitations).

### 3.3 Simple Activation of *BlueJ* Window

*BlueJ* project is presented in the project window similarly as any other *NetBeans* project and we can work with it as with any current *NetBeans* project.

If we want to open some of the packages of the given project in *BlueJ*, we can only enter the command Show in *BlueJ* in the context menu. Then *NetBeans* will open the *BlueJ* window with the class diagram of the given package (see Fig. 3).

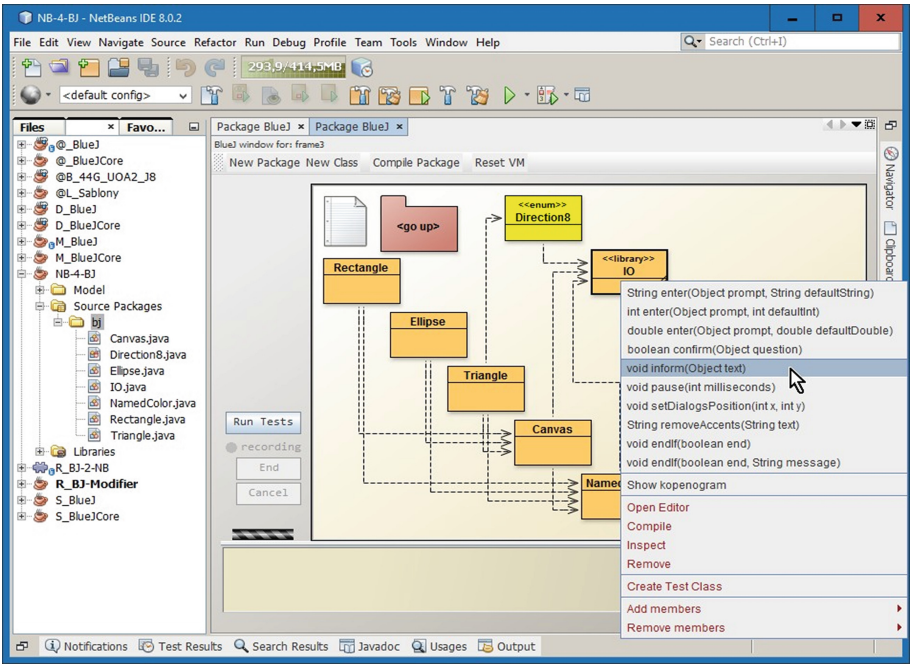



**Fig. 3.** The *NetBeans* application window with the Project window and *BlueJ* window with the opened package

### 3.4 Window Layout

Another requirement was so that the appearance of the window in *NetBeans* tab would be closely similar to that one of the *BlueJ* application window, so that the students could be familiar with the new environment. As can be seen from Fig. 3, the only difference is that the buttons from the upper part of the button panel moved to their own panel.

### 3.5 Functionality

The key requirement for the final product was to mediate the functionality of *BlueJ* environment in *NetBeans* environment, above all the ability of working in the

interactive mode and close connection of the source code and the class diagram. The user should have, in optimal case, an access to all functionality of *BlueJ* environment.

As is shown in Fig. 3, the *BlueJ* window provides all functionality of *BlueJ* environment including the possibility to work in the interactive mode.

### 3.6 Editor

Despite asking a maximum compliance of *BlueJ* functionality it was clear that it would be advantageous to prefer *NetBeans* functionality in certain areas. Such area was the source code editing. Therefore we asked *BlueJ* to use a comfortable editor inbuilt in *NetBeans* for editing the source code.

*NetBeans* editor will open independently to whether we ask for editing the source file from the *NetBeans* project window or from *BlueJ* window. Thus we can have an access not only to comfortable editor of the source code, but also to its other functionality, as is e.g. sophisticated refactoring, searching and replacing text using regular expressions, searching and replacing text in multiple files and folders etc.

## 4 Summary

The first experience with the described plugin is very good. Students can significantly easier come from the environment in which they started their learning into a more comfortable, but at the same time more demanding environment in which they develop more complex programs used in further phases of studies.

Thus, using of this plugin is more advantageous also for teachers because they do not have to switch between the environments in which the programs for teaching are designed and the environment in which they are subsequently verified. They can get quickly the feedback and can far more operatively verify the applicability of proposed procedures.

**Acknowledgment.** The paper was processed with contribution of long term institutional support of research activities by the Faculty of Informatics and Statistics, the University of Economics, Prague.

## References

1. Page About BlueJ. http://bluej.org/about.html
2. Bergin, J.: Fourteen pedagogical patterns. In: Proceedings of Fifth European Conference on Pattern Languages of Programs, (EuroPLoP™ 2000), Irsee (2000)
3. Bergin, J.: Pedagogical Patterns: Advice For Educators. CreateSpace Independent Publishing Platform (2012). ISBN:1-4791-7182-4
4. Bobusky S.: Enhancing BlueJ interactive mode. Master thesis at University of Economics, Prague (2015)

5. Chadim M., Pecinovský R.: Kopenograms and their implementation in BlueJ. In: SDOT (2015)
6. Pecinovský, R.: Principles of the methodology architecture first. In: Objekty 2012 – Proceedings of the 17th International Conference on Object-Oriented Technologies, Praha (2012). ISBN:978-80-86847-63-4
7. Pecinovský, R., Kofránek, J.: The experience with after- school teaching of programming for parents and their children. In: FECS 2013 – The 2013 International Conference on Frontiers in Education: Computer Science and Computer Engineering, Las Vegas, 22–25 July 2013. http://worldcomp-proceedings.com/proc/p2013/FEC4207.pdf
8. Pecinovský, R.: OOP – Learn Object Oriented Thinking and Programming. Eva & Tomas Bruckner Publishing, Czech Republic (2013). ISBN 80-904661-8-4

# Integration of Inertial Sensor Data into Control of the Mobile Platform

Rastislav Pirník[1], Marián Hruboš[1(✉)], Dušan Nemec[1], Tomáš Mravec[1], and Pavol Božek[2]

[1] Faculty of Electrical Engineering, University of Žilina, Žilina, Slovak Republic
{rastislav.pirnik, marian.hrubos, dusan.nemec, tomas.mravec}@fel.uniza.sk

[2] Faculty of Materials Science and Technology, Institute of Applied Informatics, Automation and Mechatronics, Slovak University of Technology, Trnava, Slovak Republic
pavol.bozek@stuba.sk

**Abstract.** The paper presents the designed algorithm, which is able to integrate of inertial sensor data into control algorithm. Autonomous operation of the mobile system requires reliable measurement of its position. Sources of such data are various; most commonly used is global satellite navigational system. However, this technique can be used only outdoors. For navigation inside building, under metal roof or underground only inertial or contact methods are available. This article analyzes possibilities of deployment of the inertial navigation in the control of the wheeled mobile platform. Experimental platform uses inertial measurement unit x-IMU manufactured by x-IO Technologies. According to our experiments inertial navigation can be reliably used only in fusion with other absolute sensors (odometers, magnetometers).

## 1 Inertial Navigation Principle

Inertial navigation uses sensors, in this case accelerometers and gyroscopes and is being used to locate objects e.g. inside buildings, in air and road traffic etc. Today, the inertial navigation systems is not used alone for short time because with the increasing time the error increases, too. Recent progress in the development is a fusion of GPS data into inertial systems using Kalman filtering. These systems can be supplemented by the barometric or magnetic sensors.

Inertial navigation [4, 5] computes position and attitude of the moving object with respect to its starting position in inertial (non-accelerating) frame of reference. With a small error we might consider surface of the Earth as an inertial base (the error will be discussed later). In the most general case the movement of the rigid object is described in six degrees of freedom – 3 degrees of translation and 3 degrees of rotation. In special case of the wheeled vehicle moving on the horizontal floor only three degrees of freedom are relevant:

- translation in two planar axes (North, East)
- rotation around vertical axis (Yaw)


J. Janech et al. (eds.), *Proceedings of the 2015 Federated Conference on Software Development and Object Technologies*, Advances in Intelligent Systems and Computing 511,
DOI 10.1007/978-3-319-46535-7_21

The other degrees of freedom are required if the surface of the floor is not planar (e.g. contains variable slopes and multiple levels).

## 2 Attitude Estimation

Attitude (rotation of the object around three axes) defines the transformation between the global coordinate system (bound with the Earth) and local coordinate system (bound with the vehicle). It can be computed in real time from the initial attitude and readings of 3-axial gyroscope (measures angular velocity in local system). Note that without knowing the initial attitude it is impossible to determine actual attitude. Inertial measurement unit x-IMU therefore contains 3-axial accelerometer for roll and pitch estimation and magnetometer for yaw estimation. Accelerometer measures the sum of the gravitational acceleration vector [7] (constant in global coordinate system, defines vertical direction) and the system's own acceleration (can be arbitrary but its mean value in long terms are zero). Magnetometer measures magnetic induction of the Earth's magnetic field (defines magnetic North). This configuration allows fully compensated attitude estimation. Basic sensor fusion [1] algorithms are implemented in default x-IMU firmware.

Attitude can be expressed in the form of Euler angles, rotational matrix or quaternion. Available x-IMU API provides conversion algorithms from quaternions to the other formalisms. Our algorithms use Euler angles in Z-Y-X convention (yaw-pitch-roll) because of their readability by humans and rotational matrix because it allows fast transformation between global and local system and can be easily used in MATLAB® environment. Coordinate systems (both global and local) are Cartesian with North-East-Down (abbr. NED) axis orientation.

## 3 Influence of the Earth's Rotation

Small portion of the error in yaw estimation is caused by rotation of the Earth around its own axis at angular velocity $\omega_{\text{Earth}} = 7.292115 \times 10^{-5}$ rad.s$^{-1}$ (approximately 15°/h). For wheeled vehicle the worst case occurs around Earth's poles, because vertical complement of the Earth's angular velocity is larger at higher latitude (see Fig. 1) and yaw estimation provided by magnetometer is less precise.

Since the angular velocity of the Earth's rotation is well known, readings from the gyroscope can be compensated by following:

$$\omega_{\text{comp}} = \omega_{\text{raw}} - \boldsymbol{R}.\omega_{\text{Earth}}[-\cos\phi \quad 0 \quad \sin\phi] \tag{1}$$

where $\boldsymbol{R}$ is rotational matrix expressing attitude with respect to local tangent plane in NED convention on the Earth's surface and $\phi$ is geographical latitude.

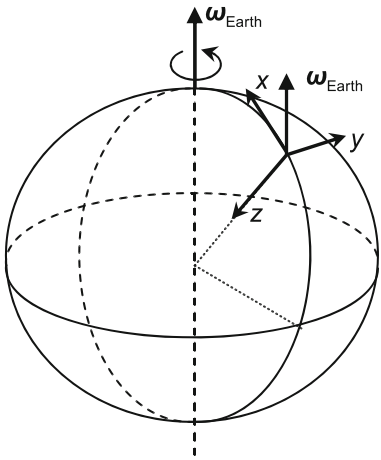



**Fig. 1.** Rotation of the Earth in local tangent plane

## 4 Position Estimation

According to the theory of inertia the position of the object can be integrated from acceleration of the object in local coordinate system $\boldsymbol{a}_{\text{local}}$, current rotational matrix $\boldsymbol{R}$, initial velocity $\boldsymbol{v}_0$ and position $\boldsymbol{d}_0$ in global inertial frame of reference. Algorithm of the integration is following:

1. Initialize state variables:
$$\boldsymbol{a} = \boldsymbol{0}, \boldsymbol{v} = \boldsymbol{v}_0, \boldsymbol{d} = \boldsymbol{d}_0 \tag{2}$$
2. Wait for new sample from accelerometer and AHRS (Attitude and Heading Reference System)
$$\boldsymbol{a}_{\text{accel}} = \boldsymbol{a}_{\text{local}} + \boldsymbol{R} \cdot \boldsymbol{g} \tag{3}$$
3. Compute new acceleration in global coordinate system:
$$\boldsymbol{a}_{\text{new}} = \boldsymbol{R}^{-1} \cdot \boldsymbol{a}_{\text{accel}} - \boldsymbol{g}, \tag{4}$$
where $\boldsymbol{g}$ is constant gravitational acceleration vector $\boldsymbol{g}$ = [0,0, g].
4. Compute new velocity in global coordinate system (trapezoidal integration):
$$\boldsymbol{v}_{new} = \boldsymbol{v} + (\boldsymbol{a}_{new} + \boldsymbol{a})\frac{T}{2}, \tag{5}$$
where $T$ is sampling period.
5. Compute new position in global coordinate system (trapezoidal integration):
$$\boldsymbol{d}_{new} = \boldsymbol{d} + (\boldsymbol{v}_{new} + \boldsymbol{v})\frac{T}{2} \tag{6}$$

6. Save state, then go back to step 2.

$$\boldsymbol{a} = \boldsymbol{a}_{\text{new}}\,,\ \boldsymbol{v} = \boldsymbol{v}_{\text{new}},\ \boldsymbol{d} = \boldsymbol{d}_{\text{new}} \tag{7}$$

Experimental results show that given theoretical algorithm is very sensitive to the bias of the accelerometer. Module x- IMU is using 12-bit accelerometer with full scale up to ±8 g. Bias equal to 1 LSB in one axis will produce velocity drift:

$$\boldsymbol{v}_{drift}(t) = \boldsymbol{a}_{bias} \cdot t = \frac{8\,\text{g}}{2^{12-1}} t \approx 0.04\,\text{m.s}^{-2} \cdot t \tag{8}$$

This drift is integrated into position:

$$\boldsymbol{d}_{drift}(t) = \boldsymbol{d}_{drift}(t-T) + \left[\boldsymbol{v}_{drift}(t) + \boldsymbol{v}_{drift}(t-T)\right]\frac{T}{2} \approx \frac{1}{2}\boldsymbol{a}_{bias}t^2 \tag{9}$$

Within $t$ = 100 s of the operation the considered minimal accelerometer bias will produce error of the position estimation $\boldsymbol{d}_{\text{drift}} \approx 200$ m. This error shows that using only accelerometer for position estimation is not reliable in longer term operation.

Real measured drift characteristics are shown in Fig. 2. Experimental results are slightly better than computed worst case.

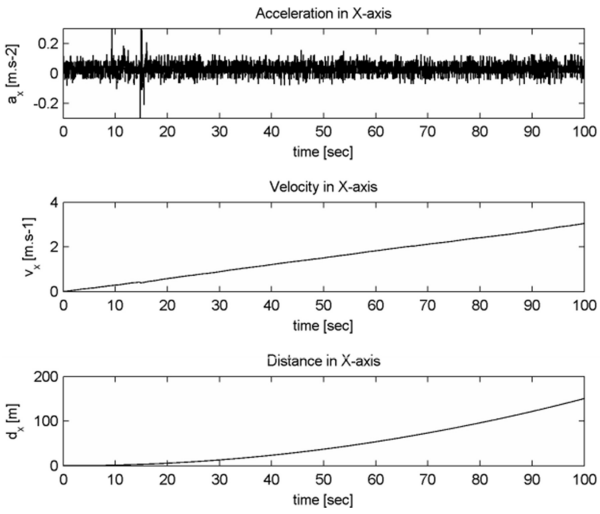



**Fig. 2.** Measured acceleration, integrated velocity and distance in steady state with respect to global frame of reference. Data are obtained right after sensor- calibration in order to eliminate thermal drift effects.

## 5 Position Estimation by Odometer

Since the double integration of the accelerometer data does not provide reliable information about distance, it is necessary to use absolute motion sensor [2, 6]. In case of wheeled mobile platform the simplest solution is to use odometer [3] bound with its wheels. We assume that wheels do not slide and difference of the distance run by wheels during turns is negligible.

Following algorithm was used for estimation of the mobile platform's position in three-dimensional space:

1. *Initialize position, reset odometer:*

$$\boldsymbol{d} = \boldsymbol{d}_0 \quad s = 0 \tag{10}$$

2. *Wait for new sample of rotational matrix* $\boldsymbol{R}$ *from AHRS.*
3. *Get current position of the odometer, compute change of odometer value:*

$$\Delta s = s_{\text{new}} - s \tag{11}$$

4. *Convert dislocation vector into global system:*

$$\boldsymbol{d} \leftarrow \boldsymbol{d} + \boldsymbol{R}^{-1}[ds \quad 0 \quad 0] \tag{12}$$

5. *Go back to step 2.*

Our experimental platform does not contain a real odometer. Since the propulsion DC electromotor moves platform at approximately constant forward speed $v_{\text{fwd}}$ we can replace measurement of the distance $s$ by measurement of electromotor on-time on $t$. Then it is valid:

$$\Delta s \approx v_{\text{fwd}} \cdot T \tag{13}$$

For improved performance of the "virtual odometer" it is possible to take the mass of the vehicle into account. The vehicle is powered by brushed DC electromotor with permanent magnets. Induced voltage in rotor winding is proportional to the rotation frequency $f$. Supply voltage $U_{\text{suppy}}$ (constant when electromotor is turned on) is equal to:

$$U_{suppy} = R_a \cdot I + \frac{f}{K_{\text{V}}} \tag{14}$$

where $R_{\text{a}}$ is the rotor winding resistance and $K_{\text{V}}$ is the rotation rate per 1 V.

Electromotor torque is proportional to its current $I$, therefore is valid:

$$\text{M} = K_M \cdot I = K_M \cdot \left( \frac{U_{suppy}}{R_a} - \frac{f}{K_V R_a} \right) = M_0 - C_{\text{M}} f \tag{15}$$

where $M_0$ is initial torque and $C_{\text{M}}$ is motor constant.

The electromotor is coupled with wheels by fixed gear, therefore the previous equation is valid also for propulsion force $F$:

$$F(t) = F_0 - C \cdot v_{\text{fwd}}(t) = \text{m}\frac{dv_{fwd}(t)}{dt} \tag{16}$$

where $F_0$ is static propulsion force and $C$ is a system constant. Since the propulsion force at maximal forward speed max v is zero we can estimate the value of the constant as:

$$C = \frac{F_0}{v_{max}} \tag{17}$$

Because the sampling period of the AHRS system is very short (maximal sampling frequency of the x-IMU module is 512 Hz), we can assume $dt = T$ and forward speed can be calculated incrementally in each step by following:

$$v_{\text{fwd}}[n+1] = v_{\text{fwd}}[n] + \frac{F_0 T}{m}\left(1 - \frac{v_{\text{fwd}}[n]}{v_{max}}\right) \tag{18}$$

In case when motor is turned off the formula is different:

$$v_{\text{fwd}}[n+1] = v_{\text{fwd}}[n] + \frac{F_0 T}{m n_{max}} \cdot v_{\text{fwd}}[n] \tag{19}$$

Previous equations allow real-time estimation of the robot forward speed without measurement of the actual wheels' speed.

## 6 Collision Detection

Great advantage of mounted inertial measurement unit is that it allows detection of the collisions of the vehicle with any obstacle. Onboard accelerometer can measure acceleration of the shock in two axes, therefore it is possible to estimate not just occurrence of the shock but also its direction. The shock will cause negative spike of the acceleration in $xy$ plane from direction $\delta$ (see Fig. 3).

$$\delta = \text{atan2}(-a_{\text{local y}}, -a_{\text{local x}}) \tag{20}$$

where $a_{\text{local x}}$, $a_{\text{local y}}$ are complements of the shock acceleration in local coordinate system.

Since the ability to accelerate of the vehicle is always limited by static friction coefficient $\mu$ between its tires and the floor ($\mu$ usually does not exceed 1), the maximal acceleration caused by vehicle propulsion or turning is given by:

$$a_{\text{max}} = \mu g \tag{21}$$

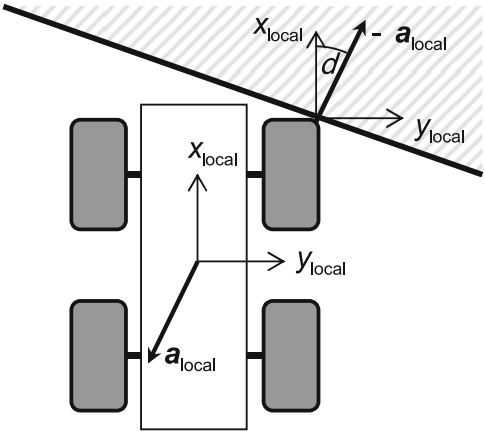



**Fig. 3.** Acceleration vector during collision with an obstacle

Threshold value of the local acceleration in *xy* plane for identifying collision should be set from 120% to 200% of $a_{max}$. Experiments show that acceleration during collision of our mobile platform with solid obstacle was approximately 8 g at full forward speed but only 2 g at 25% speed. Collision detection threshold was set to 1.2 g.

## 7 Control of the Mobile Platform

Our mobile platform was designed for massive production and allows only three-state control of the propulsion (forward – stop – backward) and three-state control of direction (leftstraight- right) (see Fig. 4). Control system is based on on-off regulator with hysteresis. User inputs several waypoints which the robot should visit (approach to it to some distance) in given order.

Every waypoint is defined in 2D space by Cartesian coordinates. We denote current waypoint as W = [$x_W$, $x_W$], current position as A = [$x_A$, $x_A$] and current heading as $\psi_A$. Distance vector to target is then:

$$D = W - A = [x_W - x_A, y_W, y_A] = [x_D, x_D], \tag{22}$$

Heading to target is:

$$\psi_A = \text{atan2}\ (x_D, x_D) \tag{23}$$

Difference between $\psi_D$ and $\psi_A$ expresses regulation error:

$$\psi_E = \psi_D - \psi_A \tag{24}$$

Note: In order to obtain angular difference $\psi_E$ from interval $(-\pi, \pi)$ it is necessary to remove period $2\pi$ out of the result:

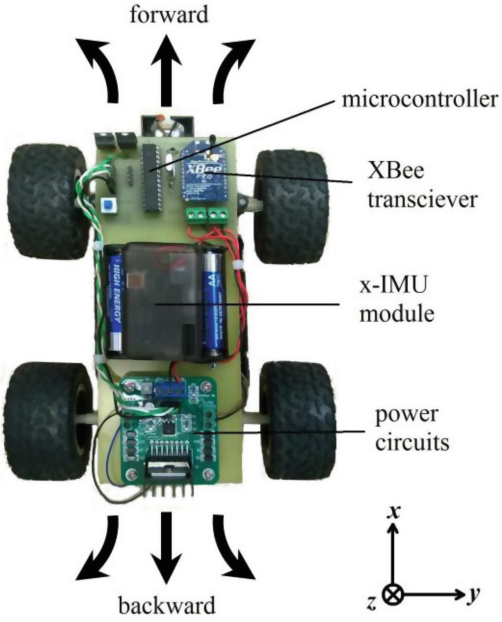



**Fig. 4.** Experimental mobile robot platform

$$\psi_{\mathrm{E}} \leftarrow \psi_{\mathrm{E}} - 2\pi \cdot \mathrm{round}\left(\frac{\psi_{\mathrm{E}}}{2\pi}\right) \tag{25}$$

Steering control algorithm is three-state on-off regulator with hysteresis. It is described by following state diagram (see Fig. 5).

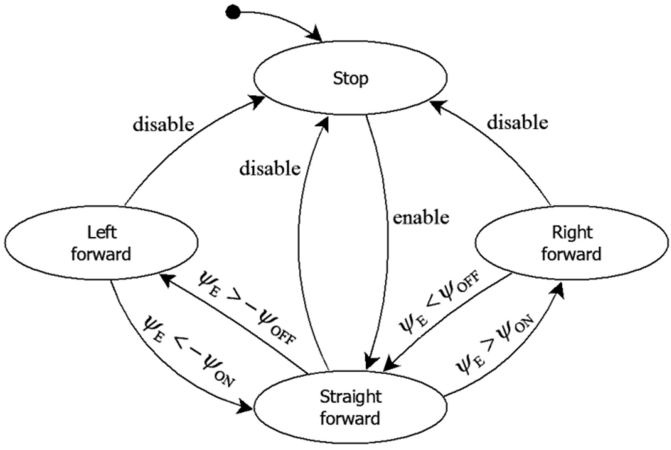



**Fig. 5.** State diagram of the simple control algorithm

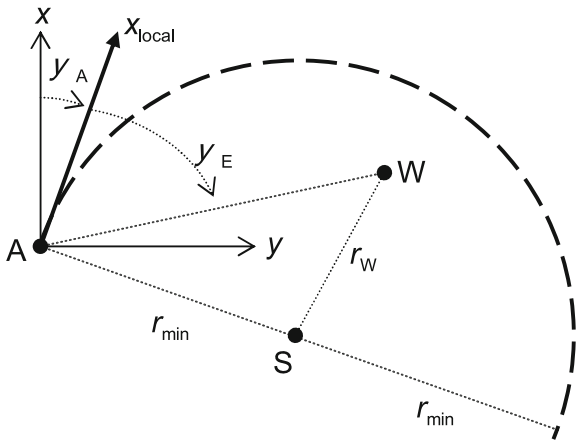



**Fig. 6.** Unreachable waypoint inside minimal radius trajectory

As can be seen, algorithm does not use backward movement. In case that robot is too close to the waypoint and heading error is too high (waypoint is inside arc with robot's minimal turn radius $r_{\min}$, see Fig. 6), algorithm will not ensure reaching the waypoint (robot will circle around waypoint).

According to Fig. 6 it is valid:

$$\mathrm{S} = \mathrm{A} + r_{\min}[-\sin\psi_{\mathrm{A}} \ \cos\psi_{\mathrm{A}}] \text{ when } \psi_{\mathrm{E}} > 0 \tag{26}$$

$$\mathrm{S} = \mathrm{A} + r_{\min}[\sin\psi_{\mathrm{A}} - \cos\psi_{\mathrm{A}}] \text{ when } \psi_{\mathrm{E}} < 0 \tag{27}$$

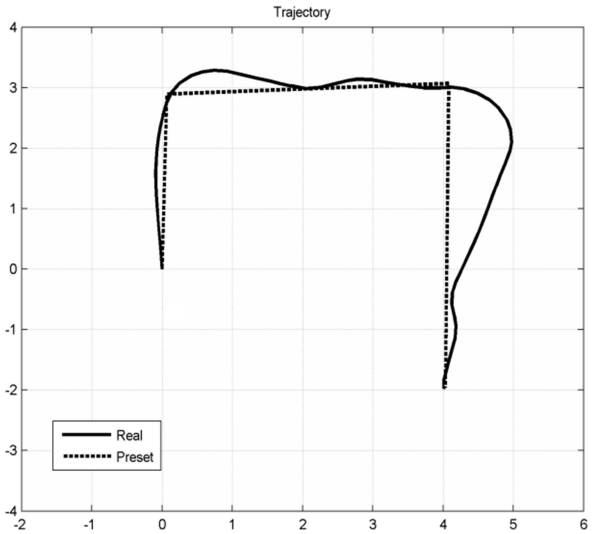



**Fig. 7.** Real trajectory of experimental mobile robot platform


**Fig. 8.** Real trajectory with bigger deviation

Distance $r_W$ is then computed as length of the segment SW = W − S. If the waypoint is unreachable ($r_W < r_{min}$), vehicle should reverse steering and electromotor direction until waypoint becomes reachable ($r_W > r_{min}$). In order to avoid oscillations it is necessary to use hysteresis in decision whether waypoint is reachable or not.

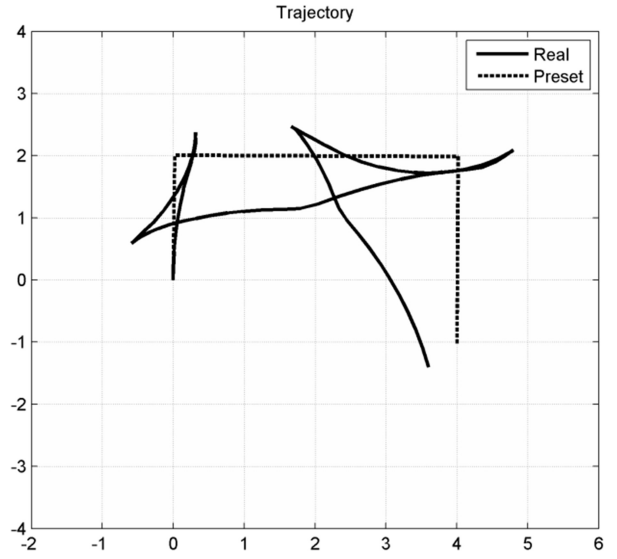



**Fig. 9.** Backward movement of experimental mobile robot platform

## 8 Experimental Results

We have applied the proposed algorithm for position control of experimental mobile robotic platform. In the Fig. 7 we can see trajectory of experimental mobile robot platform. Trajectory is defined using 4 points. As can be seen in Figure mobile platform passed all points very accurately even when we set permissible deviation from the real position.

In Fig. 8 is measurement with bigger deviation from the desired position.

In our approach is implemented algorithm which using backward movement for better results. This approach can be seen in Fig. 9.

The robot does not run in perfect circles. This effect is caused by inaccuracy of heading estimation by gyroscope (Fig. 10).

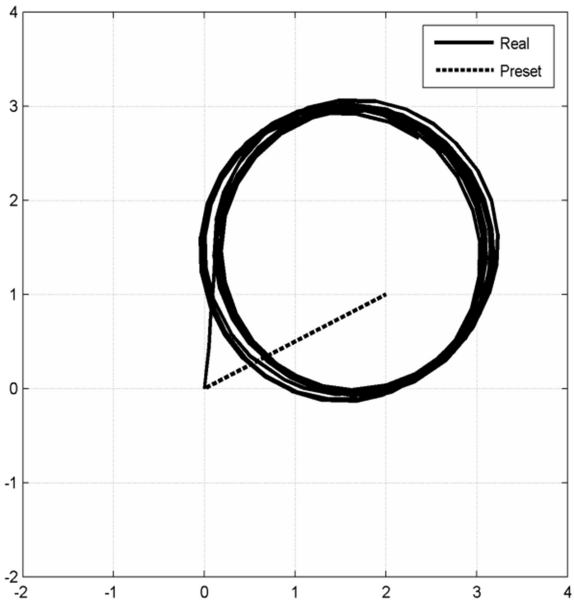



**Fig. 10.** Inaccuracy of position estimation using gyroscope

## 9 Conclusion

'Low-cost inertial sensors do not provide sufficient accuracy of the acceleration measurement therefore they cannot be used directly for position estimation. On the other hand, they can be used for very precise measurement of attitude and heading. This work designs, explains and verifies algorithm which allows integration of inertial sensor data (attitude and heading estimation) into control algorithm. This algorithm is designed universally and can be used for automatic control of movement of various devices such as mobile wheeled robots, mobile belt robots, boats or aircraft. Proposed

steering control algorithm can be parameterized according to the parameters and limitations of the real controlled vehicle (especially minimal turning radius and acceleration abilities). It also implements detection of the waypoint reachability and provides reversal control for wheeled platforms.

**Acknowledgment.** The contribution is sponsored by VEGA MŠ SR No 1/0367/15 prepared project "Research and development of a new autonomous system for checking a trajectory of a robot".

## References

1. Feng, S., Murray-Smith, R.: Fusing Kinect sensor and inertial sensors with multi-rate Kalman filter. In: IET Conference on Data Fusion & Target Tracking 2014: Algorithms and Applications (DF&TT 2014), pp. 1–8, 30 April 2014. doi:10.1049/cp.2014.0527
2. Pinney, C., Hawes, M.A., Blackburn, J.: A cost-effective inertial motion sensor for short-duration autonomous navigation. In: Position Location and Navigation Symposium, pp. 591–597. IEEE, 11–15 April 1994. doi:10.1109/PLANS.1994.303402
3. Sampaio, S., Massatoshi Furukawa, C., Maruyama, N.: Sensor fusion with low-grade inertial sensors and odometer to estimate geodetic coordinates in environments without GPS signal. IEEE Lat. Am. Trans. (Revista IEEE America Latina) **11**(4), 1015–1021 (2013). doi:10.1109/TLA.2013.6601744
4. Qasem, H., Gorgis, O., Reindl, L.: Design and calibration of an inertial sensor system for precise vehicle navigation. In: 2008 5th Workshop on Positioning, Navigation and Communication, WPNC 2008, pp. 229–231, 27 March 2008. doi:10.1109/WPNC.2008.4510379
5. Benser, E.T.: Trends in inertial sensors and applications. In: 2015 IEEE International Symposium on Inertial Sensors and Systems (ISISS), pp. 1–4, 23–26 March 2015. doi:10.1109/ISISS.2015.7102358
6. Šimák, V., Končelík, V., Hrbček, J., Folvarčík, J.: Realization and a real testing of the road–fee system. In: Proceedings of the 33rd International Conference on Telecommunications and Signal Processing, TSP 2010, Baden near Vienna, Austria, 17–20 August 2010. ISBN 978-963-88981-0-4
7. Hulsing, R.: MEMS inertial rate and acceleration sensor. IEEE Aerosp. Electron. Syst. Mag. **13**(11), 17–23 (1998). doi:10.1109/62.730613

# Measuring Maintainability of OO-Software - Validating the IT-CISQ Quality Model

Johannes Braeuer[1], Reinhold Ploesch[1(✉)], and Matthias Saft[2]

[1] Department of Business Informatics – Software Engineering Johannes Kepler, University Linz, Linz, Austria
{johannes.braeuer, reinhold.ploesch}@jku.at
[2] Corporate Technology Siemens AG, Munich, Germany
matthias.saft@siemens.com

**Abstract.** The Consortium for IT Software Quality (IT-CISQ) standard claims to provide valid measures as well as an assessment method that are suitable for properly measuring software quality. We implemented the measures for maintainability as specified by the IT-CISQ standard and wanted to find out whether both – the IT-CISQ defined assessment method and the IT-CISQ measures – are suitable for determining the maintainability of object-oriented systems. We identified a reference study that classifies the maintainability of eight open-source Java projects. This study follows a comprehensive measurement and assessment process and therefore can be used for validating the IT-CISQ approach. Due to the missing consideration of project size metrics, the IT-CISQ assessment method is not capable of properly determining the quality of projects. Even considering size metrics does not substantially enhance the result, which is an indicator that the measures proposed by IT-CISQ do not properly measure maintainability. Finally, our benchmarking approach was applied. It sets the measurements in relation to 26 projects that constitute the benchmark base. Despite a lack of statistical significance, the benchmarking results show a better correlation with the ranking published by the reference study. As our benchmarking approach is well validated, we can conclude that the measures proposed by IT-CISQ have to be considerably enhanced, i.e., additional measures have to be added to be able to determine the maintainability of object-oriented software projects.

## 1 Introduction

In the past a quality centric viewpoint has become increasingly important in software development. Nevertheless, the definition of software quality continues to be controversial and insufficiently understood. Thus, the quality of software products is often dissatisfactory, resulting in high economic impact. The consequences can be enormous and include not only spectacular failures, but also increased maintenance costs, high resource consumption, long test cycles, and delayed features [1]. To deal with this issue, software quality models focus on providing a systematic approach for modeling quality requirements. Moreover, some of the models not only propose approaches for assessing the current software quality status, but also for monitoring quality aspects as well as for improving the software based on different measures.


J. Janech et al. (eds.), *Proceedings of the 2015 Federated Conference on Software Development and Object Technologies*, Advances in Intelligent Systems and Computing 511,
DOI 10.1007/978-3-319-46535-7_22

Despite a high acceptance of quality awareness, there is still a gap between the abstract quality characteristics described in quality models and their concrete measurements [2]. Missing concrete measurements makes it difficult to explain the importance of quality problems to developers or sponsors and to quantify the potential of quality improvements. Furthermore, the relation between different quality assessment methods is not well studied [3].

The Consortium for IT Software Quality (IT-CISQ) quality model proposed by the Object Management Group (OMG) in 2012 [4] claims to close the gap between abstract quality characteristics as suggested by the ISO/IEC 25010 quality model and concrete measurements for the quality attributes maintainability, reliability, performance efficiency, and security [5]. Providing this necessary operationalization for the ISO/IEC 25010 standard pushes the IT-CISQ model in the spotlight of practitioners – either to have at least something to hold on to, or because of pressure imposed on organizations to follow this standard. It is therefore important for industry to determine whether the proposed standard is ready to be used.

In previous work we developed operational quality models that provide concrete measurements for typical software quality attributes [6, 7]. Moreover, the application of a benchmark-based assessment approach has been validated in several studies [8, 9]. These studies show that the assessment approaches have high external validity, so we were eager to validate the IT-CISQ standard with our understanding of quality assessment. A recently published work discusses the validation of maintainability-related measures of the IT-CISQ model. It highlights that the IT-CISQ quality model for maintainability could be used for quality evaluations, but is only partially suitable for improvement programs [10].

The research objective of this paper is to get more insights into the practical relevance of the IT-CISQ assessment method and its quality model for the ISO/IEC 25010 quality characteristic maintainability on object-oriented software. To achieve this objective, this paper relies on a measurement tool that has already proven its validity, as shown in [10]. Moreover, a reference study was identified, which evaluates the maintainability of multiple Java, C++, and C projects [11]. The reference study follows a comprehensive approach for quality evaluation (especially for maintainability); therefore, the results of the study can be well used to compare it with the IT-CISQ model application. Due to the fact that this work is exclusively dealing with the programming language Java, only the eight Java projects from the reference study were used for the validation. To judge whether the measures defined in the IT-CISQ quality model for maintainability can be reasonably used for assessing the maintainability of object-oriented software products, this paper first concentrates on a validation process that is guided by the evaluation method proposed in the IT-CISQ standard. Based on that result, a second validation experiment enhances the proposed method by using simple normalization with logical lines of code (LLOC) for making measurement results comparable and assessable. As both validation experiments did not lead to significant results, a third assessment approach (using the IT-CISQ measures) is applied as alternative to the IT-CISQ standard. It builds upon a benchmark suite consisting of 26 open-source Java projects, which is used to level the CISQ measures.

This paper is structured as follows: Sect. 2 presents an overview of the IT-CISQ quality model and a more detailed description of the maintainability-related parts.

Section 3 shows the approach that is used to operationalize the measurements specified by the IT-CISQ quality model for maintainability. In Sect. 4 the benchmarking-based approach is introduced, followed by a presentation of the reference study in Sect. 5. The main part of this paper is Sect. 6, which describes the validation of the IT-CISQ quality model for maintainability. Finally, Sect. 7 concludes the findings and provides an outlook for further work.

## 2 The IT-CISQ Quality Model

The goal of the IT-CISQ quality standard [4] is to provide specifications for measuring software quality. It is based on the ISO/IEC 25010 standard for software quality, meaning that IT-CISQ does not provide a model for specifying software quality but instead relies on the definitions provided by ISO/IEC 25010. However, the IT-CISQ model enhances the ISO/IEC 25010 standard by providing a specification of how to measure the defined quality attributes. It is claimed that this facilitates the definition of Service Level Agreements, the quantification of quality of business-critical applications, and supports the definition of benchmarks.

The standard comprises five sections. Section 1 introduces basic terms in the context of measurement. Section 2 defines terms with a heavy reuse of definitions from the ISO/IEC 25000 standards family. Section 3 defines requirements for implementing the standard. Section 4 is the core part of the standard and defines concrete measures for evaluating selected software quality attributes of the ISO/IEC 25010 standard. The current version 2.1 of the IT-CISQ standard covers the quality attributes reliability, performance efficiency, security, and maintainability. Finally, in Sect. 5 the standard defines a calculation model for the aggregation of individual measures to a single indicator for a quality attribute.

The IT-CISQ model does not refer to the next (finer) level of quality attributes. For example, the ISO/IEC quality attribute reliability is directly measured in the IT-CISQ model, and more differentiated measurement specifications for the sub-quality attributes of reliability (e.g., maturity or availability) are not provided. For the operationalization of quality attributes, the description of each operationalization follows the same structure. Thus, a number of issues are refined into quality rules that are operationalized by quality measure elements. Table 1 shows an example of an issue that is related to layered architectures (therefore substantiating the ISO/IEC 25010 quality attribute maintainability) and refined into two quality rules and three quality measure elements.

Quality measure elements are typically formulated as rules and the underlying metric is the number of artefacts in the source code that violate this rule. Subsequently, this work uses the general term measure, as the IT-CISQ standard also heavily relies on this phrase. Table 2 shows the measures that are defined for operationalizing software maintainability. As indicted by their description, many of these rules can only be applied in software that follows an object-oriented paradigm since they check the adequate usage of inheritance, polymorphism, and encapsulation.

**Table 1.** Issues, quality rules, and quality measure elements

| Issue | Quality rule | Quality measure Element |
|---|---|---|
| **Issue 1:** In a layered architecture, functions should strictly be allocated to layers and maintain a strict hierarchy of calling between layers (utility layers excepted) | **Rule 1:** Functions only communicate (exchange data) with functions belonging to an adjacent layer. Functions do not directly exchange data with functions that are not in an adjacent layer (no layer skipping/bridging) | **Measure 1:** # of functions that span layers **Measure 2:** # of layer-skipping calls |
| | **Rule 2:** Avoid too many horizontal layers | **Measure 3:** # of layers (threshold 4 <= 8) |

**Table 2.** IT-CISQ measures for maintainability obtained from [4]

| ID | Short description of measure |
|---|---|
| M1 | # functions that span layers |
| M2 | # of layer-skipping calls |
| M3 | # of layers (threshold 4 <= 8) |
| M4 | # files that contain 100 + consecutive duplicate tokens |
| M5 | # of unreachable functions |
| M6 | # of classes with inheritance levels > = 7 |
| M7 | # of classes with > = 10 children |
| M8 | # of instances of multiple inheritance of concrete implementation classes |
| M9 | # of methods that are directly using fields from other classes |
| M10 | # of variables declared public |
| M11 | # of functions that have a fan-out > = 10 |
| M12 | # of objects with coupling > 7 |
| M13 | # of cyclic calls between packages |
| M14 | # of functions with > 2% commented out instructions |
| M15 | # files > 1000 LOC |
| M16 | # of instances of indexes modified within its loop |
| M17 | # of GO TOs, CONTINUE, and BREAK outside the switch |
| M18 | # of functions with cyclomatic complexity > = a language specific threshold |
| M19 | # of methods with > = 7 data or file operations |
| M20 | # functions passing > = 7 parameters |
| M21 | # of hard coded literals except (–1, 0, 1, 2, or literals initializing static or constant variables) |

# 3 Measuring Maintainability

## 3.1 Assumptions for Maintainability Measures

This work measures maintainability as defined by the IT-CISQ quality model. Maintainability is chosen as evaluation target because it can have a significant impact on the costs of a software product [1]. This quality aspect influences the whole Software Development Life-Cycle (SDLC) and mainly the maintenance process [12, 13]. Furthermore, maintainability is selected because the reference study, which is used for validation, is considering the same viewpoint focuses on this quality attribute.

The previous section presented the overall structure of the IT-CISQ model, i.e., from issues to quality rules and finally to quality measure elements (measures for short). For the purpose of evaluation, this work only needs to concentrate on the section of measures defined for maintainability. Table 2 lists these measures as defined by the IT-CISQ standard. The rule number given in the first row is used in the remainder of the paper for reference.

As a first step for automating measurement, we analyzed the definitions and specifications of measures given in the standard. Then we tried to identify issues related to these specifications. The result of this analysis shows that these specifications are not exact enough for some measures:

M1, M2, M3: The standard makes the implicit assumption that each function is part of a layer. Furthermore, the specification leaves open under which circumstances a function spans a layer (e.g., just when other functions are called, or also when variables or constants are used from other layers). Additionally, layers defined in the architecture are sometimes not directly reflected in a package or directory structure. Hence, code often needs to be mapped manually to the according layer hierarchy.

M8: This measure is not generally applicable, as some object-oriented programming languages (e.g., Java, C#) do not offer multiple implementation inheritance.

M11: The specification does not provide details for calculating fan-out although there are different approaches available in the literature, e.g., [16, 17].

M12: The specification does not provide details for calculate coupling between objects although there are different approaches available in the literature, e.g., [14, 17].

M18: The language specific thresholds remain unspecified in the standard.

M19: Although the specification is clear for this measure, there is no mean to reliably detect all data or file operations in an arbitrary system since functions are often hidden in a utility layer. That is why it has to be specified manually.

Table 3 lists which assumptions we took in order to be able to provide automatic measurement for the specified measures of the IT-CISQ standard.

## 3.2 Implementation of the Measurement Tool

Figure 1 gives an overview of our tool architecture for the implementation of the necessary IT-CISQ measures. Understand is a commercial tool[1] that parses source code

[1] https://scitools.com/.

**Table 3.** Assumptions for rule automation

| | |
|---|---|
| *M1*<br>*M2*<br>*M3* | We assume that every function or method is always part of a layer. The implementation of a function or method is allowed to call functions or methods from *direct* upper or lower layers, but not from layers that are far away. In order to automatically calculate this measure, it has to be assured (by configuration) that each function or method (or its more high-level constructs like classes or packages) is assigned to exactly one layer. |
| *M8* | No meaningful implementation possible for Java. For this reason M8 is not considered anymore. |
| *M11* | The fan-out is calculated by summing up the number of called functions (methods) and the number of member variables set. |
| *M12* | For the calculation of the object coupling, we rely on the well-known metric *Coupling Between Object* (CBO) as specified in [14]. |
| *M18* | As a threshold value for the cyclomatic complexity, we assume a value of 10 as proposed by McCabe [15] and because of the widespread use of this threshold value. |
| *M19* | We allow explicit specification for each project which packages or classes contain data or file operations and that are therefore considered when calculating this measure. |

(e.g., Java Source code) and stores the extracted information about types, methods, method parameters, dependencies between classes and packages, etc. in a database. Furthermore, the tool Understand provides a Perl-based API for accessing this stored information.

MUSE (**M**USE **U**nderstand **S**cripting **E**ngine) is a library that we developed. This Perl library provides basic functionality for accessing the *Understand* database, for dealing with threshold values, for writing result files (in Fig. 1 denoted as *findings.xml*), as well as a set of helper functions to more easily access the information provided by *Understand* (e.g., getting the package of a class, getting the list of all public classes of a project, etc.).

Moreover, MUSE uses the module concept of Perl for grouping measures. These modules can be seen as small plugins.

In case a module is stored in a specific directory of MUSE, it can be loaded and the measures defined in this module can be executed. This facilitates the configuration of

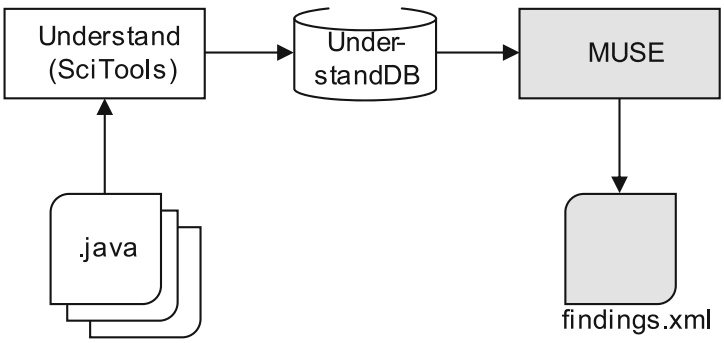



**Fig. 1.** Tool architecture for implementing IT-CISQ measures

the functionality of MUSE. For the implementation of the IT-CISQ measures, each of the 20 rules[2] is implemented as a separate Perl module in order to have maximum flexibility for the validation. The implementation of the 20 IT-CISQ rules has ∼2,000 LOC. Nevertheless, the implementation relies on some basic functionality provided by the MUSE framework as mentioned before.

## 4 Assessment Method for Maintainability

In the main part of this work, the validation of the IT-CISQ measures, we come to the conclusion that the proposed aggregation process from the IT-CISQ standard does not consider core assessment problems like different project sizes. As a consequence, we apply our benchmarking approach in the course of the validation. For keeping the validation section lean and concise, a presentation of our benchmarking method is given here.

Benchmarking is a well-established concept in many business areas where (similar) objects (e.g., products, services, processes, organizations) are checked against each other for specific purposes [18], e.g., for the evaluation of the value of an object or to derive suggestions for improvement.

In the context of software quality, Simon et al. [19] present a method for expressing the quality of a software product by calculating a single grade named the Code-Quality-Index (CQI). This method applies benchmarking for the calculation of the CQI by comparing metric values of a product with statistical data from a benchmark base. The benchmark base holds aggregated statistical data of more than 100 commercial software systems. However, the CQI calculation model suffers some conceptual, benchmarking-related, and technical deficiencies as investigated and described in [20].

Based on our experiences with the CQI, we developed an automatic assessment approach for software quality [8] that is based on the benchmarking concept. For this, in a first step we built up a benchmark database that stores the data of all reference projects included, i.e., projects that are used for comparison. This means that we store the results of the IT-CISQ quality measures for maintainability (see Sect. 3). It is sufficient in this context just to count for each rule the number of violations for each project in the benchmark base. Additionally, we store values of various size metrics for each project as needed for normalization; examples of size metrics are lines of code, number of methods, or number of classes.

Normalization of a measurement result deals with making raw results comparable with other projects. As an example, in a project with several thousand functions (e.g., 2,000), 20 undocumented functions might be just a negligible deficit. But in a project with only 50 functions, having 20 undocumented ones might be unacceptable. From this, the normalization operation eliminates the influence of the size of projects on the measurement results. After having applied normalization strategies for each measure,

[2] IT-CISQ specifies 21 measures for maintainability, but measure M8 is not applicable for Java projects (see Table 3).

the normalized data are comparable with data from other projects (e.g., 1% undocumented functions in project A versus 40% in project B).

Based on the benchmark database that can be used for several assessments, a benchmark suite needs to be generated in a second step. A benchmark suite usually is specially tailored to a specific application of benchmarking and consists of only those projects that are selected as reference projects, e.g., limited to projects of the embedded systems domain, which are written in the C programming language. In our context, the benchmarking suite consists of 26 open-source Java projects (more details will be given in Sect. 5).

To create the benchmark suite, we calculate a value distribution for each measure on the basis of their normalized measurement values of the selected projects. These value distributions reflect commonly used statistical figures, e.g., quartiles, quintiles, deciles, or percentiles.

A quartile-based distribution, as shown in Fig. 2, divides the values into four areas, Q1-area to Q4-area, where Q1-area and Q4-area are delimitated with the minimum and maximum value of the suite projects. If an investigated project is better than the benchmark minimum it is in the additional Q0-area. Vice versa, this also applies to the maximum value.

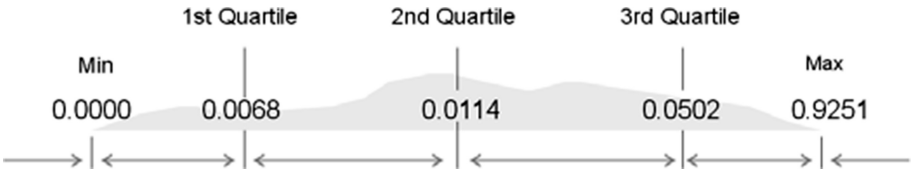



**Fig. 2.** Quartile-based value distribution for one rule (example) [8]

In order to calculate an average quality score for the project, we consider all rules/metrics and average out the number of rules/metrics falling into each quality area from Q0-area to Q5-area. Moreover, each of the Q-result has a value assigned; the better the area, the higher the number. For example, a Q-result of Q0 will get 5 points while a result of Q5 will get 0 points. Summing up the points for all rules and dividing this sum by the total number of selected rules produces the quality score. The quality score calculated based on this formula ranges from 5 to 0, with 5 representing the highest score.

## 5 Reference Study

For the validation of the IT-CISQ quality model, this paper relies on a referential data set that is published in [11]. In [11] the authors use their own quality model to certificate different open source software products. This applied quality model is based on the ISO/IEC 9126 standard [21], the predecessor of the ISO/IEC 25000 standard. Moreover, the quality model is restricted to maintainability aspects only. In other words, it maps a set of source-code measures onto the sub-characteristics of the quality aspect maintainability as defined in ISO/IEC 9126 [21].

To demonstrate the certification method based on the quality model, the referenced study uses 17 open-source projects. This set is composed of eight Java, four C/C++, and five C products. Since our work is exclusively dealing with the programming language Java, Table 4 contains only the eight Java projects including their version number, logical lines of code, and calculated maintainability value. The latter ranges from –2 to 2 and is responsible for the ranking of the projects. Hence, the project with the highest value is considered to be the most maintainable one and is therefore ranked at the top end.

The maintainability value, as shown in the last column, is calculated based on different maintainability concerns. In detail, the four sub-characteristics analyzability, changeability, stability, and testability are individually assessed for each project. Furthermore, source code duplications, test quality, complexity, unit size, and volume are five additional properties that affect the classification process proposed in [11]. In order to operationalize these different aspects that influence the maintainability value, the classification process relies on both automatic and manual measures. A software analysis toolkit, for instance, automatically delivers source code metrics whereas a manual and structured review process provides information on the test quality. At the end, all results from the different maintainability viewpoints flow together and form the maintainability value of a project. Maintainability is therefore measured comprehensively and the study can serve as reference for judging the accuracy of the IT-CISQ measures for maintainability.

According to Table 4, Axion 1.0 has the highest maintainability value compared with the other Java projects. In other words, Axion is the product that best fulfills maintainability requirements based on the quality model published in [21]. The last place of the ranking is taken by Tomcat 6.0.14. For clarification, JalistoPlus 2.0 and Derby 10.3.2.1 have the same maintainability value so that they share the same rank. In the reminder of this paper, this ranking of the projects is referenced several times in order to link the result of this study with the validation approaches discussed later.

**Table 4.** Reference projects obtained from [11]

| Rank | Product | Version | LLOC | Maintainability value [–2, 2] |
|---|---|---|---|---|
| 1 | Axion | 1.0 | 18,921 | +0.54 |
| 2 | H2 | 1.0.69 | 72,102 | +0.08 |
| 3 | SmallSQL | 0.19 | 18,402 | –0.38 |
| 4 | JalistoPlus | 2.0 | 21,943 | –0.54 |
| 4 | Derby | 10.3.2.1 | 307,367 | –0.54 |
| 6 | Jaminid | 0.99 | 1,120 | –0.63 |
| 7 | HSQLDB | 1.8.0.9 | 69,302 | –0.70 |
| 8 | Tomcat | 6.0.14 | 164,199 | –1.08 |

## 6 Execution and Validation

In order to validate the IT-CISQ measures for maintainability, this work uses the eight reference projects and calculates the number of violations for each maintainability measure specified in Sect. 3.1. Therefore the previously mentioned measurement tool

supports this initial data gathering process by computing the violations for each project in an almost automated manner. The result of this first step (without measures M1, M2, M3, and M19) is shown in Table 5.

The data summarized in Table 5 will be used for the further validation of the IT-CISQ measures. For doing so, this article applies three different validation experiments listed below. The goal of each of these experiments is to figure out whether applying IT-CISQ obtains a similar ranking as the one obtained by the reference study.

- Validation applying IT-CISQ standard
- Validation using normalization of measures with LLOC
- Validation using benchmarking approach

**Table 5.** Distribution of violations

| ID | Jaminid 0.99 | SmallSQL 0.19 | JalistoPlus 2.0 | Axion 1.0 | HSQLDB 1.8.0.9 | H2 1.0.69 | Tomcat 6.0.14 | Derby 10.3.2.1 |
|---|---|---|---|---|---|---|---|---|
| M4 | 0 | 4 | 1 | 2 | 9 | 12 | 29 | 37 |
| M5 | 1 | 0 | 1 | 3 | 18 | 1 | 29 | 98 |
| M6 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| M7 | 1 | 3 | 1 | 2 | 0 | 6 | 13 | 12 |
| M9 | 4 | 107 | 6 | 32 | 726 | 267 | 376 | 2920 |
| M10 | 0 | 0 | 2 | 28 | 90 | 59 | 240 | 598 |
| M11 | 10 | 101 | 198 | 174 | 452 | 565 | 1205 | 1809 |
| M12 | 9 | 59 | 106 | 100 | 166 | 282 | 546 | 819 |
| M13 | 1 | 0 | 61 | 35 | 25 | 189 | 78 | 404 |
| M14 | 6 | 11 | 52 | 20 | 157 | 93 | 224 | 396 |
| M15 | 0 | 4 | 0 | 5 | 30 | 14 | 58 | 131 |
| M16 | 0 | 1 | 0 | 2 | 6 | 5 | 11 | 41 |
| M17 | 2 | 39 | 6 | 59 | 266 | 325 | 536 | 629 |
| M18 | 2 | 72 | 10 | 61 | 182 | 189 | 461 | 644 |
| M20 | 0 | 2 | 0 | 1 | 16 | 18 | 48 | 448 |
| M21 | 286 | 1751 | 1467 | 3129 | 6810 | 10360 | 23184 | 33289 |
| Sum | 322 | 2150 | 1910 | 3651 | 8944 | 12373 | 27009 | 42239 |

## 6.1 Experiment 1 – Validation Applying IT-CISQ Standard

**(1) Design**

The first validation experiment is based on the evaluation method provided by the IT-CISQ standard and uses the quality characteristic calculation process proposed in [4]. This process simply sums up the number of violations to a single figure. For this task, the product is considered as a whole and not divided into layers, components, or other structuring concepts. Moreover, weighting the measures is also not recommended by the standard and therefore was not applied [4].

Due to the fact that the IT-CISQ standard deals with the four quality characteristics (reliability, performance efficiency, security, and maintainability), there may be concerns that the quality calculation process includes all of the four characteristics. However, IT-CISQ points out that the quality characteristic score can be calculated separately for each quality characteristic [4], which will be maintainability in this experiment.

**(2) Result and Discussion**

By executing the IT-CISQ specified aggregation process, it yields the ranking for the eight reference products as depicted in Table 6.

**Table 6.** Ranking based on IT-CISQ standard

| | Jaminid 0.99 | JalistoPlus 2.0 | SmallSQL 0.19 | Axion 1.0 | HSQLDB 1.8.0.9 | H2 1.0.69 | Tomcat 6.0.14 | Derby 10.3.2.1 |
|---|---|---|---|---|---|---|---|---|
| #CISQ findings | 322 | 1910 | 2150 | 3651 | 8944 | 12373 | 27009 | 42239 |
| LLOC | 1107 | 21939 | 18341 | 23821 | 65378 | 72154 | 158067 | 307127 |
| Ranking | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| Ref. | 6 | 4 | 3 | 1 | 7 | 2 | 8 | 4 |

The eight projects in Table 6 are ordered according to their obtained rank highlighted in grey. Thus, Jaminid 0.99 with the highest rank is placed on the left and Derby 10.3.2.1, as the project with the lowest rank, is placed on the right. When comparing the second row (#CISQ findings) and third row (LLOC) of the table, it is obvious that the number of findings increases with the number of LLOC. Consequently, the ranking based on the IT-CISQ specified aggregation process is strongly related to the project size. The only exception is SmallSQL 0.19, as the second smallest product, which has more violations than JalistoPlus 2.0 that represents the third smallest product.

In further discussing the result of this validation approach, it is necessary to calculate the Spearman (rank correlation) correlation coefficient since it expresses the relationship between two variables using a monotonic function. The calculated correlation coefficient ($r_s$) has a value of 0.1073. As the sample size is small (eight projects), the correlation coefficient has to have a value of 0.738 ($\alpha = 0.05$) or 0.881 ($\alpha = 0.01$) to indicate a significant correlation. Consequently, a value of 0.1073 shows that there is no relationship between the ranking based on the IT-CISQ standard and the ranking from the reference study.

One reason for this discrepancy is of course that the generic configuration for the IT-CISQ measures M1, M2, M3, and M19 could not be applied. For measuring M1, M2, and M3 this experiment would rely on a software architecture specification that defines the layering structure for each product. However, such a document could not be identified for any of the products after conducting Internet research.

Finishing the discussion of the experiment result, the validation of the IT-CISQ standard-based calculation process shows that this approach cannot be applied for ranking products according to their IT-CISQ measures. Although the IT-CISQ standard

mentions the use of normalization techniques to support comparisons for benchmarking [4], this idea is not further discussed in the standard. Consequently, in the second experiment this article continues with a validation approach that uses the LLOC to normalize the number of violations.

**(3) Threats to Validity**

Threats to internal validity concern our selection of projects, tools, and the analysis method. Thus, one threat to the internal validity of this experiment is the small set of eight reference projects. In order to explicitly address this issue, a high minimum correlation coefficient is defined in order to accept a correlation as significant (e.g., 0.738 for $\alpha = 0.05$).

In addition to the small set, another internal threat focuses on the problem that two reference projects share the same rank. More precisely, JalistoPlus 2.0 and Derby 10.3.2.1 are ranked in fourth place as shown in Table 4. Therefore the fifth rank is not occupied and the computation of the correlation coefficient is slightly influenced. In other words, it is not possible to achieve a perfect correlation coefficient of value 1 as long as no validation approach does rank the two reference projects on the fourth rank. To verify the low influence of this internal threat, the correlation coefficient was recalculated based on a ranking that occupied the fifth rank with both projects. As a result, neither of the recalculations came up with a correlation coefficient value (e.g., $r_s = 0.047$ and $r_s = 0.194$) that would lead to an acceptance of the IT-CISQ-based approach.

Threats to external validity affect the possibility of generalizing the results. As we were not able to positively validate the entire set of measures (M1, M2, M3, and M19 were excluded in advance), this experiment only partly shows the application of the IT-CISQ standard. Moreover, it is obvious that M8 does not work for Java products since multiple inheritance is not supported. The same applies to C#. Four projects (Jaminid, Jalisto Plus, HSQLDB, and Tomcat) potentially could obtain a ranking closer to the reference study, as considering the four excluded measures would increase the number of CISQ findings. However, considering these four measures would also mean for three projects (Axion, H2, Derby) that their distance to the reference projects gets even worse. Overall, considering these four measures would therefore not lead to a significant correlation between the IT-CISQ ranking and the ranking of the reference study.

Threats to reliability concern the possibility of replicating this experiment. We attempt to provide all the necessary details to replicate this experiment. In addition, the eight reference projects are publicly available. However, for a full re-execution of the validation, the architecture of the measuring tool has to be rebuilt. This means that the commercial tool Understand must be acquired and the MUSE library, which implements the IT-CISQ measures, must be used.

## 6.2 Experiment 2 – Validation Using Normalization of Measures with LLOC

**(1) Design**

Normalization is a necessary concept when measures, which count the number of rule violations, are compared with each other [22], especially when the analyzed projects

differ substantially in size. Consequently, the second validation approach relies on this idea and uses normalization for making measurement results comparable with other projects. In this particular experiment the LLOC are used to normalize the findings obtained from the IT-CISQ standard-based calculation process, which is shown in Sect. 6.1.

**(2) Result and Discussion**

Table 7 summarizes the data required for the normalization. It shows the number of violations and LLOC for each product. By putting both values into relation, the normalized value can be calculated as depicted in the third row. Consequently, the ranking represents the ascending order of the normalized number of violations. For the sake of completeness, the ranking from the reference study is shown in the last row.

Like in the discussion of the previous validation approach, the correlation coefficient between the ranking obtained by the normalization and the reference study is an important indicator for further assessment. Although the ranking is completely different compared to the first experiment, the correlation coefficient between the new ranking and the referenced ranking is exactly the same as the correlation coefficient computed in the previous validation, namely 0.1073. For the discussion of this result see Sect. 6.1.

Since normalization does not provide additional knowledge regarding the validation of the IT-CISQ measures, we believe that a simple normalization operation is not sufficient in this context. Moreover, combining the normalized values of different measures within an assessment is still questionable as they have different scales and units. Therefore, it is required to use evaluation functions in order to transfer the normalized measurement values of all measures into evaluated values with a defined semantics [22]. Such an approach, which follows a separate consideration of the IT-CISQ measures, is applied in the next experiment.

**(3) Threats to Validity**

Due to the fact that this experiment is building on the experiment presented in Sect. 6.1, the same threats to validity remain valid. Nevertheless, an additional internal threat to validity is the issue regarding the projects' LLOC. When comparing the numbers of LLOC in Table 4 with the values our tool obtained and represented in Table 7, a mismatch can be identified; especially, for project Axion 1.0, which has a high derivation with additional 25% of LLOC. Although there was the attempt to reduce the source files to the referenced set, this was not possible without losing the ability to build the product. One reason for the LLOC mismatch could be a different implementation of the counting algorithm that was applied.

### 6.3 Experiment 3 – Validation Using Benchmarking Approach

**(1) Design**

The core of this validation is to apply the maintainability measures of IT-CISQ to the eight projects using our benchmarking approach explained in Sect. 4. To prepare this benchmarking approach, it is necessary to apply our IT-CISQ measures to a benchmark

**Table 7.** Ranking normalized with LLOC

| | JalistoPlus 2.0 | SmallSQL 0.19 | HSQLDB 1.8.0.9 | Derby 10.3.2.1 | Axion 1.0 | Tomcat 6.0.14 | H2 1.0.69 | Jaminid 0.99 |
|---|---|---|---|---|---|---|---|---|
| #CISQ findings | 1911 | 2154 | 8953 | 42276 | 3653 | 27038 | 12385 | 322 |
| LLOC | 21939 | 18341 | 65378 | 307127 | 23821 | 158067 | 72154 | 1107 |
| Norm. | 0.087 | 0.117 | 0.137 | 0.137 | 0.153 | 0.171 | 0.172 | 0.291 |
| Ranking | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| Ref. | 4 | 3 | 7 | 4 | 1 | 8 | 2 | 6 |

suite composed of 26 open-source Java projects. This provides valid benchmarking distributions for all IT-CISQ measures. After that, for all reference projects all CISQ rule results are benchmarked against this distribution. This means, that for every rule the Q-area is determined for each reference project as shown in Sect. 4. Since this assignment task is implemented as a feature of the measurement tool, it can be done automatically. Afterwards, the quality score can be calculated by the arithmetic mean of the weight of all Q-areas.

**(2) Result and Discussion**

Table 8 depicts the result of the benchmarking-based approach for the eight reference projects without measuring M1, M2, M3, and M19. The row QS is the direct result using the benchmarking calculation method, i.e. for Q0 5 points, for Q1 4 points, for Q2 3 points, for Q3 two points, for Q4 one point and for Q5 zero points. Based on that value, the ranking is derived as shown in the penultimate row.

Without an in-depth analysis of Table 8, a first glance shows that the projects are almost ranked according to their LLOC. Only the first three projects do not follow this order. Especially, JalistoPlus represents an outlier under this consideration since it has the highest QS value, but is not the smallest project. The same effect regarding JalistoPlus can be seen in Table 7 as well.

**Table 8.** Ranking based on benchmarking approach

| | JalistoPlus 2.0 | SmallSQL 0.19 | Jaminid 0.99 | Axion 1.0 | HSQLDB 1.8.0.9 | H2 1.0.69 | Tomcat 6.0.14 | Derby 10.3.2.1 |
|---|---|---|---|---|---|---|---|---|
| Q0 (5) | 6 | 6 | 7 | 3 | 4 | 3 | 3 | 2 |
| Q1 (4) | 4 | 0 | 0 | 1 | 0 | 1 | 0 | 0 |
| Q2 (3) | 1 | 3 | 0 | 2 | 2 | 0 | 1 | 1 |
| Q3 (2) | 1 | 2 | 1 | 4 | 3 | 4 | 4 | 4 |
| Q4 (1) | 5 | 4 | 7 | 6 | 6 | 8 | 7 | 7 |
| Q5 (0) | 0 | 2 | 2 | 1 | 2 | 1 | 2 | 3 |
| *QS* | *3.29* | *2.76* | *2.59* | *2.29* | *2.24* | *2.06* | *1.94* | *1.65* |
| LLOC | 21939 | 18341 | 1107 | 23821 | 65378 | 72154 | 158067 | 307127 |
| Ranking | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| Ref. | 4 | 3 | 6 | 1 | 7 | 2 | 8 | 4 |

**Table 9.** Mapping of Q-area and maintainability value

| Q-area | Maintainability value range |
|---|---|
| Q0 | ]1.33, 2] |
| Q1 | ]0.66, 1.33] |
| Q2 | ]0, 0.66] |
| Q3 | ]–0.66, 0] |
| Q4 | ]–1.33, –0.66] |
| Q5 | [–2, –1.33] |

**Table 10.** Mapping of project to Q-area

| Project | Maintainability value [–2, 2] | Q-area |
|---|---|---|
| Axion 1.0 | +0.54 | Q2 |
| H2 1.0.69 | +0.08 | Q2 |
| SmallSQL 0.19 | –0.38 | Q3 |
| JalistoPlus 2.0 | –0.54 | Q3 |
| Derby 10.3.2.1 | –0.54 | Q3 |
| Jaminid 0.99 | –0.63 | Q3 |
| HSQLDB 1.8.0.9 | –0.70 | Q4 |
| Tomcat 6.0.14 | –1.08 | Q4 |

Like in the first and second experiment, the Spearman correlation coefficient is used for expressing the strength of the linear relationship between the obtained ranking and the ranking from the reference study. The calculated correlation coefficient ($r_s$) has a value of 0.2265. Even though the correlation is higher than in the aforementioned studies, this is no significant correlation as the coefficient has to have a value of at least 0.738 for $\alpha = 0.05$ (Table 11).

**Table 11.** Adapted distribution of IT-CISQ measures

| | JalistoPlus 2.0 | SmallSQL 0.19 | Jaminid 0.99 | Axion 1.0 | HSQLDB 1.8.0.9 | H2 1.0.69 | Tomcat 6.0.14 | Derby 10.3.2.1 |
|---|---|---|---|---|---|---|---|---|
| Q0 (5) | 6 | 6 | 7 | 3 | 4 | 3 | 3 | 2 |
| Q1 (4) | 4 | 0 | 0 | 1 | 0 | 1 | 0 | 0 |
| Q2 (3) | 1 | 3 | 0 | 6 (+4) | 2 | 4 (+4) | 1 | 1 |
| Q3 (2) | 5 (+4) | 6 (+4) | 5 (+4) | 4 | 3 | 4 | 4 | 8 (+4) |
| Q4 (1) | 5 | 4 | 7 | 6 | 10 (+4) | 8 | 11 (+4) | 7 |
| Q5 (0) | 0 | 2 | 2 | 1 | 2 | 1 | 2 | 3 |
| *QS* | *3.05* | *2.62* | *2.48* | *2.43* | *2.0* | *2.24* | *1.76* | *1.71* |
| Ranking | 1 | 2 | 3 | 4 | 6 (+1) | 5 (–1) | 7 | 8 |
| Ref. | 4 | 3 | 6 | 1 | 7 | 2 | 8 | 4 |

**(3) Threats to Validity**
In contrast to the previous validation approaches, this experiment builds on a benchmark distribution, which represents a threat to external validity. The 26 projects used for calculating the benchmark distribution may not reflect the reality of projects in this domain. Nevertheless these projects differ in their size, which means that they are ranging from 7,221 to 560,017 LLOC. As a result, all of the eight reference projects – except Jaminid, which is significantly smaller – are covered by this range. Furthermore the trustworthiness of the benchmark distribution has been proven in previous investigations [6, 8]. Based on the experience gained from these investigations, the benchmark distribution is considered to be reliable. Nevertheless, the external validity of this experiment could be enhanced by applying the quality model to industrial projects. Unfortunately, a larger set of industrial projects is not accessible.

The threat to validity discussion in the first experiment shows that one threat to external validity is the neglect of the IT-CISQ measures M1, M2, M3, and M19. In order to understand this external threat, the following discussion should highlight its influencing factor. In order to assess the influence of the threat, we hypothetically assume that the four missing measures affect the ranking to the same extent as the reference study assesses the projects.

Therefore, the first task is to map the maintainability value of each project to a Q-area used by the benchmarking approach. To support this task, Table 9 shows the mapping between a Q-area and the range in which the maintainability value has to fall. By using this mapping, each project gets assigned to a Q-area as depicted in Table 10. Based on that assignment, the four missing IT-CISQ measures can flow into the calculation of the ranking. The result of the modified benchmarking approach is shown in Table 10. This result highlights that the four measures M1, M2, M3, and M19 slightly change the ranking by switching the rank of HSQLDB and H2. Consequently, the Spearman correlation coefficient rises from 0.2265 to 0.3458, but is still far from being significant.

Already discussed as an external threat, but also relevant as a threat to the reliability of this experiment is the fact that a pre-calculated benchmark distribution is used. Consequently, a complete re-execution of this validation requires the calculation of the distribution including the following projects: Ant (1.7.0), ArgoUML (0.24), Azureus (3.0.4.2), FreeMind (0.8.1), GanttProject (2.0.6), hsqldb (1.8.0.9), JasperReports (2.0.5), jDictionary (1.8), JEdit (4.2), JFreeChart (1.0.9), jose (1.4.4), Junit (4.4), Lucene (2.3.1), Maven (2.0.7), OpenCMS (7.0.3), OpenJGraph (0.9.2), OurTunes (1.3.3), Pentaho (1.6.0), PMD (4.1), Risk (1.0.9.7), Spring (2.5), Tomcat (6.0.16), TuxGuitar (0.9.1), Weka (3.5.7), XDoclet (1.2.3), and XWiki (1.3).

## 7 Conclusion and Future Work

Motivated by our previous work [10] that successfully demonstrates the operationalization of the maintainability part of the IT-CISQ quality model, we were eager to verify our approach on a reference study. Hence, we used the implementation of IT-CISQ to determine whether the IT-CISQ standard specifies measures that can serve

as reliable predictors for the maintenance of object-oriented software and lead to a similar ranking of projects like the reference study.

For conducting this research, we first applied the IT-CISQ standard for the quality criterion maintainability on eight open-source Java projects, which were selected according to the reference study. Based on that result, the first experiment used the evaluation method provided-as-is by the IT-CISQ standard to evaluate the reference projects. This standard simply sums up the number of violations for deriving an evaluation for a particular project. Due to the fact that the sum is not normalized with some kind of size metrics, the first experiment comes to a ranking that correlates with the size of the projects. In other words, the software quality of a project is determined by the size of the product itself. Even though a first glance shows that the ranking of the projects is totally different to the reference study, we calculated the correlation of both rankings. Consequently, we came to the conclusion that the assessment method is not applicable for an absolute estimation of project quality (star ranking, school grades, etc. Furthermore, it is not suitable for quality comparison of projects of different sizes.

The second experiment focuses on the effect of normalization on the results of the IT-CISQ measurements. Without changing the design, the second experiment continues from the first experiment by using the LLOC of each project to normalize the number of violations. As a result, the size – defined by the LLOC – does not play a role anymore and the calculated IT-CISQ measures become comparable with each other. Although these ratio values do not contain the size factor, the obtained ranking does not significantly correlate with the result of the reference study. As normalizing measurement values for evaluation purposes is a valid approach (see [2, 6–8]), we draw the conclusion that the IT-CISQ standard-based evaluation approach cannot be reasonably used to assess the maintenance quality of Java software projects. As the second experiment eliminates the problem of normalization, we cautiously can conclude that the measures are not sufficient to predict the quality (limited to maintainability) of software projects.

In order to eliminate the impact of the (simple) aggregation method suggested by IT-CISQ and to be able to concentrate on the predictive power of the IT-CISQ measures, the third experiment uses a benchmarking-based approach that has already proven successful in [8–10]. Using this benchmarking-based approach means that the result of the IT-CISQ measures is compared with a benchmark distribution that defines six differently weighted Q-areas for each measure. Depending on the number of violations, each IT-CISQ measure becomes assigned to a Q-area for a selected project. Finally, the arithmetic mean of the weight of all Q-areas determines the software quality and the ranking of the projects. Despite a better correlation with the ranking published by the reference study, the result is not statistically significant. This strengthens our conclusion that the measures specified by IT-CISQ are not sufficient to predict the quality (limited to maintainability) of object-oriented software.

Based on the results of the three studies, this paper shows that the IT-CISQ model is weak in predicting the maintainability of software projects. This statement is underlined by the fact that both, the reference study as well as the IT-CISQ standard, define software quality measures for maintainability according to the same ISO/IEC 25000 standard and its predecessor. In discussing the differences between the quality model of the reference study and the IT-CISQ standard, it can be highlighted that the former

considers maintainability from some different quality viewpoints. Thus, the viewpoints of analyzability, changeability, stability, and testability are taken into consideration [4]. In order to assess these aspects, the proposed quality model defines the tasks of conducting code inspection, structured documentation review, and analysis of the test coverage [4]. Compared with the IT-CISQ standard, this means that the reference study concentrates on a broader project context because IT-CISQ does not consider project documents or test objects at all. Instead, the IT-CISQ measures are limited to the source code for building the product and documents for defining the layering structure. Obviously, the approach proposed by the reference study is more comprehensive and gives better hints for improvements. Nevertheless, we would expect from IT-CISQ measures at least to properly predict the maintenance quality of software projects.

The fact that a reasonable software maintainability assessment requires more than 21 measures is supported by our previous work. In [7] we use a subset of our EMISQ quality model; this subset is composed of 165 measures derived from the ISO 9126 standard (ISO 9126 is the predecessor of ISO/IEC 25010 standard). By providing much more maintainability-related measures, our model can give more hints for improving source code quality. Besides, our model proved its predictive power for maintainability (see [8]). Consequently, it reaches the coverage and depth needed to ensure thorough quality improvement activities that cannot be achieved by the IT-CISQ model.

Our validation experiments indicate that neither the assessment method nor the measures proposed by IT-CISQ can be reliably used to measure the maintainability of object-oriented software. For the assessment method we can either propose our benchmarking-based approach (see [7, 20]) or the approach developed in the Quamoco project [6]. Our benchmarking approach [23] served well to determine the maintainability of projects; nevertheless, with 165 measures used in a study [20] it is quite large and a little bit bulky to deal with.

Moreover, further experiments with industrial or commercial software would be necessary, too. Unfortunately, we were limited to open-source projects, but industrial projects would strengthen the external validity for a more robust generalization of the findings.

## References

1. Riaz, M., Mendes, E., Tempero, E.: A systematic review of software maintainability prediction and metrics. In: Proceedings of the 2009 3rd International Symposium on Empirical Software Engineering and Measurement, Washington, DC, USA, pp. 367–377 (2009)
2. Wagner, S., Lochmann, K., Winter, S., Goeb, A., Klaes, M.: Quality models in practice: a preliminary analysis. In: 3rd International Symposium on Empirical Software Engineering and Measurement, ESEM 2009, pp. 464–467 (2009)
3. Bakota, T., Hegedus, P., Ladanyi, G., Kortvelyesi, P., Ferenc, R., Gyimothy, T.: A cost model based on software maintainability. In: 2012 28th IEEE International Conference on Software Maintenance (ICSM), pp. 316–325 (2012)
4. CISQ Specifications for Automated Quality Characteristic Measures, OMG, CISQ-TR-2012-01 (2012)
5. ISO/IEC 25010 Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE), ISO/IEC, ISO/IEC 25010:2011 (2011)

6. Wagner, S., Goeb, A., Heinemann, L., Klas, M., Lochmann, K., Ploesch, R., Seidl, A., Streit, J., Trendowicz, A.: The Quamoco product quality modelling and assessment approach. In: Proceedings of the 34th International Conference on Software Engineering (ICSE 2012), Zurich, pp. 1133–1142 (2012)
7. Ploesch, R., Gruber, H., Hentschel, A., Koerner, C., Pomberger, G., Schiffer, S., Saft, M., Storck, S.: The EMISQ method and its tool support-expert-based evaluation of internal software quality. Innov. Syst. Softw. Eng. **4**(1), 3–15 (2008). Springer
8. Gruber, H., Ploesch, R., Saft, M.: On the validity of benchmarking for evaluating code quality. In: Proceedings of the Joined International Conferences on Software Measurement, IWSM/MetriKon/Mensura 2010, Aachen (2010)
9. Gruber, H., Ploesch, R., Schiffer, S., Hentschel, A.: Calculating software maintenance risks - a practical approach. In: Proccedings of the IASTED Software Engineering Conference (SE 2012), Maintenance Measures, Crete, Greece, pp. 452–455 (2012) and Proceedings of the 20th International Conference on Software Engineering, Washington, DC, USA (1998)
10. Ploesch, R., Schuerz, S., Koerner, C.: On the validity of the IT-CISQ quality model for automatic measurement of maintainability. In: Proceedings of the 39th International Computers, Software and Applications Conference (COMPSAC 2015), Taichung, Taiwan (2015)
11. Correia, J.P., Visser, J.: Certification of technical quality of software products. In: Proceedings of the Int'l Workshop on Foundations and Techniques for Open Source Software Certification, pp. 35–51 (2008)
12. Erlikh, L.: Leveraging legacy system dollars for e-business. IT Prof. **2**(3), 17–23 (2000)
13. Parnas, D.L.: Software aging. In: Proceedings of the 16th International Conference on Software Engineering, Los Alamitos, CA, USA, pp. 279–287 (1994)
14. Chidamber, S.R., Kemerer, C.F.: A metrics suite for object oriented design. IEEE Trans. Softw. Eng. **20**(6), 476–493 (1994)
15. McCabe, T.J.: A complexity measure. IEEE Trans. Softw. Eng. **SE-2**(4), 308–320 (1976)
16. Henderson-Sellers, B.: Object-Oriented Metrics: Measures of Complexity. Prentice Hall, Upper Saddle River (1996)
17. Binkley, A.B., Schach, S.R.: Validation of the coupling dependency metric as a predictor of run-time failures and maintenance measures. In: Proceedings of the 20th International Conference on Software Engineering, Washington, DC, USA, pp. 452–455 (1998)
18. Correia, J.P., Visser, J.: Benchmarking technical quality of software products. In: 15th Working Conference on Reverse Engineering, WCRE 2008, pp. 297–300 (2008)
19. Simon, F., Seng, O., Mohaupt, T.: Code-Quality-Management: technische Qualität industrieller Softwaresysteme transparent und vergleichbar gemacht. dpunkt-Verlag (2006)
20. Gruber, H., Koerner, C., Ploesch, R., Pomberger, G., Schiffer, S.: Benchmarking-oriented analysis of source code quality: experiences with the QBench approach. In: Proceedings of the IASTED International Conference on Software Engineering (SE 2008), Anaheim, CA, USA, pp. 7–13 (2008)
21. Heitlager, I., Kuipers, T., Visser, J.: A practical model for measuring maintainability. In: 6th International Conference on the Quality of Information and Communications Technology, QUATIC 2007, pp. 30–39 (2007)
22. Wagner, S., Goeb, A., Heinemann, L., Kläs, M., Lampasona, C., Lochmann, K., Mayr, A., Ploesch, R., Seidl, A., Streit, J., Trendowicz, A.: Operationalised product quality models and assessment: the Quamoco approach. Inf. Softw. Technol. **62**, 101–123 (2015)
23. Gruber, H.: Benchmarking-oriented Assessment of Source Code Quality - An Approach for Automatic Assessment using Static Code Analysis Tools, Dissertation (2010)

# Interface-Based Software Requirements Analysis

Aziz Ahmad Rais(✉) and Rudolf Pecinovský

Department of Information Technologies, University of Economics, Prague, Czech Republic
aziz.ahmad.rais@gmail.com, rudolf@pecinovsky.cz

**Abstract.** Software requirements analysis is a critical phase in the software development life cycle. It is usually carried out using one of the following: use cases, user stories or business scenarios. Use cases, user stories and business scenarios thus become inputs for the object-oriented analysis (OOA) of application software development. Most translating and mapping of software requirements from text to objects and classes creates problems with software acceptance. There is also a practical issue with object-oriented analysis, namely, that OOA and object-oriented design (OOD) both operate with the same objects and classes, and, in practice, it has not yet been possible to separate them from one another. The goal of this article is to provide a solution that will simplify software requirements analysis and separate such analysis from design, in order to give architects, developers and testers the ability to work independently through a contract (interface) that integrates their work.

## 1 Introduction

According to the ISO 12207 [1], there are two types of life cycle process: system-specific (system life-cycle) and software-specific (software life-cycle). The system-specific processes include activities that are not directly related to application software, such as, for example, project management related processes, and information system related processes, and so on. It is necessary to recognize that system life-cycle processes are important to software life-cycle processes. The most significant among these is the system requirements analysis process. The outcome of this last demonstrates how the whole organization is involved in the process that produces value and creates its core business.

Most organizations underestimate the value of enterprise-level processes [4], business requirements and business processes. Software developers must, therefore, clarify these business requirements before software requirements analysis, with the help of various business techniques, such as brainstorming, business rules, data dictionaries and glossaries, data diagrams, data modelling, decision analyses, functional decomposition, interviews, prototyping, risk analyses, scenarios and use cases, user stories, sequence diagrams, state diagrams, and so on [6]. The techniques listed by BABOK [6] can be used to elucidate business requirements at the enterprise level. This does not mean, however, that we have a well-defined idea of software requirements, which can equally be delineated using most of these techniques.


J. Janech et al. (eds.), *Proceedings of the 2015 Federated Conference on Software Development and Object Technologies*, Advances in Intelligent Systems and Computing 511,
DOI 10.1007/978-3-319-46535-7_23

Each business requirements [6] (system requirement – ISO 12207 [1]) analysis technique is used for specific purposes, and distinguishes individual business requirements from other perspectives. The techniques used for functional software requirements analysis are as follows: scenarios and use cases, user stories and sequence diagrams. They are most often used here because of a confusion of business requirements analysis and software requirements analysis.

The techniques used for functional software requirements are written in natural language, so translating them to technical language or software items [1] such as objects, classes, domain objects, components, interfaces, messages, and so on, causes a loss in consistency and a break in the relation between requirements and software items. This break is caused by the fact that the software also has to meet software quality requirements [7]. Thus, software application components have to be organized in a way other than by defined and grouped use cases, or user stories. The translation issue is a result of difficulties in identifying the domain objects, business processes or functionality, validation or limitation of inputs, formats of outputs, and so on.

ISO 12207 divides software development life cycle processes (SDLC) [1] into software implementation processes, software support processes and software reuse processes. Object-oriented analysis and object-oriented design are included in software implementation processes.

Software implementation processes are executed by software development methodology in different ways, such as incremental [8], agile [9], spiral, waterfall [10], and so on. Most such software development methodologies do not consider software support processes and software reuse processes as part of their software development life cycle. Though they model the software implementation processes in different ways, all software development methodologies basically utilize three practices: object-oriented analysis, object-oriented design, and use cases of processes such as the software requirements analysis process, the software architectural design process and the software detailed design process. Nevertheless, how these practices are used differs according to the software development methodology in question.

The diagram at Fig. 1 shows a common usage of OOA and OOD.

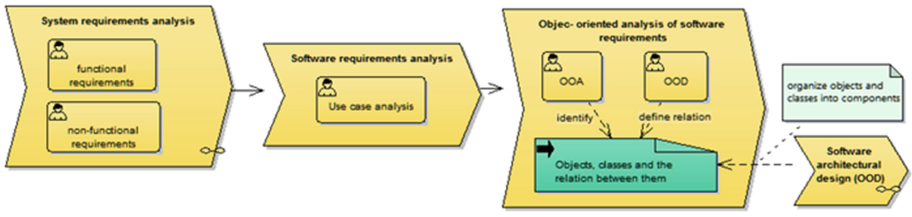



**Fig. 1.** A typical method of software requirements analysis (source: own)

Booch in [5] defines: *Object-oriented analysis is a method of analysis that examines requirements from the perspective of the classes and objects found in the vocabulary of the problem domain.*

Object-oriented analysis thus helps developers analyse software requirements written in text format (e.g. use cases) and natural language, and translate them into technical language: objects and classes. However, in order properly to understand these objects and classes, we need to place them in the context of business processes. In other words, we have to establish a relation between the objects, classes and model behaviour of relevant software.

Further Booch in [5] defines: *Object-oriented design is a method of design encompassing the process of objects oriented decomposition and a notation for depicting logical and physical as well as static and dynamic models of the system under design.*

ISO 12207 [1] defines software design as: internal and external interfaces of each software item (e.g. objects, classes) and software unit *(e.g. components); consistency and traceability are established between software requirements and software design.*

According to ISO 12207, the object-oriented design process has two aspects: the software architectural design process and the software detailed design process. The architectural design process may be considered conceptual design by identifying components and the relation between them. Software detailed design is about static and dynamic models composed of objects and classes. In other words, detailed design is very similar to object-oriented analysis. The only difference between them is that component and conceptual design ensures that detailed design is viewed simply and abstractly in order to organize the objects and classes in such a way as to meet non-functional application software requirements. As a result, many software development methodologies cannot adequately separate OOA from OOD process and so many organizations skip the design phase and move directly onto developing the software.

In order to analyse and design software, different modelling languages and tools may be used, the most common of which is UML. A modelling language is a model-driven architecture (MDA) concept [11]. Sometimes modelling is used to visualize software components, and so tools such as, for example, Microsoft Visio are put into practice to provide objects with images to show these components as natural words. The goal of such modelling reaches further than conceptualizing the software component as words, however, and also covers visualizing it within the software itself, making the model physically available as an item or unit. This means that the model can be transformed into a software item and vice versa.

Modelling is one of MDA concepts, and such transformation between the model and a software item is another one [11].

To use and interpret modelling only as the visualization of a concept makes it difficult to maintain and update such a model when application software changes during software support processes. For that reason, OOD process is skipped during software implementation processes and architecture design is done very abstractly.

There are tools available, like, for example, Spring Roo, that generate fully functional software based on domain objects [12]. Most of these tools work with domain object models, and generate Create, Read, Update and Delete operations for the domain objects. Such software is, however, usually limited in its ability to provide or cover all the functionalities required in business. Furthermore, other issues may arise with using this type of MDA to produce software, for example, the performance, maintainability or extensibility [7] of software items and units.

## 2 Interface-Based Software Requirements Analyses

Analyzing software requirements requires an understanding of how software is composed of different software parts (e.g. layers), software units (e.g. components) and software items objects or classes). Architecture and architecture layers are further MDA concepts. By applying these two concepts, software parts can be identified and divided into three layers independently of software requirements.

- Consumer layer: This layer describes how the client will interact with the software and use the service. If the client is human, then it will be either a graphical user interface or a command line interface. If the client is other software, it will be a service layer wrapper accessible remotely by consumer software.
- In use cases, the boundary of the software is not always clear. Sometimes use cases describe the end user interaction with the software, and at other times describe the system process; sometimes they describe the design. It can also be difficult to gain perspective on use case granularity.
- Service layer: This layer describes the core functionality implemented by software and provided as a service. Looking at software from the client perspective facilitates discussion regarding functional software requirements between developers and stakeholders, as the latter are not interested in the composition of the software, but rather in how to use it.
  In use cases, identifying the core functionality and service that the software in question has to provide is combined and usually described in the form of a process. It is clear that such a process is composed of related sets of activities. Thus, each activity can be implemented in one or more services. As a result, use cases become difficult to map or translate into objects, classes and components.
- Domain layer: This layer describes the type of data or information required and/or provided by the service layer.

In use cases, identifying domain objects is a difficult task for most experienced software developers. There is simply no easy trick, tip or technique to teach developers how to identify domain objects, perform domain analysis and design the domain object model. Software requirements can be divided into functional software requirements and non-functional software requirements, as is the case with business requirements [6]. This means that before analysing software requirements, business requirements must be available in the form of functional business requirements and non-functional business requirements. In order better to analyse functional software requirements with regards to interface, let us first clarify the concept of interface.

The [13] defines: *The interface of a given entity (= of a programs part – a module, a class, a method) specifies what the given entity knows and how to communicate with it. In case of an object it says which messages can be sent to it (to which messages the object understands) and how the object reacts to them. It is important to remember that an interface doesn't solve anything, it only promises, what the given entity can provide. We could say that the interface summarizes what the surrounding program should know about the given entity.* This type of interface will be called in this article conceptual interface.

This definition describes interface as physical characteristics of software items and units. However, 'interface' also has a second meaning:

*Interface is used in term of the program's construction, which we might consider as syntax representation of interface and which behaves as a class without any implementation* [13]. This type of interface will be called in this article *physical interface*; it means that programming languages like java, C#, and so on have a representation of it.

Abstraction is another foundational MDA concept, and "*deals with the concepts of understanding a system in a more general way*" [11]. Combining this MDA concept with the first definition of interface (conceptual interfaces) we can use to describe functional software requirements.

The notion of separation of concerns is a key MDA concept. By applying it to the software requirements analysis, we would be better able to identify and to manage dependencies between interfaces. Separation of concerns would make it possible to provide access to the stakeholders' perspective.

The easy way to apply separation of concerns to software requirements analysis would be to think about dividing the software architecture vertically. The result of a horizontal division of architecture is layers; with vertical division, on the other hand, we would be able to create modules of software requirements according to business concern. Please note that the vertical and horizontal division of software architecture performed during the software requirements analysis process is meant only conceptually and not physically. Physical separation is usually be done by software architecture design process (Fig. 2).

The first goal of interface-based analysis is to reduce the complexity and difficulty of translating text into objects and classes by performing such analysis only once and remaining object-oriented. The second goal is to reduce the number of analysis phases and focus on the architecture. This means carrying out the analysis from an architectural perspective and not being limited by objects and classes that will change during implementation and synchronization between code, architecture and requirements. It remains to compare user stories with interface-based analysis. Because user stories are a decomposition of use cases, they can, in fact, be regarded as mini use cases [3].

The Fig. 3 depicts how interface-based analysis different views from software architecture, software developer, software tester and functional business requirements perspective.

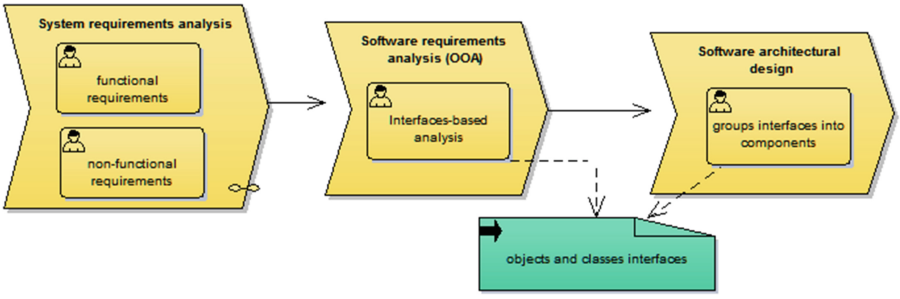



**Fig. 2.** Interface-based analysis (source: own)

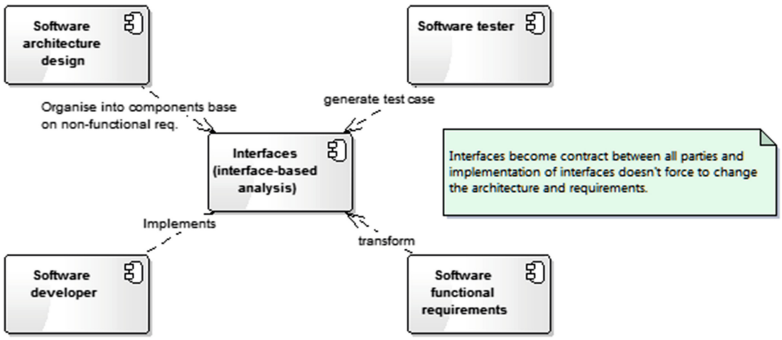



**Fig. 3.** Different views of interface-based analysis (source: own)

The UML model of interface-based analysis could be transformed to, for example, Java source code. After the transformation of UML model we can get service layer interfaces (the second definition) that can be implemented. All changes in requirements thus become easily traceable and the software developer has the freedom to implement them without limitation. The domain model can be converted into a class model, and the ATM user layer (consumer layer) can be converted into the model components of the MVC design pattern. The actors in interface-based analysis will be left out of the diagrams and instead there will be provided a matrix of interfaces and actors list and rights.

## 3 Example

In order to illustrate how interface-based analysis can work, it is useful to compare a use-case analysis model with an interface-based one. The goal is to see that what can be done with use cases can also be done with interfaces in an object-oriented way.

The example will analyse the following functionalities automated teller machine (ATM) bank:

- Customer is able to check the bank balance.
- Customer can withdraw money from ATM.
- Customer can give an order for transferring money.
- Customer can deposit funds into his bank account.

ATM machine need to be repaired and maintained thus customer should be able to use ATM machine.

### 3.1 Use Case Way

Use case at Fig. 4 diagram describes the functionalities required from ATM banking without scenarios.

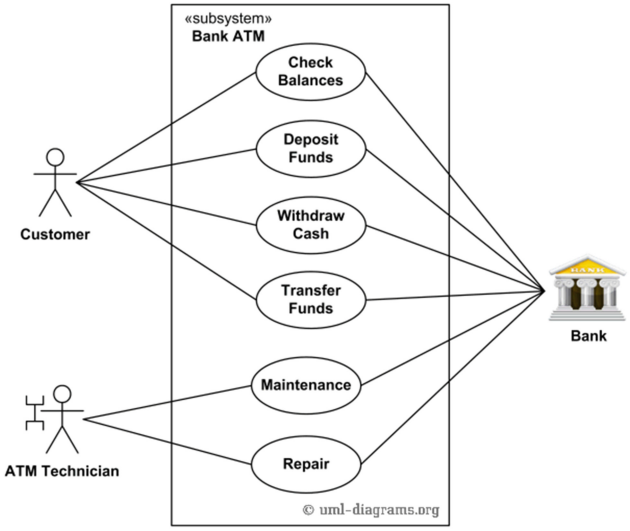



**Fig. 4.** A use-case analysis model (source: http://www.uml-diagrams.org)

### 3.2 Interface-Based Way

The ATM banking functionalities analysis with interface-based method will be divided into two business modules according to the separation of concern concept of MDA. The modules are:

- The operational module at Fig. 5 describes how customer performs cash operations with ATM interface.
- The second module depicted at Fig. 6 is maintenance module that will show change the ATM banking state into maintenance, so that the machine can be repaired (Table 1).

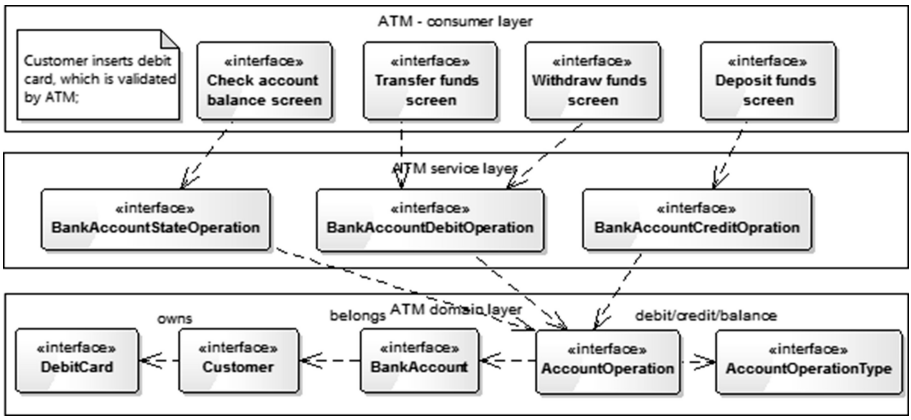



**Fig. 5.** An interface-based analysis of the operational module of an ATM (source: own)

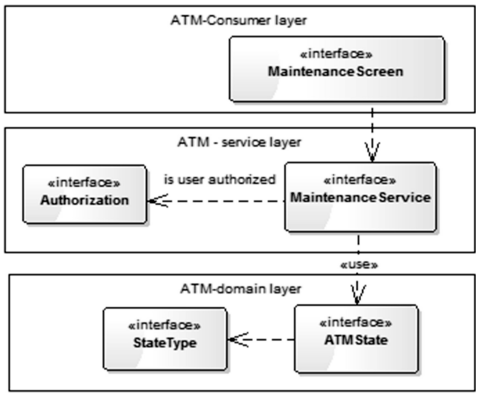



**Fig. 6.** An interface-based analysis of the maintenance module of an ATM (source: own)

**Table 1.** Interface-based actors list and rights (source: own)

| Access rights | Customer | ATM-technician |
|---|---|---|
| BankAccountDebitOperation | Read–Write–Execute | 0–0–0 |
| BankAccountCreditOpration | Read–Write–Execute | 0–0–0 |
| BankAccountStateOperation | Read–Write–Execute | 0–0–0 |
| MaintenanceService | 0–0–0 | Read–Write–Execute |

## 4 Conclusion

Use cases are not object-oriented, and analyses are performed twice in most software development methodologies (see Fig. 1). With interface-based analysis, we can perform object-oriented analyses of software requirements, and such analyses need only be performed once. Interface physically available in many programming languages, it means that there are available a representation of interface, such as, for example, Java or C#. Therefore, we can transform the conceptual interfaces from the analysis phase into a physical interface, and can force development to become agile and modular. With the use of conceptual interfaces we can clearly define analysis and design. Design will be easier with conceptual interface because architecture design works with components, and every component can have both provider and consumer conceptual interfaces. As a result, the mapping between architecture and requirements analysis will become clearer and easier.

**Acknowledgment.** The paper was processed with contribution of long term institutional support of research activities by the Faculty of Informatics and Statistics, the University of Economics, Prague.

## References

1. ISO/IEC 12207: System and software engineering – Software life cycle processes (2008)
2. Philippe, K.: Rational Unified Process, An Introduction, 3rd edn. Addison Wesley, New York (2003). ISBN 0-321-19770-4
3. Mike, C.: User Stories Applied: For Agile Software Development. Addison Wesley, New York (2004). ISBN-10: 0321205685
4. The Open Group: TOGAF, version 9.1 (2011). ISBN 978-90-8753-679-4
5. Grady, B.: Object-Oriented Analysis and Design with Applications, 3rd edn. Addison-Wesley, New York (2007). ISBN 0-201-89551-X
6. IIBA: A Guide to the Business Analysis Body of Knowledge (BABOK Guide), version 2.0. ISBN-13: 978-0-9811292-2-8
7. ISO/IEC 25030: Software engineering — Software product Quality Requirements and Evaluation (SQuaRE) — Quality Requirements (2007)
8. John, H.: Guide to the Unified Process Featuring UML, Java and Design Patterns. Springer, London (2003). ISBN-10: 1852337214
9. Orit, H., Yael, D.: Agile Software Engineering. Springer, London (2008). ISBN-10: 1848001983
10. Gerard, O.: Introduction to Software Quality. Springer, London (2014). ISBN 978-3-319-06106-1
11. OMG: Model Driven Architecture (MDA) MDA Guide rev. 2.0. OMG Document ormsc/2014-06-01
12. Craig, L.: Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and the Unified Process. Prentice Hall, Upper Saddle River (2004). ISBN-13: 978-0131489066
13. Pecinovský, R.: OOP – Learn Object Oriented Thinking and Programming. Eva & Tomas Bruckner Publishing (2013). ISBN 80-904661-8-4

# Object Metamorphism

## Type-Safe Modeling of Protean Objects in Scala

Zbyněk Šlajchrt(✉)

The Department of Information Technologies, University of Economics, Prague, Czech Republic
zslajchrt@gmail.com

**Abstract.** Modeling protean objects, i.e. objects adapting their structure and behavior dynamically with respect to changeable environment, may be challenging in traditional object oriented languages. While some dynamic languages make the implementation of changeable behavior of objects possible by injecting code into the objects at run-time, their lack of an in-depth compile-time code analysis makes the resulting code fragile. The concept of object metamorphism (OM) targets the gap between the two language types by introducing a declarative modeling of protean objects. Such a model, which is validated at compile-time, defines all possible compositions of a given object from smaller parts represented by traits. The actual composition used to instantiate the object is chosen dynamically with respect to the current environment. The result of this research will provide the theoretical concept of OM along with a proof-of-concept adoption of OM to Scala.

## 1 Introduction

Object-oriented programming (OOP) has proven successful in a wide area of applications mainly because it makes easier to model entities from real domains and the relationships among them. Although it has become a standard in today's application development, there are still some programming problems, which are difficult to solve by means of the current object-oriented languages. In this paper I am focusing on a subset of such problems emerging from the gray zone where neither statically nor dynamically typed languages fit fully the problems. In particular, this paper explores two scenarios, which the traditional OOP languages fail to fully support.

The first problem occurs in applications where the exact type and composition of an object is not known until the moment of its instantiation. The second problem occurs when the type and composition of the object may vary during its lifetime.

Although there is always a way to solve the above-mentioned problems in both kinds of languages, a developer must usually make some design concessions, which lead at least to degrading some non-functional properties of the program, such as extensibility, coupling and reusability.

The concept of object metamorphism presented in this paper attempts to overarch the gap between the static and dynamic languages by a sort of "controlled" or "statically checked" dynamism applicable in the domain of statically typed languages.


J. Janech et al. (eds.), *Proceedings of the 2015 Federated Conference on Software Development and Object Technologies*, Advances in Intelligent Systems and Computing 511,
DOI 10.1007/978-3-319-46535-7_24

OM targets the systems consisting of objects whose structure and behavior may be composed of fragmentary components, whereas the number of possible compositions may be large. An example may be a system modeling the manifestation of emotions in the human face. A relatively small number of face muscles are able to produce thousands of emotional expressions, however not every combination of the muscles represents an emotional expression. The task of such an application might be to guess the emotion according to the activity of individual muscles or, contrarily, to control virtual muscles in a dummy according to a given emotion.

Another example might be a complex service, which must apply a number of optional security constraints, whereas some may depend on others. The selection of the proper constraints is done according to the properties of the user. Such a system of constraints may also generate a large number of combinations.

In both examples OM builds a morph model producing all possible combinations, which are verified by checking their dependencies at compile-time.

The research presented in this paper is built on the ideas from Subject-oriented programming (SOP) [1, 2] and the Data-Context-Interactions paradigm (DCI) [3]. In these paradigms the objects are seen as capable of assuming different forms with respect to the context or use-case. Such forms are called subjects in SOP and roles in DCI.

## 2 Example

Let us imagine an airport luggage scanner capable of recognizing three and two-dimensional objects in baggage by their geometrical properties and material. The scanner outputs each scanned item in a form sketched in Fig. 1.

For the sake of simplicity let us assume that besides the common attributes, such as the item's position, the scanner recognizes two shapes – cuboid and cylinder – and two materials – paper and metal. It follows that the record emitted by the scanner may take on four forms corresponding to the Cartesian product of the shapes and materials sets. It also follows that with the increasing number of independent dimensions, added as the scanner is improved, the number of record forms exponentially grows.

```
{
"id": 0,
"shape": "cylinder",
"material": "metal",
"x": 234.87,
"y": 133.4,
"z": 12.94,
"radius": 13.45,
"height": 0.45,
"density": 3.8
}
```

**Fig. 1.** Scanner data sample

Now let us suppose that we have to write a program reading the item records and represent them as objects. We will examine how the three established OO languages Java, Scala and Groovy perform in modeling the scanner data.

## 2.1 Composition in Java

The most natural approach to model the items is to use composition to model the item as a two-part composite and delegation to expose the common methods from both dimensions (i.e. `volume()` and `density()`). The Fig. 2 depicts the UML diagram of such a design.

Then instantiation of an `Item` instance initialized with the data from the Listing 1 would look like this (Fig. 3):

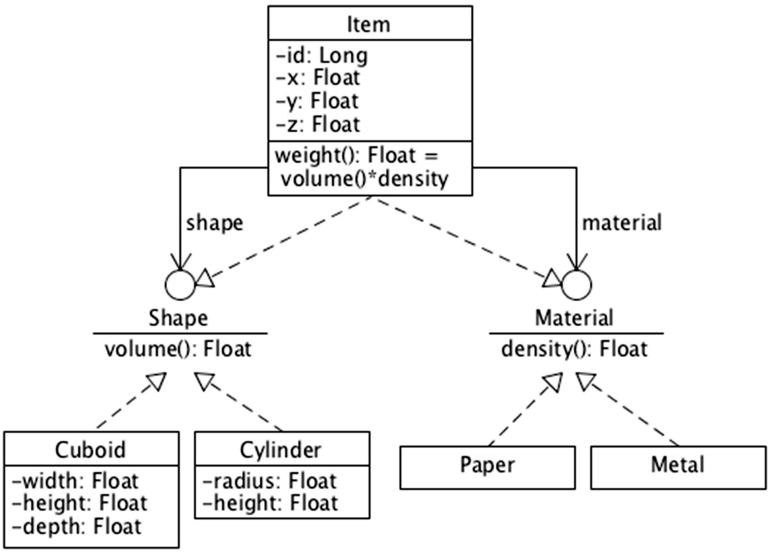



**Fig. 2.** The item model using composition

At first sight everything seems to be all right; the item object encapsulates all data from the record and we can invoke the common methods, such as `volume()`, on the item. But what if we wanted to determine whether the item is a metal cylinder, for instance? One could naturally tend to use the `instanceof` operator, since it is used to detect what an object **is** (Fig. 4).

```
Item item = new Item(0, 234.87, 133.4, 12.94, new Met-
al(3.8),new Cylinder(13.45, 0.45));
```

**Fig. 3.** Composing the item instance in Java

```
boolean isMetal = item instanceof Metal;
boolean isCyl = item instanceof Cylinder;
```

**Fig. 4.** Attempting to determine the type of the item

However, both previous statements are always false, because the objects, which may be instances of the two particular types, are wrapped as components of the `Item` class and hence they are not part of the item's type. In order to make possible to determine the item's real type from the item itself we would have to incorporate a custom management of the item types into the Item class. Such a solution would be a mere workaround fixing the deficiency of the Java type-system.

In other words, Java forces a developer to model a trait of an object (i.e. "is-a" relationship) as if the trait were a component of the object (i.e. "has-a" relationship).

Sometimes the choice of the relationship depends on the subject's perception. For example the object in Fig. 10 may be modeled as a wine *bottle box* or a *wine bottle in a box* (Fig. 5).

**Fig. 5.** Do we see a wine bottle box or a wine bottle in a box?

In the former case we see primarily a box containing a wine bottle ("has-a"), while in the latter case we perceive the box as a decoration or a "trait" of the wine bottle ("is-a"). The problem we face in Java is that we must always model this situation as the wine bottle box [4].

Such a design inherently suffers from the complication called *object schizophrenia*. The name suggests that the object's identity is scattered across the wrapper and the components (there are more "selves") [5]. This complication may be illustrated on the following example. Let us assume that some items need to be additionally corrected because of a temporary malfunction of the scanner resulting in wrong measurement of material density. This defect may be corrected in the application by decorating the affected items by the `DensityCorrection` implementation of Item.

Figure 6 shows two possible implementations of such a correction class. The left uses inheritance while the right uses composition. Let us compare the two designs with respect to the behavior of the weight method defined in Item.

Since the correction and the item share the same identity (self), the weight method in the left design reflects the overridden density by `DensityCorrection` and returns the corrected weight. On the contrary, the `weight` method invoked on the correction in the right case will return the original value instead of the corrected one, since the method invocation is delegated to the wrapped item where the method

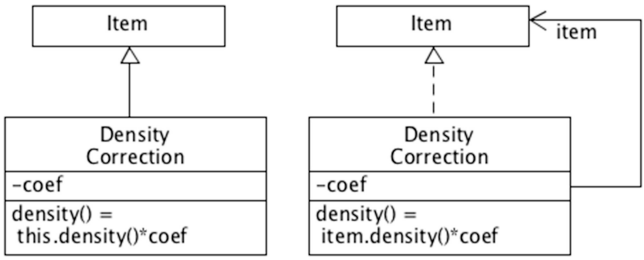



**Fig. 6.** `DensityCorrection` extension designed by inheritance and composition

calculates the weight according to the original density, which is presumably incorrect behavior.

Thus the solution based on inheritance fits better to the problem. Unfortunately, it is not possible to arbitrarily stack more such dedicated extensions by means of inheritance due to the single inheritance in Java.

The concept of traits overcomes this limitation. It makes possible to stratify an arbitrary number of extensions of a class while avoiding the multiple inheritance issues, such as the diamond problem [6], by the so-called *trait linearization* [7]. Let us examine how the problem of modeling multidimensional data may be solved in Scala and Groovy, i.e. the languages, which adopted the concept of traits.

## 2.2 Static Traits in Scala

In Scala, we can model the item quite straightforwardly; every piece of it is expressed as a trait (Fig. 7).

```
trait Shape {
def volume: Float
}
trait Cylinder extends Shape {
def height: Float
def radius: Float
def volume = PI * radius * radius * height
}
trait Material {
def density: Float
}
trait Metal extends Material {}
trait Item {
this: Shape with Material =>
val id: Int
def pos: (Float, Float, Float)
def weight: Float = density * volume
}
```

**Fig. 7.** Item elements modeled by traits in Scala

```
val item = new Item with Metal with
Cylinder {
  val pos = (234.87, 133.4, 12.94)
  def density = 3.8
  def radius = 13.45
  def height = 0.45
}
```

**Fig. 8.** Creating and initializing a metal cylinder in Scala

```
val isMetalCylinder =
  item.isInstanceOf[Metal with Cylinder]
```

**Fig. 9.** Testing whether the item is a metal cylinder in Scala

The item instance is then composed from the right traits at the moment of its creation. The initialization of the abstract members is done at the same time (Fig. 8).

The item constructed this way may be tested by means of `isInstance` to determine if it is a metal cylinder, for example (Fig. 9).

This approach also eliminates the problem with object schizophrenia, since we can design the `DensityCorrection` extension as a trait, which will share its identity with the item. This trait will be combined with the other traits only if the input data should be corrected.

Although it seems that the approach using Scala traits has coped with all problems identified above, there is in fact looming another serious problem relating to the number of all possible combinations of the item's forms. Each combination would have to be tested individually by a separate condition. Furthermore, there would have to be one class for each combination of traits. Thus the code and the classes would quickly become unmaintainable since the number of combinations grows exponentially with every additional dimension.

Since this problem is rooted in the static nature of Scala, let us examine, if the problem persists in Groovy, which provides a dynamic version of traits.

### 2.3 Dynamic Traits in Groovy

In Groovy, it is possible to attach traits to an object at run-time. Every object provides the **withTraits** method for adding one or more traits to the object. This feature allows us to initialize the item in a step-by-step way, i.e. per feature and not per combination. (Since the traits themselves are defined in a similar way as in Scala they are not showed because of the limited space in this paper.) (Fig. 10).

As in the Scala case, also here we can use the `instanceof` operator to determine what the item really is (Fig. 11).

Also the `DensityCorrection` trait could be implemented similarly. Nevertheless, the most important difference is that we have gotten rid of the combinatorial explosion of classes and code, which is the consequence of the dynamic addition of traits allowing constructing the item per feature and not per combination. Unfortunately, also

```
def item = new Object() as Item

item.x = itemData.get("x")
item.y = itemData.get("y")
item.z = itemData.get("z")

// Shape dimension
if (itemData.containsKey("cylinder")) {
  item = item.withTraits(Cylinder)
  item.radius = itemData.get("radius")
  item.height = itemData.get("height")
} else if (event.containsKey("cuboid")) {

  item = item.withTraits(Paper)
  item.width = itemData.get("width")
  item.height = itemData.get("height")
  item.depth = itemData.get("depth")
}
// Material dimension
if (itemData.containsKey("metal")) {
  item = item.withTraits(Metal)
…
```

**Fig. 10.** A step-by-step assembling in Groovy

```
def isMetal = item instanceof Metal
def isCylinder = item instanceof Cylinder
```

**Fig. 11.** Testing whether the item is a metal cylinder in Groovy

this approach suffers from some serious issues resulting from the step-by-step approach as well as from the dynamic nature of the language and its weak type system:

- **Incompleteness:** The configuration procedure may forget to choose a trait for some dimension
- **Redundancy:** The `withTraits` method can possibly add two mutually exclusive traits from the same dimension to the builder
- **Missing Dependencies:** A trait from one dimension may depend on another trait from another dimension. The configuration procedure must take these inter-trait dependencies into account, because it is beyond the capabilities of dynamic languages. This additional requirement makes the code fragile and open to inconsistencies.
- **Ambiguous Dependencies:** The configuration procedure must also guarantee that there is no ambiguity in the dependencies.

## 2.4 Summary

The traditional approach to model multidimensional data objects uses composition. The composition produces an object (item) by wrapping other objects (material, shape), which can assume various forms. The number of wrapped objects corresponds to the

number of dimensions. For each dimension the top object exposes the corresponding interface by means of delegation.

Composition hides the real shape (i.e. type) of the object. We cannot determine from the top object's type that it is a metal cylinder, for example. Instead, we just find out that the item is something of some shape and material. To determine the real type, one cannot use the `instanceof` operator only. Instead, s/he must resort to examining the object's attributes, i.e. the state, holding references to the wrapped objects (`getMaterial()`, `getShape()`) and additionally apply `instanceof` to each wrapped instance. Alternatively, the classes may be equipped with a custom type-management complementing the platform's type system.

Next, the identity of the object is scattered across the object and its components, which results in object schizophrenia. In Java there is virtually no way to cure this problem, however the concept of traits adopted by languages such as Scala or Groovy naturally solve the problem.

However, modeling multidimensional objects by means of static traits in Scala leads to the exponential explosion of code making the static traits practically unusable.

Groovy's capability to apply traits to an object at run-time efficiently solves the problem of the exponential explosion. This approach however also suffers from a couple of serious issues outlined above.

We may, therefore, conclude that the outlined problems cannot be solved ideally if they are to be implemented either in a static or dynamic OO language only. It seems that the gap between the two types of language should be filled by a static/dynamic approach combining the benefits of both language types. The following section presents one such a hybrid concept called *object metamorphism*, which has been developed by the author.

## 3 Object Metamorphism

### 3.1 Introduction

Object metamorphism may be defined as a capability of an object to assume one or more forms defined by the object's *morph model*, which defines all possible form alternatives of the object. The compiler analyses the morph model and performs various checks to guarantee that all *alternatives* are consistent. Every alternative consists of the so-called *fragments*, which are the building blocks in multidimensional compositions of objects and correspond semantically to the concept of trait as defined in Scala or Groovy. A morph fragment represents a typological, behavioral and structural element of multifaceted objects.

The run-time component of OM instantiates one of the alternatives selected with the assistance of the so-called *morphing strategy*, which usually selects the alternative according to the context or input data. The instance of the alternative, which is called a *morph*, may be subject to further re-morphing, in case the context changes and the morph should reflect the change (the morph keeps a link to its morph model).

The theoretical concept of OM is accompanied by a reference implementation (RI), whose main goal is to serve as a proof-of-concept tool. RI is designed as an extension

of the Scala compiler and may be freely downloaded from here [8]. It should be remarked that the reason why Scala was selected as the platform for RI and not Groovy, for example, is based on the experience that it should be easier to add some dynamicity to a static language than to introduce some static type checking into a dynamic language.

In order to elucidate the idea of OM we will try to apply it to the example, which we have gone through in the previous section. We will use RI to implement the example.

### 3.2 Using OM

Let us begin with designing the morph model of the scanner items. In RI, morph models are defined by means of a special type expression based on the syntax for trait composition in Scala. RI extends the Scala type system by introducing the *disjunctor* type or, which may be seen as a union operator. The disjunctor type is in fact a generic type with two parameters `or[X, Y]`, which may be written as `X or Y`. This "syntax sugar" makes the type expressions more comprehensible and allows us to express the morph models as logical formulas.

Using the disjunctor we can define the morph model type for the item as follows (Fig. 12):

```
Item with (Metal or Paper) with (Cuboid or Cylinder)
```

**Fig. 12.** The morph model type of the item

This is an almost human-like expression of all possible compositions of the item. In order to translate it to a more machine-like form, we can treat it as a logical formula and transform it to the equivalent *disjunctive normal form* (DNF) (Fig. 13):

```
(Item with Metal with Cuboid) or
(Item with Metal with Cylinder) or
(Item with Paper with Cuboid) or
(Item with Paper with Cylinder)
```

**Fig. 13.** The DNF of the item morph model

The DNF clearly reveals all four alternatives produced by the morph model.

Another useful term is the *lowest upper bound* (LUB) of the morph model. It is the most specific type yet compatible with the types of all alternatives from the morph model. It follows that the LUB of the item morph model is (Fig. 14).

```
Item with Material with Shape
```

**Fig. 14.** The LUB of the item morph model

```
val itemAsm = compose[Item with (Metal or Paper) with (Cuboid or
Cylinder)]
```

**Fig. 15.** Creating the item assembler

This type actually corresponds to the type that we obtained when modeling the item by composition.

The morph model type expression is used in the application code as the type argument to various macros such as compose, which creates the so-called *morph assembler* (Fig. 15).

The morph assembler is responsible for generating classes for all alternatives in the morph model and for creating morphs from these classes. The `morph` method creates a new morph according to the suggestion made by the morphing strategy passed as the argument. The return type of that method is the LUB of the morph model (Fig. 16).

```
val item = itemAsm.morph(strategy)

assert(item.isInstanceOf[Item with Material with Shape])
```

**Fig. 16.** Creating the item morph

In case the morph should be updated according to the current context, one may invoke the remorph method on it. The item will get reassembled to the alternative selected by the strategy specified upon the morph's creation (Fig. 17).

```
item.remorph()
```

**Fig. 17.** Remorphing the item morph

Now let us turn our attention to the validations carried out by the compiler extension. One of such validations is checking dependencies. The dependencies are specified in RI by means of the self-type of the fragment trait as illustrated on the Item fragment, which depends on the Material and Shape dimensions (Fig. 18):

The compiler extension must verify that all alternatives containing fragment `Item` must also contain fragments implementing `Shape` and `Material`. The alternatives from the model mentioned above clearly satisfy this condition. However, the following model is not valid, since its alternatives have no shape fragment as required by Item (Fig. 19).

```
trait Item {
  this: Shape with Material =>
```

**Fig. 18.** The item's dependencies expressed as the self-type

```
Item with (Metal or Paper)
```

**Fig. 19.** A model with missing dependencies

On the other hand, the compiler must also check that there is no ambiguous dependency. The next model contains two ambiguous dependencies `Metal` and `Paper` that both appear in same alternatives (Fig. 20).

```
Item with Metal with Paper with (Cuboid or Cylinder)
```

**Fig. 20.** A model with redundant fragments Metal and Paper

### 3.3 Morph Model Reification

As long as the compiler extension detects no error in the model, it performs the so-called reification transforming the model into a special syntax tree, which is injected as a synthetic variable into the syntax tree of the assembler being produced by the macro (Fig. 21).

Thanks to the reification, the morph model may be used at run-time by the assembler. The model is stored in the `model` attribute of the assembler and is publicly accessible as shown in the following statement printing the model tree (Fig. 22).

The output would look like this (Fig. 23):

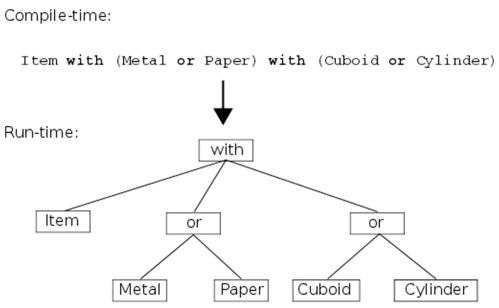



**Fig. 21.** Morph model reification

```
println(itemAsm.model.rootNode)
```

**Fig. 22.** The reified morph model is accessible through the model attribute of the assembler

```
ConjNode(List(FragmentNode(0),DisjNode(List
FragmentNode(1),FragmentNode(2))),Disj Node(List
(FragmentNode(3),FragmentNode(4)))))
```

**Fig. 23.** Textual form of the reified morph model

```
val ai = itemAsm.model.altIterator
while (ai.hasNext) {
  println(ai.next)
}
```

**Fig. 24.** Using the alternatives iterator to print the alternatives

RI implements an algorithm for transforming a morph model to the corresponding disjunctive normal form, which is used to extract the model alternatives. This functionality is available through the alternative iterator, which unfolds the model into a series of alternatives. The listing Fig. 24 shows how to print all alternatives in the model.

The first line of the output should be (Fig. 25):

```
List(FragmentNode(0,false),FragmentNode(1,false),
FragmentNode(3,false))
...
```

**Fig. 25.** Textual form of the alternatives

The ordering of alternatives is determined by the structure of the morph model tree. We can attach indices to the fragment nodes increasing from left to right (Fig. 26):

```
Item(0) with (Metal(1) or Paper(2)) with (Cuboid(3) or Cylin-
der(4))
```

**Fig. 26.** Indexing the fragments in the morph model

Then the set of alternatives can be written as this: `{(0, 1, 3), (0, 2, 3), (0, 1, 4), (0, 2, 4)}`. It allows us to define the ordering of the alternatives in the set quite easily just by comparing indices from right to left between two alternatives. The comparison of two alternatives goes from the common right-most index to the left.

In other words, given two alternatives `A1 = {a1, b1, c1}` and `A2 = {a2, b2, c2, d2}`, then `A1` precedes `A2` if and only if relation `(c1 < c2) || ((c1 == c2) && (b1 < b2)) || ((c1 == c2) && (b1 == b2) && (a1 < a2))` is true.

Applying this rule to our morph model, we obtain the following ordering of the alternatives (Fig. 27):

```
{0, 1, 3} < {0, 2, 3} < {0, 1, 4} < {0, 2, 4}
```

**Fig. 27.** Ordering the alternatives

### 3.4 Morhing Strategies

A morphing strategy is a key concept in OM determining the composition of morphs. Whenever a new morph is created or re-morphed, a morphing strategy must be supplied. Although it is possible to implement custom strategies, in practice only three are of a particular importance: promoting, masking and rating strategies. To select the winner alternative the three strategies use distinctive methods, which may be freely combined with one another. The following paragraphs will shortly introduce these strategies.

Promoting strategy swaps model tree branches in such a way that the promoted (i.e. preferred) alternatives get as high in the generated list of alternatives as possible. To illustrate this method let us use a simplified morph model consisting of four fragments only (Fig. 28):

```
(Metal or Paper) with (Cuboid or Cylinder)
```

**Fig. 28.** A simplified version of the item morph model

This model can be depicted as a tree shown on the following figure (Fig. 29).

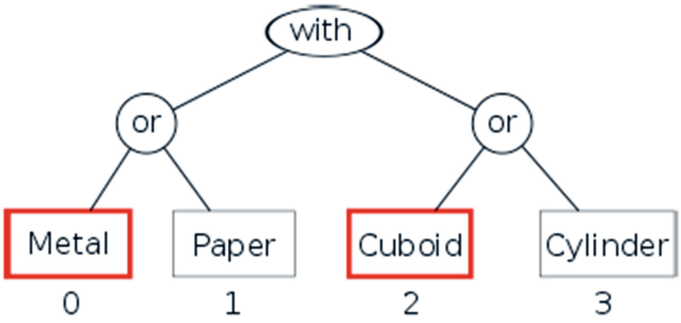



**Fig. 29.** The default morph model tree

The model has two dimensions that correspond to the two disjunctors (or) in the model. Each disjunctor contains two alternative fragments, thus the number of all alternatives defined by this model is four. A very important aspect is the order of the alternatives in the list since the **first alternative in the list is by default taken as the winner.** According to the rules described above the list of the alternatives generated by the model will be sorted in this way (Fig. 30):

The goal of the promoting strategy is to "promote" one or more independent fragments (i.e. fragments under two different disjunctors), which means to transform the morph model tree in such a way that the promoted fragments are present in the first alternative in the list generated from the tree, i.e. the default-winning alternative.

```
1. {0, 2}
2. {1, 2}
3. {0, 3}
4. {1, 3}
```

**Fig. 30.** The default list of alternatives

We will use notation (`materialCoord, shapeCoord`) to denote the fragments to be promoted. Then `(0,0)` is the default promotion leading to choosing the `{Metal, Cuboid}` alternative.

In order to promote fragments `(1, 0)`, the promoting strategy must transform the tree as follows (Fig. 31).

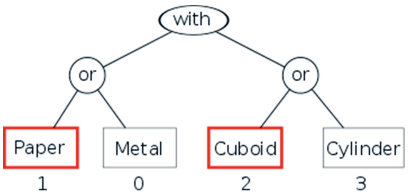



**Fig. 31.** The Paper fragment promoted

The new tree yields this list of alternatives (Fig. 32).

We can continue by promoting (1, 1). The corresponding tree will then be this (Fig. 33).

```
1. {1, 2}
2. {0, 2}
3. {1, 3}
4. {0, 3}
```

**Fig. 32.** The list of alternatives with the Paper fragment promoted

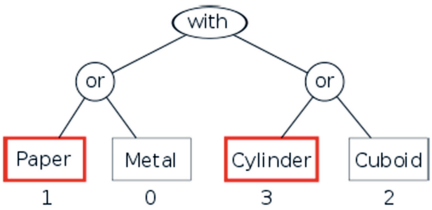



**Fig. 33.** Additional promotion of the Cylinder fragment

And the corresponding list of alternative is (Fig. 34):

```
1. {1, 3}
2. {0, 3}
3. {1, 2}
4. {0, 2}
```

**Fig. 34.** The list of alternatives after promoting Paper and Cylinder

And finally the remaining combination is (0, 1), which gives this tree (Fig. 35): with these alternatives (Fig. 36):

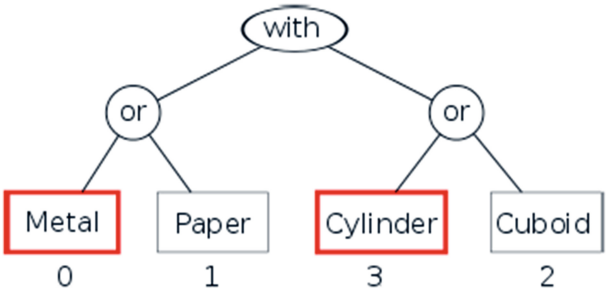



**Fig. 35.** The model tree after promoting Metal

In contrast to the promoting strategy, which only resorts the generated list of alternatives, the masking strategy is destructive as it effectively eliminates alternatives containing unwanted fragments. Masking allows suppressing alternatives, which do not match the so-called *fragment* mask. The fragment mask specifies which fragments may be present in the winning alternative. By turning some fragments off we can effectively reduce the set of the alternatives. By default, all fragments are turned on.

In order to make the explanation easier, let us introduce the following notation, which represents an alternative along with a bit vector indicating fragments contained in the alternative.

$$\{\texttt{fragments}\}[\texttt{MetalBit}, \texttt{PaperBit}, \texttt{CuboidBit}, \texttt{CylinderBit}]$$

The four alternatives from our model then look like this (Fig. 37):

```
1. {0, 3}
2. {1, 3}
3. {0, 2}
4. {1, 2}
```

**Fig. 36.** The list of alternatives after promoting Metal

```
1. {0, 2}[1, 0, 1, 0]
2. {1, 2}[0, 1, 1, 0]
3. {0, 3}[1, 0, 0, 1]
4. {1, 3}[0, 1, 0, 1]
```

**Fig. 37.** The default mask status

Now, the fragment mask for our model is a vector of four bits. A bit `fN` set to 1 indicates that the corresponding fragment may be present in the winning alternative.

$$\texttt{mask} = (\texttt{f0}\,\texttt{f1}\,\texttt{f2}\,\texttt{f3})$$

We can use this mask to reduce the original list of alternatives A by clearing some bits in the mask. The sublist $\texttt{S} \subset \texttt{A}$ consists of all alternatives whose bit vector a masked with the fragment mask (bitwise AND) gives a. It can be formulated more precisely by the following formula.

$$\forall\, \texttt{a} \in \texttt{S};\ \texttt{a}\,\&\,\texttt{mask} = \texttt{a}$$

For a better manipulation with the fragment bits, let us construct the following matrix corresponding to the previous list of alternatives (Fig. 38).

```
     |1 0 1 0|
     |0 1 1 0|
F =  |1 0 0 1|
     |0 1 0 1|
```

**Fig. 38.** The mask status matrix

When no fragment is explicitly specified, the mask is `(1 1 1 1)`. Masking matrix `F` gives the same matrix. Thus there is no reduction of the original list of alternatives.

$$\texttt{F}\,\&\,\texttt{mask} = \begin{vmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \end{vmatrix}$$

If we wish to constrain the list of alternatives only to those not containing the `Cylinder` fragment, the mask will be (1 1 1 0). Masking matrix F then will yield the following matrix.

$$\texttt{F}\,\&\,\texttt{mask} = \begin{vmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{vmatrix}$$

```
1. {0, 2}[1, 0, 1, 0]
2. {1, 2}[0, 1, 1, 0]
```

**Fig. 39.** Remaining two alternatives after the masking

Here, only the first two rows remain same after the mask is applied. The first alternative in the sublist, which preserves the order of the original list, is the winner (Fig. 39).

If we clear an additional fragment to the mask, let us say Paper, the mask will be `(1 0 1 0)`. Then the masked matrix F will be:

$$\mathtt{F\,\&\,mask} = \begin{vmatrix} 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{vmatrix}$$

Only the first row remains unchanged, which becomes the winner.

$$1.\{0, 2\}[1, 0, 1, 0]$$

The *rating strategy* is used to rate individual alternatives. It may be used as a complement to the promoting strategy to fine-tune the resulting list of alternatives. The winner alternative is selected only from those alternatives having the highest rating with respect to their order. For a better understanding let us use the following notation, which depicts the rating of an alternative.

$$\{\mathtt{fragments}\} : \mathtt{Rating}$$

By default the alternatives have no rating, i.e. their rating is 0. The four alternatives from our model then look like this (Fig. 40):

We can explicitly assign new rating to a selected sub-set of the original alternatives. Let's say we will rate alternatives 1 and 4 by 1. Then the sub-list will look like that in Fig. 41 since only the two alternatives have the highest rating. The winner is the first alternative in the sub-list.

To wrap it up, the masking strategy is used to eliminate unsuitable alternatives. If more non-orthogonal masking strategies are chained there is a risk that there will be no alternative left to create the morph. On the other hand, the promoting strategy is used to recommend suitable alternatives, i.e. promote them higher in the list, and no alternative is removed. The rating strategy may be also used to shrink the list of the alternatives by means of rating the alternatives.

```
1. {0, 2}:0
2. {1, 2}:0
3. {0, 3}:0
4. {1, 3}:0
```

**Fig. 40.** The default list of alternatives with the default rating

A key property of morphing strategies is their *composability*; they do not have to provide the definite decision on which alternative is the winner. Instead, a strategy may constrain its decision on a certain sub-model of the main morph model. The final strategy then may be composed of several partial strategies, whose sub-models are orthogonal and complete. In other words, the individual partial decisions are not mutually contradicting and unambiguous when combined. I will illustrate this topic on the following example. It follows from the above that we do not have to create one big strategy selecting the winner alternative with respect to the whole morph model. Instead, we can create two partial strategies, one for the material dimension and the other for the shape dimension. The orthogonal sub-models of those strategies will be Metal or Paper and Cuboid or Cylinder. The partial masking strategy suggesting the winner in the material dimension is created by means of the mask macro (Fig. 42):

This macro is invoked with the sub-model type and a function returning the index of the winning alternative in the sub-model. The function returns 0 if the input item data contains material and 1 if it contains paper. The partial strategy is constructed similarly except it is chained with the material strategy, which is passed as the first argument to the macro (Fig. 43).

The two strategies are complementary; each selects one pair of alternatives from the main model, while the two pairs have always one common alternative. The shape strategy linked to the material strategy may be used as a complete strategy for creating the item morph.

```
val item = itemAsm.morph(shapeStrat)
```

```
1.  {0, 2}:1
2.  {1, 3}:1
```

**Fig. 41.** The remaining alternatives having the same highest rating

```
val materStrat = mask[Metal or
Paper](itemData.get("material") match {
  case "metal" => 0
  case "paper" => 1
})
```

**Fig. 42.** Creating the mask strategy for the material dimension.

```
val shapeStrat = mask[Cuboid or Cylinder]
(materStrat, itemData.get("material") match {
    case "cuboid" => 0
    case "cylinder" => 1
})
```

**Fig. 43.** Creating the chained mask strategy for the shape dimension

### 3.5 Determining Morph Type

RI uses a special macro for determining the type of a morph. The `select` macro is invoked with the type to be tested and with the morph instance. The macro returns optional result, i.e. `None` or Some in case the item conforms to the tested type (Fig. 44).

The `select` macro provides additional static type safety, since it can reveal invalid types passed as the argument. As long as the type argument is incompatible with the morph model of the morph instance, the macro aborts the compilation. The following statement would not compile because the item can never be compatible with `Metal with String` (Fig. 45).

```
select[Metal with Cylinder](item) match {
    Some(metalCylinder) =>
    None =>
}
```

**Fig. 44.** The select macro allows determining the type of the morph

```
Metal with String.

select[Metal with String](item)
```

**Fig. 45.** An invalid invocation of the select macro.

### 3.6 Wrappers

OM distinguishes fragments, which implement the core behavior, from *wrappers*, which are used to implement stackable modifications of the underlying traits. There are two types of wrappers: *dimension* and *fragment wrappers*. The difference lies in the way they override their inherited methods. A dimension wrapper is designed to override a dimension, i.e. abstract methods only (by means of abstract override), while a fragment wrapper is used to override a fragment's methods (by means override).

The `DensityCorrection` extension is an example of a dimension wrapper, since it overrides the `density` method defined in the Material dimension trait. In RI the `DensityCorrection` wrapper would be implemented as shown in the following listing (Fig. 46).

The wrapper should be incorporated into the item morph model as an optional fragment, since it is to be applied to some items only. Optional fragments are expressed

```
@dimension @wrapper
trait DensityCorrection extends Material{
  val coef: Float
  override def density() = coef * super.density()
}
```

**Fig. 46.** The `DensityCorrection` extension as a dimension wrapper

as a disjunction of the fragment and the neutral type Unit, i.e. (Unit or Fragment) or (Fragment or Unit).

The order determines whether the fragment will be by default active or not, since the left operand has a lower index and thus it will be in higher alternatives (Fig. 47).

```
val itemAsm = compose[Item with (Metal or Paper) with (Cuboid or
Cylinder) with (Unit or DensityCorrection)]
```

**Fig. 47.** Incorporating DensityCorrection to the item morph model

In order to activate the wrapper selectively with respect to the input data, we have to create an additional morphing strategy, which will activate the wrapper for some item data only. This new strategy will be chained with the shape strategy and become the top strategy (Fig. 48).

```
val densCorStrat = mask[Unit or DensityCorrection]
(shapeStrat,
  if (shouldCorrect(itemData)) 1 else 0)
```

**Fig. 48.** The mask strategy controlling the presence of DensityCorrection in morphs.

Note: By adding the optional wrapper we made the morph model three-dimensional.

### 3.7 Fragment Factories

In RI, a morph instance is in fact a dynamic proxy delegating invocations to the so-called fragment stubs. These *fragment stubs* correspond to the fragments in the model alternative used as the template for creating the morph. By default, RI uses default stub factories, which are analogous to default constructors, i.e. they do not allow passing arguments. The default stub factory creates a new stub instance on every morph instantiation. If one wishes to reuse the same stub instance when re-morphing, s/he can override the default stub factory by declaring an implicit variable prior to the statement creating the assembler. This implicit variable holds an alternative stub factory, such as one created by the `single` macro, which produces the singleton factory (Fig. 49).

The same mechanism may be used to initialize the fragment stub by some external data, i.e. to mimic a constructor with arguments. There is an overloaded single macro with one argument for the so-called configuration object. The following listing

```
implicit val cylFact = single[Cylinder]

val itemAsm = compose[Item with (Metal or Paper) with (Cuboid or
Cylinder)]
```

**Fig. 49.** Overriding the default fragment factory.

illustrates how to pass a configuration object to the singleton factory creating the Cylinder fragment stub. The configuration object is actually injected into the new stub instance in order for the corresponding stub's methods to delegate on it [9] (Fig. 50).

```
implicit val cylFact = single[Cylinder](
  CylinderInitData(itemData.get("radius"),
        itemData.get("height")))
```

**Fig. 50.** Supplying the configuration object to the fragment factory.

The configuration object must implement an interface, which is also implemented by the fragment. In this case the common interface is `CylinderData`. There is also a helper case class `CylinderInitData` implementing it to facilitate the creation of the configuration object (Fig. 51).

```
trait CylinderData {
  val radius: Float
  val height: Float
}
case class CylinderInitData(radius: Float, height: Float) ex-
tends CylinderData
```

**Fig. 51.** The configuration interface and the helper configuration case class.

The Cylinder fragment trait extends `CylinderData` decorated with the `dlg` marker to indicate that the abstract methods from the interface are to be automatically implemented by delegating on the configuration object injected on the creation of the fragment proxy (Fig. 52).

```
trait Cylinder extends Shape with dlg[CylinderData] {
def volume = PI * radius * radius * height }
```

**Fig. 52.** The Cylinder fragment trait declaring the configuration interface.

### 3.8 Putting the Pieces Together

The following listing wraps up the pieces scattered across the preceding text. The procedure overrides the default fragment stub factories to initialize the stubs with the input data and then it creates the item assembler and chains three partial strategies, which are used to create the item morph instance. In the end, the item instance is tested whether it is a metal cylinder.

## 4 Other OM Features

Because of the limited space in this paper I could not present in detail other interesting features of OM. If the reader is interested in them s/he may visit the home site of the project [8], where a lot of additional resources are available. In this section I am going to briefly explain only one of them.

### 4.1 Extending Morph Models

OM allows extending existing morph models by other models. The extending model is typically incomplete, which means that not all alternative it produces are valid because of missing dependencies. The model being extended is supposed to deliver those missing dependencies.

```
implicit val cylFact = single[Cylinder](
CylinderInitData(itemData.get("radius"),
                                    itemData.get("height")))
implicit val cubFact = single[Cuboid](
CuboidInitData(itemData.get("width"),
                                    itemData.get("height"),
                                    itemData.get("depth")))
implicit val metFact = single[Metal](
MetalInitData(itemData.get("density"))

implicit val papFact = single[Paper](
PaperInitData(itemData.get("density"))

val itemAsm = compose[Item with (Metal or Paper) with (Cuboid or
Cylinder) with (Unit or DensityCorrection)]

val materStrat = mask[Metal or Paper](
itemData.get("material") match {
                                    case "metal" => 0
                                    case "paper" => 1
})

val shapeStrat = mask[Cuboid or Cylinder](materStrat,
itemData.get("material") match {
                                    case "cuboid" => 0
                                    case "cylinder" => 1
})

val densCorStrat = mask[Unit or DensityCorrection](
shapeStrat,
  if (shouldCorrect(itemData)) 1 else 0)
val item = itemAsm.morph(densCorStrat)
select[Metal with Cylinder](item) match {
    Some(metalCylinder) =>
                    // use the metal cylinder None =>
}
```

**Fig. 53.** Putting the pieces together

To illustrate the extension of a morph model let us consider a model describing currencies recognized among the scanned items. There would be two types of currencies, coins and banknotes, represented by fragments Coin and Banknote extending the dimension trait Currency (Figs. 53 and 54).

```
trait Currency {
  …
}
trait Coin extends Currency {
  this: Metal with Cylinder =>
  …
}
trait Banknote extends Currency {
  this: Paper with Cuboid =>
  …
}
```

**Fig. 54.** Currency traits

Assuming that every coin corresponds to a metal cylinder item, we express this relationship by the self-type of the Coin fragment. In a similar way we express the binding between banknotes and paper cuboids (very thin).

The morph model for the currencies among items is very simple:

```
type CurrencyModel =
  Coin or Banknote or Unit
```

The model includes the neutral fragment Unit addressing the metal cylinders and paper cuboid not recognized as coins or banknotes.

The model is evidently incomplete because of the missing dependencies. However, by joining it with the item morph model we obtain a complete model, of which DNF is shown in the following listing (Fig. 55).

Two morph models are joined through the so-called morph references: `&[TargetModel]`.

```
val curRef : &[CurrencyModel] = item
```

The curRef variable is a reference to a new assembler initialized with the joined morph model, which is created at compile-time. The new assembler also contains the so-called *default morphing strategy*, which adapts the strategy used to create the item to the joined model. This adapted default strategy is automatically chained with any other strategy used to create morphs from the new joined model.

```
(Coin with Metal with Cylinder) or
(Banknote with Paper with Cuboid) or Unit
```

**Fig. 55.** The joined morph model

By dereferencing the reference we obtain the new assembler:

```
val curAsm = *(curRef)
```

Then we may use the assembler as usual, i.e. to create a currency morph:

```
val currency = curAsm.morph(CurrencyValidatingStrategy)
```

The `CurrencyValidatingStrategy` custom strategy validates if the proportions of metal cylinders and paper cuboids correspond to coins or banknotes. If not, the strategy selects the Unit alternative indicating an invalid currency. This strategy is chained with the above-mentioned default strategy adapted to the new model. In case the item is neither a metal cylinder nor a paper cuboid the default strategy selects the Unit alternative and eliminates the others. The `CurrencyValidatingStrategy` cannot override this decision, since it is a follow-up strategy in the chain.

The currency morph may be tested for its actual type by the select macro as usual:

```
select[Coin](currency) match {
  case Some(coin) => // use the coin
  case None =>
}
```

## 5 Conclusion

One of the goals of this paper was to show that modeling protean objects might be surprisingly difficult in established OO languages such as Java, Scala and Groovy as three representatives.

Java as a non-trait language has proven to be the least suitable language for the multidimensional modeling. Because of the lack of traits, one must resort to composition, which forces us to model "is-a" relationship as "has-a" one. The complications stemming from this approach are known as object schizophrenia.

In Scala, as a representative of the static languages, each form of a given object must be declared as a class. Since the number of forms grows exponentially with the number of dimensions, the number of class declarations and lines of the code quickly becomes unsustainable.

Groovy, which is a dynamic language, can cope with this problem by means of the dynamic traits applied to objects at runtime, however in contrast to Scala, its weak type system is not able to guarantee the consistency of trait compositions made in a step-by-step way.

It seems that neither static nor dynamic OO languages are capable of coping with the multidimensional modeling. As suggested in this paper, the problem could be solved by a hybrid approach combining some features from the dynamic and static languages and adding new ones.

As a candidate to such a hybrid approach is presented object metamorphism, which describes all possible forms of an object by the so-called morph model, which is verified at compile-time. This approach guarantees that all possible forms of the object, called alternatives, are consistent. At run-time, these alternatives are used as templates for assembling instances from fragment stubs.

The application of the reference implementation of OM to the example showed that OM is able to resolve all major issues detected in the analysis.

Additionally, OM comes up with other concepts such as morph model extensions, which allows joining incomplete morph models with complete ones. This concept may be used for domain mapping or for raising the level of abstraction.

## References

1. Harrison, W., Ossher, H.: Subject-oriented programming (a critique of pure objects). In: Proceedings of the Conference on Object-Oriented Programming Systems, Languages and Applications (OOPSLA 1993), Washington, D.C. ACM, September 1993
2. Harrison, W., Budinsky, F., Simmonds, I.: Subject-oriented programming: supporting decentralized development of objects, IBM TJ Watson Research Center (1995)
3. Coplien, J.O., Reenskaug, T.M.H.: The data, context and interaction paradigm. In: Proceedings of the 3rd Annual Conference on Systems, Programming, and Applications: Software for Humanity (SPLASH 2012), pp. 227–228. ACM, New York (2012)
4. Alam, O., Kienzle, J.: designing with inheritance and composition. In: Proceedings of the 3rd International Workshop on Variability & Composition (VariComp 2012), pp. 19–24. ACM, New York (2012)
5. Herrmann, S.: Demystifying object schizophrenia. In: Proceedings of the 4th Workshop on MechAnisms for SPEcialization, Generalization and inHerItance (MASPEGHI 2010). ACM, New York (2010)
6. Malayeri, D., Aldrich, J.: CZ: multiple inheritance without diamonds. In: Proceedings of the 24th ACM SIGPLAN Conference on Object Oriented Programming Systems Languages and Applications (OOPSLA 2009). ACM, New York (2009)
7. Odersky, M., Spoon, L., Venners, B.: Programming in Scala, Artima, 0981531644 (2008)
8. Šlajchrt, Z.: Morpheus: a scala extension introducing metamorphism of objects (2015). https://github.com/zslajchrt/morpheus
9. Wright, K.: A Taste of Scala: The Autoproxy Plugin (2009). http://www.artima.com/weblogs/viewpost.jsp?thread=275135

# Using Interactive Card Animations for Understanding of the Essential Aspects of Non-recursive Sorting Algorithms

Ladislav Végh(✉) and Ondrej Takáč

Department of Mathematics and Informatics,
J. Selye University, Komárno, Slovakia
{veghl, takaco}@ujs.sk

**Abstract.** Animations can help students to comprehend computer science algorithms. Previous experiments, mentioned in this paper, show that interactivity is very important in educational animations. In this contribution we also describe three categories of algorithm animations with different views, and we introduce our interactive card animations that belong to the first group (animations with conceptual views). These card animations of sorting algorithms were used in our experiment, where first-year computer science students were asked to fill out a pre-test, use the animations, and fill out a post-test. In the third part of the paper we discuss the obtained results, which proved that the interactive card animations can help students to understand the essential aspects of different sorting algorithms. Finally, we draw conclusions and introduce our future plans.

## 1 Introduction

Understanding algorithms is one of the hardest tasks for novice computer science students. One of the reasons why algorithms are difficult to understand is that they use abstract concepts. Algorithm animations may help to create a bridge between these abstract notions and real-world situations [1].

We used interactive card sorting animations in our experiment described in this paper. Our goal was to figure out if these interactive animations can help students to understand the essential aspects of different sorting algorithms. For this purpose, we used a pre-test and a post-test during the experiment, and we analyzed the obtained results.

Students prefer to use animations and visualizations in their study. Algorithm animations were used in computer science education since the 80$^{th}$ years of the twentieth century. Many experiments were conducted in this area during the last 30-35 years. It was surprising to researchers when these experiments showed mixed results [2–6].

After ambiguous results, researchers started to investigate in which circumstances is learning with animations and visualizations effective. Mayer in his book "Multimedia Learning" [7] defined twelve principles that could be helpful for creators of multimedia learning materials.

In the field of using animations for teaching and learning computer science algorithms were also made some recommendations. These suggestions were published in [2, 8].


J. Janech et al. (eds.), *Proceedings of the 2015 Federated Conference on Software Development and Object Technologies*, Advances in Intelligent Systems and Computing 511,
DOI 10.1007/978-3-319-46535-7_25

Several different research results suggest that visualizations and animations are more helpful in understanding computer science algorithms when students are not passive viewers of the animations, but they actively participate in visualization processes [2, 9]. Participation in the animations can be different: students can answer some questions, modify input data of the algorithms, move some objects of the animations, create the animations, explain the animations in front of their peers [9].

A meta-study comparing 21 different research papers also showed that the interactivity is very important in animations for learning (Fig. 1). From 9 experiments focusing on the graphical representation only 33% showed significant results; however, from 12 experiments focusing on the interactivity of the animations 83% showed significant results. The complete meta-study was published in [10].

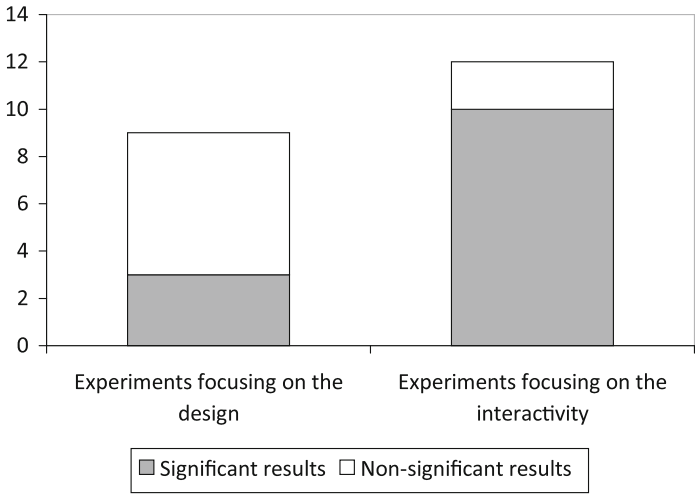



**Fig. 1.** Comparing the results of the researches focusing on the graphical design of the animations with the results of studies focusing on the interactivity of the animations [2]

The referred studies suggest that the computer science algorithm animations should be interactive for active learning. Students can experiment with interactive animations, they can change input data and observe the behaviors of the algorithms in different situations – all these activities support constructive learning.

Hansen et al. [4] proposed to use three different kinds of algorithm animations during the programming courses. These three animations show different views of the same computer science algorithm.

### 1.1 Animations with Conceptual View

Animations in this first category introduce the computer science algorithms using real world examples, e.g. sorting algorithms by using cards or crates [11]. Models from real world help students to connect the real world situations to the abstract concepts of the

algorithms. These animations provide students understanding of the essential aspects of the visualized algorithms [4]. In this paper, we focus on the animations that belong to this category.

### 1.2 Animations with Detailed View (Micro-Level)

Animations in this second category describe the algorithms at a very detailed level on small data sets (6-8 elements). Usually, a pseudocode is also included in the animations, where the animated steps are highlighted. An example of this kind of animation is on the Fig. 2.

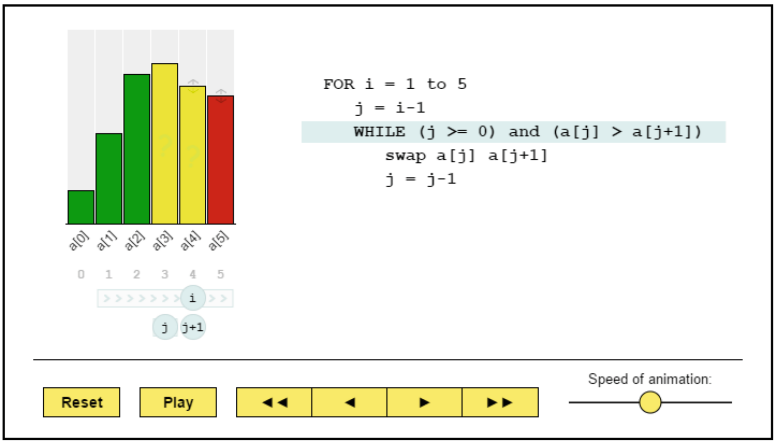



**Fig. 2.** Interactive animation with detailed view of the insertion sort algorithm

### 1.3 Animations with Populated View (Macro-Level)

Algorithms are demonstrated on large data sets (> 50 elements) in this third category. Many of the details of the animations should be removed so that the students can concentrate on the performance and behaviors of the algorithms at macro-level.

## 2 Materials and Methods

To help students comprehend the main features and differences of sorting algorithms we developed interactive algorithm animations using playing cards which are available on the web page http://anim.ide.sk/sortingcards.php. This collection of interactive game-based animations contains five non-recursive sorting algorithms: simple exchange sort, bubblesort, insertion sort (Fig. 3), and two types of the selection sort algorithm - minsort and maxsort. The reason we have chosen these algorithms is that these algorithms occur in the programming courses before any other more complex sorting algorithms.

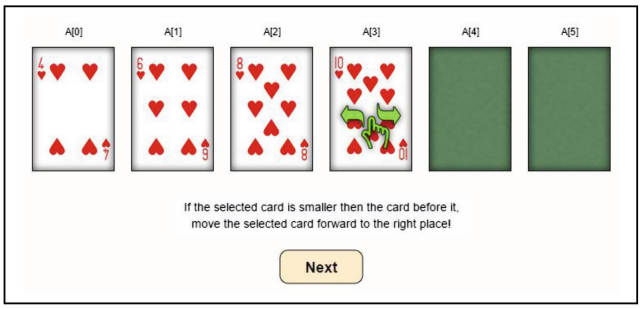



**Fig. 3.** Interactive game-based animations with conceptual view of the insertion sort algorithm

The animations were developed in HTML5 and Javascript programming language, using CreateJS libraries (http://www.createjs.com/). One of the reasons why we have decided to use these technologies is that they are starting to become a standard for creating web-based animations since the appearing of the HTML5 in October, 2014. Furthermore, usage of these techniques allows the developers to embed the animations into their web pages without any plugin. Creators of learning materials can easily integrate these types of animations with their electronic textbooks. Our interactive card animations were published on the portal http://algoanim.ide.sk/ as well, where the embed code is available for every animation.

In spite of the fact that we did not add any textual explanation to these animations in our experiment, we believe that the animations can help the most for the learners when they are integrated with learning materials, e.g. electronic textbooks, and followed by another, more detailed animations of the same algorithms. Adding explanations and embedding algorithm animations into the learning environment was proved by many experiments, e.g. [1, 5, 6].

We tried to create the animations as much interactive as they were possible but also wanted to guide students through the correct solutions. Students can grab the selected cards and exchange them with the other cards, but they have to follow the rules of the sorting algorithms. Moving or exchanging those cards that are not consistent with the actual steps of the algorithms are not allowed. In this way students construct the whole solution by learning the rules of the games, which are the main steps of the sorting algorithms.

In this phase of the research we wanted to prove that the interactive animations we created are effective in learning. The main question was if students can understand the essential differences between the selected sorting algorithms by using only our game-based animations.

The experiment was conducted during the "Algorithms and programming" course at J. Selye University, in the second semester of the academic year 2014/15. In the experience were involved 39 first-year students of computer science.

Students have not learned about any sorting algorithm in the "Algorithms and programming" course before the experiment. However, some of the students might have learned these sorting algorithms during other programming courses, in high school, or on their own.

All of the students were familiar with control structures, and they solved many problems on arrays, e.g. searching the minimum, maximum, counting some elements, calculating the sum, average, mirroring the array. They were also able to read and understand pseudocodes, but they did not learn the pseudocodes of the sorting algorithms.

At the beginning of our experiment, we asked our students to fill out a pre-test, which contained seven statements about the behaviors of the sorting algorithms. Students' task was to mark with X those cells in the table, where the statement- algorithm combinations are true. This pre-test is shown in Table 1.

**Table 1.** Test about the main differences between the selected sorting algorithms (For Easier Referring to the Cells of the Table in this Paper, We Added Letters to the Columns and Numbers to the Rows. These Marks were not Included in Students' Tests.)

| | **A. Simple exchange sort** | **B. Bubblesort** | **C. Insertion sort** | **D. Selection sort: Minsort** | **E. Selection sort: Maxsort** |
|---|---|---|---|---|---|
| 1. The algorithm always compares two neighboring elements in the array. | | | | | |
| 2. The algorithm compares every element with all elements located behind it. | | | | | |
| 3. First, the algorithm chooses one element from the unsorted part; next, the algorithm exchanges the selected element with the first or last element of the unsorted part. | | | | | |
| 4. In the unsorted part of the array, **the smallest element** is always moved to **the beginning** (the sorted sequence is starting to form in the beginning of the array). | | | | | |
| 5. In the unsorted part of the array, **the largest element** is always moved to **the end** (the sorted sequence is starting to form in the end of the array). | | | | | |
| 6. Elements in the sorted part of the array (in the beginning or in the end of the array) **are not modified (not moved)** during the sorting. | | | | | |
| 7. Elements in the sorted part of the array (in the beginning or in the end of the array) **can be modified (moved)** during the sorting. | | | | | |

Because most of the students were not familiar with any or some of these sorting algorithms, they were asked to mark only statements of sorting algorithms they knew. In this way we wanted to minimize the number of students' guesses.

In the next part of the pre-test, students had to connect the names of the five sorting algorithms with the pseudocodes of the algorithms. In this part of the pre-test students were asked again to pair only those sorting algorithms they knew.

After filling out the pre-test, students used the game-based animations, where they had to sort cards, following the rules of the algorithms. All five animations were accessible on a web page, in the following order: simple exchange sort, bubblesort, insertion sort, selection sort: minsort, selection sort: maxsort. The animations did not contain any explanation, only the names of the sorting algorithms, and the task to sort the cards in ascending order.

While students played with the animations, we gave them the post-test and asked them to fill it out. The post-test contained the same task as the pre-test, but, in addition, it also had a questionnaire to rate the clarity, user-friendliness, and the graphic quality of the animations from 1 to 10.

In the post-test we asked our students again to mark only those statement-algorithm combinations and pair only those pseudocodes with algorithms they knew, or learned by experimenting with the interactive card animations.

## 3 Results and Discussion

After the experiment we counted the number of X marks in the pre-test and post-test papers for every statement-algorithm combination.

On Fig. 4 is illustrated how many answers were marked correctly and how many were marked incorrectly by the students in the whole test.

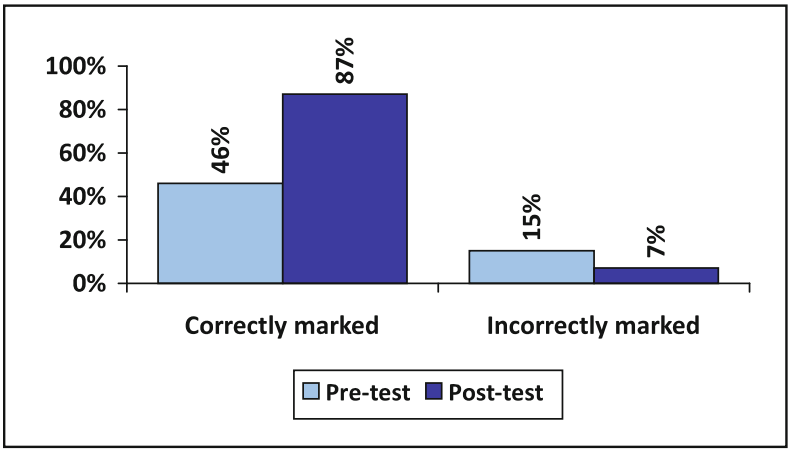



**Fig. 4.** Percentage of correctly and incorrectly marked answers in the whole test

The percentage of correctly marked answers increased by 90%, from 46% to 87%; while the percentage of incorrectly marked answers decrease by 51% from 15% to 7% during the experiment.

We also wanted to know how students answered for each of the seven statements.

Figure 5 shows the percentage of correctly marked answers, grouped by the statements. We can see in the graph that students marked significantly more right answers in the post- test, after experimenting with the interactive card animations.

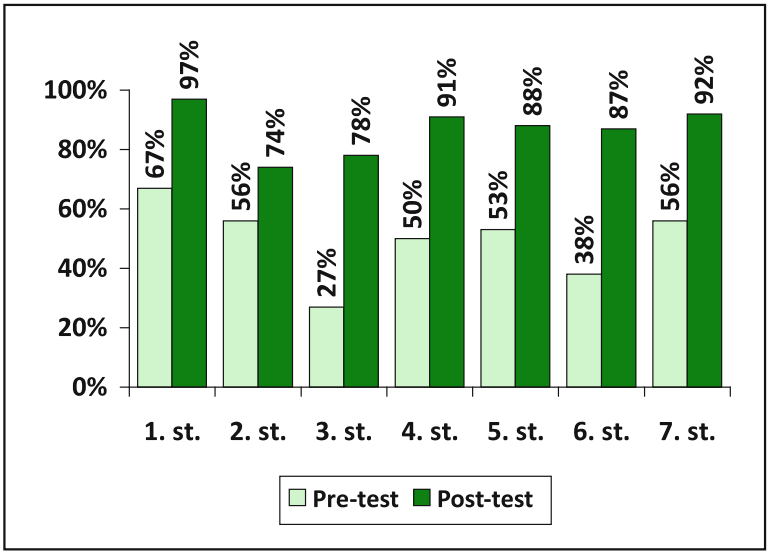



**Fig. 5.** Percentage of correct answers in the pre-test and post-test for true statement-algorithm combinations grouped by the statements

Figure 6 shows the percentage of incorrectly marked answers, grouped by the statements. The graph shows an overall decrease in students' wrongly marked answers in the post-test.

A more detailed view of correct answers is illustrated in Fig. 7. This graph shows the number of right answers in the pre-test and post-test for every true statement-algorithm combination. These combinations are those cells in Table 1, which students should have marked with X.

It is evident from the Fig. 7 that students identified more true statements after using the game-based animations. An interesting situation can be seen in the column 1-C, where no significantly more learners identified this statement as true in the post-test. The possible reason for this occurrence we will discuss later when trying to examine the results in more detail.

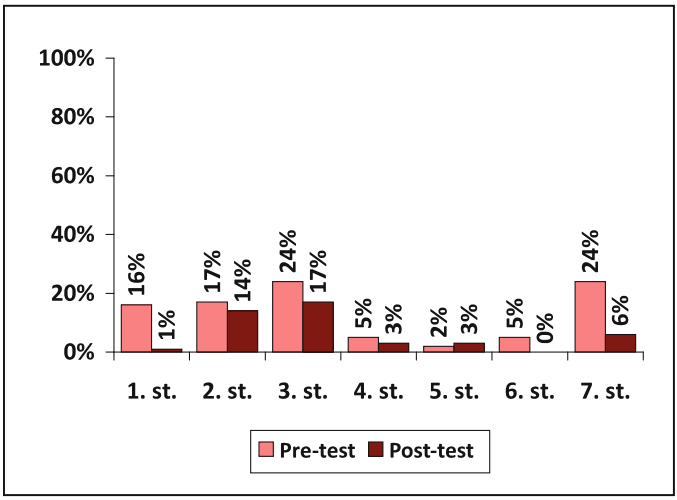



**Fig. 6.** Percentage of incorrect answers in the pre-test and post-test for false statement-algorithm combinations grouped by the statements

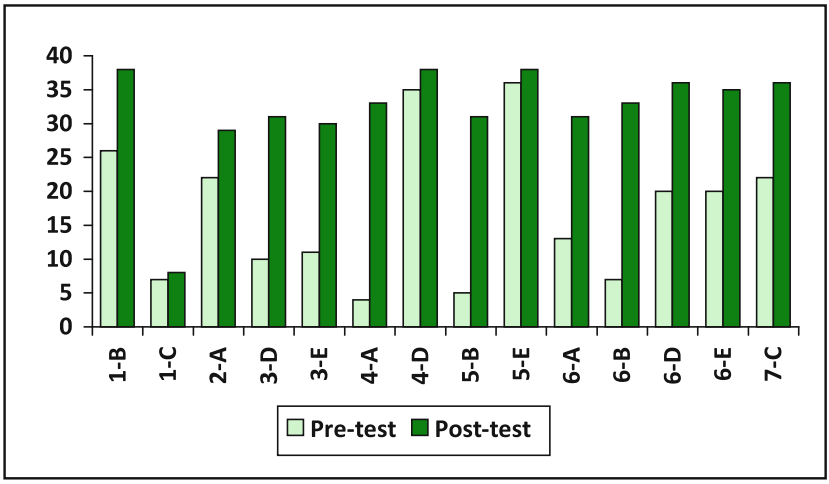



**Fig. 7.** Number of correct answers in the pre-test and post-test for true statement-algorithm combinations

In Fig. 8 the number of incorrectly marked answers are shown in the pre-test and post-test for every false statement-algorithm combination. These are those cells in Table 1, which students should have left blank.

This figure shows that students overall recognized false statements and marked them less as true in the post-test. However, in the columns 2-C, 3-A and 4-C significantly more learners identified the false statements as true. The possible reasons for these occurrences we will discuss later when trying to examine the results in more detail.

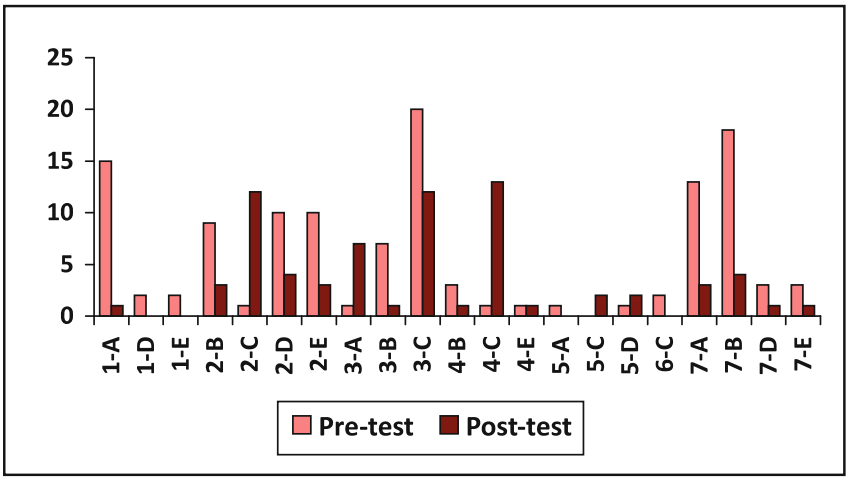



**Fig. 8.** Number of incorrect answers in the pre-test and post-test for false statement-algorithm combinations

The difference in the number of answers in the pre-test and post-test is shown in Table 2. In this table, we marked the true answers with "X", the disputably true answers with "X?", and the disputably false answers with "?".

**Table 2.** Difference in the number of answers in the pre-test and post-test ("X" – true answer, "X?" – disputably true answer, "?" – disputably false answer)

| | A | B | C | D | E |
|---|---|---|---|---|---|
| 1. st. | -14 | +12 X | +1 X? | -2 | -2 |
| 2. st. | +7 X | -6 | +11 ? | -6 | -4 |
| 3. st. | +6 | -6 | -8 | +21 X | +19 X |
| 4. st. | +29 X | -2 | +12 ? | +3 X | 0 |
| 5. st. | -1 | +26 X | +2 | +1 | +2 X |
| 6. st. | +18 X | +26 X | -2 | +16 X | +15 X |
| 7. st. | -10 | -14 | +14 X | -2 | -2 |

Table 2 shows that the number of correctly placed X marks in the post-test were 12 more than the number of correctly placed X marks in the pre-test for the first statement. The number of incorrectly placed X-marks for the first statement decreased by 18 (-14 -2 -2) in the post-test after using the game-based animations. Putting an X mark into the cell 1-C in the table can be evaluated as correct for simple insertion sort, but as incorrect for the improved insertion sort. In the game- based animation, there was not explicitly defined if students have to think about the improved or not improved version of the insertion sort algorithm. For this reason, we did not take into consideration the result from cell 1-C in the Figs. 4, 5 and 6.

In the second row of Table 2 we can see that the number of correctly place X marks increased by 7, while the number of incorrectly placed X marks decreased by 5 (-6 + 11-6 -4) in the post-test for the second statement. Cell 2-C of the table shows that the number of incorrect answers increased by 11 for the second statement when students thought about insertion sort algorithm. One reason they might believe that this statement is true for the insertion sort is that they did not think in details how this algorithm works. After all, thinking in details is not the goal of these animations with a conceptual view. We assume that, after understanding the animation of insertion sort with a detailed view, students would change their answer. More research is needed to confirm or refute this assumption.

The third row of the table shows that the number correctly place X marks increased by 40 (+21 +19) and the number if incorrectly placed X marks decreased by 8 (+6 -6 -8) in the post-test for the third statement. We do not know, why the number of incorrect answers increased by 6 for the third statement when learners thought about the simple exchange sorting algorithm (cell 3-A).

In the fourth row of Table 2 we can see that the number of correctly placed X marks increased by 32, and unfortunately, the number of incorrectly placed X marks also increase by 10 (-2 +12) in the post-test for the fourth statement. We have another disputable statement here, in the cell 4-C. The statement 4 is as follows: "In the unsorted part of the array, the smallest element is always moved to the beginning (the sorted sequence is starting to form in the beginning of the array)." The first part of this statement is false for insertion sort algorithm because the elements are not moved to the beginning of the unsorted part. However, the second part of the statement in the parentheses is true for insertion sort algorithm because the sorted sequence is starting to form in the beginning of the array. Because our statement 4-C was not clear enough, we did not take into consideration this result in the Figs. 4, 5 and 6.

The fifth row of the table shows that the number correctly placed X marks increased by 28 (+26 +2) in the post-test, but unfortunately the number of incorrectly placed X marks also increased by 2 (-1 +2 +1) in the post-test for the fifth statement. The increase of incorrectly placed X marks is not significant in this case.

In the sixth row of the table the number of correctly placed X marks increased by 75 (+18 +26 +16 +15), while the number of incorrectly placed X marks decreased by 2 in the post-test for the sixth statement.

The seventh row of Table 2 shows that the number of correctly placed X marks increased by 14, while the number of incorrectly placed X marks decreased by 28 (-10 -14 -2 -2) in the post-test for the seventh statement.

All these results suggest that students were able to recognize the main differences of the sorting algorithms easily, when they used interactive card animations.

In the second part of the pre-test and post-test, we asked students to pair the names of five sorting animations to their pseudocodes. The result is shown in Table 3.

As we see in this table, the overall number of correct answers is only slightly higher in the post-test. Surprisingly, high number of students paired some algorithms correctly in both tests, even though most of them did not learn about the sorting algorithms and the goal of these animations with conceptual view was not the understanding of the pseudocodes in detail. We assume, the reason of the similarly high number of correct answers in the pre-test and post-test for some sorting algorithms is not that students

**Table 3.** Percent's of correctly paired names to their pseudocodes in pre-test and post-test

| | **Simple exchange sort** | **Bubblesort** | **Insertion sort** | **Selection sort: Minsort** | **Selection sort: Maxsort** |
|---|---|---|---|---|---|
| pre-test | 51% | 41% | 59% | 97% | 97% |
| post-test | 69% | 46% | 56% | 97% | 97% |

knew in detail how these algorithms work, but they were able to recognize the names of the variables and/or some of the typical control structures in the pseudocodes. This assumption supports the previous part of the experiment, where most of the learners were not able to assess the essential features of the algorithms, only after using the interactive card animations.

Finally, we were interested in students' opinion about the interactive card animations. The result of this questionnaire is shown in Table 4.

**Table 4.** Students' ratings of the interactive card animations of sorting algorithms (10-scale rating: 1 – low, 10 – high)

| | **Simple exchange sort** | **Bubblesort** | **Insertion sort** | **Selection sort: Minsort** | **Selection sort: Maxsort** |
|---|---|---|---|---|---|
| ***Clarity*** | | | | | |
| Average: | 9.72 | 9.72 | 9.59 | 9.64 | 9.64 |
| StdDev: | 0.50 | 0.55 | 0.71 | 0.66 | 0.66 |
| ***User-friendliness*** | | | | | |
| Average: | 9.49 | 9.49 | 9.33 | 9.28 | 9.28 |
| StdDev: | 0.75 | 0.75 | 1.05 | 1.08 | 1.08 |
| ***Graphic Quality*** | | | | | |
| Average: | 9.56 | 9.56 | 9.49 | 9.56 | 9.56 |
| StdDev: | 0.74 | 0.74 | 0.84 | 0.74 | 0.74 |

Students gave high scores in the questionnaire, they liked the animations and were interested in learning algorithms using interactive animations and visualizations.

## 4 Conclusion and Future Plans

The obtained results show that interactive card animations can help students to understand the essential features of sorting algorithms. After comparing the percentages of correctly and incorrectly marked answers in the pre-test and post-test we can see improvement in students' understanding.

However, there is only a slight difference in the results of that part of the pre-test and post-test, where students had to pair the names of the sorting algorithms to their pseudocodes. This proves that more detailed animations are needed for deeper understanding of sorting algorithms.

Students enjoyed experimenting with interactive algorithm animations. They said they would like to learn more algorithms by using animations.

In future, we plan to integrate these card animations into an interactive online electronic book that can enhance students' comprehension. After each card animation, we plan to add another animation with a detailed view containing the pseudocode, and some exercises; thus learners will be able to accomplish the cognitive requirements for the first three levels in revised Bloom taxonomy [12] (the levels of remembering, understanding, and applying).

**Acknowledgment.** The contribution was published thanks to grant KEGA, 010UJS-4/2014 Modeling, simulation and animation in education (Modelovanie, simulácia a animácia vo vzdelávaní).

## References

1. Rudder, A., Bernard, M., Mohammed, S.: Teaching programming using visualization. In: Proceedings of the Sixth IASTED International Conference on Web-Based Education, pp. 487–492 (2007)
2. Naps, T.L., Rößling, G., Almstrum, V., Dann, W., Fleischer, R., Hundhausen, C., et al.: Exploring the role of visualization and engagement in computer science education. SIGCSE Bull. **35**, 131–152 (2002)
3. Byrne, M.D., Catrambone, R., Stasko, J.T.: Evaluating animations as student aids in learning computer algorithms. Comput. Educ. **33**, 253–278 (1999)
4. Hansen, S., Narayanan, N.H., Hegarty, M.: Designing educationally effective algorithm visualizations. J. Vis. Lang. Comput. **13**, 291–317 (2002)
5. Kann, C., Lindeman, R.W., Heller, R.: Integrating algorithm animation into a learning environment. Comput. Educ. **28**, 223–228 (1997)
6. Kehoe, C., Stasko, J., Taylor, A.: Rethinking the evaluation of algorithm animations as learning aids: an observational study. Int. J. Hum Comput Stud. **54**, 265–284 (2001)
7. Mayer, R.E.: Multimedia Learning, 2nd edn. Cambridge University Press, New York (2009)
8. Fleischer, R., Kucera, L.: Algorithm animation for teaching. Softw. Vis. **2269**, 113–128 (2002)
9. Grissom, S., McNally, M.F., Naps, T.: Algorithm visualization in CS education: comparing levels of student engagement. In: Presented at the Proceedings of the 2003 ACM Symposium on Software Visualization, San Diego, California (2003)
10. Hundhausen, C.D., Douglas, S.A., Stasko, J.T.: A meta-study of algorithm visualization effectiveness. J. Vis. Lang. Comput. **13**, 259–290 (2002)
11. Végh, L.: From Bubblesort to Quicksort with Playing a Game (Hravou formou od bublinkového triedenia po rýchle triedenie). In: XXIX International Colloquium on the Management of Educational Process, Brno, CZ, pp. 539–549 (2011)
12. Anderson, L.W., Krathwohl, D.R., Airasian, P.W., Cruikshank, K.A., Mayer, R.E., Pintrich, P.R., et al.: A Taxonomy for Learning, Teaching, and Assessing. A Revision of Bloom's Taxonomy of Educational Objectives. Addison Wesley Longman Inc, USA (2001)

# An Incremental Approach to Semantic Clustering Designed for Software Visualization

Juraj Vincúr(✉) and Ivan Polášek

Faculty of Informatics and Information Technologies,
Slovak University of Technology, Bratislava, Slovakia
{juraj.vincur,polasek}@fiit.stuba.sk

**Abstract.** In this paper, we introduce an incremental approach to semantic clustering, designed for software visualization, inspired by behavior of fire ant colony. Our technique focus on identification of equally sized but natural clusters that provides better hindsight of software system structure for development participants. We also address performance issues of existing approaches by maintaining similarities based on global weights incrementally, using subspaces and covariance matrix. Effectivity of visualization is improved by representing multiple documents with precise medoid approximation.

## 1 Introduction and Related Work

Understanding the structure of large and complex software systems is difficult task that demands huge effort. A certain level of understanding by software engineers is necessary when they perform daily development and maintenance tasks. Many approaches have been proposed to minimize the cost of such a comprehension, but most of them ignore huge amount of developer knowledge hidden in identifier names since they only focus on program structure [1] or documentation [2].

To enrich software analysis by this valuable source of information, Semantic Clustering has been proposed [3]. According to its evaluation this technique provides a useful first impression of an unfamiliar software system by grouping source code artifacts (e.g. class types, functions, methods) to clusters. Main problem of this approach identified by authors themselves lies in tendency of identifying one oversized and dozens of small clusters. Such a distribution is not suitable for software visualization and it is caused by use of average- linkage clustering algorithm. The authors also do not take in mind frequent software system changes (e.g. by adding features, bug fixing or refactoring) that each require recalculation of whole vector representation. This behavior leads to huge performance over-head as identified in [4, 21]. Use of incremental techniques should be considered to minimize time required for obtaining up-to-date representation since decisions based on non-up-to-date visualization of software system may be wrong and lead to degradation of its structure. Different approaches [2, 5] try to increase quality of identified topics by use of probabilistic model called Latent Dirchlet Allocation [6]. However, this method is very sensitive to user pre-defined number of resulted topics which are also harder to visualize.


J. Janech et al. (eds.), *Proceedings of the 2015 Federated Conference on Software Development and Object Technologies*, Advances in Intelligent Systems and Computing 511,
DOI 10.1007/978-3-319-46535-7_26

It is obvious that robustness and quality of visualization based on Semantic Clusters is highly related to choice of clustering algorithm. An exhausting review of clustering algorithms may be found in [7]. We will focus on 3 types of cluster analysis techniques since our method is partially related to them.

First set of methods are members of hierarchical cluster analysis which aims to build hierarchy of clusters by using generally two strategies. Agglomerative, also known as bottom-up approach, which in initial phase considers each observation as cluster. Pairs of clusters are iteratively merged until only one huge cluster remains or the other defined criterion is met. Divisive, top-down approach, treats data in opposite direction. An efficient and scalable representative of agglomerative methods is BIRCH (Balanced Iterative Reducing and Clustering) [9] which requires two input parameter: the distance threshold and the tree branching factor. With respect to these parameters the BIRCH algorithm build CF-tree consisted of summary information (clustering features) about candidate clusters. Then, instead of using full data set, another agglomerative clustering algorithm is applied on these features to refine clusters. This approach is suitable for very large data sets, however, since each node in CF-tree can contain only limited number of data items obtained clusters may not correspond to natural clusters. BIRCH also requires significant effort to tune its parameters [8]. Non spherical clusters trouble BIRCH as well since it uses diameter or radius to control boundaries of clusters.

Second set of algorithms are based on minimum or maximum spanning tree. Multiple non-incremental approaches are described in [10]. This type of methods may be considered as subset of hierarchical clustering. Analogy between them is well described in [11], where MST implementation of single- linkage clustering is presented. We are not aware of any incremental approach based on spanning tree even though their base implementations are relatively poor in performance with respect to BIRCH.

Third group consists of ant colony algorithms. These algorithms, members of swarm intelligence family, are inspired by multiple species and behavior models of ants. A brief survey may be found in [12].

With respect to identified limitations, this paper aims at providing incremental approach to semantic clustering that identifies equally sized but natural clusters. Section 2 explains concepts of our approach. Section 3 presents our adaption of circle packing layout to visualize obtained semantic clusters. Results and evaluation are presented and discussed in Sect. 4. Section 5 summarizes this paper and its contribution.

## 2 The Fire Ant Approach to Clustering

Our clustering approach is inspired by behavior of fire ant colony during floods. Fire ants have unique ability to gather together and form rafts on the surface of water [14]. This formation in which they can retain for weeks, allows them to survive. We assume that each ant has limited vision range in which it can sense the other ants. We also assume that during gathering ants are dispersed in space so first clusters of ants are formed and then these clusters are merged to one big raft (hierarchy). Moving

principles of these ants are often compared to fluids. The behavior we described is similar to oil spreading on the surface of water. In this paper we work with simplified model that considers only one possible way of grabbing another ant (instead of 7), by its jaw. Number of survivors then depends on number of connected ants (raft size). To maximize it, we add restriction to suppress cycle connections between ants, thus we get spanning tree representation, which is close to minimum.

In this section, we will first present our domain specific observations that were considered in design and implementation. Next, we will discuss main steps of our approach in following order: preprocessing (building corpus representation), spanning tree construction, identification of cluster-points, assigning documents to cluster-points and merging cluster-points to clusters.

### 2.1 Domain Observations

Most approaches treat source code snippets as textual documents although they are produced differently. Source code projects are usually implemented in incremental manner by applying batches of changes. Dimensionality of space representing documents affected by single batch should be relatively low since these documents are usually related (e.g. classes affected by single commit). To support our hypothesis we conduct an experiment in which we simulate development of few open source projects handled by Git version control system. Description of chosen projects may be found in Table 2. Results of experiments are shown in Fig. 1. Data are summarized in Table 1. As we can see, average values of dimensionality of such a subspace lie between 172.07 and 297.14, which is relatively small compared to total number of dimensions (see Table 2). Moreover, it doesn't seem to be strongly related to project size. Trend lines in Fig. 1 even suggest that this dimensionality may be constant (in average). Another

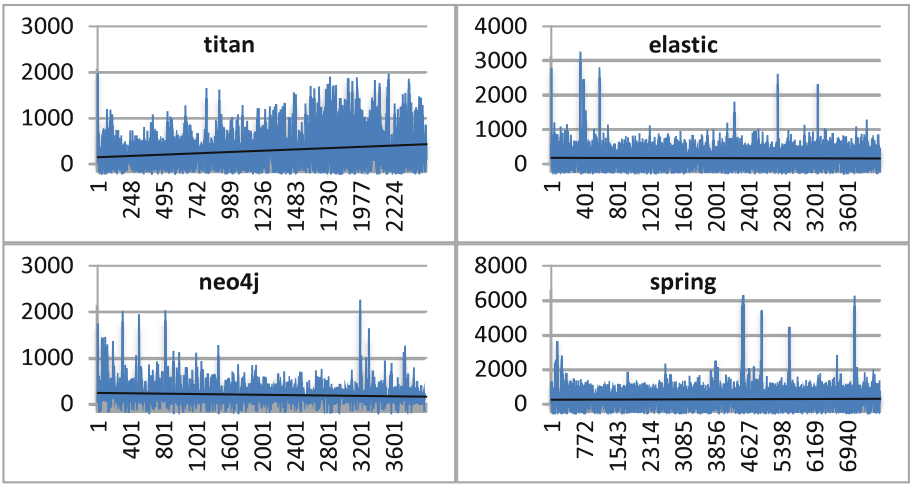



**Fig. 1.** Number of dimensions per commit subspace (with trend lines)

**Table 1.** Average value[a] in covariance matrix of increment

| | Titan | Elastic | Neo4j | Spring | Total |
|---|---|---|---|---|---|
| avg sim | 0.35 | 0.48 | 0.34 | 0.39 | 0.39 |
| med dim | 146 | 132 | 200 | 196 | 199 |
| avg dim | 201.92 | 172.07 | 270.07 | 297.14 | 235.30 |

[a]Diagonal values set to 0.

**Table 2.** Properties of DTM of chosen projects

| | Titan | Elastic | Neo4j | Spring |
|---|---|---|---|---|
| row | 1238 | 5593 | 5753 | 9833 |
| col | 3226 | 5297 | 5257 | 6562 |
| nnz | 63009 | 249107 | 225114 | 414638 |
| density | 1.58% | 0.84% | 0.74% | 0.64% |
| avg | 50.90 | 44.54 | 39.13 | 42.17 |

aspect we were interested in is the density of document-term matrix (DTM) which is comparable to domain of textual documents (see Table 2).

## 2.2 Corpus Representation

Quality of clusters is highly dependent on precision of similarity measure obtained in preprocessing step. In our work we use cosine similarity [17] and corpus representation that exploits global weights and low density of document-term matrix. Primary corpus representation is constructed as document-term matrix with term frequency (TF) weighting. No normalization vector or global scale is applied because these may change in time by editing corpus. However, vector of document lengths and average document length are updated for further usage. Document lengths are used for calculation of IDF [15] and average document length for pivot normalization [19]. TF weights are used for serialization purposes only. In order to calculate document similarity, TF-IDF model has to be computed by applying IDF vector to normalized primary matrix. We consider global weight in TF-IDF essential due to its tendency to suppress false cluster identification caused by crosscutting concerns [16]. For example, using raw TF weights, term "map" widely spread across classes in source code of distributed database may cause identification of oversized cluster that will contain all these classes. Problem with TF-IDF model in terms of incremental approaches is that weights change by adding, removing or editing documents. To overcome this behavior or even employ it we compute covariance matrix. We assume that documents similar at some particular not too early point of project development should be relatively equally similar at later point of development as well. Moreover, we assume that earlier point similarity is more precise because it takes in account aspect of time. If two documents were highly similar before some words become widely spread (by adding some module or by merging branches) they are probably similar even if new TF-IDF weights say otherwise. To add new documents to covariance matrix, their similarities to all

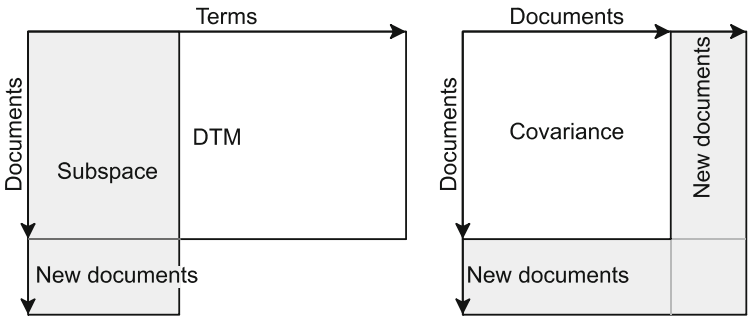



**Fig. 2.** Adding increment to covariance matrix.

documents have to be calculated (see Fig. 2). However, since the subspace for these documents tends to be constant in number of dimensions (term dimension of DTM), only one dimension of matrix is growing (document dimension of DTM). Moreover, document dimension may be filtered as well because there is high probability that relatively low dimensional vector (rounded average of 44 dimensions for our sample) extracted from high-dimensional space (rounded average 5085) representing single class has no common dimensions with constructed subspace (rounded average 235).

Since both matrices, TF weights and TF-IDF model are sparse, we address memory efficiency problem by storing them using sparse matrix schema. More specifically, TF weights are stored using dictionary of keys format, which assigns to each row, column tuple corresponding value. This structure is efficient for constructing sparse matrices incrementally, for row and column slicing, but not efficient for arithmetic operations. On the other hand, TF-IDF model is represented by compressed sparse row matrix format. This format consists of indices (array of column indices), data (array of nonzero values) and pointer array whose items points to row starts in indices and data. It is an efficient representation for arithmetic operations and dot product, but very slow representation for column slicing or changing structure.

### 2.3 Spanning Tree Construction

Simplified model based on spanning tree structure has been selected due to several benefits. It is easy to maintain, analyze and visualize. Moreover, spanning tree clustering methods are capable of identifying clusters with irregular boundaries (e.g. not spherical) [13]. In our algorithm, inspired by swarm intelligence, we represent each document by single agent. Each agent has limited vision of k nearest/most similar nodes based on cosine similarity, obtained from covariance matrix (storing full covariance matrix isn't necessary). Since ants are capable of making max. 7 connections we set k to this value. The steps required to compute spanning tree are shown in Fig. 3. This approach has been chosen over other algorithms with concurrency [20] due to the time complexity, its natural incrementalism and fact that using nearly minimum spanning tree has no significant impact on quality of resulted clusters. Note that

```
INIT graph
FOR I = 1 to K
  FOR each document
    SET doci to its I-th most similar document
  IF edge between nodes representing document and doci will not pro-
     duce cycle in graph THEN add edge between these nodes
  ENDIF END FOR
END FOR
```

**Fig. 3.** Pseudocode of spanning tree construction algorithm

representing each document as single agent brings certain level of rivalry that leads to tendency of equally sized clusters, but not for price of having non-natural clusters.

### 2.4 Cluster-Points

In created spanning tree, we select nodes that have more than 2 links. We call them cluster-points and they are good approximation of medoid for group consisted of selected node and nodes adjacent to him. Note that this group represents group of mutually similar documents since it was obtained from spanning tree close to minimum. Table 3. presents measured number of incorrectly assigned medoids, total number of identified cluster-points, total error (sum of differences between total dissimilarity of real medoid and approximated medoid) and average error. As we can see, only few medoids are approximated incorrectly and even these bad approximations yield low average error. Nodes with degree 2 will be referred as connectors.

**Table 3.** Average value in covariance matrix of increment

| | JHotDraw | Titan | Elastic | Neo4j | Spring |
|---|---|---|---|---|---|
| missed | 4 | 8 | 73 | 68 | 156 |
| total | 86 | 310 | 1336 | 1396 | 2349 |
| error | 0.07 | 0.22 | 3.96 | 3.34 | 4.58 |
| avg error | 0.02 | 0.03 | 0.05 | 0.05 | 0.03 |

### 2.5 Forming Clusters

Selecting cluster-points in spanning tree revealed high probability of forming groups of adjacent cluster-points. These groups refer to cluster candidates. Different strategies may be applied to obtain real clusters. The one we present could be described by pseudo code in Fig. 5. Each one of resulted clusters is represented by list of cluster-points where each cluster-point has list of assigned nodes. Example of spanning tree and obtained hierarchy tree structure may be found in Fig. 4.

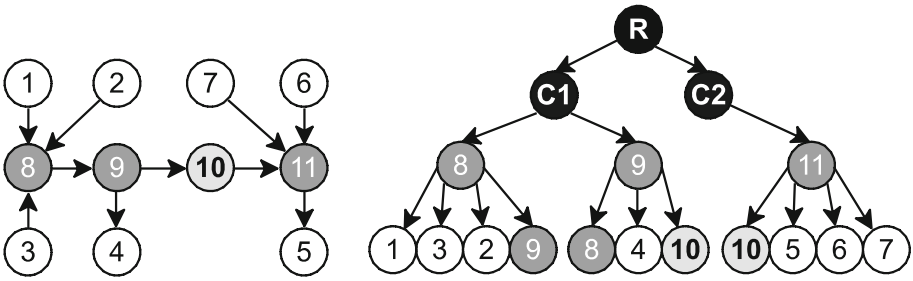



**Fig. 4.** Spanning tree (left) and corresponding hierarchy (right). Black nodes - root and identified clusters; dark gray – cluster-points; gray – connectors

### 2.6 Updating Structure

Two main strategies have to be considered in terms of maintaining spanning tree and cluster hierarchy structure. In first, building of vector space model and covariance matrix is managed incrementally and construction of spanning tree and cluster hierarchy is recalculated each time since it requires only single pass over documents. This strategy will probably yield more accurate results, but it causes certain level of overhead.

```
SET subg to subgraph of spanning tree that contains only cluster-points
   FOR each component in connected components of subg
  IF number of nodes in component is smaller than defined level of abstrac-
     tion THEN
    exclude nodes of component from cluster-points ENDIF
   END FOR
   SET level to 0
WHILE not all nodes of spanning tree are traversed SET level = level +
  1
     FOR each cluster-point
    SET traversed to nodes traversed in spanning tree by BFS at current
        level initialized from current cluster-point
       FOR each node in traversed
      IF node is not cluster-point and not already assigned THEN
        assign node to current cluster-point ELSE IF level == 1
      THEN
          assign copy of node to current cluster-point
        to prevent medoid representation END IF
    END FOR END FOR
   END WHILE
   SET subg to subgraph of spanning tree that contains
       only cluster-points
   SET clusters to connected components of subg
```

**Fig. 5.** Pseudocode for cluster-forming strategy

Second strategy maintains spanning tree and cluster hierarchy incrementally as well by reducing recalculations to affected subtrees only. Formed clusters will be more stable, but less accurate.

## 3 Data Visualization

Most companies develop large software projects using version control system. Source code is physically located on one or multiple servers what allows us to create web based visualization tools. Main benefit of this approach is the possibility to see the results everywhere, immediately without any installation required. There are many visualization JavaScript libraries suitable for this purpose. Most of them require only JSON dump in specified format. Our software visualization, inspired by [24], use d3.js library which provides robust documentation and large database of examples. From these examples we adapt circle packing layout (see Fig. 6).

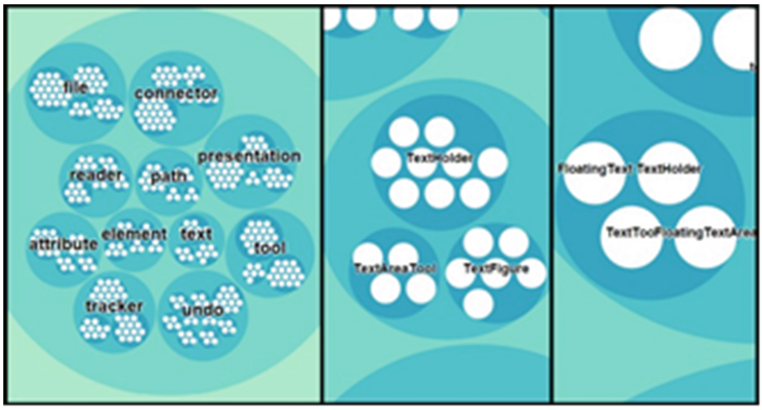



**Fig. 6.** Circle packing layout (visualization of JHotDraw, zoomed views after clicking "text" cluster and "TextAreaTool" cluster-point)

## 4 Results and Evaluation

Discussing our results we will mainly focus on quality of clusters from view of software engineer trying to understand the structure of existing software system. Use of external validity techniques would be the most suitable, but since we are not aware of any evaluated data set from source code domain relevance judgments [18] are applied.

Relatively small project as JHotDraw yield with abstraction level 1 11 clusters. As we can see in Fig. 6 (left side of picture) these clusters are relatively equally sized. We judge the using of cluster-points as representative of group of documents very positively (see Table 4). A very good example is class AlignAction that represents classes North, East, West, South, Vertical and Horizontal. Similar behavior is observed in all

**Table 4.** Example of identified cluster-points in JHotDraw

| Cluster-point | Classes that cluster-point represents |
|---|---|
| AlignAction | East, West, South, North, Vertical, Horizontal |
| Handle | NullHandle, CloseHandle, HandleMulticaster, HandleListener |
| OpenAction | NewAction, OpenRecentAction, Project, … |
| DuplicateAction | CopyAction, PasteAction, CutAction, DeleteAction |

sample projects. To proof our spanning tree based concept in terms of natural clusters, we compare our approach with selected algorithms (Birch and average linkage clustering, or UPGMA) in two-dimensional space with well separated clusters. Without use of any cluster-forming strategy or tune of parameters we were able to correctly identify all natural clusters (see Fig. 7). Such an experiment is not considered as sufficient evaluation since there's no proof of relation between two and multidimensional space but it serves as an appropriate example to present potential of proposed technique.

To briefly evaluate our approach in multidimensional space we create a table of clusters with corresponding cluster-points that were identified in JHotDraw project (see Table 5). With our basic knowledge about JHotDraw architecture, we were able to identify four core parts of system (see Table 6). By comparing these components to obtained clusters we observed that part "Drawing, figure" corresponds to cluster "Undo, redo, decorator", parts "Handle" and "Tool" to "Tracker, mouse, evt" and "Command" to "File, project, open". In addition we identified two clusters related to XML since clipboard operations and external storage in JHotDraw use this format. First, cluster "Reader, xml, entity", is related to XML parsing while second, "Element, and attribute, open", and focus on XML DOM manipulation. Cluster "Path, node, bezier" is related to fact that support for lines and complex shapes in JHotDraw is based

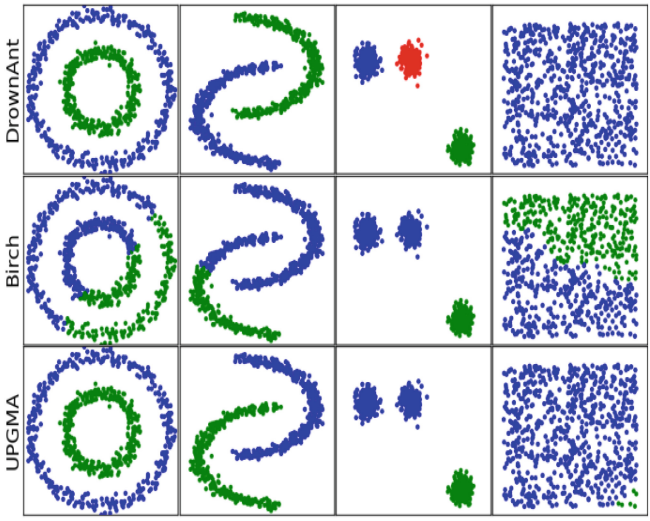



**Fig. 7.** Comparission of our approach (DrownAnt) with selected algorithms in 2D space

**Table 5.** Identifiet clusters and cluster-points in jhotdraw

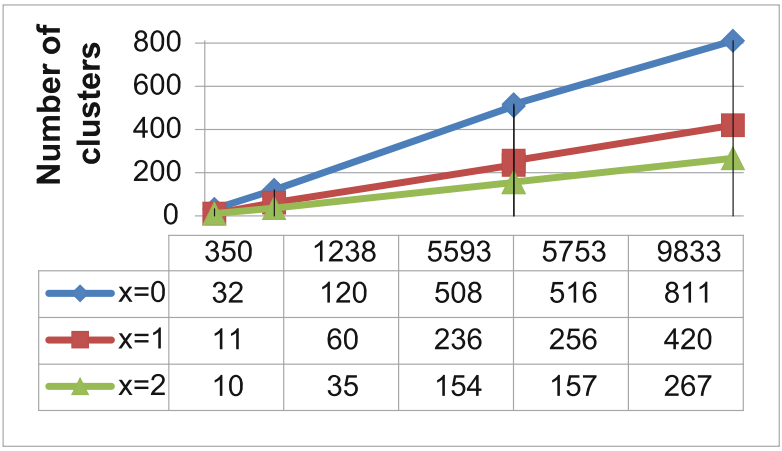
| Cluster | Cluster-points of cluster |
|---|---|
| Undo, redo, decorator | Figure, FigureEvent, AbstractFigure, AbstractFigureListener, UndoRedoManager, AttributeChangeEdit, SetBoundsEdit, RestoreDataEdit, RunnableWorker |
| Tracker, mouse, evt | SelectionTool, FormListener, HandleTracker |
| File, project, open | LoadRecentAction, NewAction, OpenAction, OpenRecentAction |
| Reader, xml, entity | XMLUtil, StdXMLParser, IXMLParser, IXMLValidator, NonValidator |
| Element, attribute, open | JavaxDOMOutput, DOMOutput, NanoXMLLiteDOMOutput, NanoXMLDOMInput, DOMInput |
| Connector, target, connection | ChopBoxConnector, Connector, ChopEllipseConnector, BidirectionalConnectionTool, ChangeConnectionHandle |
| Path, node, bezier | BezierTool, BezierPath, BezierFigure, BezierControlPointHandle |
| Text, origin, layout | TextFigure, TextAreaTool, TextHolder |
| Presentation, figure, graphical | HorizontalLayouter, AbstractLayouter, ListFigure, GraphicalCompositeFigure |
| Attribute, key, forbid | ColorChooserAction, DefaultAttributeAction, AttributeAction, AbstractAttributedCompositeFigure |
| Tool, bar, toolbar | ToggleToolBarAction, DrawApplicationModel, ToolBarPrefsHandler |

**Table 6.** Identified core parts of JHotDraw

| Part | Responsibility |
|---|---|
| Drawing, figure | Drawing represents two-dimensional space and consists of figures |
| Tool | DrawingView inputs are delegated to its current tool |
| Handle | Used to change a figure by direct manipulation |
| Command | Classes related to command design pattern that provide basic project actions (save, exit, open) |

on bezier paths (instead of polygons). “Text, origin, layout” corresponds to classes handling text areas, “Presentation, figure, graphical” to layout and composition of figures, “Attribute, key, forbid” to setting attributes of figures and “Tool, bar, toolbar” to toolbars and panels of drawing view.

During experiments we did not expect frequent occurrence of the term “Map” in project Titan (distributed database) across clusters. After short analysis we realized that term “map” occurs in 345 classes from total number of 1238 and so it is related to cross-cutting topic. IDF works correctly in this example. During the browsing of documents of one cluster, we realize that sometimes their vocabulary is not similar. This is caused by use of vector enrichment as proposed in [4]. Such an observation may be confusing for user as well, thus some abstraction of this relation should be added to visualization.




| | 350 | 1238 | 5593 | 5753 | 9833 |
|---|---|---|---|---|---|
| x=0 | 32 | 120 | 508 | 516 | 811 |
| x=1 | 11 | 60 | 236 | 256 | 420 |
| x=2 | 10 | 35 | 154 | 157 | 267 |

**Fig. 8.** Number of identified clusters

Size of the clusters is considered rational with respect to total number of documents. According to Fig. 8 the number of clusters may scale linearly or even logarithmic, but to support these assumptions statistically we need more significant number of samples to analyze.

In Fig. 9 the results of performance evaluation of computing covariance matrix are presented. Incremental approach is compared with calculation of full covariance matrix each time, to see how both of them scale by number of applied commits. As we can see, time required by base approach grows significantly faster than the time required by

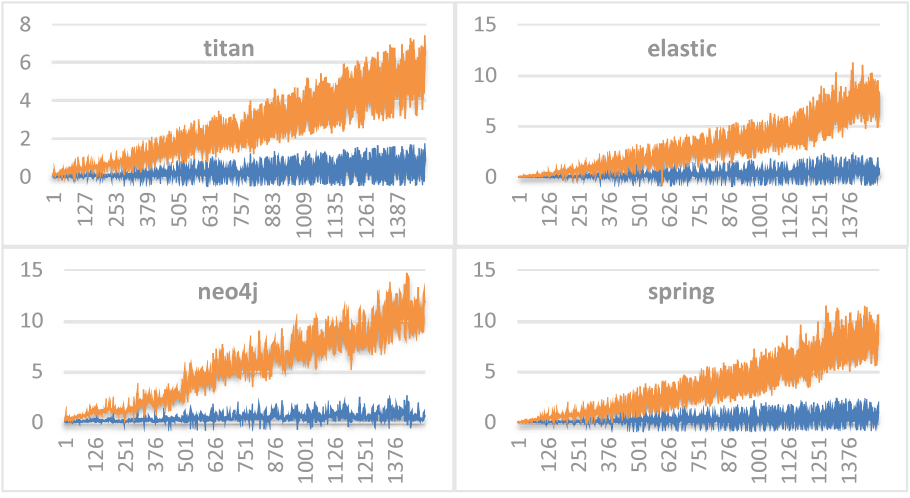



**Fig. 9.** Comparison of computational time between base (orange) and incremental (blue) approach of computing covariance matrix

incremental. Moreover, incremental approach will scale even better after applying document filtering. Since our current implementation of adding increments to covariance matrix scales linearly, there is yet no point of managing spanning tree and cluster hierarchy incrementally (we can recalculate them by single pass over documents).

## 5 Conclusion and Future Work

In this paper we propose new incremental technique for semantic clustering, designed for software system visualization, inspired by behavior of fire ant colony, slightly sensible to optional tuning of single parameter, which is capable of identifying natural, but relatively equally sized clusters even with irregular boundaries. We employed low density of DTM to minimize performance overhead by calculating document to document similarities in low dimensional subspaces and by storing them to incrementally maintained covariance matrix. We reduced size of browsing space by applying three-layer hierarchy visualization with medoid based abstraction. To provide efficient preview of documents, we added to our visualization source code browsing. As we showed in evaluation, we achieve intuitive representation that provides good hindsight of software system structure and functionality.

In future work, our primary goal is to design and implement 3D visualization of our software system representation that utilize benefits of the third dimension as suggested in [22, 23]. Moreover, we would like to integrate resulted tool in the IDE and enrich it by mixed reality features and advanced interaction techniques.

**Acknowledgments.** This work was supported by the Scientific Grant Agency of Slovak Republic (VEGA) under the grant No. VG 1/1221/12. This contribution is also a partial result of the Research & Development Operational Programme for the project Research of Methods for Acquisition, Analysis and Personalized Conveying of Information and Knowledge, ITMS 26240220039, co-funded by the ERDF.

## References

1. DeLine, R., Rowan, K.: Code canvas: zooming towards better development environments. In: Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering (ICSE 2010), vol. 2, pp. 207–210. ACM, New York (2010)
2. Asuncion, H.U., Asuncion, A.U., Taylor, R.N.: Software traceability with topic modeling. In: Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering (ICSE 2010), vol. 1, pp. 95–104. ACM, New York (2010)
3. Kuhn, A., Ducasse, S., Gírba, T.: Semantic clustering: identifying topics in source code. Inf. Softw. Technol. **49**(3), 230–243 (2007)
4. Uhlár, M., Polasek, I.: Extracting, identifiyng and visualisation of the content in software projects. In: Proceedings of the 4th World Congress on Nature and Biologically Inspired Computing (NaBIC 2012), November 2012, pp. 72–78. IEEE Press (2012)
5. Linstead, E., Rigor, P., Bajracharya, S., Lopes, C., Baldi, P.: Mining concepts from code with probabilistic topic models. In: Proceedings of the 22nd IEEE/ACM International Conference on Automated Software Engineering (ASE 2007), pp. 461–464. ACM, New York (2007)

6. Blei, D. M., Ng, A. Y., Jordan, M. I.: Latent Dirichlet allocation. J. Mach. Learn. Res. 3 (March 2003), 993–1022March
7. Jain, A.K., Murty, M.N., Flynn, P.J.: Data clustering: a review. ACM Comput. Surv. **31**(3), 264–323 (1999)
8. Ackermann, M.R., Märtens, M., Raupach, C., Swierkot, K., Lammersen, C., Sohler, C.: StreamKM++: a clustering algorithm for data streams. J. Exp. Algorithmics **17**, 1–31 (2012). Article 2.4
9. Zhang, T., Ramakrishnan, R., Livny, M.: BIRCH: an efficient data clustering method for very large databases. SIGMOD Rec. **25**(2), 103–114 (1996)
10. Grygorash, O., Zhou, Y., Jorgensen, Z.: Minimum spanning tree based clustering algorithms. In: Proceedings of the 18th IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2006), pp. 73–81. IEEE Computer Society, Washington, DC (2006)
11. Gower, J.C., Ross, G.J.S.: Minimum spanning trees and single linkage cluster analysis. Appl. Stat. **18**, 54–64 (1969)
12. Jafar, O.M., Sivakumar, R.: Ant-based clustering algorithms a brief survey. Int. J. Comput. Theor. Eng. **2**(5), 787–796 (2010)
13. Zahn, C.T.: Graph-theoretical methods for detecting and describing gestalt clusters. IEEE Trans. Comput. **20**(1), 68–86 (1971)
14. Mlot, N.J., Tovey, C.A., Hu, D.L.: Fire ants self-assemble into waterproof rafts to survive floods. Proc. Natl. Acad. Sci. USA **108**(19), 7669–7673 (2011)
15. Paltoglou, G., Thelwall, M.: A study of information retrieval weighting schemes for sentiment analysis. In: Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL 2010), pp. 1386–1395. Association for Computational Linguistics, Stroudsburg (2010)
16. Kiczales, G., Lamping, J., Mendhekar, A., Maeda, C., Lopes, C., Loingtier, J.-M., Irwin, J.: Aspect-oriented programming. In: Akşit, M., Matsuoka, S. (eds.) ECOOP 1997. LNCS, vol. 1241, pp. 220–242. Springer, Heidelberg (1997). doi:10.1007/BFb0053381
17. Salton, G., McGill, M.J.: Introduction to Modern Information Retrieval. McGraw-Hill Inc., New York (1986)
18. Voorhees, E.M.: Variations in relevance judgments and the measurement of retrieval effectiveness. In: Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 1998), pp. 315–323. ACM, New York (1998)
19. Singhal, A., Buckley, C., Mitra, M.: Pivoted document length normalization. In: Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 1996), pp. 21–29. ACM, New York (1996)
20. Nešetřil, J., Milková, E., Nešetřilová, H.: Otakar Borůvka on minimum spanning tree problem translation of both the 1926 papers, comments, history. Discrete Math. **233**, 1–3, 3–36 (2001)
21. Polášek, I., Uhlár, M.: Extracting, identifying and visualisation of the content, users and authors in software projects. In: Gavrilova, M.L., Tan, C.J.K., Abraham, A. (eds.) Transactions on Computational Science XXI. LNCS, vol. 8160, pp. 269–295. Springer, Heidelberg (2013). doi:10.1007/978-3-642-45318-2_12
22. Gregorovic, L., Polasek, I.: Analysis and design of object-oriented software using multidimensional UML. In: Proceedings of the 15th International Conference on Knowledge Technologies and Data-Driven Business (i-KNOW 2015). ACM, New York (2015)

23. Gregorovič, L., Polasek, I., Sobota, B.: Software model creation with multidimensional UML. In: Khalil, I., Neuhold, E., Tjoa, A.M., Da Xu, L., You, I. (eds.) CONFENIS/ICT-EurAsia - 2015. LNCS, vol. 9357, pp. 343–352. Springer, Heidelberg (2015). doi:10.1007/978-3-319-24315-3_35
24. Polasek, I., et al.: Information and knowledge retrieval within software projects and their graphical representation for collaborative programming. Acta Polytech. Hung. **10**(2), 173–192 (2013)

# Feature Extraction Methods in JEM-EUSO Experiment

Michal Vrabel[1(✉)], Jan Genci[1], Jozef Vasilko[1], Pavol Bobık[2], Blahoslav Pastircak[2], and Marian Putis[2]

[1] Faculty of Electrical Engineering and Information Technologies, Technical University of Kosice, Kosice, Slovakia
{michal.vrabel, jan.genci}@tuke.sk
[2] JEM-EUSO Collaboration Institute of Experimental Physics, Slovak Academy of Sciences, Kosice, Slovakia
bobik@saske.sk

**Abstract.** The article summarizes activities and results regarding pattern recognition for the JEM-EUSO experiment done by Slovak group of JEM-EUSO collaboration. The activities include estimation of trigger probability of false positives and reconstruction of simulated UHECR showers in UV background using Euso Simulation and Analysis Framework (ESAF). The Hough transform-based techniques are presented as methods to find UHCER showers in the JEM-EUSO detector recorded data. Additionally, the article describes structure and data flow of the framework.

## 1 Introduction

The JEM-EUSO telescope is planned to be placed on the International Space Station to look into the Earth's atmosphere to spot Extensive Air Showers (EAS) caused by ultra-high energy cosmic rays (UHECR). UHECR is represented by a high energy particle, usually a proton or a heavier nucleus, moving with velocity close to the light speed. When such a primary particle collides with molecules of atmosphere, secondary particles are created forming giant cascades of particles. These EAS can be detected by measuring the light emitted during their evolution through nitrogen fluorescence and cherenkov radiation mechanisms both observable in the UV spectrum. Because of the speed of UHECRs, very high imaging rate is required in JEM-EUSO to detect them.

This paper provides an overview of the Euso Simulation and Analysis Framework and work done in cooperation of students from Technical University of Kosice and Institute of Experimental Physics of Slovak Academy of Sciences Kosice. All this work is concerned with the detection of EAS on the focal surface of the JEM-EUSO telescope. This is done with two different approaches. From one side by analysing the images of pure UV background to compute the probability of detecting false positives, called in the following fake showers. On the other hand an analysis is conducted on simulated EAS on the JEM-EUSO detector to find methods that produce the most precise results in the estimation of the angular resolution of the incoming primary particles.


J. Janech et al. (eds.), *Proceedings of the 2015 Federated Conference on Software Development and Object Technologies*, Advances in Intelligent Systems and Computing 511,
DOI 10.1007/978-3-319-46535-7_27

## 2 JEM-EUSO Detector

The JEM-EUSO is a wide field of view telescope planned to be located in one of the ports provided by the External Exposure Facility of Kibo module on the International Space Station. From there it will observe atmosphere to spot EAS as a bright spot moving along a line at the light speed. The brightness of the shower image is related with the energy of the cosmic ray and the direction of the line shows the arrival direction of the cosmic ray. These are important parameters for the study of cosmic ray origins. Conceptual drawing of the experiment principle is in Fig. 1.

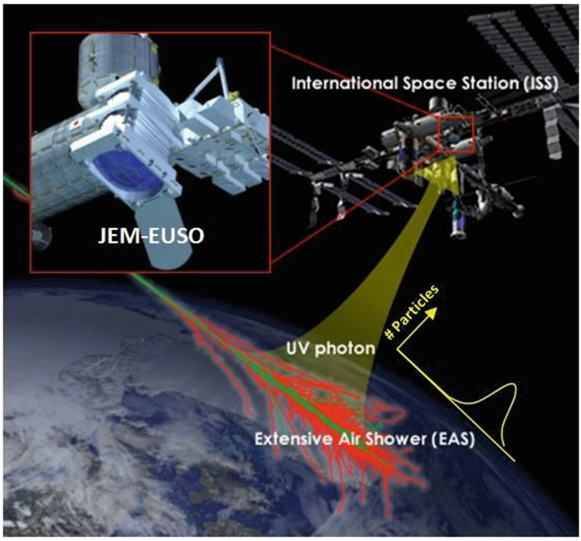



**Fig. 1.** Principle of the JEM-EUSO telescope to detect UHECR [3].

The detector's focal surface has 300,000 photo-sensitive pixels for purpose of counting photons. Individual photons are counted using photomultiplier tubes (PMTs). In the current design, PMTs have 8 × 8 pixel resolution (3 mm each). Groups of four near-by PMTs form an elementary cell (EC); 3 × 3 elementary cells are grouped in the so-called photo detector module (PDM); 9 PDMs form one cluster. The whole focal surface contains 137 PDMs (15 clusters). Recording counts of photons for long period is difficult due to storage capacities and bandwidth availability. The photons are accumulated in fixed time frames called Gate Time Units (GTU). In its standard mode a GTU lasts 2.5 μ seconds [3]. A GTU determines, therefore, the duration between resets of pixel counters. To discriminate noise, the first level of the analysis is done on-board in real-time using schemes of triggers directly on the detector. A two-level triggering system decreases data rate from 9.6 GB/s to 297 kbps [5].

The First-level trigger searches for count excess lasting a few GTUs on a few near-by pixels. This trigger is called Persistent Track Trigger. The second-level trigger

decides about the presence of an interesting feature using Linear Track Trigger algorithm. The algorithm integrates photon counting values of the pixels along the track over some predefined time. Integrated value is compared to a threshold value. Set of directions for integration covers complete parameter space. Size of an integration "box" depends on configuration and its location varies with GTU and direction [3].

Triggered events are stored to be analyzed on the ground using more precise and independent methods which require much higher computational resources. Here, the data reconstruction chain starts; a detection algorithm selects from the all downloaded data those records which are expected to come from EAS, excluding fake ones due to fluctuations of the UV light intensity.

## 3 Overview of ESAF Framework

To simulate the whole process of an EAS development, the Euso Simulation and Analysis Framework (ESAF) [2] has been developed. In its simulation application the frame- work provides the environment to develop UHECR simulation stages by dedicated interfaces. The second part of ESAF is the reconstruction application which provides interfaces to implement the reconstruction chain. The third part provides utilities for a data visualization. Nowadays, the whole simulation and reconstruction chain are implemented; in many cases they use alternative options. Figure 2 presents an overview of the ESAF workflow.

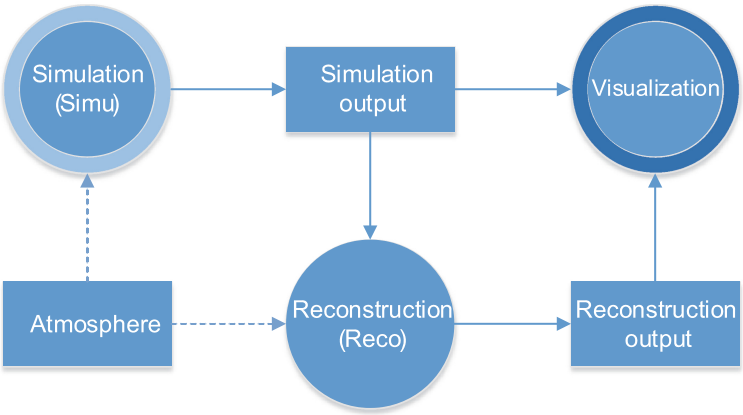



**Fig. 2.** Overview of the ESAF workflow.

The framework is developed using C++ programming language with an atmosphere propagation modeling done using *lowtran* simulation written in FORTRAN. Compilation is done using gcc compiler conforming to C++98 standard. ESAF is based on CERN ROOT framework utilizing many of its libraries and coding style. Also, output

files are in the ROOT framework output format. Both *Simu* and *Reco* applications are single-threaded. Typical use case is to run multiple independent simulation or reconstruction processes in parallel by varying values of the parameters under study.

### 3.1 Simulation in ESAF

Simulation part of ESAF covers the process from a primary particle generation to the detector response. Simulation has two significant parts - light production and the detector response. Figures 3, 4 and 5 visualize the process as data flow diagrams. Processes in the diagrams represent C++ method calls and data stores are either data structures - objects of certain class - or simulation input files containing table data. The first step in the simulation process is to generate an ultra-high energy particle. This generation is encapsulated in an interface provided by the abstract class `EventGenerator`. The generator produces shower track that contains shower steps. The track is processed by a photon generator represented by the abstract class `ShowerLightSource`. For every step of the track two bunches of photons are produced according to the step properties of the shower track. First bunch is generated to model photons produced by nitrogen fluorescence, while the second to model photons produced by cerenkov radiation. All produced bunches are aggregated in an object of class `PhotonsInAtmosphere`. Photons generated in atmosphere are transferred to the detector's pupil using radiative transfer represented by the abstract class `RadiativeTransfer`. In the current configuration, the `BunchRadiativeTransfer` is used. This implementation utilizes fortran library *lowtran*. In the first part, bunches of photons are propagated through the simulated medium, where reflections are also included. Result of these transfers is a list of single photons. Bunch propagation is done by its transfer through different mediums until the bunch gets to the final state. In the current configuration, the clear sky propagation implemented in class `AlongTrack_CSPropagation` is the most significant part of this propagation. In the second part, the object of class `ListPhotonsInAtmosphere` aggregates objects representing single photons which are propagated on the detector's pupil. The whole process of radiative transfer here described uses propagation functionalities declared by abstract class `RadiativeProcessCalculator`, currently configured to *lowtran* library-based calculator.

Atmosphere properties are included in steps from the primary particle generation to the photons on pupil. Singleton object of class Atmosphere wraps this functionality which includes retrieving values such as air density or humidity and calculations requiring these values such as the determination of the slant depth or positions of the shower impact. In present configuration `LowtranAtmosphere` implementation of atmosphere is used. Atmospheric model is defined in tables which are loaded in run-time.

The second step of the simulation process is the detector's response to photons arrived on pupil. In the beginning of processing, photons need to be transported through the detector's optics onto PMTs. This process is handled by `Detector TransportManager` method of class `DetectorTransportManager`. The position of each pho- ton is evaluated and processed by the relevant part of the

telescope represented by an object of a proper class. The optics is described by `OpticalSystem` interface. A photon represented by an object of class `Photon` is exchanged between parts of the simulation until it is considered non relevant by a particular part of the simulation - for example absorbed by the focal plane or the telescope's wall. Depending on position, a photon may get captured on a PMT described by abstract class Photomultiplier. Adding photons to photomultiplier results in new objects of class `PmtSignal`, which are aggregated in the object of this photomultiplier by channel (pixel) hit. The whole described photon transport process includes random generation to reproduce real fluctuations.

After all photons are transported to PMTs as signals, the electronics response is simulated. An electronics is represented by class `EusoElectronics` which in its `EusoElectronics::Simulate()` method calls simulations of macrocell and PMT operations. PMT simulation itself only consists of adding pointers to an object of PmtSignal to list associated with an object of class `FrontEndChip`. This class represents front end chip. Simulation continues with macrocell simulation described by class MacroCell. Method `Macrocell::Simulate()` calculates the number of GTUs for the simulated event, and for every associated front end chip, its signals are saved as counts on channels in specific GTUs. Also background noise is simulated and added to this data. These data are aggregated in objects of class MacroCellData which are analysed for trigger response. Trigger simulation consists of processing the data using the expected trigger response described by class `TriggerEngine`. Finally, macrocell data are added to telemetry which is stored in the output file.

In whole described process, modules store also many of their intermediate data to the simulation output files.

### 3.2 Reconstruction in ESAF

Shower reconstruction process is much more straightforward than the simulation. Its main complexity lays in the reconstruction algorithms.

The process starts by reading input data. The data are usually expected to be in simulation output file, but input module is described by interface of abstract class `InputModule` which can be implemented to support different methods of input. For example, we have implemented an input module which parses external UV background simulation data. Processing of the event is done in modular sequence. An EAS event processing module is described by the interface of the abstract class `RecoModule`. Notable information about this abstract class is that is requires implementation of `RecoModule::Process(RecoEvent*)` method which takes pointer to the object of `RecoEvent` that contains all data of the processed event. Another interesting method is `RecoModule::SaveRootData(RecoRootEvent*)` which takes pointer to the object which is after processing written into an output file. Each module can make its results accessible through an map-like data structure associated with an processed event (global data). Also, the module interface itself provides similar capability (local data).

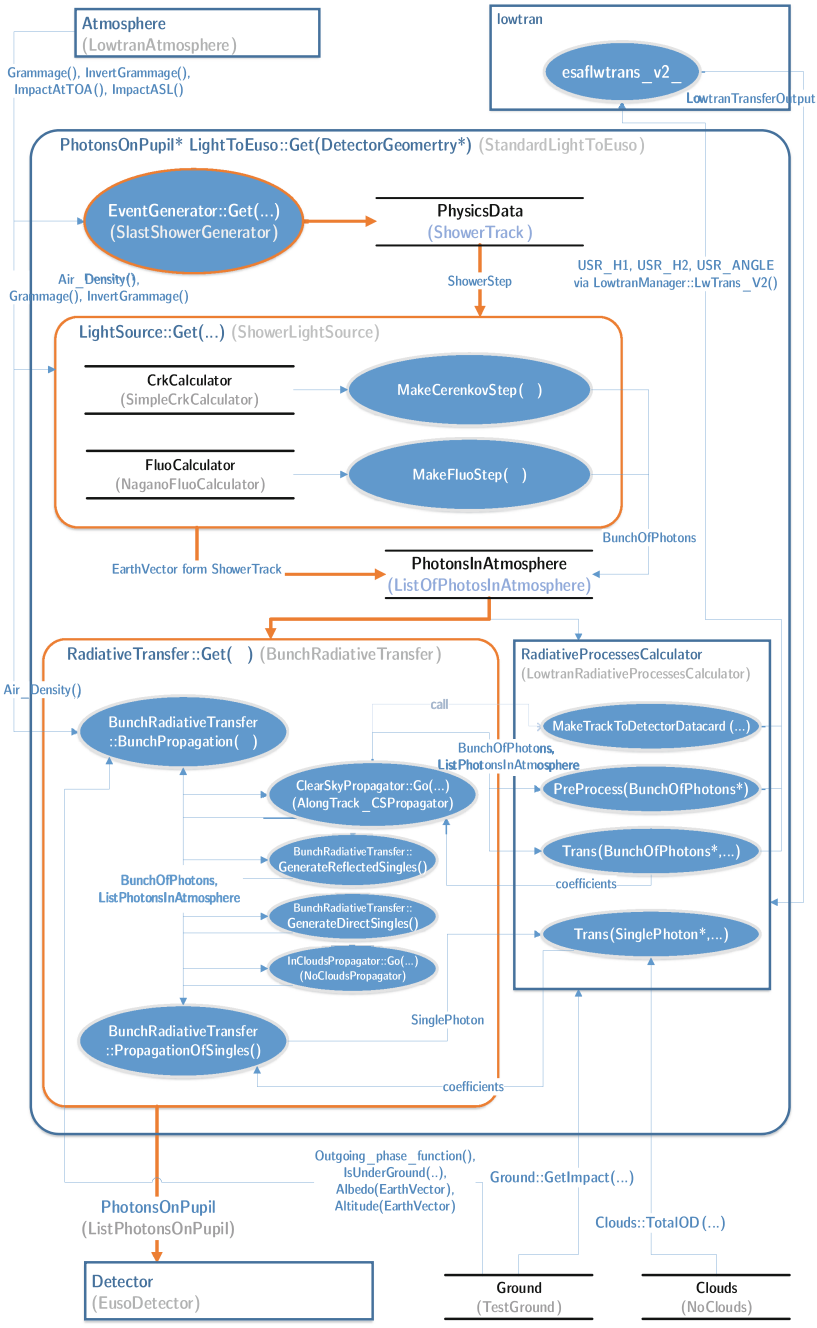



**Fig. 3.** Photons on the pupil production in ESAF

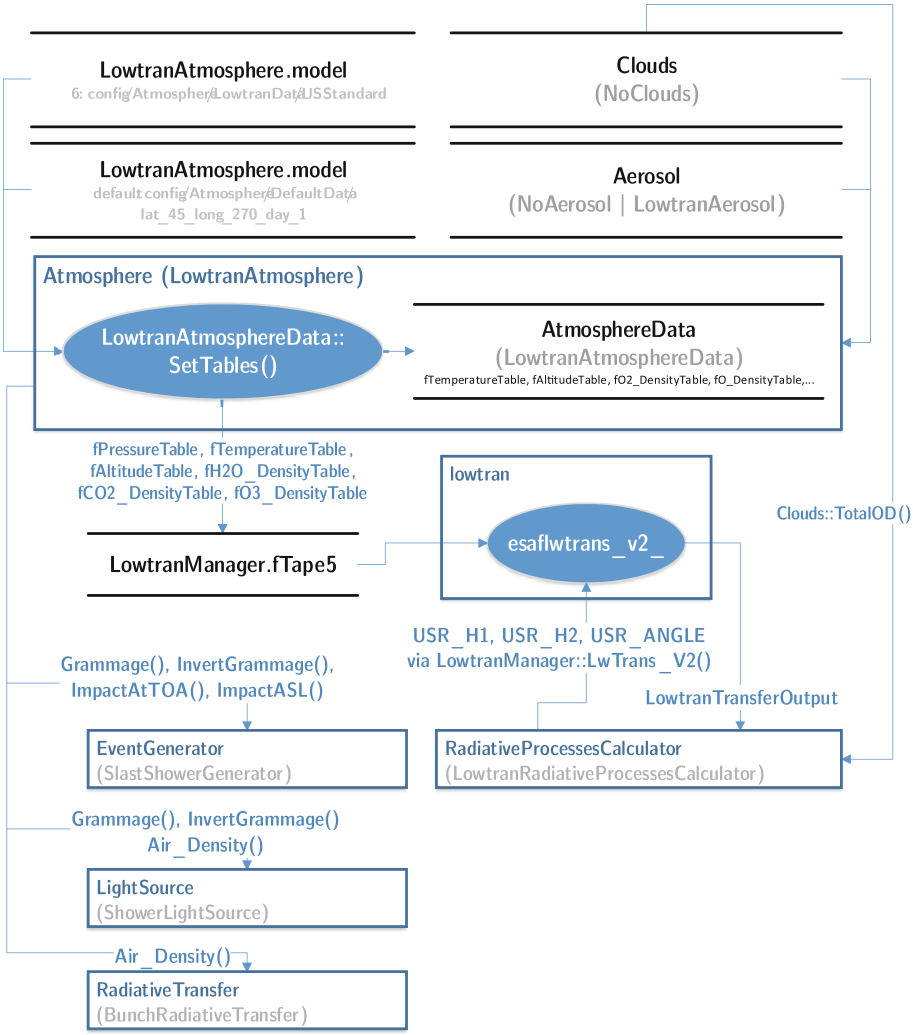



**Fig. 4.** Atmosphere model in ESAF.

The first step of the shower processing is expected to be pattern recognition which selects only pixel records that are most probably associated to shower's photons. Pixel record is represented by an object of class `RecoPixelData` containing information such as pixel identifier, pixel counts, and GTU from the start of the EAS event. The most widely used method is PWISE [7]. Currently, pattern recognition modules are expected to store data in the global data structure of the processed event.

The task of the angular reconstruction is to determine the direction of the shower. These angles are zenith angle $\Theta$ and azimuth angle $\Phi$. A unit vecto $\hat{\Omega}(\Theta, \Phi)$ describes shower direction. Its origin is at shower core (impact position on ground) pointing along the shower axis into sky [10]. To evaluate the performance of the angular

reconstruction, the separation angle $\gamma$ is defined, which is the angle between simulated and reconstructed shower axis.

Currently used angular reconstruction module `TrackDirecition2` starts with the estimation of Track Detector Plane (TDP). This plane contains shower track and detector itself. TDP is inferred from x-t and y-t fit of the signal track. For further information about TDP mathematics reader is directed to [10]. Task of a track reconstruction algorithm is to determine angle $\boldsymbol{\beta}$ between $\hat{\boldsymbol{\Omega}}$ and $\hat{\mathbf{W}}$ Vector $\hat{\mathbf{W}}$ is the perpendicular vector to a normal vector of TDP $\hat{\mathbf{V}}$. To reconstruct shower direction there are multiple techniques available in ESAF - *Analytical approximate algorithm AA1 and AA2, and Numerical exact algorithm NE1 and NE2*.

Final part of reconstruction is energy and shower maximum reconstruction. The module responsible for this task is `PmtToShowerReco` developed by F. Fenu [9]. The first step is reconstruction of *cout curve* which means reconstructing the signal intensity as a function of time. An important factor to be taken into account in this step is the proximity of the pixel to the edge of the focal surface, where only a portion of the signal is recorded. At the end the UV background counts are subtracted as an average amount for each pixel record.

The second step is the identification of the maximum and of the cherenkov peak in the reconstructed light curve. Both of the steps are not trivial due the nature of the detected signals. After cleaning of the signal by removing fake peaks, the valid peaks are retrieved and ranked. Such ranking is used for the analysis of the shower maximum and for timing purposes.

In the next step, the photon curve at the entrance of the pupil is obtained by using the detector's parameterization and correction for the optics response. The optics response map is calculated by simulating point like sources at predefined FOV angles and wavelengths.

F. Fenu proposed two methods to determine the altitude of shower maximum. The first uses the cherenkov reflection mark and the second assumes a parameterization for the Xmax and relies on direction provided by the algorithms for direction reconstruction.

Finally, after the determination of a tri-dimensional position of the maximum, the shower position at each time can be calculated and the luminosity of the shower is calculated. From position and age of the shower, energy distribution of secondary electrons is used to estimate the fluorescence and cherenkov yields. This module uses lowtran library to model atmospheric transmission.

## 4 Probability of False Shower Detection

The JEM-EUSO detector is expected to take 400 thousands frames (shots) in one seond, i.e. 34 billions per day. Taking into account the energy spectrum of UHECR, exposure of the detector, including the duty cycle [8], the rate at which an EAS with energy above GZK limit will be observed, is roughly once every 1–2 days. This EAS will appear in a few (mostly 1 or 2) PDMs for a few tenths of GTUs. Therefore, one PDM will observe a real event approximately every 100 billions PDM shots.

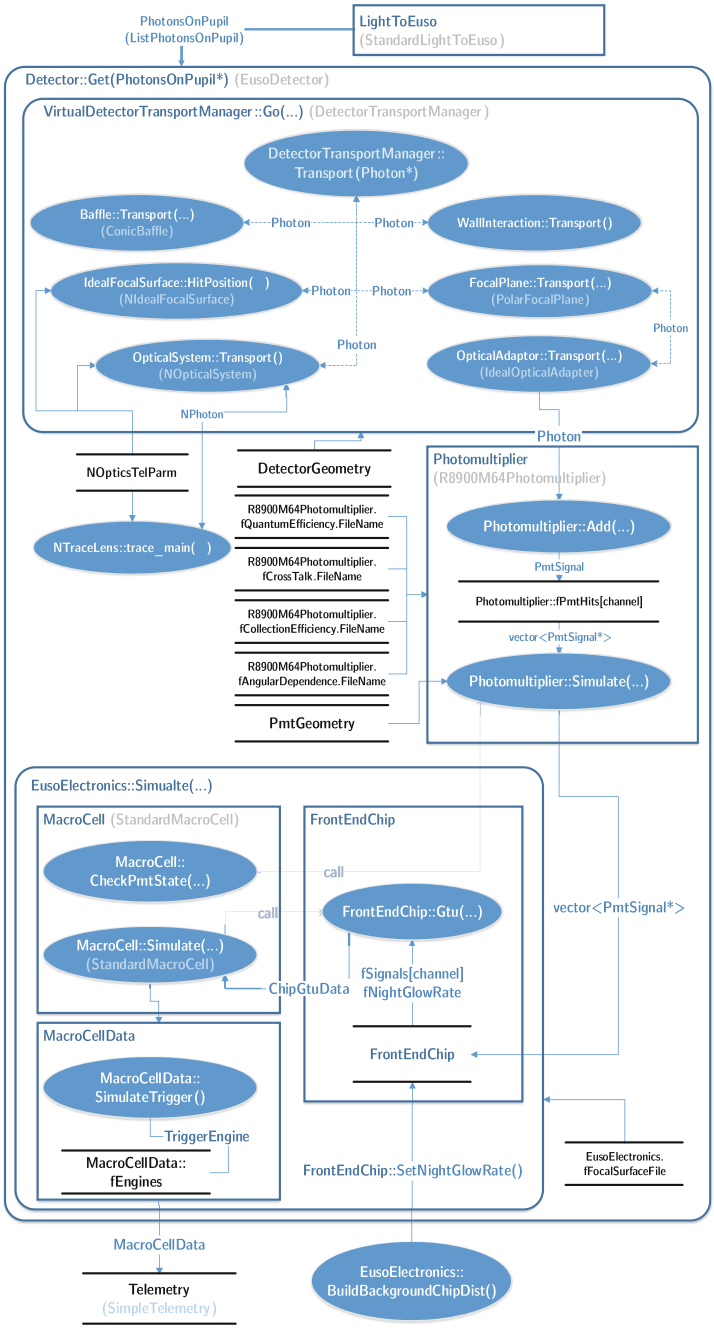



**Fig. 5.** Detector simulation in ESAF.

In other words, the order of magnitude of the ratio is approximately 1 to 1011 between a shot with real EAS and shots without EAS, which means only with UV background noise. Analysis of fake patterns was done for one PDM. UV background was simulated as Poissonian noise and then triggered by 1st and 2nd level of trigger scheme [3]. Only frames which passed trigger were analyzed. Hough transform method (simpler than one presented for pattern recognition) was used to find patterns and length of patterns. The number of found patterns as a function of a pattern length is presented in the Fig. 6 [6]. The Pattern length is in pixels. An example of the analysis result for 109 GTUs run, equivalent to 2500 s measurements for one PDM of the detector, is presented by the blue line with diamonds. The result from full analysis of 3.3 h measurements of all detector's PDMs is presented by magenta line with triangles. These results from simulation can be fitted by a statistically motivated function in Eq. 1 in [6]. If we scale the curve to consider one day measurement of all detector's PDMs (green line on Fig. 6.), we can find only a few patterns with length of 11 and maybe one with length of 12 pixels. Further approximation scaling to the full planned 3 years of JEMEUSO operation, gives one pattern with length equivalent to 15 pixels. This lengths (approx. 15 pixels) are close to the length of the shortest real shower projections observed by JEMEUSO detector. Moreover, the number of fake patterns whose length is comparable to real EAS will be reduced by analyzing the time evolution of their signal. Usual fake pattern has noisy characteristics, while real showers have time evolution described by a typical shower profile. The conclusion of this analysis is that the probability to find a fake shower is very low, at the level lower than 0,1 percent, when compared to the rate of real EASs. Thus only one among one thousand or more detected showers could be created by noise. This result confirms the reliability of the trigger processes and following patter recognition to select very rare signal in overwhelmingly big data recorded by the detector. Results were published in the proceedings of ICRC 2013 conference [6]. Those results were confirmed by an alternative pattern recognition methods based on modified clustering method.

## 5 Shower Pattern Recognition Methods

The standard method for pattern recognition/feature extraction is Peak Window Searching technique (PWISE). Our proposed alternative method is based on Hough transform.

### 5.1 PWISE Module

The PWISE module selection is based on the signal-to-noise ratio (SNR) calculation. In the first step, PWISE selects pixels above *peak threshold*. Then, among such pixels, a search is performed to select those with the highest SNR for a time window (range of GTU). If SNR within this time window is higher than SNR *threshold*, pixel records from this time window are selected. For more detailed description the reader is pointed to the original paper [7].

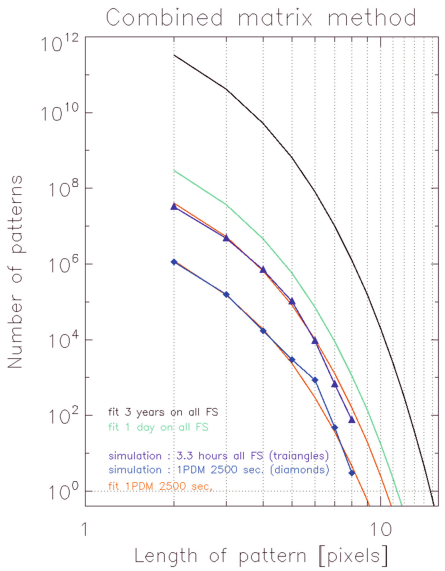



**Fig. 6.** Probability of false shower detection [6]

### 5.2 Hough Transform

A high energy particle is moving in straight line which is the reason to expect the highest intensity pixels of the shower projected on the detector's focal surface to be positioned in a straight line too. The Hough transform is a technique easily used for detection of lines (and other shapes). The purpose of the transformation is to construct accumulator matrix representing parameter space where a voting procedure is carried out. Line represented by equation $y = ax + b$ in image space can be in parameter space represented by pair $(a, b)$. However, this representation is unbounded and cannot represent vertical lines. Normal parameterization describes lines by pair $(\phi, \rho)$. $\phi$ is an angle of a line normal and $\rho$ is an algebraic distance from the origin [1]. Using these parameters line equation can be written as indicated by the following equation:

$$y = -\left(\frac{cos\,\phi}{sin\,\phi}\right)x + \frac{\rho}{sin\,\phi} \tag{1}$$

From this equation $\rho$ is expressed as such:

$$\rho = x\,\cos\phi + y\,\sin\phi \tag{2}$$

Figure 7 illustrates parameters $\phi$ and $\rho$.

Normal parameters for a straight line are unique in an interval $[0, \pi]$. A line in $x$–$y$ space (image space) corresponds to a point in $\phi - \rho$ space (parameter or Hough space). On the other hand, a point in the image space corresponds to a sinusoidal curve in $\phi - \rho$ parameter space. From these statements it can be deduced that points laying on the same straight line in the image space correspond to curves through single point

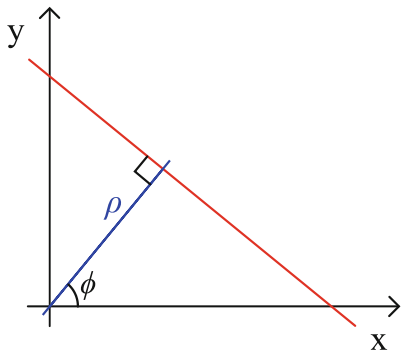



**Fig. 7.** Hough transform parameters

in the parameter space. Similarly, curve in the parameter space correspond to lines from single point in picture space.

Our basic implementation of the feature extraction technique consists of two steps - two separate loops. Input parameters are the size of the step between consecutive $\phi$ angles, optionally the range of the evaluated $\phi$ angles, the step between consecutive normal distances $\rho$, and optionally their range. By default the range of $\phi$ is $[-\pi/2, \pi/2]$ and the range of $\rho$ is $[-\rho_{\max}, \rho_{\max}]$, $\rho_{\max}$ is the hypotenuse of the right-angled triangle which sides are absolute values of the greatest x and y pixel locations. These default values are also shown in the presented Algorithm 1. These four parameters define size of the parameter space (Hough space) which is represented by an accumulator matrix. Additionally, there is precision adjustment parameter which determines range of $\rho$ cells incremented for calculated $\rho$ value.

The first step is the construction of the accumulator matrix. Rows correspond to $\phi$ and columns to $\rho$, although this is just preference of the implementation. To fill the matrix $\rho$ is calculated for every pixel of the image space and every $\phi$ of the parameter space using Eq. 2. Then indexes of $\rho \pm adjustment$ adjustment in the array of all $\rho$ values are calculated. This means that for single point and all $\phi$ of parameter space single sinusoidal curve is created. In calculated $\rho$ indexes range, the accumulator matrix field value is incremented with a weight given by the value of the pixel counts. Using counts as an increment value adds weight to pixels with higher counts value discriminating noise pixel records. The new field value is compared with maximum value that has already been set. If it is greater, then a new maximum value is set and size of the list of the maximum value positions is set to 0. This list is added records whenever an incremented field value is equal or greater the maximum accumulator matrix value. Storing these locations in the list removes need of later searching in the whole parameter space.

The second step is the selection of pixels belonging to the field with the highest value in the accumulator matrix. This field correspond to the most significant line. Current implementation of the algorithm expects only one significant line - the EAS detected on the analyzed area. Because the most significant pixels themselves are not stored when testing for the maximum value, $\rho$ calculation is repeated for every pixel and $\phi$ from the list of maximum value positions.

1. *Three-dimensional Hough transform*: Expecting a shower development to be approximately constant speed, one of the pixel position dimensions and its time fit a linear function. There were two alternatives of three-dimensional Hough transform considered.
   (a) *Hough Plane 3D*: The approach is similar to simple Hough transform, but definition of the line is added additional dimension, resulting in Hough space (parameter space) having three dimensions. This method is much more memory space demanding because of the accumulator matrix size which is $num_\rho \times num_\phi \times num_\theta$.$num_\theta$ is the size of the additional dimension - additional angle defining line in three dimensions. In our implementations the manageability of memory requirements is achieved by limiting ranges of parameters of Hough space. The ranges are determined by using a simple Hough transform in the way that maximum and minimum value of parameter comes from lines with maximum accumulator field value.
   (b) *Hough Line 3D*: The approach consists of two consequent calls of simple Hough transform. In the first step, Hough transform is used to find a line in a plane XT (T is the time dimension measured in GTU) and in the second step Hough transform is used to find line in a plane YT using pixels selected in the first, also range of normal distances of the second Hough space is determined by range of the most significant lines from the first. Because the *Hough Plane* 3D algorithm didn't show any significant improvements over this less memory and processing demanding method, this is the method predominantly used in shower reconstructions. Any future algorithm enhancements will most probably be based on this approach.
2. *Hough 1*: The first version of algorithm selects pixels in two phases. To use this method efficiently, two thresholds are needed to be configured for analyzed UV background. Hough transform method is implemented as described in the previous section and it is used in summary three times to select a line of pixels. Main idea of the algorithm is to select rough line from pixels with the highest counts, in the first phase. And in the second, the described three-dimensional Hough transform is applied to select a line form all pixels above threshold.

The input parameters of the algorithm:

| | |
|---|---|
| Pattern threshold | Only pixel records above this threshold will be considered for se lection of unique pixels used in the first Hough transform. |
| Data Threshold | A pixel record above this threshold will be added to pixel records above Pattern threshold to be selected and some of them eventually handled by Hough transform. |
| X,Y Adjustment | Adjustment value used in the first phase's Hough trans form. |
| X,Y,Gtu Adjustment | Adjustment value used in the second phase's Hough trans forms. |
| $\rho_{step}$ | Step between consecutive normal distances from the origin in Hough space. |
| $\phi_{step}$ | Step between consecutive normal angles in Hough space. |

**Algorithm 1** Selection of pixels using Hough transfom.

**Require:** $PX[n]$: Array of pixel records
**Require:** $\rho_{step}$: Real
**Require:** $\phi_{step}$: Real
**Require:** *adjustment*: Real
$\rho_{max} \leftarrow findMaxRho(PX)$
$num_{\rho} \leftarrow \lceil 2\rho_{max}/\rho_{step} \rceil$
$num_{\phi} \leftarrow \lceil \pi/\phi_{step} \rceil$
$\rho_{first} \leftarrow -\rho_{max}$
$corr_{lower} \leftarrow -adjustment - \rho_{first}$
$corr_{upper} \leftarrow adjustment - \rho_{first}$
$acc_{max} \leftarrow 0$
$MAX\ POS \leftarrow$ [] {Empty list of pairs, maximum values}
$ACC \leftarrow [num_{\rho} \times num_{\phi}]$ {Matrix of $num_{\rho}$ rows and $num_{\phi}$ columns}
$OUT \leftarrow$ [] {Empty list of pixel records}
**for all** $px$ in $PX$ **do**
  **for** $\phi_{index} = 0$ to $num_{\phi}$ **do**
    $\phi_{value} \leftarrow -\pi/2 + \phi_{index} * \phi_{step}$
    $\rho_{value} \leftarrow px.x \cdot cos(\phi_{value}) + px.y \cdot sin(\phi_{value})$
    $\rho_{lowerindex} \leftarrow round(\rho_{value} + corr_{lower})$
    $\rho_{upperindex} \leftarrow round(\rho_{value} + corr_{upper})$
    **if** $\rho_{lowerindex} < 0$ **then**
      $\rho_{lowerindex} = 0$
    **end if**
    **if** $\rho_{upperindex} \geq size(\rho_{array})$ **then**
      $\rho_{upperindex} \leftarrow size(\rho_{array})$
    **end if**
    **for** $\rho_{index} = \rho_{lowerindex}$ to $\rho_{upperindex}$ **do**
      $ACC[\phi_{index}][\rho_{index}] \leftarrow ACC[\phi_{index}][\rho_{index}] + px.counts$
      **if** $ACC[\phi_{index}][\rho_{index}] \geq acc_{max}$ **then**
        **if** $ACC[\phi_{index}][\rho_{index}] > acc_{max}$ **then**
          $acc_{max} \leftarrow acc_{max}$
          *clear* (*MAXPOS*)
        **end if**
        $\rho_{curr} \leftarrow \rho_{first} + (\rho_{step} \cdot \rho_{index})$
        *append* (*MAX POS*, $(\phi_{value}, \rho_{curr})$)
      **end if**
    **end for**
  **end for**
**end for**
**for all** $px$ in $PX$ **do**
  **for all** *maxpos* in *MAXPOS* **do**
    $\rho_{value} \leftarrow px.x \cdot cos(maxpos.\ \phi) + px.y \cdot sin(maxpos.\ \phi)$
    $\rho_{lowerindex} \leftarrow round(\rho_{value} + corr_{lower})$
    $\rho_{upperindex} \leftarrow round(\rho_{value} + corr_{upper})$
    **if** $\rho_{lowerindex} \leq maxpos.\ \rho \leq \rho_{upperindex}$ **then**
      *append* (*OUT*; *px*)
      break
    **end if**
  **end for**
**end for**
**return** OUT

Phases of the algorithm:

(a) *Phase 1. - Selection of a shower pattern*: From all input pixel records for each pixel ID, the pixel record with GTU with the highest number of counts is considered. The number of counts has to be above pre-defined threshold (*Pattern threshold*). If it is not, no GTU record is selected for such a pixel ID. Hough transform is applied on these pixel records on the plane XY (space defined by axes x and y) of detector focal surface and is used to select pixels (with unique ID) on a line with the highest sum of counts.
(b) *Phase 2. - Selection of pixel records from all GTU*: From all input pixel records above threshold (*Data threshold*), only those are selected which ID is present in pixel records selected in Phase 1. Hough transform is applied to select pixels on a line with the highest sum of counts on the plane XT (defined by axes x and GTU). Similarly, Hough transform on plane YT (defined by axes y and GTU) is applied on the result of selection on the plane XT. Pixels on a line with the highest sum of counts are selected as a result of this method.

3. Hough 2: This algorithm is less straightforward than the first one and it is still under development. The main idea of the algorithm is principally similar to the first one - selection of the shower's pattern and then selection from all pixel records. A difference is that the threshold for the pattern pixels is determined dynamically depending on number of pixels with the highest counts. Also, those pixels are conditioned to be in "close" proximity and there is a method of adding weights to more probable pixels by multiplying their number of counts by constant. This makes them more preffered in the selection from the Hough space. The second phase is processing of all pixel records above threshold, but pixels are grouped by counts. Additional phase is the filtering of selected pixels.

Input parameters of the algorithm include adjustments for Hough transforms, percentage of all pixels to be selected out of pixels with the highest counts, counts threshold for the second phase (*threshold lower limit*), maximum counts group size, number of counts threshold groups, rectangle selection dimensions.

Following description is focused on the description of the basic algorithm and does not include all implemented capabilities.

(a) *Phase 1. - Selection of pixels belonging to shower maximum*: From all input pixel records for each pixel ID, a pixel record with GTU having the highest number of counts is selected. Pixel records with counts above computed threshold are selected. This threshold is connected to number of pixels needed for pattern recognition. Neighbouring pixels in the plane XY with the highest sum of counts from this set are added a weight to be preferred within this phase. Hough transform in the plane XY is applied on these pixel records with counts above the threshold to find a line with the highest sum of counts (including weights). Then 3D Hough transform in space XYT (similar to Phase 2 of method Hough 1) is applied on pixel records in an area around the line (weights are not included). Selected line of pixels is filtered to contain only neighbouring pixels that have limited maximum relative distance. Rectangle around this set of pixels is selected from all input pixel records.

(b) *Phase 2. - Selection of pixel records from all GTU*: 3D Hough transforms in space XYT are applied on pixel records around rectangle selection from the selection of shower's maximum. The transforms are applied separately for each count value over a pre-defined threshold (*threshold lower limit*). The ranges of angles defining Hough space are ranges of a line with the highest sum of counts selected by Hough transform from previous iteration with higher counts. Starting range is from Hough transform of shower maximum. Sets of pixel records selected by the use of these transforms are joined to the final set.

(c) *Phase 3. - Filtration*: Additionally, the filtration rules based on shower characteristics were defined to reduce the number of pixel records with redundant GTUs. The rules are applied to each set of pixel records with the same ID. Pixel counts must be increasing with increasing GTU up to the GTU with the highest counts followed by a decreasing phase. GTU values of pixel records must be continuous. Uninterrupted sequence containing pixel records with the highest counts and the highest number of pixel records is selected.

### 5.3 EAS Angular Reconstruction Results

The shower reconstruction analysis was done by analysis of same set of simulated events using different pattern recognition methods. For each selected incident zenith angle $\Theta_{truth}$ more than 2000 events were simulated. Energy of simulated primary UHECR particle was set to $10^{20}$ *eV* for all simulated events. Positions of the showers were generated uniformly in the detector field of view. Simulated incident zenith angles $\Theta_{truth}$ are in range from 30° to 75°. The step between angles is 5°.

To analyze quality of angular reconstruction for each simulated shower, the separation $\gamma$ angle is evaluated, which is defined as an angle difference between the reconstructed shower axis (determined by $\Theta_{reco}$ and $\Phi_{reco}$) and the injected shower axis (determined by $\Theta_{truth}$ and $\Phi_{truth}$ of a primary particle). Sets of reconstructed showers are compared by $\gamma_{68}$. This is the value by which the cumulative distribution of reaches 0,68. Parts of this analysis were also published in [11].

The Fig. 8 presents the comparison of $\gamma_{68}$ for PWISE and both described Hough-based methods in case of nominal background. Reference PWISE results are taken from [10] and green line labeled "PWISE" presents results achieved by reconstructing the same set of events as Hough-based techniques. The PWISE configuration in this study: *absolute threshold = 9, second threshold = 1, minimum SNR = 5, minimum height of any pixel's peak = 8*, second iteration (PWISER) was not used.

Please notice that, for the purpose of this paper, the relative value comparisons are more important than their absolute value. The reconstruction result depends on simulation and reconstruction parameterization.

On the chart, the *Hough 1* method is presented by the red line. Hough transform adjustment value described in Sect. 5-B was set to 2 mm for the plane XY, and 4 for the planes XT and YT. The minimal Hough space parameter values were both set to 0:1, but preliminary analyses show that these values can be increased and achieve comparable results. Decrease of resolution of a Hough space decreases computation

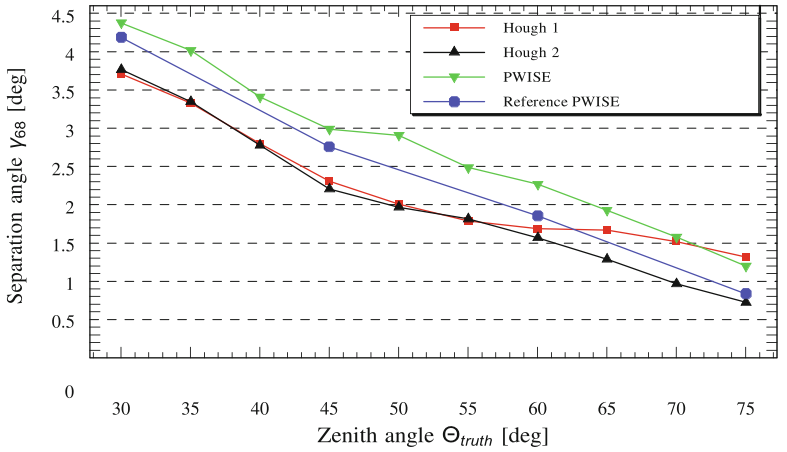



**Fig. 8.** Separation angle γ68 for the energy $10^{20}$ eV

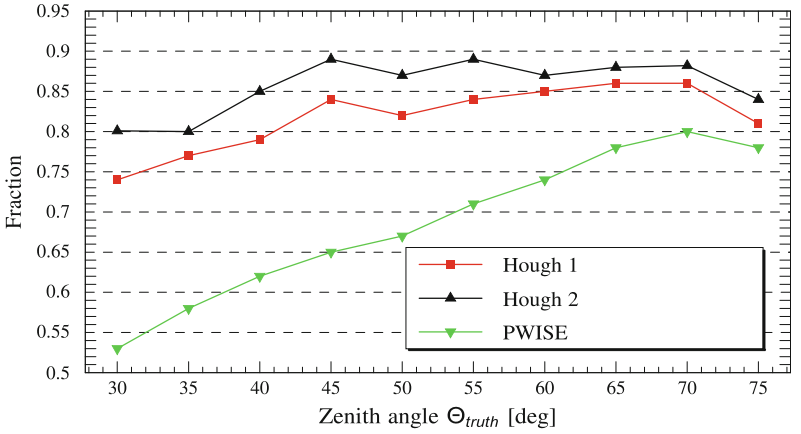



**Fig. 9.** Fraction of reconstructed events of energy $10^{20}$ eV.

and space resource demands. In case of nominal background, threshold parameters for the method were set to 5 for *pattern threshold* and similarly 5 for *data threshold*. These two thresholds were adjusted for an analysed UV background level.

The *Hough 2* method is on the chart presented by the black line. The *lower threshold limit* parameter was set to 5. Value of this parameter was adjusted to fit analyzed background intensity. Values of parameters which are adjusting resolution of the Hough space were same as they were in analysis of Hough 1 method. Both filters were always applied. In almost all cases, adjustment size of Hough transform in phase 2. was determined from analyzed data by the method. Distance limit, between two pixels which are added weight, is 5 mm and distance limit between two pixels of shower maximum is 12 mm.

The analysis of the dataset shows that the angular reconstruction, with data from the proposed Hough transform-based methods, has more precise estimation of particle incident Θ angle than PWISE method. In comparison to the *reference* PWISE, the *Hough 1* method provides more precise estimation for angles under 65°. The *Hough 2* method results are comparable with the *Hough 1*, but for higher incident zenith angles angular reconstruction is more precise with data from the second method.

Not all simulated events captured by the detector's front end electronics were reconstructed - accepted by angular reconstruction module. In this analysis, this module accepted only pattern recognition results with at least 10 pixel records. The fraction of reconstructed events is presented in Fig. 9. The proposed Hough-based pattern recognition methods tends to reconstruct more events in comparison to PWISE because the Hough-based methods usually select more pixel records than PWISE in analyzed configuration.

Results of angular reconstruction of data from Hough transform-based pattern recognition module were also analyzed for different background levels. The evaluation was made for primary particle zenith angles 30°, 45°, 60°, 75°. Background levels started at the nominal background 500$ph/(m^2 \cdot ns.sr)$ and increased up to 5000 $ph/(m^2 \cdot ns.sr)$ with steps of 500$ph/(m^2 \cdot ns.sr)$. Analyzed datasets for angle 30° contain more than 2800 events, for higher angles they contain at least 500 events. Because with higher background noise level the detection gets more difficult, less events are reconstructed and $\gamma_{68}$ estimation gets less precise. In our analysis, we have reconstructed same set of simulated events multiple times. Each time the threshold configuration of the Hough method was different. Following figures present the threshold combinations where $\gamma_{68}$ value was the lowest for particular background level and simulated zenith angle $\Theta_{truth} = 30°$. Table 1 present's threshold configuration for the *Hough 1* method[1] and Table 2 presents threshold lower limit parameter for the *Hough 2* method. Let us remind again that angular reconstruction results depend on parameterization of simulation and reconstruction and for purpose of this paper their relative comparison is more important. For every figure presenting $\gamma_{68}$, the fraction of reconstructed events is associated. Lowering the quality of arrival direction determination would allow to reconstruct a higher fraction of events.

The Fig. 10 presents $\gamma_{68}$ for simulated $\Theta_{truth} = 30°$. The methods deliver similar $\gamma_{68}$ values except the sharp upward spike at the highest analyzed background level of the *Hough 1*. The spike might be caused by low efficiency (small number of reconstructed events) of this method in case of higher backgrounds as presented in Fig. 11. The *Hough 2* provides comparable $\gamma_{68}$ but higher fraction of reconstructed events.

The Figs. 12, 13, 14, 15, 16 and 17 show results for simulated incident zenith angles 45°, 60°, 75°. In the figures for $\Theta_{truth} = 45°$ $\gamma_{68}$ is similar, but the *Hough 2* produces higher fraction of reconstructed events. The figures for $\Theta_{truth} = 60°$ show the *Hough 2* having better angular estimation and sightly higher fraction of events. In the end, results for $\Theta_{truth} = 75°$ show that if reconstructed fraction of events is almost similar, the *Hough 2* provides more precise angular reconstruction.

[1] A slash between numbers in Table 1 separates numbers after and before additional analysis. Higher background analysis for $\theta_{truth} = 30°$ uses the first numbers. Other angles use second numbers.

**Table 1.** Thresholds for Hough 1 method

| Background | Pattern threshold | Data threshold |
|---|---|---|
| 500 | 5 | 5 |
| 1000 | 8 | 8 |
| 1500 | 10 | 10 |
| 2000 | 12 | 12/11 |
| 2500 | 14 | 14 |
| 3000 | 16 | 16 |
| 3500 | 18 | 18 |
| 4000 | 20/21 | 20/19 |
| 4500 | 21 | 20 |
| 5000 | 23 | 22 |

**Table 2.** Threshold lower limit for Hough 2 method

| Background | Threshold lower limit |
|---|---|
| 500 | 5 |
| 1000 | 7 |
| 1500 | 8 |
| 2000 | 10 |
| 2500 | 13 |
| 3000 | 16 |
| 3500 | 16 |
| 4000 | 18 |
| 4500 | 20 |
| 5000 | 20 |

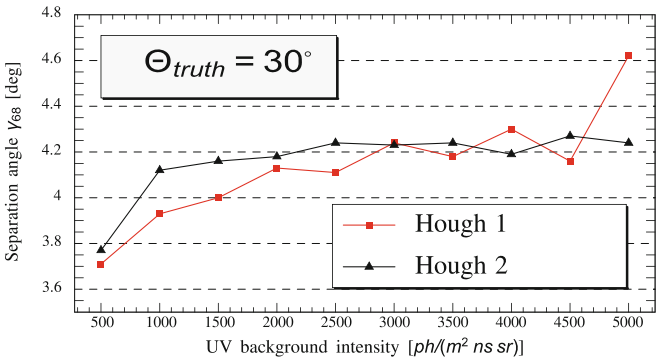



**Fig. 10.** Separation angle $\gamma_{68}$ for $\Theta_{truth} = 30°$

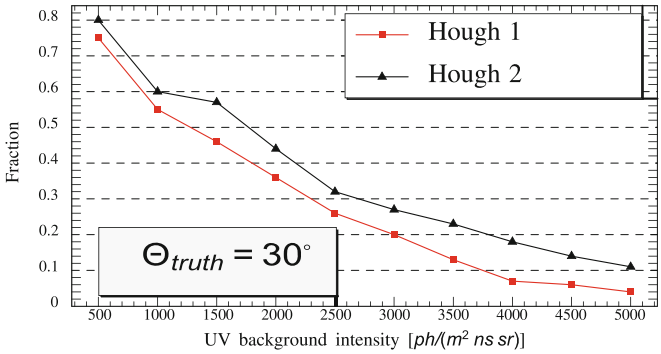



**Fig. 11.** Fraction of reconstructed events for $\Theta_{truth} = 30°$

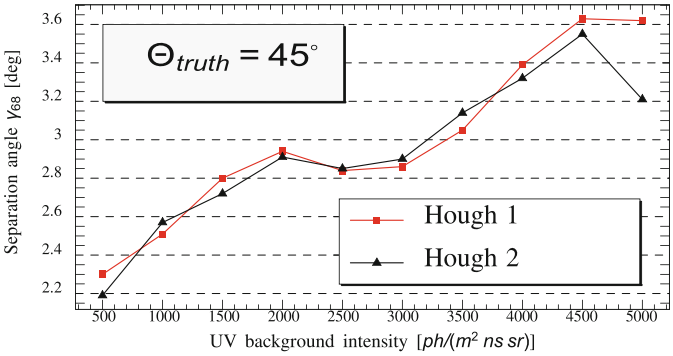



**Fig. 12.** Separation angle $\gamma_{68}$ for $\Theta_{truth} = 45°$

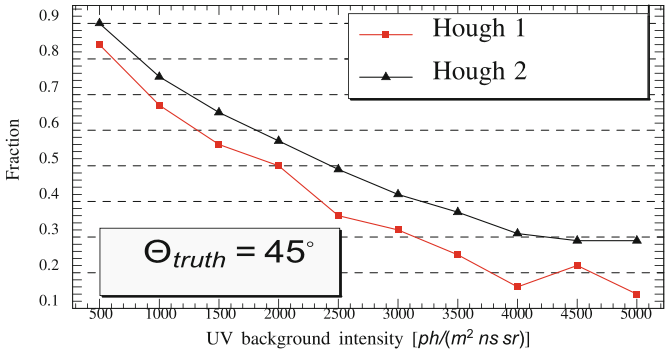



**Fig. 13.** Fraction of reconstructed events for $\Theta_{truth} = 45°$

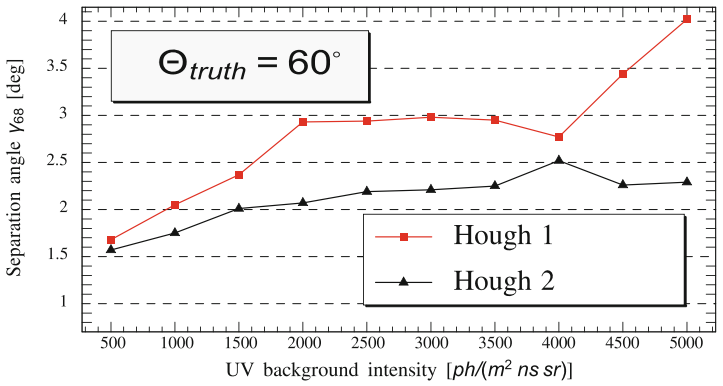



**Fig. 14.** Separation angle $\gamma_{68}$ for $\Theta_{truth} = 60°$

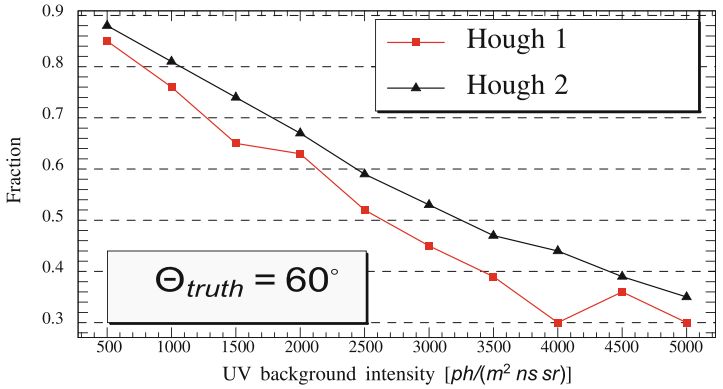



**Fig. 15.** Fraction of reconstructed events for $\Theta_{truth} = 60°$

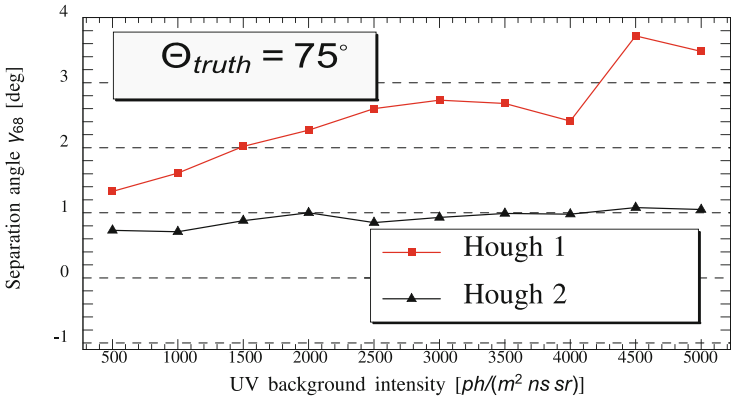



**Fig. 16.** Separation angle $\gamma_{68}$ for $\Theta_{truth} = 75°$

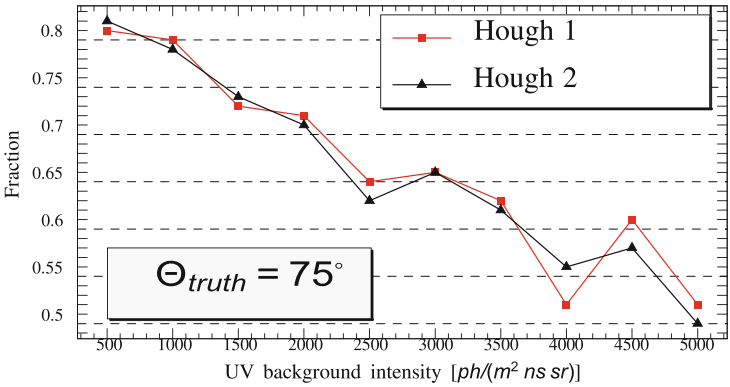



**Fig. 17.** Fraction of reconstructed events for $\Theta_{truth} = 75°$

## 6 Conclusion

This paper attempted to describe the control and data flow of The Euso Simualtion and Analysis Framework in a form that would not require deep understanding of physics behind simulations. The purpose of the description is to help the reader identify part of application where his module or extension should be implemented. Our interest was pattern recognition reconstruction module.

Presented pattern recognition methods satisfy mission requirements [4] for angular reconstruction which requires $\gamma_{68}$ to be under 2,5° for primary particle at incident zenith angle 60° with energy $10^{20}$ *eV*. Although the *Hough 1* method is simple, it provides satisfactory results either for nominal or higher background. This is thanks to the fact that combination of thresholds can be "tuned" to for particular background level. The *Hough 2* method is mainly better from the view of reconstructed events fraction. The main advantage of the second method is better initial shower detection but there are still enhancements to be made.

The lessons learned from these two methods will be applied in implementation of the third method. New version will try to incorporate expected "triangular" shape of a shower, introduce new thresholds to allow better control over excluded pixel records, and there will be added enhanced proximity dependent selection of pixels by incorporating more information about focal surface geometry.

**Acknowledgment.** The paper was supported by KEGA grant 062TUKE 4/2013, granted by the Cultural and Education Grant Agency of the Slovak Ministry of Education.

This work was partially supported by Basic Science Interdisciplinary Research Projects of RIKEN and JSPS KAKENHI Grant (22340063, 23340081, and 24244042), by the Italian Ministry of Foreign Affairs and International Cooperation, by the 'Helmholtz Alliance for Astroparticle Physics HAP' funded by the Initiative and Networking Fund of the Helmholtz Association, Germany, and by Slovak Academy of Sciences MVTS JEM-EUSO as well as VEGA grant agency project 2/0076/13. Russia is supported by the Russian Foundation for Basic Research Grant No 13-02-12175-ofi-m. The Spanish Consortium involved in the JEM-EUSO Space Mission is funded by MICINN & MINECO under the Space Program projects:

AYA2009-06037-E/AYA, AYA-ESP2010-19082, AYA-ESP2011-29489-C03, AYA-ESP2012-39115-C03, AYA-ESP2013-47816-C4, MINECO/FEDER-UNAH13-4E-2741, CSD2009-00064 (Consolider MULTIDARK) and by Comunidad de Madrid (CAM) under projects S2009/ESP-1496 & S2013/ICE-2822.

## References

1. Duda, R.O., Hart, P.E.: Use of the hough transformation to detect lines and curves in pictures. Commun. ACM **15**(1), 11–15 (1972). http://doi.acm.org/10.1145/361237.361242
2. Berat, C., et al.: Full simulation of space-based extensive air showers detectors with ESAF. Astropart. Phys. **33**(4), 221–247 (2010). http://arxiv.org/abs/0907.5275
3. The JEM-EUSO collaboration. Report on the phase A study 2010, Collaboration Mission Report, December 2010
4. Bertaina, M., et al.: Requirements and expected performances of the JEM-EUSO mission. In: Proceedings of 33rd International Cosmic Ray Conference, Rio de Janeiro, Brazil (2013). http://dx.doi.org/10.7529/ICRC2011/V03/0991
5. Bayer, J., et al.: Second level trigger and Cluster Control Board for the JEM-EUSO mission. In: Proceedings of 33rd International Cosmic Ray Conference, Rio de Janeiro, Brazil, pp. 99–102 (2013). http://arxiv.org/abs/1307.7071
6. Biktemerova, S., Pastirčák, B., Bertania, M., et al.: Simulations and the analysis of fake trigger events background in JEM-EUSO experiment. In: Proceedings of 33rd International Cosmic Ray Conference, Rio de Janeiro, Brazil, pp. 59–62 (2013). http://arxiv.org/abs/1307.7071
7. Guzman, A., et al.: The peak and window searching technique for the EUSO simulation and analysis framework: impact on the angular reconstruction of EAS. J. Phy. Conf. Ser. **409** (2013). http://dx.doi.org/10.1088/1742-6596/409/1/012104
8. Adams, J.H., et al.: An evaluation of the exposure in nadir observation of the JEM-EUSO mission. Astropar. Phys. **44**, 76–90 (2013). http://arxiv.org/abs/1305.2478
9. Fenu, F.: A simulation study of the JEM-EUSO mission for the detection of ultrahigh energy Cosmic Rays. Ph.D. dissertation, Faculty of Mathematics and Natural Sciences, The Eberhard Karls Universität Tübingen (2013). http://hdl.handle.net/10900/49955
10. Biktemerova, S., Guzman, A., Mernik, T.: Performances of JEM-EUSO: angular reconstruction. Exp. Astron. **40**(1), 1–25 (2014). http://dx.doi.org/10.1007/s10686-013-9371-0
11. Vasilko, J., Vrabel, M., Bobik, P., et al.: Pattern recognition study for different levels of UV background in JEM-EUSO experiment. Presented at 34th International Cosmic Ray Conference, The Hague, The Netherlands, 30 July–6 August 2015. http://pos.sissa.it/archive/conferences/236/661/ICRC2015_661.pdf

# Author Index





J. Janech et al. (eds.), *Proceedings of the 2015 Federated Conference on Software Development and Object Technologies*, Advances in Intelligent Systems and Computing 511,
DOI 10.1007/978-3-319-46535-7