LNBIP 116

Karl Aberer
Ernesto Damiani
Tharam Dillon (Eds.)

Data-Driven Process Discovery and Analysis

First International Symposium, SIMPDA 2011
Campione d'Italia, Italy, June/July 2011
Revised Selected Papers

ifip

Springer

Lecture Notes in Business Information Processing 116

Karl Aberer
Ernesto Damiani
Tharam Dillon (Eds.)

Data-Driven Process Discovery and Analysis

First International Symposium, SIMPDA 2011
Campione d'Italia, Italy, June 29 – July 1, 2011
Revised Selected Papers

Springer

Volume Editors

Karl Aberer
EPFL Lausanne
Lausanne, Switzerland
E-mail: karl.aberer@epfl.ch

Ernesto Damiani
Università degli Studi di Milano
Crema CR, Italy
E-mail: ernesto.damiani@unimi.it

Tharam Dillon
University of Technology
Sydney, NSW, Australia
E-mail: tharam@it.uts.edu.au

ISSN 1865-1348 e-ISSN 1865-1356
ISBN 978-3-642-34043-7 e-ISBN 978-3-642-34044-4
DOI 10.1007/978-3-642-34044-4
Springer Heidelberg Dordrecht London New York

Library of Congress Control Number: 2012948778

ACM Computing Classification (1998): H.4, J.1, H.3

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Preface

The rapid growth of organizational and business processes supported by ICT is making available a huge amount of process data. As a consequence, there is a growing demand for effective and efficient process analysis techniques. The International Symposium on Data-Driven Process Discovery and Analysis (SIMPDA), jointly organized by IFIP WG 2.6 and W.G 2.12, was conceived as a forum where researchers from different scientific communities – to name but a few, process representation and modelling, data warehousing and analysis and data mining – could freely share their insights and discuss them with potential industrial partners in open sessions. The first symposium took place in June 2011 in the unique location of Campione d'Italia, between Italy and Switzerland; it featured a number of advanced keynotes, presentations on recent research, a competitive PhD seminar, and a small number of selected research and industrial demonstrations.

After the symposium, a call for papers was circulated to solicit submissions to this LNBIP volume. The call was open to selected SIMPDA participants – both paper presenters and keynotes – as well as to the research community at large. There were 31 submissions and 11 papers were accepted for publication, covering a wide range of topics, from theoretical issues related to process representation to practical experience in process discovery and analysis.

In the paper "Towards Distributed Collaborative Workflow Management for Mobile Devices", Anna Kocurova et al. present their approach to implementing a workflow management system supporting content management. The main goal of this work is to study the implications related to scenarios where people can cooperate via mobile systems. The design of a conceptual reference framework for the definition, management and execution of mobile context-aware content-centric workflows is introduced.

The paper by Francesco Arigliano et al., "Monitoring Business Processes in the Networked Enterprise", addresses the problem of defining frameworks and languages for measuring processes. The paper proposes a framework that enables the specification and measurement of metrics, supporting the definition of relationships among metrics at different organizational levels, providing a way to bind strategic objectives to organizational and technological performance.

The paper by Wil van der Aalst et al., "Towards Improving the Representational Bias of Process Mining", describes a technique to reduce internal inconsistencies of process models created by process mining techniques. The authors propose a new generation of genetic process discovery algorithms that limit the search space to improve the quality of the results discovered and speed up the search process.

The paper by Claudio Di Ciccio et al., "MailOfMine: Analyzing Mail Messages for Mining Artful Collaborative Processes", studies the problem of an efficient approach to automatically building, on top of a collection of email messages, a set of workflow models that represent the processes underlying the knowledge workers' activities.

The paper by Gianluca Demartin et al., "Benchmarking RDF Analytics", presents a novel benchmark for evaluating and comparing the efficiency of Semantic Web data management systems on analytic queries. The benchmark models a real-world setting derived from the Bologna process and includes a broad set of queries reflecting a large panel of concrete, data-intensive user needs.

The paper by Ioana Ciuciu et al., "Towards Evaluating an Ontology-Based Data Matching Strategy for Retrieval and Recommendation of Security Annotations for Business Process Models", develops a method to provide semantic support to process modellers during the design of secure business process models.

The paper by Jerzy Korczak et al., "FP-Growth in Discovery of Customer Patterns" describes an algorithm for finding association rules and develops an experiment for evaluating the performances of the algorithm.

The paper by Marcello Leida et al., "Case Study in Process Mining in a Multinational Enterprise" deals with one such case study and describes the issues that were encountered and overcome.

The paper by Rafael Accorsi et al., "Discovering Workflow Changes with Time-Based Trace Clustering", proposes an algorithm for trace clustering, a vital preliminary step to process mining. The approach includes a time-oriented clustering of process logs that directly reflects the process dynamics.

The paper by Vera Kunzle et al., "Striving for Object-Aware Process Support: How Existing Approaches Fit Together", introduces the fundamental features of object-aware processes. The proposed approach supports a tighter integration of processes and data along these characteristics.

Finally the paper by Yishuai Lin et al., "Scrum Conceptualization Using K-CRIO Ontology", presents an ontology that allows the description of specific types of business processes: those that are dedicated to the design of a product. The ontology provides a means for annotating resources, reasoning, monitoring design processes, enabling searches and proactively proposing tips and proper contents.

Putting together a volume like this is always a team effort. We are grateful to the research community that gathered around the symposium and acknowledge the high quality of their research work, which is hopefully reflected in the papers of this volume. Hopefully, the lively discussions that took place at SIMPDA helped authors to improve their work and fostered interesting extensions. We also would like to express our deep appreciation for the referees' hard work and dedication. Above all, however, heartfelt thanks are due to the authors for submitting the best results of their work to this publication.

August 2012

Karl Aberer
Ernesto Damiani
Tharam Dillon

Symposium Committee

Co-chairs

Karl Aberer	EPFL Lausanne, Switzerland
Ernesto Damiani	Università degli Studi di Milano, Italy
Tharam Dillon	University of Technology Sydney, Australia

Steering Committee

Erich Neuhold	University of Vienna, Austria
Lionel Brunie	Institut National des Sciences Appliquées (INSA) de Lyon, France
Paolo Ceravolo	Università degli Studi di Milano, Italy
Tharam Dillon	Curtin University, Australia
Elizabeth Chang	Curtin University, Australia
Giuseppina Passiante	Università del Salento, Italy

Conference Program Committee

Peter Spyns	Vrije Universiteit Brussel - STAR Lab, Belgium
Daniele Bonetta	Università della Svizzera Italiana, Switzerland
Philippe Cudre-Mauroux	MIT-CSAIL, USA
Maurice Van Keulen	University of Twente, The Netherlands
Avigdor Gal	Israel Institute of Technology, Israel
Mohand-Said Hacid	Université Claude Bernard Lyon 1, France
Angelo Corallo	Università del Salento, Italy
Irene Vanderfeesten	Eindhoven University of Technology, The Netherlands
Rafael Accorsi	University of Freiburg, Germany
Farookh Khadeer Hussain	Curtin University, Australia
Thomas Risse	L3S Research Center, Germany
Wolfgang Klas	University of Vienna, Austria
Davide Storelli	Università del Salento, Italy
Marcello Leida	EBTIC (Etisalat BT Innovation Centre), UAE
Gabriele Ruffatti	Engineering Group, Italy
Jerzy Korczak	Wroclaw University of Economics, Poland
Abder Koukam	University of Technology UTBM, France
Renato Iannella	Semantic Identity, Australia
Manfred Reichert	University of Ulm, Germany
Wei-Chiang Hong	Oriental Institute of Technology, Taiwan (China)
Mustafa Jarrar	Birzeit University, Palestinian Territories

Schahram Dustdar	Vienna University of Technology, Austria
Mohamed Achemlal	Orange Labs, France
Jose M. Alcaraz Calero	Hewlett-Packard, UK
Mohamed Mosbah	University of Bordeaux, France
Eduardo Fernández-Medina	University of Castilla-La Mancha, Spain
Meiko Jensen	University of Bochum, Germany
Haris Mouratidis	University of East London, UK
Manfred Reichert	University of Ulm, Germany
Debasis Giri	Haldia Institute of Technology, India
Helen Balinsky	Hewlett-Packard Laboratories, UK
Valentina Emilia Balas	University of Arad, Romania
Antonio Mana Gomez	University of Malaga, Spain
Frédéric Cuppens	Télécom Bretagne, France
Nora Cuppens	Télécom Bretagne, France
Mihaela Cardei	Florida Atlantic University, USA
Eduardo Fernandez	Florida Atlantic University, USA
Andreas Wombacher	University of Twente, The Netherlands
Karima Boudaoud	Ecole Polytechnique de Nice Sophia Antipolis, France
George Spanoudakis	City University of London, UK
Artur Hecker	Télécom ParisTech, France
Etienne Riviere	Université de Neuchâtel, Switzerland
Richard Chbeir	University of Bourgogne, France
Chi Hung	Tsinghua University, China
Anas Abouelkalam	Institut National Polytechnique de Toulouse, France
Luis Soares	Barbosa Universidade do Minho, Portugal
Adrian Pasarariu	Florida Atlantic University, USA
Gregorio Martinez Perez	University of Murcia, Spain
Ioana Georgiana Ciuciu	Free University of Brussels, Belgium
Alfredo Cuzzocrea	Università della Calabria, Italy

Keynote Speakers

Wil van der Aalst	Technische Universiteit Eindhoven, The Netherlands
Florian Kerschbaum	SAP, Germany
Dragan Gasevic	Athabasca University, Canada
Barbara Pernici	Politecnico di Milano, Italy

Table of Contents

Towards Distributed Collaborative Workflow Management for Mobile Devices

Anna Kocurova[1], Samia Oussena[1], Peter Komisarczuk[1],
Tony Clark[2], and Dean Kramer[1]

[1] School of Computing and Technology
University of West London, UK
[2] School of Engineering and Information Sciences
Middlesex University, London, UK
{Anna.Kocurova,Samia.Oussena,Peter.Komisarczuk,Dean.Kramer}@uwl.ac.uk,
T.N.Clark@mdx.ac.uk

Abstract. Using mobile devices enhances overall collaboration and communication between team workers. Co-workers can collaborate and share content regardless of their location. Workflow management is a widely accepted technology that supports collaboration; however, existing workflow approaches provide only a limited support for content management. Moreover, constantly changing collaborators' contexts and needs demand more adaptable and flexible workflows. The aim of this paper is to examine how collaborative workflows can be adapted for mobile devices, be capable of dealing with current collaborators' context and support content manipulation. As a result, a conceptual reference framework for the definition, management and execution of mobile context-aware content-centric workflows is introduced.

Keywords: Distributed workflow management system, context-awareness, content lifecycle, mobile collaboration.

1 Introduction

The demand for an intelligent electronic environment which responds to people′s needs is growing with worldwide proliferation of portable mobile electronic products. With mobile devices such as smartphones, tablets or PDAs becoming ubiquitous, people are more and more dependent on using mobile computing capabilities in everyday's life. The unique mobile's benefits are that the mobile devices are the always-on and always-carried mass media with ability to create and distribute content, take and upload a picture, capture an important event and publish the video in near real time [1]. People often want to share content or information privately by using mobile devices in various situations such as in business meetings. They expect to have tools to collaborate, share and publish content as required by the situation [2].

Workflow management is a technology that can support collaboration and content sharing between participants in mobile settings. Mobile devices reside

K. Aberer, E. Damiani, and T. Dillon (Eds.): SIMPDA 2011, LNBIP 116, pp. 1–20, 2012.

in extremely dynamic contexts. Frequent changes of user, device or environment state can influence behaviour of mobile information systems. Mobile workflow capabilities can be enhanced by making workflow systems more context-aware and adaptable to changing conditions. Context awareness might have a number of meanings, depending on the domain to which it is applied. We use the following definition of context [3]:

> *'Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves. Context is typically the location, identity, and state of people, groups, and computational and physical objects.'*

In this paper, context is mainly related to two workflow concepts: content-related context and context driven collaboration. The concepts are examined in details below.

Content-Related Context: Content shared by mobile devices differs from content circulated in traditional systems. Mobile content such as image, document or video/audio file is usually user-generated or adapted for use on mobile devices and can be context-aware. Context-aware content is content semantically enriched by context. For example, a picture is associated with information about the physical location or time when it was taken. Creating, processing and disposition of context-aware content are the basic stages of its lifecycle. However, content can flow through a more complex management process. The process might involve content editing, reviewing or publishing. In addition, context information associated with content can be changed at any time. The context change can trigger an action, cause content movement between two different states or influence content behaviour. In process-centric workflows, there is only a little support for recognition of context-aware content lifecycle. Integration of the lifecycle into workflows and extending mobile workflow management systems by additional context-aware content management functionalities can enhance content manipulation and sharing. In this paper, a context-aware content-centric approach with focus on key collaboration-related context-aware contents, their lifecycles and management is described.

Context Driven Collaboration: Content sharing between collaborators might be a costly operation in terms of transfer cost, devices resource usage and users time consumption, especially when task can be performed by a number of participants with the same role but not all of them can really accomplish it within a required time. Moreover, collaborators expect to have solutions for content sharing adapted for their current needs, preferences and situation. Collaboration driven by collaborator's current context can improve team work. However, the underlying management topology has a significant impact on the way how context can drive collaboration. In centralised workflow management approaches, management decisions are made by servers and context driven collaboration would be hardly achievable. In peer-to-peer (P2P) workflow management, management decisions are made by mobile devices. The decision can be based on local context-related information. By using the P2P management topology, workflows can become more context oriented processes adapted for collaborators needs.

The P2P management topology requires the development of a mobile distributed management system that consists of a set of mobile devices, connected through wireless links. High-speed wireless connections (Wi-Fi) and open wireless technology standards such as Bluetooth enable connection between devices in various environments. In P2P workflow management, data and content are exchanged directly; administration and responsibility for operations are distributed among the devices [4]. No device has complete information about the workflow state. By using P2P management technology, communication processes can be accelerated and collaboration costs are reduced through the ad hoc administration of working groups [5]. Moreover, each device can consider a variety of local information to determine what will happen next.

In this paper, a context-aware content-centric collaborative workflow tailored for mobile P2P collaboration is introduced and a reference framework for its definition, management and execution is elaborated. This framework can bring benefits to designers and developers of mobile collaborative applications who will be able to define own workflows according to their needs. In particular, the integration of a context-aware content-centric perspective into a workflow process is examined. Content behaviour is expressed by its lifecycle. The content lifecycle is composed of content states and transitions between content states. The transitions are driven by context guard conditions. A logical architecture of a workflow management system which is capable of executing such workflows is outlined.

The structure of the paper is as follows. Section 2 describes related work. The research methodology is highlighted in Section 3. A case study is described in Section 4. Context categorisation and management are presented in Section 5. Section 6 offers the definition of context-aware content-centric workflows. The logical architecture of the workflow management system is discussed in Section 7. The final section 8 summarizes the current research stage and results achieved so far.

2 Related Work

The need for adaptive workflow management demanded by practical situations is widely known and acknowledged. We review related work to workflow adaptation according to three categories: context-aware workflow modelling, context-aware workflow adaptation and artifact-centric workflow approaches.

Wieland et al. pointed out that workflow meta-models should support context modelling and its use in workflows [6]. Earlier works such as Context Toolkit [3] provided rather simple frameworks for context modelling. Later, more structured and complex frameworks that facilitate the development of context-aware applications were presented [7,8,9]. The frameworks address concepts of context management such as differentiation of lower-level raw context data from higher-level derived context information, context aggregation and context querying. Recently with increased context data dissemination, shared global context models for context-aware applications in large-scale scenarios such as Nexus approach have been developed [10]. However, we believe that a context modelling approach need to be customised and adapted for the problem it is applied to. In this paper, no new domain concept is introduced but a customised modularisation approach to context management is proposed.

Workflow contextualisation has been addressed in a number of works, however, there are many research challenges to make context-aware workflow systems ready for practical use. From the modelling perspective, COMPRO, a methodological approach for context-aware business process modelling and discovering relevant properties and variants has been developed [11]. A more role-driven process contextualisation and an explicit definition of the context related knowledge for adequate role-based business process modelling has been addressed in another work [12]. The works offer methodological, general or abstract approaches for context awareness which are tailored to a wide spectrum of business processes. However, context awareness should be domain-related with focus on the execution environment. Thus we propose a contextualised workflow definition designed for mobile devices.

In early stages, adaptive workflows have been described as workflows able to deal with certain changes. Later, process flexibility has been seen as the ability to deal with foreseen and unforeseen changes and four types of flexibility have been recognised: flexibility by design, flexibility by deviation, flexibility by underspecification and flexibility by change [13]. Three objectives for context-aware workflow adaptation have been identified: customisation, correction and optimisation [14]. Process flexibility can be related to design time, configuration time, instantiation time or run time. No modification of process definition at build-time is needed in the case of the flexibility by selection (a priori), as opposed to the flexibility of adaptation (a posteriori) in which modification of the process definition or its instances at run-time is allowed [15]. It has been pointed out that despite to a large number of proposals for flexible workflow support, the workflow models are still composed of activities and flexibility is usually achieved by two specific approaches: allowing runtime deviations or minimalistic specification of flow dependencies [16]. Various techniques for workflow flexibility have been designed. We do not propose another approach, rather we combine existing techniques in a specific way. Our objective is to customise workflows to user's needs with focus on workflow flexibility at run-time.

Context adaptation in a distributed collaborative execution environment is challenging because a context-aware data dissemination technique needs to be considered. Reachability, availability and roles of workflow participants are the factors that should be monitored. Ghayam et al. adopted a mobile agent paradigm and proposed a distributed solution for context management by using a decision tree-based approach [17]. A notification-token mechanism for dynamic data dissemination for Vehicular Ad Hoc Networks (VANETs) is based on opportunistic publish/subscribe system [18]. However, the mechanisms are not entirely role-based. Importance how local mobility and context awareness can influence interactions among the hospital workers is shown in [19]. The work offers a conceptual insight and outlines the significance of the inquiry for context aware collaboration but lacks in providing more details.

Workflows have become complex and better solutions are needed to manage complexity and event-driven behaviour. Whilst process-centric models are still used for most workflows, a data-centric approach to workflows has become more popular. Focus on key business-relevant objects, their lifecycles and how services invoke on them has emerged [20], howbeit, object-awareness in workflow processes is still very limited [21,22]. The duality of information-centric and activity-centric workflow models has been emphasised as a sophisticated solution for workflow management [23]. Content integration is usually limited on the inputs and outputs of certain workflow activities, and impact on its content lifecycle would be neglected or hardly visible; therefore, a basis for an artifact-centric workflow model comprising four concepts has been proposed [24]. The four concepts are: business artifact information model, business artifact macro-level lifecycle, services (tasks), and the association of services to business artifact. In our work, the aim is to integrate a context-aware content-centric perspective into the mobile collaborative workflow process in order to support content management and context-driven behaviour.

So, in summary, there is no existing mobile workflow approach for day-to-day collaborative activities that would support context-aware content manipulation, integrate a context-aware content lifecycle and operate in a P2P context driven manner. We present our conceptual framework that can be used to define, manage and execute mobile context-aware content-centric workflow processes.

3 Research Methodology

The objective of the paper is to introduce a conceptual reference framework. The framework outlines information needed for the mobile context-aware content-centric workflow process definition, management and execution.

Firstly, we describe a case study to illustrate the use of the framework and targeted class of workflows. Secondly, context related to that class of workflows is determined. A modularisation approach to context management and aggregation is proposed. Thirdly, a workflow can be seen as an abstraction of real work. A workflow definition is basically an abstraction of phenomena in the real world and is used to create actual workflow operational instances. Workflow needs to be well defined because the success of a workflow management system is based on the quality of the workflow models put into it [25]. In this paper, we describe all constituents that form the context-aware content-centric workflows at a metamodel level. Finally, workflows are carried out by a mobile workflow management system. Therefore, we describe a logical architecture of the system.

The framework is expressed at a higher level of abstraction, therefore, it can be used as a tool for application domain experts who will be able to define the concrete workflow cases for a particular domain. The area of framework's application is exemplified in the following scenario.

4 Case Study

In this section, we present an example scenario to illustrate the motivation for utilising workflow management technology for mobile devices. The framework is tailored specifically to a certain class of workflows and this simplified scenario describes a particular workflow case that belongs to the class.

Consider a small business which focuses on residential and commercial interior design of houses, schools, offices, hotels and shops. Design projects are carried out by a team of six interior designers. Business success depends on close co-operation between designers and close collaboration with customers. Designers usually work out in the field and communicate with each other by using their smart phones. With mobile devices, designers can make important decisions right away, share images of design patterns with as little delay as possible.

Designers work on a number of concurrent projects. Although each project is assigned to a particular designer, design decisions are never done by a single person. Jane, as a specialist on private houses, has been assigned to a project to redesign a living room for a private client. The client wishes to optically enlarge the room by repositioning the furniture. In order to successfully complete the project, the following steps need to be taken:

1. Jane knows few tricks how to optically enlarge rooms so the furniture is repositioned according to her ideas. Jane takes a picture of the new room design by using her smart phone.
2. A simple rating system is used to quickly assess design ideas. Jane adds her own rating to the picture.
3. The picture is sent to her fellow co-workers. They work out in the field so receive the picture on their phones. Each of them can review the picture.
4. Reviewer's subjective opinion can be captured by adding a comment.
5. Reviews and comments are sent back to Jane. She finally reassesses her idea according to opinions of other designers.
6. If the idea is good, the picture is sent for final approval to client.
7. Approved picture is added to Jane's completed work.

From a user's point of view, the real pattern can be abstracted into a collaborative workflow as illustrated in the middle in Fig. 1.

It illustrates how a picture lifecycle can be integrated with the collaborative workflow and how the workflow can be context-driven. For example, context information about *location* and *time* are needed when the picture is taken. Latitude and longitude are coordinates that are usually captured as default by most mobile devices. However, high-level context information such as name of place or building might be more useful. Secondly, the picture goes through a number of states such as *Initial, Reviewed, Assessed, Approved or Final.* Jane can specify that only pictures with rating larger than her *Rating User Preference* can move

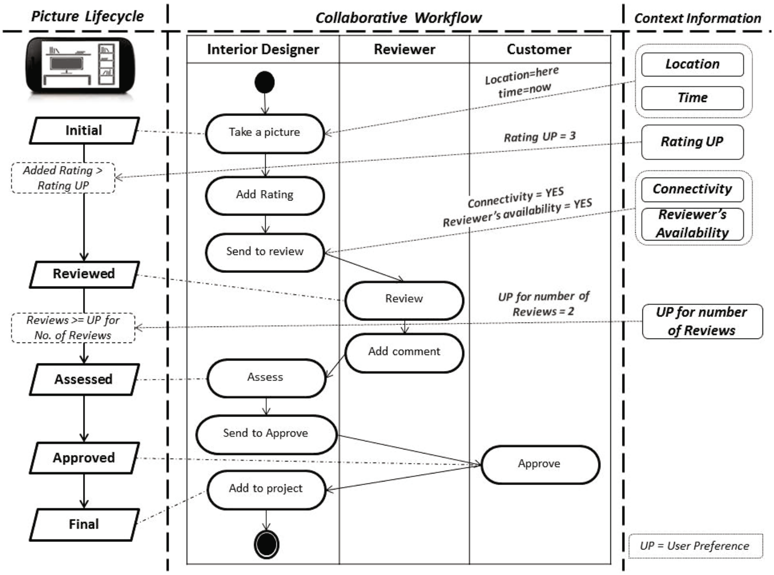


Fig. 1. Collaborative Workflow Case

from the *Initial* state to the *Reviewed* state. So if the *Rating User Preference* is set to 3 and the rating added to the picture is 4, then the picture can be sent for review. A similar situation is between the *Reviewed* and the *Assessed* states. The picture can move to the *Assessed* state only if a number of obtained reviews has reached Jane's requirement. Jane sets *User Preference for no. of reviews* to 2, so two reviews are needed for her to assess the picture. Finally, sending the picture only to reviewers who are currently available would make content sharing more efficient (*Reviewer's Availability* = *YES*).

5 Context

5.1 Context Categorisation

People use mobile devices in a more personalised way and mobile information systems need to be more customisable and adaptable to user behaviour and needs. Personalisation and adaptation of information systems is usually accomplished by considering additional information about user, for example, by including more detailed information based on personal preferences [26]. Using user preferences in the workflow concept can lead to more flexible and dynamic workflow structures that can respond to collaborators' needs.

Secondly, smart mobile phones are capable of capturing various types of contextual information. Context information can be related to the context of device or environment. Connectivity and battery level are examples of device's context.

Environmental context might be time, location or information about surrounding devices. Finally, significant context to establish cooperative effort is social context. Social context awareness relies on knowing the work context of fellow collaborators, such as their availability, current activity and location [27].

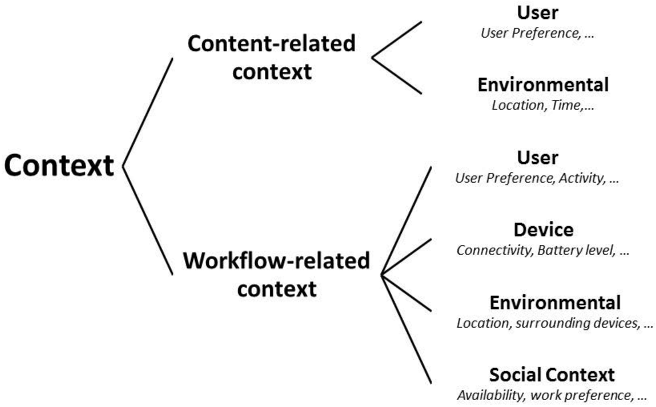


Fig. 2. Context Categorisation for the Framework

Context categorisation considered in the framework is shown in Fig. 2. We assume that whilst environmental and device's context information can influence overall workflow execution, coordinated social context-dependent workflow activities can effectively mediate the constraints of content sharing between workflow participants.

5.2 Context Management

Context management itself is not a trivial concept. We have separated context acquisition and aggregation from context adaptation. Whilst context adaptation can be considered workflow specific, context acquisition and aggregation can be workflow independent. Therefore, context acquisition and aggregation can be handled by a generic standalone mechanism that monitors, manages, aggregates and disseminates contextual information to a workflow management system and other systems running on the same mobile device. Describing the mechanism is out of scope of the paper but can be found in our previous work [28]. In the mechanism, context is described by its name and is associated with a finite set of context values. For example, context values for Bluetooth state are {*ON, OFF*}. In case of numeric data, high level context values are derived from raw values, e.g. if battery level is between 0 - 10%, its context value is defined as *LOW*. Context values for Battery level can be defined as {*LOW, MEDIUM, FULL*}. High-level context values can be defined by user. Context information is broadcasted in a unified form as a *(KEY-VALUE)* pair.

5.3 Context Aggregation

Context aggregation is not a new concept in the research field. This section describes its accommodation in this framework by illustrating the following examples (Fig. 3).

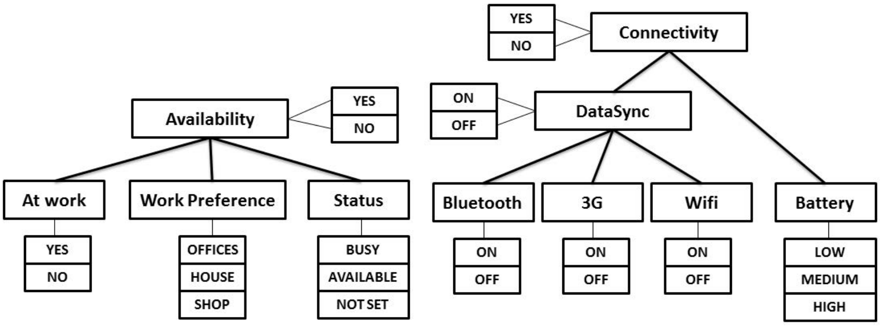


Fig. 3. Examples of Context Aggregation

Work context of fellow interior designers is expressed as a context aggregation: At work, Work Preference and Status. Context values for the 'At work' context are *YES* and *NO* representing the designer's work status. Each designer can specify own work preference, for illustration purposes only *OFFICES*, *HOUSES* and *SHOPS* are shown. The 'Status' is used to show whether collaborator is currently busy. *Availability* of the designer has an informative character to indicate whether the task can be taken by the person. The context value of *Availability* is determined by using context values of children and associated rules. The aggregated context value is broadcasted as follows: *(Availability-YES)* or *(Availability-NO)*.

The other example shows a context hierarchy for *Connectivity*. The *DataSync* component is composed of *Bluetooth*, *3G* and *WiFi*. The value sets for all of them are specified as {*ON,OFF*}. In case that any of the context is *ON*, the context value of *DataSync* is *ON* and content can be shared between devices. Content sharing might consume battery, so ideally, this operation should be performed only if battery level is not *LOW*. Thus the *Connection* component is aggregated of the *DataSync* and the *Battery* contexts. The context value of *Connectivity* is set to *YES* iff *DataSync* is *ON* and *Battery* is *MEDIUM* or *FULL*. The workflow management system gets a notification only about the aggregated context information, therefore, either *(Connectivity-YES)* or *(Connectivity-NO)*.

These examples of context aggregations are illustrative and the context aggregation hierarchy can be uniquely defined for each workflow case as shown later.

6 Context-Aware Content-Centric Workflow Process Definition

This section describes a context-aware content-centric workflow process definition designed for mobile P2P workflows. The anatomy of the workflow definition is depicted in Fig. 4. The workflow definition is composed of a role-independent part which is same for all workflow participants and a role-specific part which needs to be explicitly defined for each participating role. The parts are in details described in the remainder of this section.

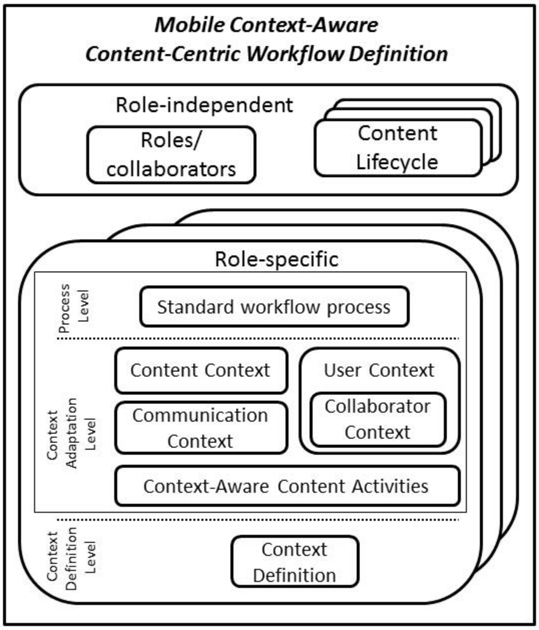


Fig. 4. Anatomy of the workflow definition

6.1 Role-Independent Part

The role-independent part of the workflow definition includes a list of roles, collaborators, and content lifecycles. These workflow constituents are same for all participants.

Roles/Collaborators: In collaborative workflows, tasks are performed by collaborators. In this concept, workflow collaborator is a person who uses a mobile device to collaborate, share content and communicate with other team members in order to achieve a common goal. Each collaborator has a mobile device which is identifiable by its phone number, Wi-Fi and Bluetooth MAC addresses

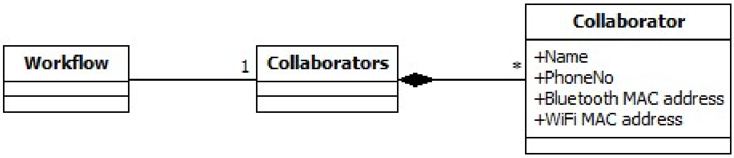


Fig. 5. Collaborators

(assuming that SIM card is not changed). In order to execute workflows, the participating devices are aware of other fellow devices and their identifiable elements. A list of participating collaborators and their devices is predefined and available on each device. The workflow definition includes a set of collaborators as shown in Fig. 5.

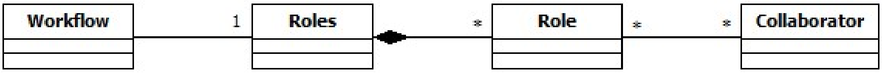


Fig. 6. Roles

A collaborator can play a number of roles in various workflow processes. For instance, Jane with the role of *Interior Designer* takes a picture and creates a workflow instance. In the meanwhile, another team member can take a picture that needs to be reviewed by Jane and creates another workflow instance. Jane will have a role of *Reviewer* in the coexisting workflow instance. Our workflow definition includes a list of roles. Each role is associated with a certain group of collaborators who can play the role (Fig. 6).

Content Lifecycle: A number of content pieces can be managed in a workflow. A set of states that each content moves through is described by a content lifecycle (Fig. 7). Transitions between a source and a target content state are associated with conditions. A condition depends on particular context. By using this structure, content management and behaviour can be context driven. A content lifecycle describes content processing across multiple mobile devices.

6.2 Role-Specific Parts

The workflow definition includes a number of role-specific parts specified for each participating role. A role-specific part in the workflow definition is defined for each participating role and comprises three levels: process level, context definition level and context adaptation level. The process level contains a standard workflow process definition. We use the Business Process Execution Language

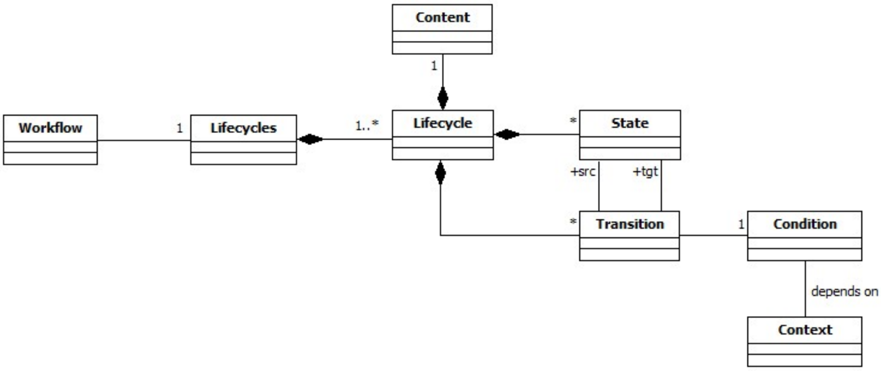


Fig. 7. Content Lifecycles

(BPEL) to describe our workflow processes but any workflow executable language can be used. Describing the process level in details is out of scope of the paper.

Context Definition: Context is described by its name, type and a context values set as shown in Fig. 8. For example, context can have name: *Status*, type: *User Preference* and context values set: {*Busy, Available, Not Set*}. However, in certain cases, only raw context data are obtained from context sources and high level context values need to be derived. We use *Values Descriptor* to associate high context values with raw data. Two examples: *Range* and *Coordinates* are outlined. For instance, *LOW* as a high-level context value for battery can be defined for a range: 0 to 10%. Composite context, dedicated for context aggregation, is designed as a context container that inherits all constructs of context. The role of the context definition constituent is to specify all atomic and aggregated contexts which influence workflow behaviour. Corresponding context values sets can be also identified. Using this constituent, context structures and hierarchies can be modelled.

Context Adaptation: As mentioned, often only aggregated context values influence behaviour of workflows. We call contexts which influence workflow execution as *active contexts*. We defined four active context types: content context, communication context, user and collaborator context (Fig. 9). Content context specifies context associated with content and its lifecycle. Communication context influences interaction between devices. User context comprises user preferences. Collaborator context specifies user's availability to perform tasks.

Context Adaptation-Content Context: Integration of content lifecycles with a particular workflow process is done by specifying content aware activities. A content-aware activity is a special type of a workflow activity. It processes content according to an action and changes its content state (Fig. 10).

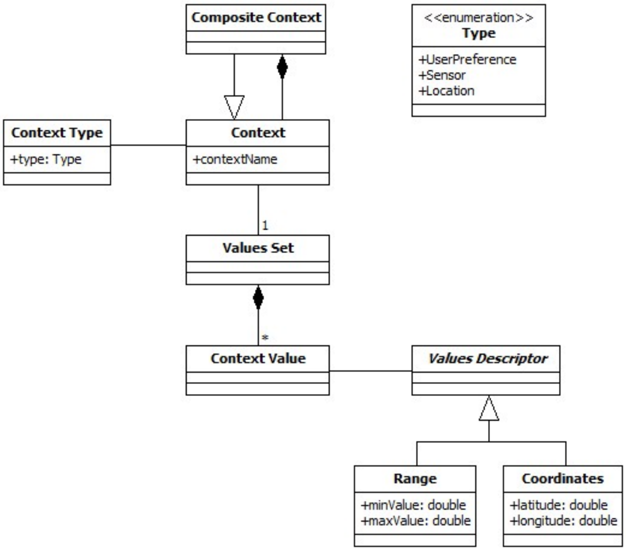


Fig. 8. Context Definition

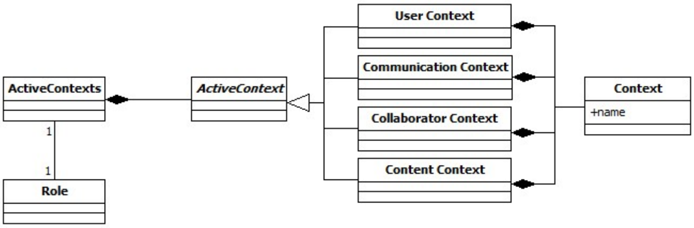


Fig. 9. Active contexts

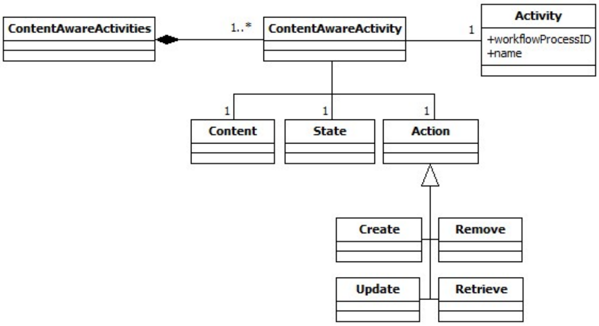


Fig. 10. Content Activities

7 Workflow Management

The context-aware content-centric workflows are interpreted and carried out by a system that needs to be deployed on each participating device. The system is based on cooperation between a mobile workflow management system and a context provisioning engine. The interaction model, as an extended version of the reference model defined by Workflow Management Coalition [29], is outlined in Fig. 11. In the remainder of this section, we describe its logical architecture and key components.

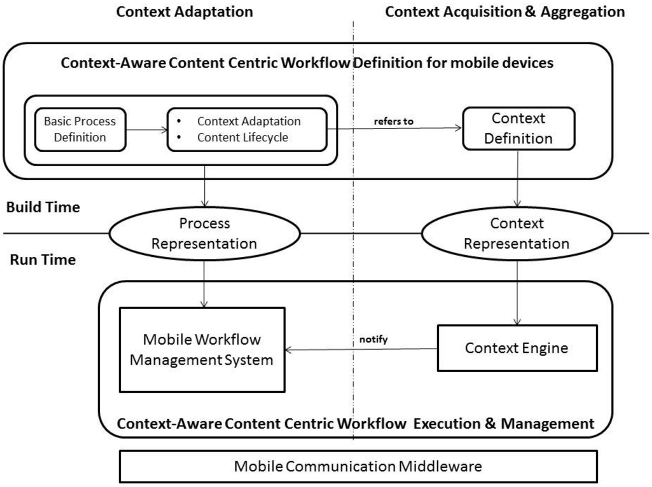

Context Adaptation
Context Acquisition & Aggregation
Context-Aware Content Centric Workflow Definition for mobile devices
Basic Process Definition
• Context Adaptation
• Content Lifecycle
refers to
Context Definition
Build Time
Run Time
Process Representation
Context Representation
Mobile Workflow Management System
notify
Context Engine
Context-Aware Content Centric Workflow Execution & Management
Mobile Communication Middleware

Fig. 11. Interaction between Workflow Management System and Context Engine

7.1 Context Provisioning Layer

A context provisioning engine is built upon the context acquisition and aggregation mechanism. As mentioned, the details about the mechanism can be found in our previous work. The engine interprets the context definition part of workflow and operates as a context provisioning platform. The context engine supports asynchronous and synchronous communication. If a context change occurs, it is broadcasted to all listening systems and applications. On the other hand, other systems and applications can query context values at any time. A prototype of the context provisioning engine is being developed on the Android platform and is available as an open source software[1].

7.2 Workflow Management System

A workflow management system needs to be deployed on each participating device. This section offers a logical architectural overview of the system. Describing

[1] `https://github.com/deankramer/ContextEngine`

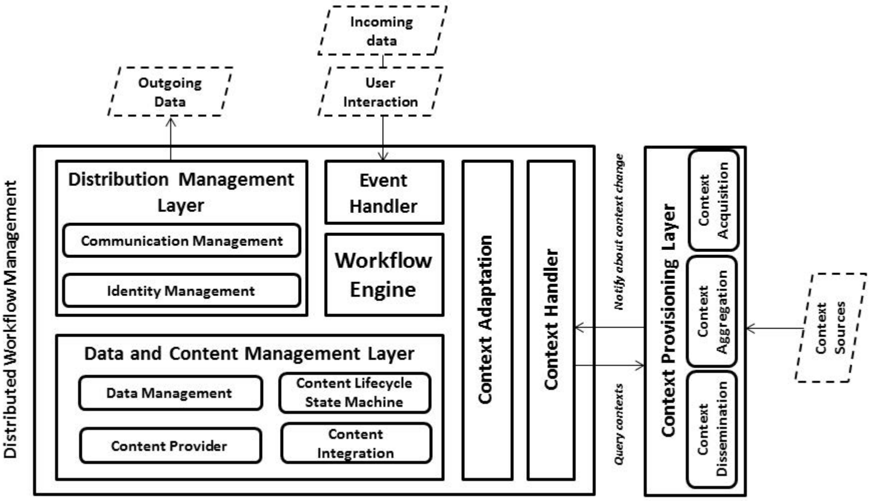


Fig. 12. The Architectural Model

the details of its run-time architecture is out of scope of the paper. P2P workflow management is more complex, therefore, key components grouping semantically related functions and data have been identified. The independent but interoperable components of the mobile distributed workflow management system are shown in Fig. 12. The system needs to detect, consume, and react to context events. To cope with context events, the architecture is based on an event-driven architecture pattern.

Context Handler: Interaction with the context provisioning layer is ensured by a context handler. The handler supports synchronous and asynchronous communication. The synchronous communication mechanism enables real-time interaction and context querying. Asynchronous communication is achieved by using a listener. The listener receives messages broadcasted by the context engine.

Context Adaptation: The component processes the received context messages further. Its role is to provide a run-time context adaptation support to all dependent components.

Data and Content Management Layer: In P2P workflow management, each device needs to administer the following data and content sets:

Data Management: Most likely a mobile device will participate in a reasonable number of small-scaled workflow processes. Workflow definitions are pre-loaded on each device. As outlined, a workflow consists from a role-independent part which has to be deployed to all participating devices and a role-based part which is deployed only to the devices assigned to the role.

Each workflow can have more operational instances running at the same time. In P2P collaboration, each device executes only an allocated partition of the workflow. Each device persists operational, instance and control data.

Content Provider: Sharing and manipulating of mobile content becomes a part of workflow execution. To prevent content duplication on a single device, content is stored in a read-only mode in a mobile′s default file system. This ensures that content can be seen by user at any time but cannot be modified. Only the content provider can encapsulate contents and manage access to them.

Content Lifecycle State Machine: A state machine is used to interpret a content lifecycle part of workflow and manages the content lifecycle.

Content Integration: Extended management functionalities for context-aware content are provided. Content-related context information is associated with content.

Event Handler: This component handles incoming text/media messages or events triggered by user.

Workflow Engine: The mobile workflow management system needs to be based on a good execution engine. Havey recommends the adoption of the Business Process Execution Language (BPEL) [30]. A workflow execution engine interprets workflow process definitions, instantiate them and provides logistic facilities for their completion. An existing mobile BPEL workflow engine called Sliver has been adapted [31]. Adding context awareness means that the engine must react on context changes and act upon them. This component must be flexible in order to enable realization of workflow modifications.

Distribution Manager: Distribution Manager is a component that manages collaboration between participants and dissemination of data and content. It functions as a task allocation manager and scheduler, therefore, it makes decisions what content and data need to be sent and when to send it.

Identity Management: In general, certain tasks within the process can be performed only by collaborators with a specified role. A list of collaborators is available on each device. This component interprets roles/collaborators part of workflow and provides access management to information about workflow participants.

Communication Manager: Communication manager cooperates with the distribution manager and communication middleware. Its role is to define a preferred communication method between mobile devices and decide how content or data should be sent.

The manager contains a mechanism to serialise and deserialise data. A mechanism for serializing structured data has been created by using Protocol Buffers - Google's data interchange format. By using Protocol Buffers, structured data are encoded in an efficient, language-neutral and platform-neutral format. The data structure for the workflow management system has been identified in the .proto file as shown in Listing 1.

Listing 1: Protocol buffer data structure described by using .proto file syntax

```
message TaskData {

  enum State {
    NEW = 0;
    RUNNING = 1;
    END = 2;
  }

 enum DataType {
    CONTEXT = 0;
    CONTROL = 1;
    VARIABLE = 2;
  }

  required string processInstanceID = 1;
  required string processID = 2;
  required string senderPhoneID = 3;
  optional State type = 4;
  optional string contentName = 5;
  optional string contentType = 6;

  message Data {
    required string dataName = 1;
    required string dataValue = 2;
    optional DataType dataType = 3;
  }

  repeated Data workflowData = 7;
}

message TaskDataCollection {
  repeated TaskData taskData = 1;
}
```

The data structure includes the following fields:

enum State **-** identifies the workflow state: NEW - informing about a new process instance; RUNNING - indicating that the message is related to an existing instance; END - informing about termination of an instance;

enum DataType **-** specifies a type of data attached to the message: CONTEXT - context data such as location or time; CONTROL - data that drives the correct execution of decentralized workflow; VARIABLE - inputs or outputs of the tasks;

processInstanceID and processID - unique identifiers of process and process instance;

senderPhoneID - identifies the sender of the message;

contentName and contentType - are optional. If a piece of content is shared between devices, these fields are filled with corresponding information about content name and its type like picture, video etc.;

workflowData - data associated with the message; the 'repeated' word indicates that more than one workflowData can be added;

message Data - determines the structure of workflowData;

message TaskDataCollection - allows packing of more than one TaskData within one message.

Communication middleware is used to exchange messages between mobile devices. It needs to support various communication protocols for information passing and content sharing over Wi-Fi, Bluetooth or by using SMS/MMS. However, communication middleware is not part of the workflow management system. The mobile workflow management systemuses only services provided by communication middleware.

8 Conclusion

This paper presents a conceptual reference framework for definition, management and execution of mobile context-aware content-centric collaborative workflows. The workflow definition has been specified in a form of metamodel. The logical architecture of the workflow management system has been described.

The development of the framework is in progress. As a proof of concept that the theory is functional, a realistic artifact is being developed on the Android platform. Its run-time architecture will be described in our future work.

The integration of content lifecycles with a workflow process presented in this work has been only an initial step towards more advanced content integration. In our future work, our aim is to extend the workflow process in the following way. The workflow process should be capable of reacting on particular changes in content lifecycles in two ways. Firstly, querying of a content lifecycle should be supported at specific places in the workflow process. Based on the obtained current content state, decisions should be made. Secondly, certain activities in the workflow process should be able to register their interest and act only when content reaches a required content state.

References

1. Fling, B.: Mobile Design and Development. O'Reilly Media, Inc., Sebastopol (2009)
2. Erickson, J., Rhodes, M., Spence, S., Banks, D., Rutherford, J., Simpson, E., Belrose, G., Perry, R.: Content-Centered Collaboration Spaces in the Cloud. IEEE Internet Computing, 34–42 (September/October 2009)

3. Dey, A.K., Abowd, G.D., Salber, D.: A Conceptual Framework and A Toolkit For Supporting the Rapid Prototyping of Context-Aware Applications. Hum.-Comput. Interact. 16(2), 97–166 (2001)
4. Androutsellis-Theotokis, S., Spinellis, D.: A survey of peer-to-peer content distribution technologies. ACM Computing Surveys 36(4), 335–371 (2004)
5. Schoder, D., Fischbach, K.: Peer-to-peer prospects. Comm. ACM 46(2), 27–29 (2003)
6. Wieland, M., Kopp, O., Nicklas, D., Leymann, F.: Towards Context-aware Workflows. In: CAiSE 2007 Proceedings of the Workshops and Doctoral Consortium (2007)
7. Henricksen, K., Indulska, J.: Developing context-aware pervasive computing applications: Models and approach. Pervasive and Mobile Computing 2(1), 37–64 (2006)
8. Reichle, R., Wagner, M., Khan, M.U., Geihs, K., Lorenzo, J., Valla, M., Fra, C., Paspallis, N., Papadopoulos, G.A.: A Comprehensive Context Modeling Framework for Pervasive Computing Systems. In: Meier, R., Terzis, S. (eds.) DAIS 2008. LNCS, vol. 5053, pp. 281–295. Springer, Heidelberg (2008)
9. Bardram, J.: The Java Context Awareness Framework (JCAF) – a service infrastructure and programming framework for context-aware applications. Pervasive Computing, 98–115 (2005)
10. Grossmann, M., et al.: Efficiently managing context information for large-scale scenarios. In: Third IEEE International Conference on Pervasive Computing and Communications, pp. 331–340 (2005)
11. De la Vara, J.L., Ali, R., Dalpiaz, F., Sánchez, J., Giorgini, P.: COMPRO: A Methodological Approach for Business Process Contextualisation. In: Meersman, R., Dillon, T.S., Herrero, P. (eds.) OTM 2010. LNCS, vol. 6426, pp. 132–149. Springer, Heidelberg (2010)
12. Saidani, O., Nurcan, S.: Context-awareness for adequate business process modelling. In: Third International Conference on Research Challenges in Information Science, RCIS 2009, pp. 177–186. IEEE (2009)
13. Schonenberg, H., Mans, R., Russell, N., Mulyar, N., Aalst, W.: Process Flexibility: A Survey of Contemporary Approaches. In: Dietz, J.L.G., Albani, A., Barjis, J. (eds.) Advances in Enterprise Engineering I, pp. 16–30. Springer, Heidelberg (2008)
14. Smanchat, S., Ling, S., Indrawan, M.: A survey on context-aware workflow adaptations. In: Proceedings of the 6th International Conference on Advances in Mobile Computing and Multimedia, pp. 414–417 (2008)
15. Nurcan, S.: A survey on the flexibility requirements related to business processes and modeling artifacts. In: Proceedings of the 41st Annual Hawaii International Conference on System Sciences (2008)
16. Redding, G., Dumas, M., Hofstede, A.H.M., Iordachescu, A.: A flexible, object-centric approach for business process modelling. Service Oriented Computing and Applications 4(3), 191–201 (2010)
17. El Ghayam, Y., Erradi, M.: Distributed Context Management in Collaborative Environment. In: 11th Annual International Conference on New Technologies of Distributed Systems (NOTERE), pp. 1–8. IEEE (2011)
18. Wu, L., Liu, M., Wang, X., Gong, H.: Dynamic distribution-aware data dissemination for Vehicular Ad Hoc Networks. In: 2010 2nd International Conference on Future Computer and Communication (ICFCC), pp. 353–360. IEEE (2010)
19. Mejía, D.A., Favela, J., Morán, A.L.: Understanding and Supporting Lightweight Communication in Hospital Work. IEEE Transactions on Information Technology in Biomedicine 14(1), 140–146 (2010)

20. Hull, R.: Artifact-Centric Business Process Models: Brief Survey of Research Results and Challenges. In: Meersman, R., Tari, Z. (eds.) OTM 2008, Part II. LNCS, vol. 5332, pp. 1152–1163. Springer, Heidelberg (2008)
21. Künzle, V., Reichert, M.: PHILharmonicFlows: towards a framework for object-aware process management. Journal of Software Maintenance and Evolution: Research and Practice (2011)
22. Vanderfeesten, I.T.P., Reijers, H.A., van der Aalst, W.M.P.: Product Based Workflow Support: Dynamic Workflow Execution. In: Bellahsène, Z., Léonard, M. (eds.) CAiSE 2008. LNCS, vol. 5074, pp. 571–574. Springer, Heidelberg (2008)
23. Kumaran, S., Liu, R., Wu, F.Y.: On the Duality of Information-Centric and Activity-Centric Models of Business Processes. In: Bellahsène, Z., Léonard, M. (eds.) CAiSE 2008. LNCS, vol. 5074, pp. 32–47. Springer, Heidelberg (2008)
24. Bhattacharya, K., Hull, R., Su, J.: A data-centric design methodology for business processes. In: Handbook of Research on Business Process Modeling, ch. 23 (2009)
25. Aalst, W., Hee, K.: Workflow Management: Models, Methods, and Systems. MIT Press, Massachusetts (2004)
26. Bierig, R.: Event and map content personalisation in a mobile and context-aware-environment. In: ACM SIGIR Forum, Thesis (2010)
27. Bardram, J.E., Hansen, T.R.: The AWARE Architecture: Supporting Context-mediated Social Awareness in Mobile Cooperation. In: Proceedings of the 2004 ACM Conference on Computer Supported Cooperative Work, Chicago, Illinois, USA, pp. 192–201 (2004)
28. Kramer, D., Kocurova, A., Oussena, S., Clark, T., Komisarczuk, P.: An Extensible, Self Contained, Layered Approach to Context Acquisition. In: 3rd International Workshop on Middleware for Pervasive Mobile and Embedded Computing at Middleware 2011, Lisbon, Portugal (2011)
29. Hollinsworth, D.: The Workflow Reference Model. Workflow Management Coalition (1994)
30. Havey, M.: Essential business process modeling. O'Reilly Media, Inc., Sebastopol (2005)
31. Sliver - A SOAP and BPEL execution engine for mobile devices, http://mobilab.cse.wustl.edu/projects/sliver/

Monitoring Business Processes in the Networked Enterprise

Francesco Arigliano[4], Devis Bianchini[5], Cinzia Cappiello[3], Angelo Corallo[2], Paolo Ceravolo[1], Ernesto Damiani[1], Valeria De Antonellis[5], Barbara Pernici[3], Pierluigi Plebani[3], Davide Storelli[2], and Claudia Vicari[4]

[1] Dipartimento di Tecnologie dell'Informazione
Università degli Studi di Milano, Italy
{firstname.lastname}@unimi.it

[2] Centro Cultura Innovativa d'Impresa, Università del Salento, Lecce, Italy
{firstname.lastname}@unisalento.it

[3] Dipartimento di Elettronica ed Informazione, Politecnico di Milano, Italy
{firstname.lastname}@polimi.it

[4] Research & Development Laboratory - Engineering, Ingegneria Informatica S.p.A., Italy
{firstname.lastname}@eng.it

[5] Dipartimento di Elettronica per l'Automazione
Università degli Studi di Brescia, Italy
{firstname.lastname}@ing.unibs.it

Abstract. The Object Management Group (OMG) is promoting the Model Driven Architecture (MDA) approach to support interaction among enterprises based on business process models. Based on this approach, we discuss in this paper how to specify performance indicators among the levels with different degree of abstraction suggested in MDA. These indicators will drive the monitoring activities to check the execution of business processes involving networked enterprises. Connecting the different levels we also decrease the cost of implementing metrics as the measurement of the entities at one level can be based on the lower level.

Keywords: performance indicators, model driven architecture, rules, violations, trends.

1 Introduction

Business Process Management (BPM) [1] is a systematic approach for driving the flow of business activities in accordance with strategic analysis. The idea is that this can be achieved only with a high level of coordination among the activities composing a business process. In particular, design, execution, and monitoring must be interrelated to verify if the running process is in compliance with requirements and to understand which part can be modified for improvement. One of the core functions of any BPM is represented by the Business Activity Monitor (BAM) that checks at run-time if a business process works according to the initial specifications by means of rules. These rules predicate on

K. Aberer, E. Damiani, and T. Dillon (Eds.): SIMPDA 2011, LNBIP 116, pp. 21–38, 2012.

data, messages and activities that have to be known in advance by the process designer and to be accessible by the BAM at run-time.

Although there are several BAM implementations, they suffer of two main limitations. Firstly, current BAMs usually focus on business processes that involve a single organization and manage the whole process by relying on a single environment [2,3]. Secondly, rules feeding a BAM predicate on elements that refer to the technological description of the business process. Thus, the BAMs assume that someone is in charge of defining these rules starting from the business needs. In addition, the continuous update of the business process to better fit the business objectives requires tools able to review the strategies adopted, the effectiveness of the information flow implemented, or the services implementing the activities.

The aim of the TEKNE (Towards Evolving Knowledge-based interNetworked Enterprise) project is to create an integrated framework for supporting and guiding Internetworked Enterprises (IEs) focusing on process modeling and rule-based monitoring. IEs, as defined in [4], are borderless organizations that share applications, services and knowledge and whose processes are transformed and integrated with the ones of their partners. The framework supports a methodology that plays a fundamental role in organizations modeling and in the measurement of their performance, in order to allow business managers and IT-specialists to control the organizational complexity in an integrated way. The methodology adopts the *MDA (Model Driven Architecture)* [5] approach in that it allows the representation of organizations and of their business at different levels of abstractions and defines a set of transformations between the models of such levels: *CIM (Computation Independent Model)* level, *PIM (Platform Independent Model)* level, and *PSM (Platform Specific Model)* level. According to this structure, companies are modeled spanning progressively from their strategies and business objectives to the enabling technological services. In this way, monitoring rules can be defined with higher level of abstraction and then transformed to the rules suitable for BAMs.

This implies the management of *directives* and *Key Performance Indicators* (KPI) both at business and technological level [6]. *Directives* are intended to define business rules the IE processes have to be compliant with, whereas *KPIs* are quantifiable measures that are usually used to analyze the trend of a process and its quality. Directives and KPIs are used to define properties that the IE business processes and the related implementation should satisfy. Usually, each organizational level defines the rules and indicators by using its own models and data and there is a lack of connection between indicators defined at different levels.

This paper proposes a framework that enables the specification and measurement of automatable metrics at each level of the modeled IE organizations, allowing as well the definition of relationships among metrics of different levels, thus providing a means to drill down the abstraction levels and bind strategic objectives to organizational and technological performance [7]. Connecting the different levels will also decrease the cost of implementing metrics as the measurement of the entities at one level can be based on the lower level. Metrics at CIM level are used to monitor process compliance to strategic directives and financial goals, thus mostly highlighting tendencies exhibited by some aggregated values extracted from enacted processes. Metrics at PIM level are directly related to operational aspects of business processes, measuring violations of business rules that affect specific activities. Metrics at PSM level are focused on process performance in terms of timing and QoS. Although each level is focused on different aspects, one of the goals of this paper is to describe how it is possible to relate a metric defined at one level to the other levels. Also the method discussed in Section 5 provides an important instrument to evaluate the feasibility of a monitoring program on a set of distributed services that constitutes the Internetworked Enterprise.

The paper is structured as follows. In Section 2 we illustrate the TEKNE framework and the approaches for rule modeling at different levels. In Section III a case study is presented. In Section IV a formal definition of Directives and KPIs is proposed and discussed. Section V discusses KPI complexity. In Section VI related work is illustrated.

2 The TEKNE Framework

The aim of the TEKNE project is to create a unique framework for BPM addressing specification, execution, and monitoring of internetworked enterprise processes. The project was also aimed at ensuring compatibility with software engineering standard approaches, such as the MDA. Basically, the company models are represented at different levels of abstractions that progressively span from the definition of business strategies to the technological platform that will implement them in concrete executions. The *Computer Independent Model* (CIM) defines the conceptual elements that are required at a business level, such as for instance actors, resources, and overall strategies and tactics of the business. The *Platform Independent Model* (PIM) describes the activities to be implemented and the flow of information driving activity execution. The *Platform Specific Model* (PSM) defines the technological platform specifying the services and the software components to be implemented. All these models are integrated with each other by appropriate mappings and are related to the methodology of organization change that is proposed in the project[1]. With this approach, we can create a complete framework that could grant a real co-design of organization and technology.

In TEKNE, we adopt three different notations for describing the business processes: one for each of the levels. More precisely, SBVR (Semantic of Business Vocabulary and Business Rules) is used at CIM level, BPMN (Business Process Modeling Notation) is used at PIM level, and finally XPDL (XML Process Definition Language) is used at PSM level.

The combination of these models is a key element for addressing the final objective of the work presented in this paper: the definition at design-time of

[1] The discussion about this methodology is out of the scope of this paper that focuses on technological aspects, further details can be found in [8].

automatable metrics for internetworked processes that can be referred directly to business policies and business rules. Indeed, the architecture allows coupling elements of a business domain (i.e., SBVR business rules) with activities of a business process, both at design time and at run time, thus binding generic logical assertions to generic operations and their instances.

Probes are defined for all the three levels in order to measure the performance of the business process execution starting from the data available at each level. In case a business rule is violated during the process execution, the run-time execution environment has the ability of adapting the process execution as a composition of services that are dynamically selected with respect to quality-of-service parameters and to their context of execution. The execution environment is based on the PAWS approach [9] for adaptive service-based processes.

2.1 Rules Modeling with SBVR

SBVR (Semantics of Business Vocabulary and Business Rules) is a formalism proposed by OMG [2] that allows for the formalization of business vocabularies and rules through a combination of syntactical and text formatting rules. According to this formalism, at design-time, the business engineer is able to conceptually model (i.e., at CIM level) a business process together by its context and directives. The adoption of SBVR is motivated by two features:

- SBVR models can be expressed by means of SBVR Structured English (SBVR-SE), a controlled English notation that has been designed to enable process/knowledge owners to directly and easily represent their tacit or explicit knowledge;
- the SBVR metamodel is compliant with and mapped to first order logic, thus fully supporting automatic interpretation and reasoning upon its assertions.

As an example, a *rule* indicating that a manager may book hotels in any category can be expressed in SBVR as follows:

Permissibility:
a manager **may** book a Hotel that has-category 5stars, 4stars, 3stars

Thanks to the mapping between elements in the BPMN and SBVR models, it is possible to derive rules from actual instances of process activities. Given a certain rule defined in a vocabulary, a rule can be defined as a real-world instance of another rule. For example "Mario Rossi books the Hotel Morning" is a rule that is an instance of the rule "Role book Hotel". Assuming that the workflow-engine is aware of roles, one can for instance derive that "Mario Rossi is a Manager", from the login logs. The resolution of rules is of responsibility of the third major component of the proposed architecture: the `Metrics Framework`. Here, data from the design-time and run-time repositories are retrieved, merged, and transformed into a logical formalism, thus constituting a knowledge base where logical assertions can be evaluated and metrics can be measured. Moreover, a dashboard

[2] http://www.omg.org/spec/SBVR/1.0/

is provided in order to allow business managers to directly monitor the actual values of the metrics they have previously defined.

2.2 Monitoring in Process Modeling Using BPMN

The *Business Process Modeling Notation* (BPMN) [10] has been adopted for specifying the business process at PIM level.

BPMN is the standard notation used for business process modeling. It is a simple graphical specification for defining the control flow of the business process. In particular determining the ordering of activities, and clearly defining pre- and post-conditions for their execution. The primary goal of developing BPMN diagrams is to provide a notation that is readily understandable by all business users, that is the analysts that create the initial drafts of processes, the technical developers who are responsible for implementing the technology automating those processes, and, finally, the business people who will manage and monitor those processes.

With BPMN business process can be seen as the collection of activities that are designed to produce the intended business behavior for a specific customer. In this way, the business engineering can capture both the broad outline and the specific procedures governing a business.

In the process models, activities represented by dotted rectangles are introduced to represent points of monitoring, where Directives or KPIs have to be verified. In addition, in the BPMN business process alternative paths can be introduced at different points of control, to check the defined directives and enforce them, defining where to read data and how to analyze them.

In order to automate the BPM for Directives and KPI definition any measure must be associated with a concrete data model providing the information required to implement that measure. As we are going to discuss in Section 5, this is the preliminary condition to reduce the cost of implementation of any measure. The integrated representation of business process models provided by `TEKNE` is motivated by the idea of providing a system supporting such a definition of KPIs.

2.3 Flexible Process Execution with XPDL and PAWS

At PSM level we adopt XPDL 2.0 (XML Process Definition Language). This XML-based language is a widely adopted standard maintained by the WfMC (Workflow Management Coalition) and supported by a number of workflow-engines. As an alternative solution, XPDL can be substituted with WS-BPEL (Web Service Business Process Execution Language) to specify the business process in terms of how to orchestrate the Web services able to perform the activities belonging to the process.

These languages allow designers modeling the process as the combination of abstract services. The main assumption is that several services can be used to perform the same activities. Such services can be, in fact, provided by alternative providers and they may present different characteristics, both from the semantic point of view (e.g. selecting expensive *vs.* cheap hotels) and for the quality of service of the provider (e.g., considering the response time).

Automatic process execution based on XPDL is complemented by the ability of dynamically selecting services to be invoked during process execution. The execution environment is based on the PAWS approach [9] for adaptive service-based processes, in which services are dynamically selected on the basis of process global constraints, service interface similarity evaluation, and QoS is negotiated. Services are retrieved from a service registry in which the service interfaces can be semantically annotated [11].

3 Case Study

A simple but real example is illustrated in Fig. 1 and Fig. 2, about an organization for managing reimbursement requests submitted by workers traveling for business. In particular, Fig. 1 and Fig. 2 represent in BPMN respectively the two processes of hotel reservation and board expenses reimbursement.

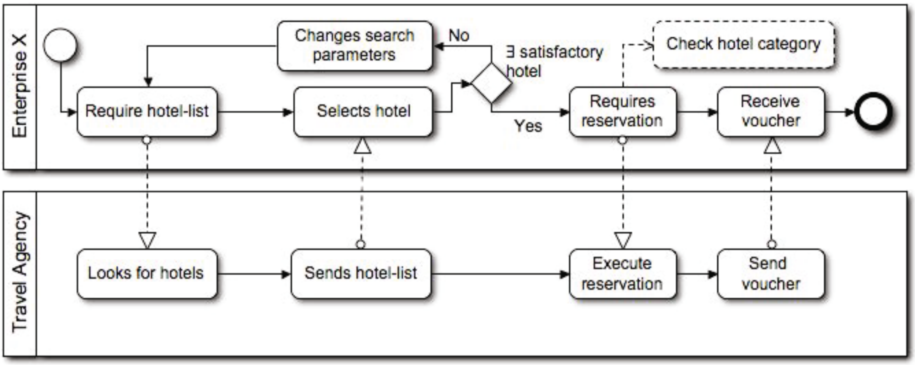


Fig. 1. An example of BP modeling *Hotel Booking*

The example is based on an internal regulation of a real SME regarding missions of its workers. The regulation identifies two kinds of workers: employees and consultants. The regulation is composed of many rules. Here few of such rules are considered for the sake of simplicity. For example, hotels booked for consultants should not be of category higher than 3 star, whereas employees do not have such a restriction. Moreover, both employees and consultants have an upper limit on the reimbursable daily board expenses.

Note that at an operational level it is permitted that the first rule is violated in some exceptional cases (e.g. only four-stars hotels are available at the location of the mission), while the second should never be violated: any exceeding expense should be systematically ignored by the administration of the organization. On the contrary at strategic level monitoring is not applied to the single activity

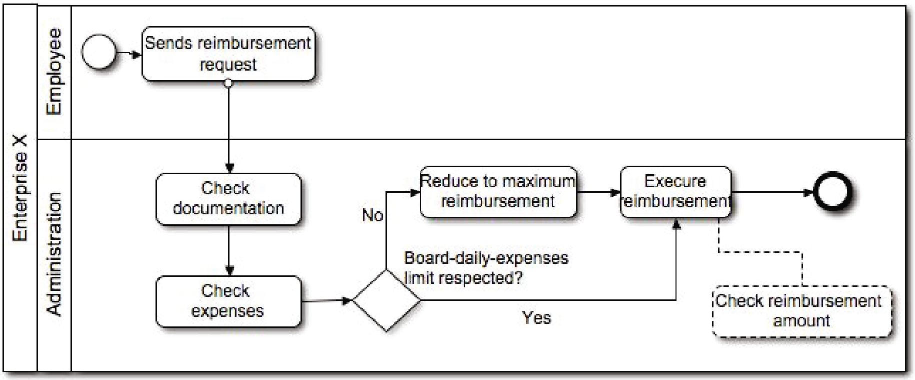

Enterprise X
Employee
Administration
Sends reimbursement request
Check documentation
Check expenses
No
Reduce to maximum reimbursement
Execure reimbursement
Board-daily-expenses limit respected?
Yes
Check reimbursement amount

Fig. 2. An example of BP modeling *Request of Reimbursement*

but on trends and aggregation of information. The rules applied insist on the comparison among a specific execution of the process and an objective to be achieved. For example a rule insisting on strategic level could state that the number of missions with daily board expenses grater than 50 euro cannot exceed the 80% of the total number of missions. Note that this rule is based on the assessment of a KPI defined as the ratio between the number of missions that violate the rule defined at the operational level and the total amount of missions.

In Fig. 1 the *Check hotel category* represents a monitoring activity; in Fig. 2 a monitoring activity is *Check reimbursement amount*, while the alternative path to *Reduce maximum reimbursement* is introduced if a business rule about the maximum reimbursement is violated in the process.

We assume that services such as *Look for hotels* in the Hotel Booking process may be dynamically selected based on the given directives (selecting hotel with limited categories), or based on timing constraints, such as for instance *Execute reimbursement* in Process Request of Reimbursement, where a faster service could be selected in the maximum time for executing the process is strict and at risk of being violated.

Given the BPMN process models and the applied rules expressed in SBVR-SE, it is possible to define KPIs and metrics for strategic, operational and execution levels, as discussed in the following of the paper.

4 Defining Directives and KPIs

As previously said, a primary aim of our framework is to provide a means for monitoring the process at different levels. For this reason, our monitoring system requeires the definition of Directives and KPIs at strategic, operational, and executive level.

Directives and KPIs are defined extending the approach described in [12]. First, we distinguish among different components of the specifications. F is a specification of the flow of activities and information that must result from the execution of the process to be developed; `W` represents a set of world properties, i.e. the context where the process is executed; `L` is the log tracking the execution of the activities described in `F`; and `R` represents a set of requirements to be met by the process, our Directives and KPIs. In TEKNE the requirements are expressed using the SBVR rules. All these components are related by the following implication:

$$F, W, L \rightarrow R \tag{1}$$

To prove that the specifications will satisfy the requirements (our rules), it is necessary to show that this implication holds. This can be done under the condition of providing an algorithm which is able to conjunctively evaluate the components, maintaining their independence. This algorithm has to work as a black box taking in input `F`, `W`, `L`, and `R`, and computing the consistency of the knowledge base. In case of inconsistency a violation has to be pointed out. Such an algorithm is compatible only with a representation of `F`, `W`, `L`, and `R` in declarative form, and assuming a uniform naming space.

Taking this approach, the definition of a Directive or a KPI reflects the requirement the system has to comply with. Of course, the hard part is achieving the uniformity in representing the specifications, maintaining the independence of the components. Another key point is to provide definitions detailing the points of control to be activated in order to equip our representation with the data values required to the resolution of the implication 1.

In addition, a definition is enriched by other information required to set up the rule resolution. In particular:

- the components of the specification to be involved, in correspondence to the related MDA level;
- the points of control to be activated to extract data values.

Based on this approach, considering the example represented in Fig. 1 and Fig. 2, we now provide the definitions of the rules previously described in Section 3 using SBVR.

Strategic Level. KPIs and directives aimed at monitoring strategic goals insist on trends exhibited by the process, as shown for instance by Rule 1a and Rule 1b.

Obligation:

Rule 1a: **It is obligatory that the number of** reimbursement **that** *have* board-daily-expense **greater than** 50 euro *is* **less or equal than** 80%

Specification: `R`, `F`, `L`	KPI type: `Strategic`
Point of control: `Check reimbursement amount`	

Obligation:

Rule 1b: **It is obligatory that the number of** reimbursement processes **that** *have* duration **greater than** 7 days *is* **less or equal than** 95%

Specification: `R`, `F`, `L`	KPI type: `Strategic`
Point of control: `Tasks Execution Time`	

Operational Level. At Operational level the aim is to implement a control on the flow of activities that has to comply to specific constrains. Rule 2 is an example of acting only on the contextual knowledge (`W`), without involving the process flow or its execution. Rules 3 and 4 are instead examples of rules requiring to know data values created at execution time.

Rule 2: Necessity:

a worker **has to** *be* an employee **or** a consultant

Specification: `R`, `W`	Directive type: `Operational`
Point of control: `none`	

Rule 3: Obligation:

board-daily-expense *is* **at most** 50 euro

Specification: `R`, `F`, `L`	Directive type: `Operational`
Point of control: `Check reimbursement amount`	

Rule 4: Obligation:

a consultant *books* a Hotel **that** *has-category* 1stars, 2stars, 3stars

These rules described above are strictly related to the domain. Rules that can be generally applied to the execution of any process are related to the duration of process or activities, as described in Rule 5.

Specification: `R`, `F`, `L`	Directive type: `Operational`
Point of control: `Check hotel category`	

Specification: `R`, `F`, `L`	Directive type: `Executive`
Point of control: `Process execution time`	

Rule 5: Obligation:
a reimbursement-process *has-duration* duration **that is less than** 7 days

Rule 5 is a directive linked to Rule 1b, where a KPI is defined based on execution time for the process. While the operational level rule can be used to check processes during execution and might be violated in exceptional cases, Rule 1b defines a criterion to be checked over multiple executions of the process, defining a KPI linked to the CIM level of IE process definitions.

In general, correspondences may be defined between rules at strategic level and operational level, with data derived from monitoring points defined in processes at the PIM level, using the extended BPM notation.

4.1 Rule Validation

To better understand how these rules can be processed in the TEKNE `Metrics Framework` let us detail how the predicates forming our knowledge base are structured. In `F` we have the predicates describing the process flow. These predicates are a specialization of the predicates in (2).

$$\begin{aligned}\rightarrow \mathit{Process}(p) \wedge \mathit{executeActivity}(p,a)\\ \wedge\ \mathit{hasStartEvent}(p,s)\\ \wedge\ \mathit{hasEndEvent}(p,e)\\ \wedge\ \mathit{hasDecisionPoint}(p,d)\\ \wedge\ \mathit{involveRole}(p,r) \wedge \mathit{Activity}(a) \wedge \mathit{Role}(r)\end{aligned} \tag{2}$$

For example the process in Fig. 1 is represented using predicates as in (3).

$$\begin{aligned}\rightarrow \mathit{Process}(p) \wedge \mathit{executeActivity}(p,a)\\ \wedge\ \mathit{hasStartEvent}(p,s)\\ \wedge\ \mathit{executeDecisionPoint}(p,d)\\ \wedge\ \mathit{involveRole}(p,r)\\ \wedge\ hasName(r, TravelAgency)\\ \wedge\ hasName(a, Looksforhotels)\\ \wedge\ hasName(d, BoardDailyExpense)\\ \wedge\ hasName(s, Start1)\end{aligned} \tag{3}$$

In `W` we can have any predicate extending the representation of the process flow with facts and constraints pertinent to the business domain. For example Rule

2 prescribes that we can refer to the union of employees and consultants using the term worker, as in (4).

$$\begin{aligned} Employee(x) &\rightarrow Worker(x) \\ Consultant(x) &\rightarrow Worker(x) \end{aligned} \tag{4}$$

In L we have the predicates describing the concrete execution of a process, i. e. the specific user executing the activities, time of execution, input and output produced. An example of the predicates we can use is in (5).

$$\begin{aligned} \rightarrow Process(p) \wedge executeActivity(p, a) \\ \wedge\ hasStartEvent(p, s) \\ \wedge\ hasEndEvent(p, e) \\ \wedge\ involveRole(p, r) \\ \wedge\ hasUserName(r, John) \\ \wedge\ hasStartTime(s, 12:02:34) \\ \wedge\ hasEndTime(e, 12:53:38) \end{aligned} \tag{5}$$

To represent the rules set R expressing Directives and KPIs we have to distinguish predicates insisting on the antecedent or on the consequent. Our task is to check if one interpretation of the knowledge base, formed by the union of W, F and L, is implied by R. Any rule is organized according to an antecedent defining a specific state of the interpretation of the predicates in the knowledge base plus a consequent that must be true if the antecedent is true. Because the theorem prover we are using implements the material implication, rules are valid if both antecedent and consequent are true or if the antecedent is false. Without distinguishing the two part of the rule we could turn up to have a valid rule only because the antecedent is false. In 6, 7 we provide an example of the representation of Rule 3 and Rule 5 according to this distinction.

$$\begin{aligned} Process(p) \wedge executeDecisionPoint(p, d) \\ \wedge\ hasName(d, BoardDailyExpense) \\ \rightarrow ValidCondition(p) \\ Process(p) \wedge executeDecisionPoint(p, d) \\ \wedge\ hasName(d, BoardDailyExpense) \\ \wedge\ hasOutput(d, 65) \\ \rightarrow ValidConsequence(p) \end{aligned} \tag{6}$$

$$\begin{aligned} &Process(p) \wedge executeActivity(p, a) \\ &\wedge\ hasName(a, ReimbursementProcess) \\ &\rightarrow ValidCondition(p) \\ &Process(p) \wedge\ hasDuration(p, d) \\ &\wedge\ lessThen(d, 7) \\ &\rightarrow ValidConsequence(p) \end{aligned} \tag{7}$$

4.2 Rule Monitoring

The PSM level receives the BPMN process representation and translates it in term of the technology that has to be used to implemenent the IT platftorm. As described in Section II, our approach is based on service-based processes. Thus, applications and related business processes can be modeled as the composition of different services provided by the organizations involved in the IE network. Formally, the process $\mathcal{P}$ specified and modeled at the PIM level that contains a set of tasks $\mathcal{T} = \{t_i\}$ can be associated with a set of services $\mathcal{S} = \{s_j\}$ to be performed. Each service s_j can execute one or more tasks t_i specified in the process model. Note that in a real business process not all the tasks can be executed by using a Web service (e.g., manufacturing activities) since they manipulate physical objects and produce tangible outputs. Thus, considering the set of services $\mathcal{S}$ and the set of tasks $\mathcal{T}$, it is possible to assume that for every process $\mathcal{S} \subseteq \mathcal{T}$.

Considering the example Hotel Booking we can identify the services as represented in Figure 3.

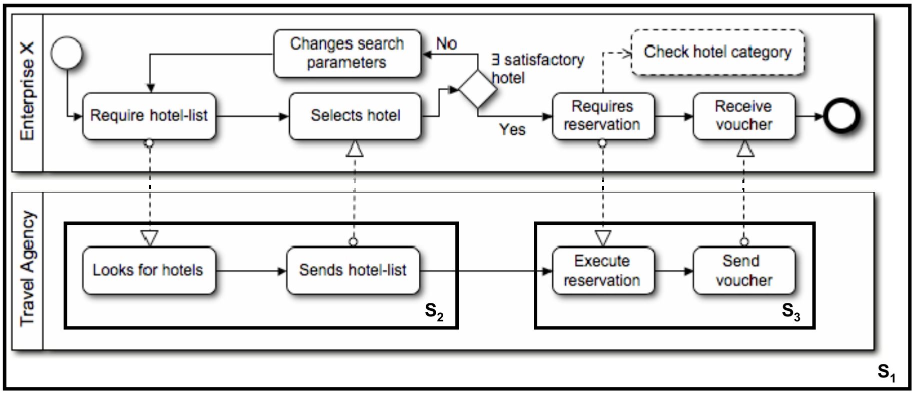


Fig. 3. *Hotel Booking* services

Each service is associated with a set of tasks as described in Table 1 (we assume that services are numbered starting from 1 inside each process description).

Table 1. Hotel Booking Service description

Service No.	Tasks Name
s1	All tasks in the process
s2	Looks for hotels and Send hotel-list
s3	Execute Reservation and Send Voucher

Considering the example Request of Reimbursement we can identify the services as represented in Figure 4.

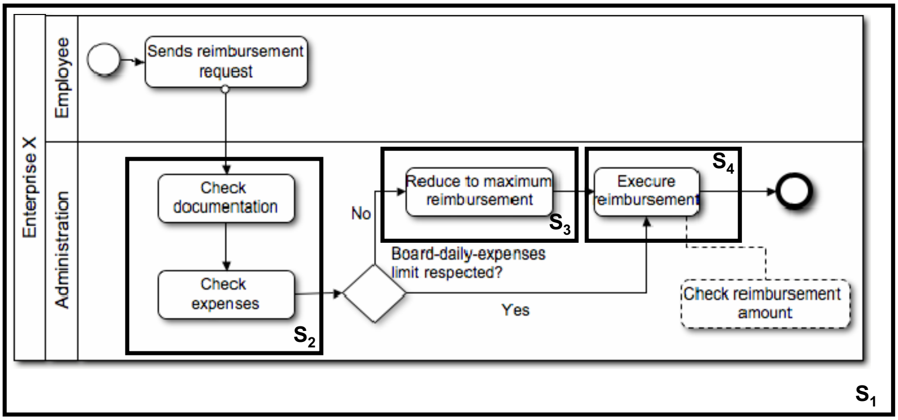

Enterprise X
Employee
Administration
Sends reimbursement request
Check documentation
Check expenses
S2
No
Reduce to maximum reimbursement
S3
Execure reimbursement
S4
Board-daily-expenses limit respected?
Yes
Check reimbursement amount
S1

Fig. 4. *Request of Reimbursement* services

Each service is associated with a set of tasks as described in Table 2.

Table 2. Hotel Booking Service description

Service No.	Tasks Name
s1	All tasks in the process
s2	Check documentation and Check expenses
s3	Reduce to maximum reimbursement
s4	Execute reimbursement

Each service s_j is characterized by functional and non-functional aspects. To represent functional aspects, a service can be modeled as

$$s_i =< name, in_j, out_k, f, r, o, QD >$$

where the service is described by the *name*, the set of input data in_j and the set of output data out_k, the related transformation function $f : in \rightarrow out$, and the role r inside the organization o responsible for the task. Non-functional aspects are related to quality attributes specified in the set QD. The quality of a Web service is defined by a set of quality dimensions QD in which each dimension qd_i associated with a given quality aspect and represented as:

$$qd_i =< qdname, V >$$

The *name* uniquely identifies the quality dimension. The element V corresponds to the quality dimension value that could be either categorical, numerical or an interval of admissible values. Both functional and non functional aspects can be used in the verification of the expressed requirements (rules validation and KPI assessment). In fact, for example the Rule 4 for the verification of the Hotel Category can be validated by considering input and output data associated with the s_2 in the Hotel Booking process. Rule 5 instead requires the analysis of non-functional properties. In particular, the assessment of the execution time of the service Request of Reimbursement has to be performed. In particular, the evaluation has to consider the *response time* dimension values associated with the services that compose the process analyzed. In formula:

$$s1.ExecutionTime \approx s2.ResponseTime + s3.ResponseTime.$$

For the assessment of Rule 1b we can consider the result of this formula and compare the number of times in which the results satisfy the Rule 5 with respect to the number of times in which the process is executed.

As regards Rule 1a we can not use any functional or not functional service properties. In this case, the number of reimbursements with limited expenses can be calculated by looking at the service execution. In fact, in the Request of Reimbursement process, this parameter can be calculated as the differences between the number of invocations of s_1 and the number of invocations of s_3.

During execution, monitoring at PSM level can support process adaptivity with respect to the defined business rules and KPI. For instance, during hotel selection, the proposed hotels may depend on the category of the employee for which the reservation is being made. Considering quality of service, different services can be employed (or selected) depending on the probability of violating a timing constraint (e.g., if the reimbursement process has been running for more than a given threshold time, a faster execute reimbursement service may be selected for s_4).

Monitoring at the PSM level is therefore needed to perform adaptivity and constraint enforcement functionality. In addition, monitoring rules can be defined to select the messages to be logged in order to be able to perform monitoring at the above levels. For instance, if the daily limit for board expenses is violated, the Request for reimbursement process is executed taking the "no" path as shown in Figure 4. However, to analyze the requirements satisfaction at PIM level, as noted above, information about the number of reimbursements above the daily limit and the total number of reimbursements is to be collected by the monitoring system. Monitoring rules can be defined using a monitoring system able to analyze the exchanged messages, such as defined in [13].

5 KPI Complexity

The literature on Business Process Monitoring has achieved today huge dimensions. Around this problem we have studies involving BPM infrastructures, Performance Indicators categorizations, BAM methods. In spite of this strong development very few lines were wrote on criteria for comparing the performances of two monitoring systems. The main reasons behind this lack are related to the difficulty to identify a uniform term of comparison. For instance the performance of a system are dependent on the dimension of the data in analysis or on the computational power in use.

To avoid being affected by all these variable factors, we propose here to evaluate the complexity of implementation of a measure, reducing its representation to the set of inputs required by the measure itself. The complexity of implementation is the effort required to transform the primitive data in our system into the inputs involved in the given measure. More concretely we can represent the complexity by the following formula:

$$C_m = \sum_{i=1}^{n} c_{im} * K. \tag{8}$$

Where c_i are the single inputs used in the measure m and K is a generic complexity of integration of distinct inputs. The complexity represented by K can vary according to the degree of homogeneity the data we are integrating have. Any c_i must be mapped to a specific set of data available in output from the system. To understand how much this mapping is complex we need to define:

- d_i for the single input data required by the measure under evaluation;
- a_l for output data that can be mapped without any manipulation;
- $b_j = f(a_j, \ ... \ a_n)$ for output data that must be manipulated by a single function;
- $e_k = g(a_l, \ ... \ a_n, b_j, \ ... \ b_n)$ for output data that must be manipulated by combination of functions.

Now we need to assign a value to this data according to a unit of measurement of information storage, and to assign a multiplicative value to these functions, for instance staying in the range $[1 - 5]$. This way the complexity of mapping one input of our measure with the output data from the system can be see as the ratio among the dimension of output data and the dimension of the output data multiplied by the required manipulations. In formula:

$$c = 1 - \frac{\sum d_i}{\sum a_l + \sum b_j + \sum e_k}. \tag{9}$$

The last passage is to show how this formula works. Let take as an example the Rule 3 and Rule 5. Using the TEKNE infrastructure Rule 3 insists on data that are directly produced by the system, when we estimate the *Board_day_expense* we directly have the information to verify that rule. Its cost is equal to 0 because in the ratio we have two factors with equal value:

$$c = 1 - \frac{d_1}{a_1}. \tag{10}$$

This is because $d_1 = a_1 = dim(Board\_day\_expense)$. With Rule 5 the result will be different because the predicate *ReimbursementProcess* is not mappable as a simple output of our system but we have to evaluate the time difference among the *start event* of a process and the *end event*. In this case the cost will be positive. The function will be a ratio among one simple input and one composite function of outputs.

$$c = 1 - \frac{d_1}{b_1}. \tag{11}$$

For example if the $b_1 = f(a_1,\ a_2)$ and $f = 3$, $b_1 = 3(a_1 + a_2)$, and the complexity of implementation of Rule 5 is equal to $c = 1 - \frac{1}{6} = 0.83$.

6 Related Work

In [14], SBVR rules are used for defining policies to be conformed, but this work is limited to access control. In paper [15] authors introduce the assertion language XSAL whose main purpose is to express business rules in the form of assertions over business processes in order to verify this rules when executing a business process. Another work [16], leveraging semantic control of BP, is focused on access control on transaction execution and on supporting financial control objectives. The work in [17] aims at modeling compliance measures based on policies and presents a framework for managing and enforcing compliance policies on enterprise models and BPs. Such work suggests to utilize SBVR to state the policies through business rules. In [18] authors define an approach to check business processes against rules emerging from business contracts. In [19] the logical language PENELOPE is presented. Such language enables verification of temporal constraints arising from compliance requirements on effected business processes. Compared with the existing research, our work explores a peculiar approach based on the idea of directly use business rules to define indicators in the monitoring activity. Such indicators can be referred to different levels (strategic, operative and execution) and are described using languages and notations familiar to business people.

Monitoring of service-based compositions is being studied in software and services areas. In [20], rules are inserted in a process to monitor its functionalities and performance, focusing on the execution of a single service. In [13], the monitoring infrastructure allows analyzing logs of messages according to monitoring rules that are specified by the designer of the monitoring functionality. In this case the focus is on analyzing series of messages arising from multiple executions of the same process model. In the approach propose in TEKNE, the goal is to be able to monitor the process at different architectural levels, and therefore analyzing at the same time both short term and long term behavioural patterns.

Furthermore, the relationships between business requirements, expressed by the definition of KPIs, and QoS have been scarcely addressed in past contributions. KPI and services have been initially compared in [21]. In this work, author indicates that business requirement definition at the PIM level should drive QoS, which in turn should drive the infrastructure requirements. In [22], the need for mapping KPIs to SLA parameters of services and infrastructure is expressed, while in [23] a service requirement can be associated with many KPIs so as to be able to move from the service space to the KPI space and introduce constraints on the KPIs. The approach presented in this paper considers the linkage between srvice characteristics and business requirements already analyzed in these contributions but also proposes the Web service monitoring as a method to enable the KPI assessment.

In conclusion we remark that today market provides professional Business Process Management (BPM) tools to monitor, administrate, measure, and analyze the performance of individual and end-to-end processes. Anyway these solutions are often limited to centralized organizations. In the TEKNE project we explored the potentialities of SW technologies for BPM, especially in distributed environments. Our framework supports design and monitoring at strategical, operational, and executive level, allowing the definition of ad-hoc Directives and KPIs.

7 Concluding Remarks

Current technology provides professional Business Process Management (BPM) tools to monitor, administrate, measure, and analyze the performance of individual and end-to-end processes. Anyway these solutions are often limited to centralized organizations. Making BPM available to the Networked Enterprises is still an open issue. Networked Enterprises are decentralized by definition, with disparate business units managing different business platforms. In this context the definition of standards supporting data portability, but maintaining the independence of the data sources is a basic requirement. In the TEKNE project we are exploring the potentialities of SW technologies for Internetworked Enterprises based on BPM, focusing on metrics for monitoring processes at different levels defined according to the MDA approach. Our framework supports design and monitoring at strategic, operational, and executive level, allowing the definition of ad-hoc Directives and KPIs for internetworked processes. We assume that monitoring rules can also result on adaptation during process execution, through dynamic service selection to support the enforcement of business rules. Also we provided a method for evaluating the complexity of implementation of a measure. This method is based on the comparison among the inputs required by the measures and the data provided in output by the informative system. This method provides an important instrument to evaluate the feasibility of a monitoring program on a set of distributed services that constitutes the Internetworked Enterprise.

Acknowledgments. This work has been partly funded by the Italian Ministry of Research Fund for Basic Research (FIRB) under project RBNE05FKZ2_004 TEKNE.

References

1. van der Aalst, W.M.P., ter Hofstede, A.H.M., Weske, M.: Business Process Management: A Survey. In: van der Aalst, W.M.P., ter Hofstede, A.H.M., Weske, M. (eds.) BPM 2003. LNCS, vol. 2678, pp. 1–12. Springer, Heidelberg (2003)
2. Oracle: Business Activity Monitoring
3. Workflow Management Consortium: The workflow reference model
4. O'Brien, J.A.: Introduction to Information Systems: Essentials for the Internetworked Enterprise. McGraw-Hill Education (2000)

5. Aßmann, U., Aksit, M., Rensink, A. (eds.): MDAFA 2003/2004. LNCS, vol. 3599. Springer, Heidelberg (2005)
6. Ceravolo, P., Damiani, E., Fugazza, C.: Representing and validating digital business processes. In: Proc. of the 3rd International Conference on Web Information Systems and Technologies, WEBIST (2007)
7. Arigliano, F., Ceravolo, P., Fugazza, C., Storelli, D.: Business Metrics Discovery by Business Rules. In: Lytras, M.D., Damiani, E., Tennyson, R.D. (eds.) WSKS 2008. LNCS (LNAI), vol. 5288, pp. 395–402. Springer, Heidelberg (2008)
8. Corallo, A., Taifi, N., Passiante, G.: Strategic and managerial ties for the new product development. In: The Open Knowlege Society. A Computer Science and Information Systems Manifesto, pp. 398–405 (2008)
9. Ardagna, D., Comuzzi, M., Mussi, E., Pernici, B., Plebani, P.: PAWS: A framework for executing adaptive web-service processes. IEEE Software 24(6), 39–46 (2007)
10. BPMN: Business process modeling notation (bpmn). Misc, OMG (2006), http://www.bpmn.org/Documents/BPMN%20V1-0%20May%203%202004.pdf
11. Plebani, P., Pernici, B.: URBE: Web service retrieval based on similarity evaluation. IEEE TKDE (November 2009)
12. Jackson, M.: Problem Frames: Analyzing and Structuring Software Development Problem. Addison-Wesley (2001)
13. Buccafurri, F., Meo, P.D., Fugini, M.G., Furnari, R., Goy, A., Lax, G., Lops, P., Modafferi, S., Pernici, B., Redavid, D., Semeraro, G., Ursino, D.: Analysis of QoS in cooperative services for real time applications. Data Knowl. Eng. 67(3), 463–484 (2008)
14. Goedertier, S., Mues, C., Vanthienen, J.: Specifying Process-Aware Access Control Rules in SBVR. In: Paschke, A., Biletskiy, Y. (eds.) RuleML 2007. LNCS, vol. 4824, pp. 39–52. Springer, Heidelberg (2007)
15. Lazovik, A., Aiello, M., Papazoglou, M.: Associating assertions with business processes and monitoring their execution. In: Proceedings of the 2nd International Conference on Service Oriented Computing (2004)
16. Namiri, K., Stojanovic, N.: A formal approach for internal controls compliance in business processes. In: 8th Workshop on Business Process Modeling, Development, and Support (2007)
17. Kharbili, M.E., Stein, S., Pulvermller, E.: Policy-based semantic compliance checking for business process management. In: CEUR Workshop Proceedings, Gemischter Workshop zu Referenzmodellierung und Semantische Geschftsprozessmodellierung, Saarbrcken, Germany, vol. 420, pp. 178–192 (November 2008)
18. Governatori, G., Milosevic, Z., Sadiq, S.: Compliance checking between business processes and business contracts. In: Proc. EDOC 2006, pp. 221–232 (2006)
19. Ghose, A., Koliadis, G.: Auditing Business Process Compliance. In: Krämer, B.J., Lin, K.-J., Narasimhan, P. (eds.) ICSOC 2007. LNCS, vol. 4749, pp. 169–180. Springer, Heidelberg (2007)
20. Baresi, L., Guinea, S.: Towards Dynamic Monitoring of WS-BPEL Processes. In: Benatallah, B., Casati, F., Traverso, P. (eds.) ICSOC 2005. LNCS, vol. 3826, pp. 269–282. Springer, Heidelberg (2005)
21. Wustenhoff, E.: Service level management in the data center. Technical report (2002)
22. Wetzstein, B., Karastoyanova, D., Leymann, F.: Towards management of SLA-aware business processes based on key performance indicators. In: 9th Workshop on Business Process Modeling, Development, and Support, BPMDS 2008 (2008)
23. Motta, G., Pignatelli, G.: Performing business processes knowledge base. In: First International Workshop and Summer School on Service Science (2007)

Towards Improving the Representational Bias of Process Mining

Wil van der Aalst, Joos Buijs, and Boudewijn van Dongen

Department of Mathematics and Computer Science,
Technische Universiteit Eindhoven, The Netherlands
{W.M.P.v.d.Aalst,J.C.A.M.Buijs,B.F.v.Dongen}@tue.nl

Abstract. Process mining techniques are able to extract knowledge from event logs commonly available in today's information systems. These techniques provide new means to *discover*, *monitor*, and *improve processes* in a variety of application domains. Process discovery—discovering a process model from example behavior recorded in an event log—is one of the most challenging tasks in process mining. A variety of process discovery techniques have been proposed. Most techniques suffer from the problem that often the discovered model is *internally inconsistent* (i.e., the model has deadlocks, livelocks or other behavioral anomalies). This suggests that the *search space should be limited* to sound models. In this paper, we propose a *tree representation* that ensures soundness. We evaluate the impact of the search space reduction by implementing a simple *genetic algorithm* that discovers such *process trees.* Although the result can be translated to conventional languages, we ensure the internal consistency of the resulting model while mining, thus reducing the search space and allowing for more efficient algorithms.

1 Introduction

More and more events are being recorded. Over the last decade we have witnessed an exponential growth of event data. Information systems already record lots of transactional data. Moreover, in the near future an increasing number of devices will be connected to the internet and products will be monitored using sensors and RFID tags. At the same time, organizations are required to improve their processes (reduce costs and response times) while ensuring compliance with respect to a variety of rules. *Process mining* techniques can help organizations facing such challenges by exploiting hidden knowledge in event logs. Process mining is an emerging research discipline that provides techniques to *discover, monitor and improve processes based on event data* [4].

Starting point for process mining is an *event log.* All process mining techniques assume that it is possible to *sequentially* record *events* such that each event refers to an *activity* (i.e., a well-defined step in the process) and is related to a particular *case* (i.e., a process instance). Event logs may store additional information such as the *resource* (i.e., person or device) executing or initiating an activity, the *timestamp* of an event, or *data elements* recorded with an event (e.g., the size of an order). We often distinguish three main types of process mining:

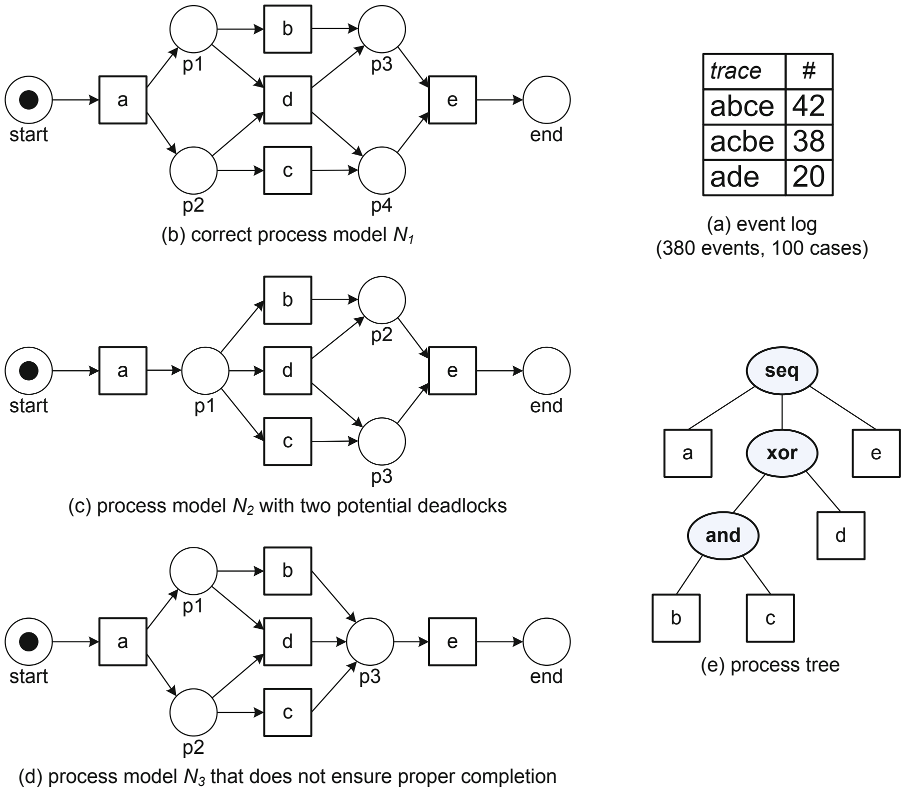
K. Aberer, E. Damiani, and T. Dillon (Eds.): SIMPDA 2011, LNBIP 116, pp. 39–54, 2012.

- *Discovery*: take an event log and produce a model without using any other a-priori information. There are dozens of techniques to extract a process model from raw event data. For example, the classical α algorithm is able to discover a Petri net by identifying basic process patterns in an event log [10]. For many organizations it is surprising to see that existing techniques are indeed able to discover real processes based on merely example executions recorded in event logs. Process discovery is often used as a starting point for other types of analysis.
- *Conformance*: an existing process model is compared with an event log of the same process. The comparison shows where the real process deviates from the modeled process. Moreover, it is possible to quantify the level of conformance and differences can be diagnosed. Conformance checking can be used to check whether reality, as recorded in the log, conforms to the model and vice versa. There are various applications for this (compliance checking, auditing, six-sigma, etc.) [30].
- *Enhancement*: take an event log and process model and extend or improve the model using the observed events. Whereas conformance checking measures the alignment between model and reality, this third type of process mining aims at changing or extending the a-priori model. For instance, by using timestamps in the event log one can extend the model to show bottlenecks, service levels, throughput times, and frequencies [4].

In this paper we focus on *process discovery*. However, we would like to stress that process discovery is only the starting point for other types of analysis. After linking events to process model elements it becomes possible to check conformance, analyze bottlenecks, predict delays, and recommend actions to minimize the expected flow time.

To illustrate the notion of process discovery see Fig. 1(a-b). Based on the event log shown in Fig. 1(a), we can discover the Petri net shown in Fig. 1(b). For simplicity we use a rather abstract description of the event log: process instances are represented by sequences of activity names (traces). For example, there are 42 cases that followed trace *abce*, 38 cases that followed trace *acbe*, and 20 cases that followed trace *ade*. This small event log consists of 380 events describing 100 process instances. There are 80 events corresponding to the execution of activity *b*. Here we abstract from additional information such as the person executing or initiating an activity, the timestamp of an event, and associated data elements. The Petri net shown in Fig. 1(b) describes a process model able to explain the observed behavior.

Process discovery can be seen as a search process, i.e., given an event log search for the model that describes the observed behavior best. When using Petri nets to describe process models, the search space consists of *all possible Petri nets*. However, even when we discard the event log, we can identify Petri nets that are clearly undesirable. Figure 1(c-d) shows two additional candidate models. Model N_2 has two potential deadlocks. After executing a and b we reach the state with just a token in place *p2*. Transition e is not enabled because there have to be tokens in both input places (*p2* and *p3*) in order for e to occur. Hence, N_2 gets



trace	#
abce	42
acbe	38
ade	20

(a) event log
(380 events, 100 cases)

(b) correct process model N_1

(c) process model N_2 with two potential deadlocks

(d) process model N_3 that does not ensure proper completion

(e) process tree

Fig. 1. A small event log (a) and four process models (b-e)

"stuck" after executing the partial trace *ab*. A similar deadlock is encountered after executing *ac*. Only after executing partial trace *ad*, transition *e* becomes enabled and the process can successfully terminate with a token in *end*.

N_3 in Fig. 1(d) has another problem. It is possible to execute trace *abe* that puts a token in place *end*. However, a token is left in *p2*. Although the process seems to have completed (token in *end*), it is still possible to execute *c*. Whereas N_2 was unable to replay the event log in Fig. 1(a), N_3 is able to replay the event log but 80 of the 100 cases do not reach the desired final state with just a token in place *end*.

The anomalies illustrated by N_2 and N_3 in Fig. 1(c-d) are *not* specific for Petri nets. Any of the main business process modeling languages (EPC, BPMN, UML, YAWL, etc.) [34] allows for deadlocks, livelocks, and improper termination. These anomalies exist independent of the event log, e.g., the event log is not needed to see that N_2 has a deadlock. Nevertheless, most process discovery techniques consider such incorrect models as possible candidates. This means that the search space is composed of both correct and incorrect models. It is not easy to limit the search space to only correct models. For common notations such as Petri nets, EPCs, BPMN, UML activity diagrams, and YAWL models it is only possible to check correctness afterwards. Note that deadlocks and livelocks are non-local properties. Hence, simple syntactical correctness-preserving restrictions are not possible without severely crippling expressiveness.

In earlier work, we used *genetic algorithms* to discover process models [7,28]. However, these algorithms suffered from the problem that the majority of process models considered during the search process has anomalies such as deadlocks, livelocks, and improper termination. Therefore, we propose to use *process trees* for process mining. Process trees such as the one shown in Fig. 1(e) cannot have any of the anomalies mentioned before (deadlocks, livelocks, etc.) as they are sound by design [27]. Process trees are discussed in more detail in Section 3. Using process trees as a *new representational bias*, we propose a new generation of genetic process discovery algorithms. *By limiting the search space to process trees, we can improve the quality of the discovered results and speed up the search process.*

The remainder of this paper is organized as follows. Section 2 elaborates on the importance of selecting the right representational bias. Section 3 formalizes the representational bias used in this paper and Section 5 introduces a new genetic algorithm. The first experimental results are presented in Section 6. Section 7 concludes the paper.

2 On the Representational Bias of Process Mining

In this section we discuss challenges related to process discovery and explain why an appropriate representational bias [3] needs to be selected.

2.1 Process Discovery as a Search Problem

Starting point for process mining is an event log composed of individual events. Each event refers to a case (process instance). Events corresponding to a case are ordered. Therefore, a case can be described by a trace, i.e., a sequence of activity names. Recall that in this paper we abstract from attributes such as timestamps, resources and additional data elements, and focus on activity names only. Different cases may have the same trace. Therefore, an event log can be formalized as a *multiset* of traces (rather than a set). A denotes the set of *activities* that may be recorded in the log. $\sigma \in A^*$ is a *trace*, i.e., a sequence of events. $L \in \mathbb{B}(A^*)$ is an *event log*, i.e., a multiset of traces. For example, the event log shown in Fig. 1(a) can be formalized as follows: $L = [abce^{42}, acbe^{38}, ade^{20}]$. Note that the trace *abce* appears 42 times in this event log.

A process discovery algorithm can be seen as a function f that, given an event log L, produces a model M, i.e., $f \in \mathbb{B}(A^*) \rightarrow \mathcal{M}$ where $\mathcal{M}$ is the class of process models considered. $\mathcal{M}$ is the *representational bias*, i.e., the set of possible candidate models. For example, the α algorithm can be seen a function that produces Petri net N_1 shown in Fig. 1(a) based on event log $L = [abce^{42}, acbe^{38}, ade^{20}]$. In this example, the representational bias $\mathcal{M}$ is the class of Petri nets where all transitions have unique labels (there cannot be two transitions with the same label).

Since the mid-nineties several groups have been working on techniques for process mining [8,10,12,15,17,19,20,32], i.e., discovering process models based on observed events. In [5] an overview is given of the early work in this domain. The idea to apply process mining in the context of workflow management systems was introduced in [12]. In parallel, Datta [17] looked at the discovery of business process models. Cook et al. investigated similar issues in the context of software engineering processes [15]. Herbst [26] was one of the first to tackle more complicated processes, e.g., processes containing duplicate tasks.

Most of the classical approaches have problems dealing with concurrency. The α-algorithm [10] is an example of a simple technique that takes concurrency as a starting point. However, this simple algorithm has problems dealing with complicated routing constructs and noise (like most of the other approaches described in literature). In [19,20] a more robust but less precise approach is presented.

Region-based approaches are able to express more complex control-flow structures without underfitting. State-based regions were introduced by Ehrenfeucht and Rozenberg [22] in 1989 and generalized by Cortadella et al. [16]. In [9,21] it is shown how these state-based regions can be applied to process mining. In parallel, several authors applied language-based regions to process mining [13,33]. In [14] a related approach based on convex polyhedra is presented.

For practical applications of process discovery it is essential that *noise* and *incompleteness* are handled well. Surprisingly, only few discovery algorithms focus on addressing these issues. Notable exceptions are heuristic mining [32], fuzzy mining [24], and genetic process mining [7,28].

See [4] for a more elaborate introduction to the various process discovery approaches described in literature.

2.2 Balancing between Quality Criteria Such as Fitness, Simplicity, Precision, and Generalization

Generally, we use four quality dimensions for judging the quality of the discovered process model: *fitness*, *simplicity*, *precision*, and *generalization*. As illustrated by Fig. 2(a), the different criteria may be competing.

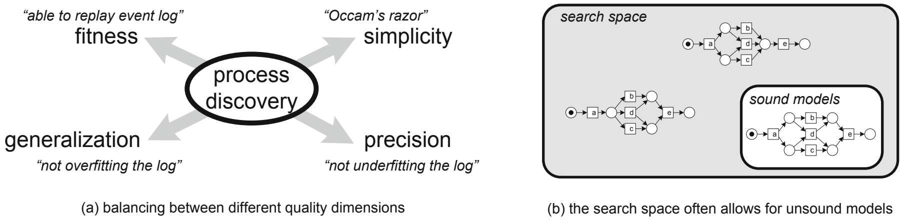
A model with good *fitness* allows for the behavior seen in the event log. A model has a perfect fitness if all traces in the log can be replayed by the model from beginning to end. There are various ways of defining fitness [4]. It can be defined at the case level, e.g., the fraction of traces in the log that can be fully replayed. It can also be defined at the event level, e.g., the fraction of events in the log that are indeed possible according to the model.

Fitness is not sufficient as it is easy to construct process models that allow for all imaginable behavior ("underfitting") or simply encode the example behaviors stored in the event log ("overfitting").

A model is *precise* if it does not allow for "too much" behavior. A model that is not precise is "underfitting". Underfitting is the problem that the model over-generalizes the example behavior in the log, i.e., the model allows for behaviors very different from what was seen in the log. For example, a Petri net without places and just transactions $\{a, b, c, d, e\}$ is able to replay the example event log, but also any other event log referring to the same set of activities.

A model should *generalize* and not restrict behavior to the examples seen in the log. A model that does not generalize is "overfitting". Overfitting is the problem that a very specific model is generated whereas it is obvious that the log only holds example behavior, i.e., the model explains the particular sample log, but a next sample log of the same process may produce a completely different process model.

The *simplicity* dimension refers to *Occam's Razor*; the simplest model that can explain the behavior seen in the log, is the best model. The complexity of the model can be defined by the number of nodes and arcs in the underlying graph. Also more sophisticated metrics can be used, e.g., metrics that take the "structuredness" or "entropy" of the model into account.



(a) balancing between different quality dimensions

(b) the search space often allows for unsound models

Fig. 2. Process discovery can be viewed as a search problem with possibly competing quality criteria in an enormous search space mostly populated by incorrect models

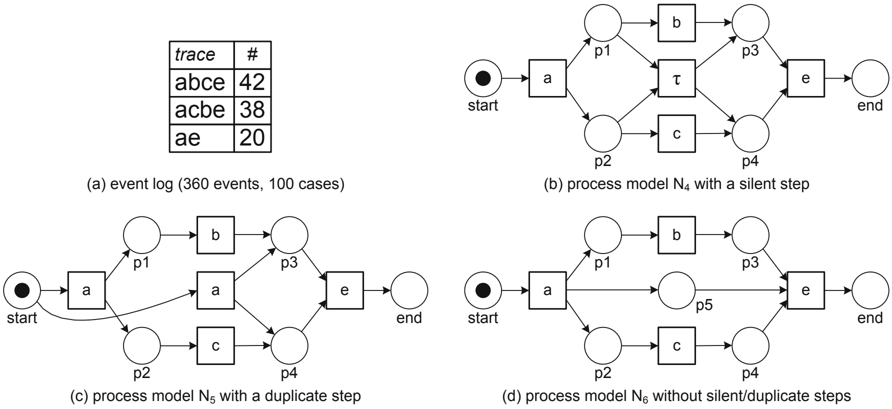
2.3 Choosing the Right Representation Bias

The four main quality dimensions mentioned in Fig. 2(a) illustrate that process discovery is a non-trivial problem. For an event log there may be a simple model with a fitness of 80% and a more complex model with a fitness of 95%. Both models can be useful. Therefore, most process discovery algorithms provide parameters to guide the discovery process. However, the search process is bounded by the representational bias $\mathcal{M}$ [3].

It is important to separate the visualization of process mining results from the representation used during the actual discovery process. The selection of the representational bias $\mathcal{M}$ should be a conscious choice and should not be (only) driven by the preferred graphical representation. To illustrate the importance of this choice, we discuss implications related to *correctness* and *expressiveness*.

The three Petri nets shown in Fig. 1(b-d) are so-called *WF-nets* (workflow nets) [1]. A WF-net is a Petri net with a designated source place (*start*) and sink place (*end*). All nodes in the net need to be on a path from *start* to *end*. A commonly used correctness notion for WF-nets is *soundness* [6]. A WF-net is sound if from any reachable state it is possible to reach the desired final state with just a token in *end*. Moreover, there should be no dead parts that can never be executed. WF-net N_1 in Fig. 1 is sound. WF-net N_2 is not sound because of the potential deadlocks: after executing b or c it is no longer possible to reach the desired final state. WF-net N_3 is not sound because only the trace *ade* results in the desired final state. The notion of soundness is not specific for WF-nets and similar notions can be defined for all mainstream process modeling languages [1,6]. Similar anomalies can be encountered in EPCs, BPMN models and the like [34].

Figure 2(b) shows the implications of having a representational bias $\mathcal{M}$ that allows for unsound models. Consider for example a genetic process mining algorithm. Initially, process models are generated randomly. Obviously, most of such models will not be sound. In each generation of a genetic process mining algorithm new models (called individuals) are created using mutation and crossover. When using conventional languages, such genetic operators are likely to create models with anomalies such as deadlocks and livelocks. As a result, the search process may take unnecessarily long because irrelevant models are considered (cf. Fig. 2(b)).



trace	#
abce	42
acbe	38
ae	20

(a) event log (360 events, 100 cases)

(b) process model N_4 with a silent step

(c) process model N_5 with a duplicate step

(d) process model N_6 without silent/duplicate steps

Fig. 3. One event log (a) and three process models (b-d)

The representational bias $\mathcal{M}$ also has implications on the *expressiveness* of the resulting model. To illustrate this, consider event log $L' = [abce^{42}, acbe^{38}, ae^{20}]$ shown in Fig. 3(a). Note that this is original event log where activity d is filtered out. WF-net N_4 in Fig. 3(b) has a silent step τ to model that after executing a it is possible to immediately enable e, i.e., the execution of τ is invisible and not recorded in the event log. WF-net N_5 has two a labeled transitions to model the three observed scenarios. Both N_4 and N_5 are able to reproduce the behavior seen in event log L'. However, there is no WF-net with unique visible labels having the visible behavior reflected in L'. Hence, any process discovery algorithm that has a representational bias limited to WF-nets with unique visible labels is destined to fail. The α algorithm [10] uses such a representational bias. Therefore, it is unable to discover the underlying process properly. The α algorithm produces WF-net N_6 shown in Fig. 3(d). This model does not allow for trace *ae*.

The example shows that the representational bias $\mathcal{M}$ may exclude desirable models. It is important that $\mathcal{M}$ supports at least the basic workflow patterns.

3 Process Trees

In this paper, we choose to use a representational bias called *process trees* that satisfy two important properties: (i) all process trees correspond to sound models, (ii) even the most basic process trees support the basic control flow patterns.

A process tree is a directed connected graph without cycles. A node V in the graph is either a branch node or a leaf node. Each leaf node represents an activity from the collection of activities $\mathcal{A}$. Each branch node, or operator node, has two children. These children can be either another operator node or a leaf. The labeling function ℓ assigns each operator node an operator from $\mathcal{O}$ and each leaf node an activity from $\mathcal{A}$. Currently, we have defined basic operators for the sequence ($\rightarrow$), exclusive choice ($\times$) and parallel ($\wedge$) constructs. Furthermore, operators for loop ($\circlearrowleft$) and OR ($\vee$) are available. The order of the children matters for the operators sequence and loop. The order of the children of a sequence operator specify the order in which they are executed (from left to right). For a loop, the left child is the 'do' part of the loop. After the execution of this 'do' part the right child, the 'redo' part, might be executed. After this execution the 'do' part is again enabled. The loop in Fig. 4 for instance is able to produce the traces $\langle A \rangle$, $\langle A, B, A \rangle$, $\langle A, B, A, B, A \rangle$ and so on. Therefore, the children of an operator node are ordered using a sorting function s. All operator nodes represent both the split and the join construction in other process modeling languages. The sequence, exclusive choice and parallel operators together cover all five basic Control Flow Patterns [2] which is one of the requirements as discussed in Section 2.3. Furthermore, by adding the loop and or operators, process trees are able to express any event log.

In contrast to Petri nets, process trees always represent sound models [27] and a straightforward translation from process trees to Petri nets exists. Figure 4 shows how each of the operators can be translated to a Petri net construct. The children of the sequential operator are ordered from left to right. In order to correctly translate the parallelism operator to a Petri net, new transitions need to be added. Since these transitions do not represent an observable activity these are marked as invisible or τ-transitions (as is for example the case in the Petri net of Figure 3(b)).

Unlike frequent pattern mining and episode mining [25], process tree discovery aims to discover end-to-end processes rather than frequent patterns. The goal is to find "complete process models" and not just process fragments that are executed frequently. Moreover, traditional data mining techniques are not considering all four quality dimensions and tend to focus on just two dimensions (e.g., fitness and simplicity).

To determine the quality of a process tree, we need metrics to measure the four dimensions. While plenty of metrics exist [18,31] to measure the quality of a Petri net, we are not aware of any metrics for process trees. Therefore, to measure the

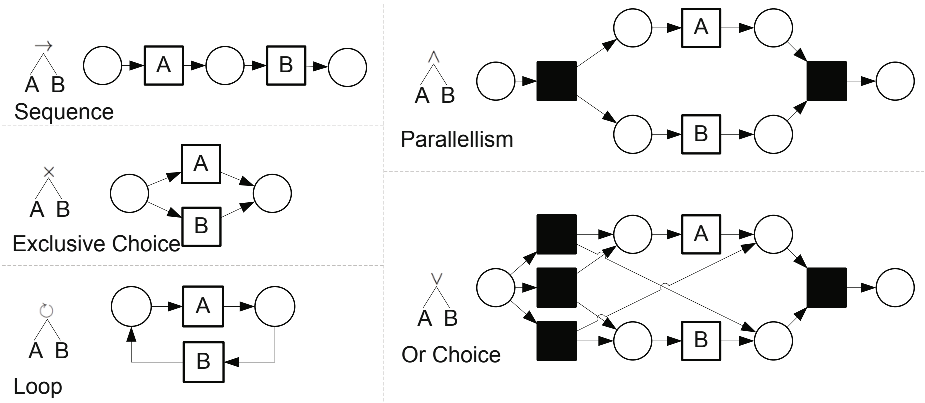


Fig. 4. Translation of Tree Operators to Petri net constructs

quality of process trees, we measure the quality of the corresponding Petri net translation, where we focus on fitness and precision. For the fitness dimension we use the cost-based fitness as defined by [11] and precision is covered by behavioral appropriateness metric which is currently under development by the same authors.

The overall quality of a process tree is computed by taking the harmonic mean of the fitness and precision metric. For two inputs (x_1 and x_2) the harmonic mean H is defined as $H = \frac{2x_1x_2}{x_1+x_2}$. The harmonic mean ensures a ‘pessimistic’ fitness value when the two quality metrics are more apart. This ensures that if one metric scores very high but the other very low, the overall quality is relatively low since the two metrics should be more balanced.

4 Searching for Process Trees

In the introduction, we stated that the goal of selecting the right representational bias was to limit the size of the search space. Therefore, when comparing process trees to Petri nets, we should compare the size of the search space for a given number of activities (i.e. a given number of transitions in a Petri net, or a given number of leaf nodes in a process tree).

Since a place in a Petri net can have any transition as input or as output, or is not connected to that transition, there are 3^n different places and a Petri net contains any number of places (where we neglect the initial marking).

Definition 1 (Number of Petri nets on n activities). *The number of different Petri nets on n activities is defined as:*

$$\#PNets(n) = 2^{3^n}$$

The number of possible binary process trees having 5 types of operations (AND,XOR,SEQ,OR and $LOOP$) and n leafs for the activities depends on the number of operator nodes $(n-1)$, the selection of the operator type of each node (5^{n-1}) and the possible orderings of the leaf nodes $(n!)$, as well as the number of ordered rooted trees with n leaves:

Definition 2 (Number of binary process trees). *The number of binary process trees with three types of operator nodes can be defined as a function of the number of leaf nodes n as follows:*

$$\begin{aligned}\#Trees(n) &= \#StructuralCombinations \cdot \#OperatorChoices \cdot \#LeafOrders \\ &= C(n-1) \cdot 5^{(n-1)} \cdot n! \\ &= \frac{(2(n-1))!}{(n)!\,(n-1)!} \cdot 5^{(n-1)} \cdot n!\end{aligned}$$

where $C(n)$: the Catalan number sequence [29] which specifies the number of ordered rooted trees with n operator nodes.

Table 1 shows the number of different Petri nets and process trees for a number of activities ranging from 1 to 6. It shows the number of process trees that are possible using three operators (SEQ,XOR and AND) and when using all 5 operators. The table clearly shows that there are far less possible trees than there are Petri nets, even when using all 5 operator types. Furthermore all of the process trees represent sound models, while of the Petri nets, only a fraction is actually sound.

Table 1. Size of the search space for varying number of activities

number of activities	number of Petri nets	number of process trees	
		3 operators	5 operators
1	8	1	1
2	512	6	10
3	134,217,728	108	300
4	2,41785 $\cdot 10^{24}$	3,240	15,000
5	1,41347 $\cdot 10^{73}$	136,080	1,050,000
6	2,82401 $\cdot 10^{219}$	7,348,320	94,500,000

Since any process tree that can be generated represents a sound model, a naive process mining algorithm could simply generate random trees, test their quality and if the quality is not good enough, try again. To see how many of such experiments would have to be conducted, we look at a simple example with 6 activities and three different logs. Furthermore, we only consider the SEQ, XOR and AND operators. The first event log contains only one trace which describes a sequential process model, $L_{SEQ} = [abcdef]$. The next event log describes 6 activities in an exclusive choice, $L_{XOR} = [a, b, c, d, e, f]$. The third event log contains all 720 possible permutations of the 6 activities $A-F$ and thus describes the process model where these 6 activities are executed in parallel.

For the sequential log containing 6 activities we iterated over all $7,348,320$ possible trees and found there are 42 trees describing this behavior with an overall quality of 1. This can be also be calculated using Definition 2 by observing that there is only one correct order of the leaf nodes (namely A to F from left to right) and that all operator nodes should be of the type sequence. For both the exclusive choice and parallel event logs there are $30,240$ trees that describe this behavior. This implies that for the sequential case one out of $174,960$ trees is correct where for the exclusive choice and parallel case this is one out of 243 trees.

Generating random trees until the first tree is found with quality 1 can be seen as the repetition of a Bernoulli experiment until the first success, and hence this process follows a geometric distribution. Table 2 shows how many trees we expect to generate, when doing such an experiment 100 times, i.e. in order to randomly find a process tree with perfect quality for our sequential log, the number of trees that we expect to need to generate is $174,960$ with a 99% confidence interval of 4506.66.

Table 2. Results for Process Trees with 6 Activities

	Sequential	**Exclusive Choice**	**Parallel**
Trees considered	174,960	243	243
Variance	306,108,266	5,881	5,881
99% Confidence Interval	4506.66	19.75	19.75

Clearly, just randomly searching for a process tree is not a good idea when mining a process model. Therefore, in Section 5 we investigated the use of genetic algorithms to more efficiently search for a process tree.

5 Genetic Mining for Process Trees

Genetic process mining is a technique to discover process models from an event log of observed behavior. Instead of using a deterministic approach, as most discovery algorithms do, an evolutionary algorithm is applied [23]. This approach has first been applied to process mining in [7,28].

The main idea of evolutionary algorithms is that one of the following is unknown [23]: the input to the problem, the model or algorithm, or the desired output. In the case of genetic mining the input, the event log, is known and provided. The desired output given the input is not known exactly, but we can determine which solutions are better or worse than others by providing a fitness value for each candidate model. Therefore the goal of the genetic mining algorithm is to provide a (process) model that describes the observed behavior in the event log ‘best’ given a certain fitness. The fitness function can be used to emphasize desired characteristics of the resulting process model.

A genetic algorithm is a search heuristic where a suitable solution needs to be found among possible candidates. The search space is searched by (semi-) randomly creating and changing candidates until certain stop criteria are fulfilled. In general a genetic algorithm follows the flow as shown in Figure 5. The initial population, or set of candidate solutions, can be created completely random or using some simple heuristics. In the next step the fitness of each candidate is calculated. If the algorithm should continue then the candidates are changed. In general there are two types of change operations: *crossover* mixes elements of two candidates creating two children; *mutation* changes one or more details of a candidate. These change operations are applied to a selection of the fitter candidates. Furthermore, for a small group with the highest fitness values a copy is created, which is not changed, to make sure that a copy of the fittest candidates survive. Each cycle of fitness calculation and population changes is called a generation. When a specified minimum fitness value is achieved, or when a specified maximum number of generations or execution time has been reached, the algorithm stops. The fittest candidate is now returned as the output of the algorithm.

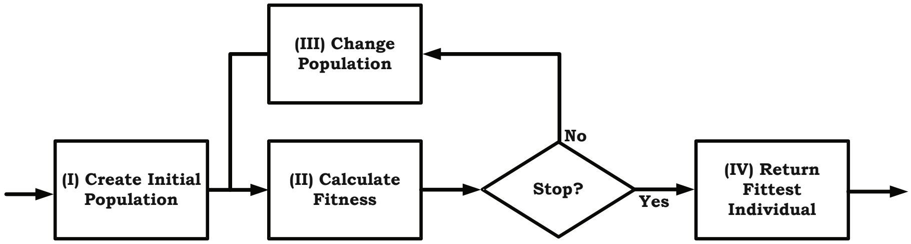


Fig. 5. Genetic Algorithm flow

In [7,28], a genetic algorithm is presented to mine Petri nets. This genetic algorithm uses an internal matrix representation that can easily be translated to a Petri net and vice versa. In essence the authors represent a Petri net as a matrix which makes it easier to define correct mutation and crossover functions. Since the matrix representation is almost as expressive as Petri nets and is able to describe unsound Petri nets, the search space is equivalent in size as the search space of Petri nets, shown in Table 1.

The fitness function, which specifies how 'good' a certain candidate is considering the event log, is the second most important part of a genetic algorithm. The fitness function determines the main characteristics of the output. It can for instance consider all, or a selection of, the quality dimensions shown in Figure 2. It is important however to correctly balance the different quality dimensions. If the fitness function for instance only considered the 'fitness' dimension, the resulting model is likely to have a very low 'precision' on the event log. Furthermore, if the event log contains a lot of noise, aiming for a perfect 'fitness' might not result in the desired process model. For simplicity, our fitness calculation directly uses the quality metrics discussed in Section 3 which cover the fitness and precision dimensions.

Another important aspect of the genetic algorithm are the crossover and mutation functions specified on the internal representation. The main purpose of the crossover function is to combine two good parts from two candidates together in a single candidate. The mutation function randomly modifies a candidate to introduce possibly new behavior that might be beneficial for the fitness. Together the crossover and mutation functions need to make sure that all possible candidates *can* be discovered.

In order to change the process tree candidates we define 4 mutation functions on process trees. The single node mutation selects a single vertex and modifies the labeling function ℓ on that node. This means that an operator node gets a different operator assigned and a leaf node represent a different activity. The second mutation function adds a leaf node with a randomly selected activity to a randomly selected operator node. In a similar way the third mutation function removes a randomly selected vertex, which might be an operator node. The most complicated mutation function is the 'internal crossover' mutation which swaps subtrees within a single process tree.

Our implementation currently does not use crossover since initial experiments showed that this was not beneficial for the performance. For the future we plan to add a guided crossover.

The performance of any genetic algorithm is determined mainly by the size of the search space and the time needed to compute the fitness of an individual. By using process trees as the internal representation we drastically reduce the search space as we only consider sound models and therefore we improve performance of the genetic algorithm. In this paper, we did not try to optimize the fitness computations.

6 Experimental Results

In this section we discuss the first experimental results of the initial version of the genetic algorithm. For these experiments we ran the genetic mining algorithm 100 times and recorded how many trees it created in order to find a candidate with an overall fitness of 1. We used a small population of 10 candidates in each generation. In each next generation the 2 fittest candidates of the last generation are copied without mutation. The 8 other candidates are created by first copying one of the candidates of the previous generation and then applying one of the mutation functions. Initialization of the 10 trees in the first generation was done completely random in order to test the mutation functions.

We applied the genetic algorithm on the three event logs introduced before.

We compare the performance of our genetic algorithm to the case where trees are created completely random until a suitable candidate has been found. All of our experiments are independent and identically distributed. Therefore we can compare the random case, where we would create trees completely random, with our experiments. In the random case we would on average need to create $174,960$

process trees for the sequential or 243 process trees for the other 2 event logs. The 99% confidence interval is +/- 4507 and +/- 19.75, respectively, as shown in Table 3 (note that we copied the results of Table 2 for easy comparison).

Table 3. Results for Process Trees with 6 Activities

	Sequential		**Exclusive Choice**		**Parallel**	
	Random	Genetic	Random	Genetic	Random	Genetic
Trees considered	174,960	459.84	243	100.96	243	100.88
Variance	306,108,266	95,042	5,881	8,454	5,881	18,515
Standard dev	17,495.95	308.29	76.69	91.95	76.69	136.07
99% Confidence Interval	4506.66	79.41	19.75	23.68	19.75	35.05

The experimental results shown in Table 3 clearly show an improvement over the random case. In all cases the number of trees that needed to be generated before finding a tree with perfect fitness was far less than the random case, even when considering the 99% confidence intervals.

This shows that by first drastically reducing the search space, followed by an application of a simple genetic mining algorithm, provides perspectives for a new genetic mining algorithm. Of course, performance can be further increased by initializing the trees in a smart way. Another important future improvement is the definition of a custom fitness function directly on process trees. This could drastically reduce the time required for a fitness calculation, which currently is the main performance issue. These initial results do conform our intuition that process trees are a good candidate for genetic mining.

7 Conclusion

Most process mining discovery algorithms suffer from the problem that the discovered model is potentially unsound. Considering unsound process models in a search space suggests that improvements are possible by eliminating these. In this paper, we propose an alternative representation for process models, *process trees*. Process trees are inherently sound and can easily be translated to other process modeling languages, such as Petri nets, EPCs or BPMN.

Process mining algorithms can be seen as search algorithms. By using process trees as a representation, the search space of all possible process models is drastically reduced, compared to Petri nets. In this paper we show that genetic algorithms can be defined to search this reduced search space. One of the main benefits of genetic mining algorithms is the flexibility. The desired characteristics of the resulting process model can be defined in the fitness function. This makes the genetic algorithm a versatile discovery algorithm that can be applied in many situations. In this paper we used existing Petri net based metrics to calculate fitness. As a result, our current approach is rather slow and not very suitable for real-life processes. However, it seems possible to define quality metrics directly on trees that make use of their unique characteristics. We expect that this will dramatically speed up the discovery process.

References

1. van der Aalst, W.M.P.: The Application of Petri Nets to Workflow Management. The Journal of Circuits, Systems and Computers 8(1), 21–66 (1998)
2. van der Aalst, W.M.P.: Workflow Patterns. In: Liu, L., Tamer Özsu, M. (eds.) Encyclopedia of Database Systems, pp. 3557–3558. Springer, Berlin (2009)
3. van der Aalst, W.M.P.: On the Representational Bias in Process Mining (Keynote Paper). In: Reddy, S., Tata, S. (eds.) Proceedings of the 20th Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE 2011), Paris, pp. 2–7. IEEE Computer Society Press (2011)
4. van der Aalst, W.M.P.: Process Mining: Discovery, Conformance and Enhancement of Business Processes. Springer, Berlin (2011)
5. van der Aalst, W.M.P., van Dongen, B.F., Herbst, J., Maruster, L., Schimm, G., Weijters, A.J.M.M.: Workflow Mining: A Survey of Issues and Approaches. Data and Knowledge Engineering 47(2), 237–267 (2003)
6. van der Aalst, W.M.P., van Hee, K.M., ter Hofstede, A.H.M., Sidorova, N., Verbeek, H.M.W., Voorhoeve, M., Wynn, M.T.: Soundness of Workflow Nets: Classification, Decidability, and Analysis. Formal Aspects of Computing 23(3), 333–363 (2011)
7. van der Aalst, W.M.P., Alves de Medeiros, A.K., Weijters, A.J.M.M.: Genetic Process Mining. In: Ciardo, G., Darondeau, P. (eds.) ICATPN 2005. LNCS, vol. 3536, pp. 48–69. Springer, Heidelberg (2005)
8. van der Aalst, W.M.P., Reijers, H.A., Weijters, A.J.M.M., van Dongen, B.F., Alves de Medeiros, A.K., Song, M., Verbeek, H.M.W.: Business Process Mining: An Industrial Application. Information Systems 32(5), 713–732 (2007)
9. van der Aalst, W.M.P., Rubin, V., Verbeek, H.M.W., van Dongen, B.F., Kindler, E., Günther, C.W.: Process Mining: A Two-Step Approach to Balance Between Underfitting and Overfitting. Software and Systems Modeling 9(1), 87–111 (2010)
10. van der Aalst, W.M.P., Weijters, A.J.M.M., Maruster, L.: Workflow Mining: Discovering Process Models from Event Logs. IEEE Transactions on Knowledge and Data Engineering 16(9), 1128–1142 (2004)
11. Adriansyah, A., van Dongen, B., van der Aalst, W.M.P.: Conformance Checking using Cost-Based Fitness Analysis. In: Chi, C.H., Johnson, P. (eds.) IEEE International Enterprise Computing Conference, EDOC 2011, pp. 55–64. IEEE Computer Society (2011)
12. Agrawal, R., Gunopulos, D., Leymann, F.: Mining Process Models from Workflow Logs. In: Schek, H.-J., Saltor, F., Ramos, I., Alonso, G. (eds.) EDBT 1998. LNCS, vol. 1377, pp. 469–483. Springer, Heidelberg (1998)
13. Bergenthum, R., Desel, J., Lorenz, R., Mauser, S.: Process Mining Based on Regions of Languages. In: Alonso, G., Dadam, P., Rosemann, M. (eds.) BPM 2007. LNCS, vol. 4714, pp. 375–383. Springer, Heidelberg (2007)
14. Carmona, J., Cortadella, J.: Process Mining Meets Abstract Interpretation. In: Balcázar, J.L., Bonchi, F., Gionis, A., Sebag, M. (eds.) ECML PKDD 2010, Part I. LNCS, vol. 6321, pp. 184–199. Springer, Heidelberg (2010)
15. Cook, J.E., Wolf, A.L.: Discovering Models of Software Processes from Event-Based Data. ACM Transactions on Software Engineering and Methodology 7(3), 215–249 (1998)
16. Cortadella, J., Kishinevsky, M., Lavagno, L., Yakovlev, A.: Deriving Petri Nets from Finite Transition Systems. IEEE Transactions on Computers 47(8), 859–882 (1998)
17. Datta, A.: Automating the Discovery of As-Is Business Process Models: Probabilistic and Algorithmic Approaches. Information Systems Research 9(3), 275–301 (1998)

18. De Weerdt, J., De Backer, M., Vanthienen, J., Baesens, B.: A critical evaluation study of model-log metrics in process discovery. In: Business Process Management Workshops: BPM 2010 International Workshops and Education Track, Hoboken, NJ, USA, September 13-15, vol. 66, p. 158 (2011) (Revised Selected Papers)
19. van Dongen, B.F., van der Aalst, W.M.P.: Multi-phase Process Mining: Building Instance Graphs. In: Atzeni, P., Chu, W., Lu, H., Zhou, S., Ling, T.-W. (eds.) ER 2004. LNCS, vol. 3288, pp. 362–376. Springer, Heidelberg (2004)
20. van Dongen, B.F., van der Aalst, W.M.P.: Multi-Phase Mining: Aggregating Instances Graphs into EPCs and Petri Nets. In: Marinescu, D. (ed.) Proceedings of the Second International Workshop on Applications of Petri Nets to Coordination, Workflow and Business Process Management, pp. 35–58. Florida International University, Miami (2005)
21. van Dongen, B.F., Busi, N., Pinna, G.M., van der Aalst, W.M.P.: An Iterative Algorithm for Applying the Theory of Regions in Process Mining. In: Reisig, W., van Hee, K., Wolf, K. (eds.) Proceedings of the Workshop on Formal Approaches to Business Processes and Web Services (FABPWS 2007), pp. 36–55. Publishing House of University of Podlasie, Siedlce (2007)
22. Ehrenfeucht, A., Rozenberg, G.: Partial (Set) 2-Structures - Part 1 and Part 2. Acta Informatica 27(4), 315–368 (1989)
23. Eiben, A.E., Smith, J.E.: Introduction to Evolutionary Computing. In: Natural Computing, Springer, Berlin (2003)
24. Günther, C.W., van der Aalst, W.M.P.: Fuzzy Mining – Adaptive Process Simplification Based on Multi-perspective Metrics. In: Alonso, G., Dadam, P., Rosemann, M. (eds.) BPM 2007. LNCS, vol. 4714, pp. 328–343. Springer, Heidelberg (2007)
25. Han, J., Cheng, H., Xin, D., Yan, X.: Frequent pattern mining: current status and future directions. Data Mining and Knowledge Discovery 15(1), 55–86 (2007)
26. Herbst, J.: A Machine Learning Approach to Workflow Management. In: Lopez de Mantaras, R., Plaza, E. (eds.) ECML 2000. LNCS (LNAI), vol. 1810, pp. 183–194. Springer, Heidelberg (2000)
27. Kopp, O., Martin, D., Wutke, D., Leymann, F.: The Difference Between Graph-Based and Block-Structured Business Process Modelling Languages. Enterprise Modelling and Information Systems Architecture 4(1), 3–13 (2009)
28. Alves de Medeiros, A.K., Weijters, A.J.M.M., van der Aalst, W.M.P.: Genetic Process Mining: An Experimental Evaluation. Data Mining and Knowledge Discovery 14(2), 245–304 (2007)
29. The On-Line Encyclopedia of Integer Sequences. Sequence a000108 (October 2011), Published electronically at http://oeis.org
30. Rozinat, A., van der Aalst, W.M.P.: Conformance Checking of Processes Based on Monitoring Real Behavior. Information Systems 33(1), 64–95 (2008)
31. Weidlich, M., Polyvyanyy, A., Desai, N., Mendling, J.: Process Compliance Measurement Based on Behavioural Profiles. In: Pernici, B. (ed.) CAiSE 2010. LNCS, vol. 6051, pp. 499–514. Springer, Heidelberg (2010)
32. Weijters, A.J.M.M., van der Aalst, W.M.P.: Rediscovering Workflow Models from Event-Based Data using Little Thumb. Integrated Computer-Aided Engineering 10(2), 151–162 (2003)
33. van der Werf, J.M.E.M., van Dongen, B.F., Hurkens, C.A.J., Serebrenik, A.: Process Discovery using Integer Linear Programming. Fundamenta Informaticae 94, 387–412 (2010)
34. Weske, M.: Business Process Management: Concepts, Languages, Architectures. Springer, Berlin (2007)

MailOfMine – Analyzing Mail Messages for Mining Artful Collaborative Processes*

Claudio Di Ciccio[1], Massimo Mecella[1],
Monica Scannapieco[2], Diego Zardetto[2], and Tiziana Catarci[1]

[1] SAPIENZA – Università di Roma
Dipartimento di Ingegneria Informatica, Automatica e Gestionale Antonio Ruberti
Via Ariosto 25, Roma, Italy
{cdc,mecella,catarci}@dis.uniroma1.it
[2] Istituto Nazionale di Statistica
Via Balbo 16, Roma, Italy
{scannapi,zardetto}@istat.it

Abstract. Artful processes are informal processes typically carried out by those people whose work is mental rather than physical (managers, professors, researchers, engineers, etc.), the so called "knowledge workers". In this paper we propose the MailOfMine approach, to automatically build, on top of a collection of email messages, a set of workflow models that represent the artful processes laying behind the knowledge workers activities.

Keywords: process mining, email analysis, object matching, visual representation of processes, knowledge workers, artful processes, declarative workflows.

1 Introduction

For a long time, formal business processes (e.g., the ones of public administrations, of insurance/financial institutions, etc.) have been the main subject of workflow related research. Informal processes, a.k.a. "artful processes", are conversely carried out by those people whose work is mental rather than physical (managers, professors, researchers, engineers, etc.), the so called "knowledge workers" [1]. With their skills, experience and knowledge, they are used to perform difficult tasks, which require complex, rapid decisions among multiple possible strategies, in order to fulfill specific goals. In contrast to business processes that are formal and standardized, often informal processes are not even written down, let alone defined formally, and can vary from person to person even when those involved are pursuing the same objective. Knowledge workers create informal processes "on the fly" to cope with many of the situations that arise in

[*] This work has been partly supported by Sapienza – Università di Roma through the grants FARI 2010 and TESTMED, and by the EU Commission through the FP7 project Smart Vortex.

K. Aberer, E. Damiani, and T. Dillon (Eds.): SIMPDA 2011, LNBIP 116, pp. 55–81, 2012.

their daily work. Though informal processes are frequently repeated, they are not exactly reproducible even by their originators – since they are not written down – and can not be easily shared either. Their outcomes and information are exchanged very often by means of email conversations, which are a fast, reliable, permanent way of keeping track of the activities that they fulfill. Understanding artful processes involving knowledge workers is becoming crucial in many scenarios. Here we mention some of them:

- *personal information management (PIM)*, i.e., how to organize one's own activities, contacts, etc. through the use of software on laptops and smart devices (iPhones/iPads, smartphones, tablets). Here, inferring artful processes in which a person is involved allows the system to be proactive and thus drive the user through its own tasks (on the basis of the past) [1,2];
- *information warfare*, especially in supporting anti-crime intelligence agencies: let us suppose that a government bureau is able to access the email account of a suspected person. People planning a crime or an act out of law are used to speak a language of their own to express duties and next moves, where meanings may not match with the common sense. Although, a system might build the processes that lay behind their communications anyway, exposing the activities and the role of the actors. At that point, translating the sense of misused words becomes an easier task for investigators, and allows to infer the criminal activities of the person(s) under suspicion;
- *enterprise engineering*: in design and engineering, it is important to preserve more than just the actual documents making up the product data. Preserving the "soft knowledge" of the overall process (the so-called product life-cycle) is of critical importance for knowledge-heavy industries. Hence, the idea here is to take to the future not only the designs, but also the knowledge about processes, decision making, and people involved [3,4,5].

The objective of the approach, proposed here, is to automatically build a set of workflow models that represent the artful processes laying behind the knowledge workers activities, on top of a collection of email messages. There are many advantages out of this work. First of all, the unspecified agile processes that are autonomously used become formalized: since such models are not defined *a priori* by experts but rather inferred from real-life scenarios that actually took place, they are guaranteed to respect the true executions (often Business Process Management tools are used to show the discrepancy between the supposed and the concrete workflows). Moreover, such models can be shared, compared, preserved, so that the best practices might be put in evidence from the community of knowledge workers, to the whole business benefit. Finally, an analysis over such processes can be done, so that bottlenecks and delays in actual executions can be found out.

The approach we want to pursue involves many research fields at a time, each one concerning a phase of the overall processing. We make use of *object matching* algorithms to obtain clusters of related email conversations. Every cluster is subsequently treated by *text mining information extraction* procedures, in order to find out which tasks email messages are about. *Process mining* is used to abstract process models representing the workflows, which the sets of subsumed tasks were considered traces of.

Our approach is named MailOfMine.

Here we also discuss how we addressed the challenge of showing artful processes to users, that are knowledge workers mainly, through a graphical interface which is flexible, functional and easy to understand. It is designed to be validated and further improved in collaboration with them. A user-centered methodology is going to be applied, not only to the development of the graphical interface, but also to the specification of the visual notation that describes the processes.

The remainder of the paper is organized as follows. Section 2 describes the adopted process model, and Section 3 presents the approach. Section 4 focuses on the techniques adopted for email clustering and their validation. Section 5 presents the visual notation for artful processes and the proposed user interface of the tool. Section 6 discusses and compares the relevant related work. Finally, Section 7 concludes the paper and outlines the future activities.

2 The Process Model

In this section, we present the process model adopted for describing the mined artful processes. After a clarifying example, we will discuss technical and theoretical implications of our assumptions.

2.1 Definitions

Definition 1 (Actor). *An* actor *is the subject directly or indirectly taking part in the progress of a work. She is called* contributor *when her collaboration is either proven by the participation to the collaborative communications or by the production of outcomes. Otherwise, she is named* spectator*.*

The spectator is the person who receives messages but does not reply, i.e., is among the recipients but is never a sender during the discussion. Nonetheless, her presence is perhaps fundamental, since she is a stakeholder, or a manager controlling the thread though having no reason to intervene ("no news, good news").

Definition 2 (Task). *A* task *is an elementary unit of work. Each task is connected to* (i) *its expected* duration*,* (ii) *zero or more* outcome*s, and* (iii) *one or more* actor*s. A task is called* productive *when linked to one or more outcomes. Otherwise, it is named* clarifying*.*

Clarifying tasks are not less important than productive ones. Often they are crucial to define, e.g., aim and methodology of the further steps. The outcome is whatever adds a bit of information to the exchanged knowledge among actors. In our context we simplify the definition by considering *document oriented* outcomes, i.e., we consider as new products of the tasks only those documents which are attached to the email body.

Definition 3 (Activity). *An* activity *is a collection of tasks or (recursively) other activities.*

Definition 4 (Key Part). *A* key part *is each* unique *piece of text belonging to the email messages exchanged in a communication. Thus, given a collection of duplicated pieces of text (coming from the same or different email messages in the thread), just a single representative is selected as key part.*

For instance, HTML signatures used by the sender of the email, quotations of previous email messages used in replies, etc., are all examples of redundant information that may appear in a communication thread, but will be filtered out by means of the *key part* concept. On the other hand, any piece of text not appearing in any other email message in the discussion thread will be interpreted as *key part*.

Definition 5 (Indicium). *An* Indicium *is any communication thread, or part of it, attesting the execution of a task, an activity, or a process instance.*

Indicia are key parts sets (or sets of key parts sets) containing any evidence that a task, or an activity, or a process instance, has been performed or is being performed.

Definition 6 (Process Scheme (Process) and Process Describing Grammar ($\mathcal{PDG}$)). *A* process scheme *(or* process *for short) is a semi-structured set of activities, where the semi-structuring connective tissue is represented by the set of constraints stating the interleaving rules among activities or tasks. Such a set of constraints, formulated on top of activities and tasks, is named* Process Describing Grammar ($\mathcal{PDG}$)*.*

Constraints do not force the tasks to follow a tight sequence, but rather leave them the flexibility to follow different paths, to terminate the enactment, though respecting a set of rules that avoid illegal or non-consistent states in the execution.

2.2 On the Process Describing Grammar

Let us consider every possible task [1] in a process as a character of an alphabet. Activities are thus structured actions which are either allowed or not to be executed in a process, as characters may or may not appear inside a string. The execution of some activities can imply others to be implied, at any point in time, as well as production rules enable the modification of the string expression in case a given sequence of characters is reached.

In this paper, we argue that each constraint useful to define declarative models for artful processes is expressible through regular grammars [6]. Regular grammars are recognizable through Finite State Automata (FSA) [7] (either deterministic or non-deterministic [8]). Constraints on the process always hold and must

[1] In the following, we focus on task identifiers, omitting, for sake of readability, details about duration, actors and outcomes. Hence, we identify tasks by their names only.

be respected altogether at each point in time. This means that the intersection of all the constraints must always hold. In other words, the FSA recognizing the correct traces for processes (i.e., accepting the valid strings) is the intersection of all the FSAs composing set of constraints (it is known that such grammars are closed to the operation of intersection [9]). Thus, we consider each task as a terminal character in the $\mathcal{PDG}$. Each constraint on tasks is a $\mathcal{PDG}$ itself. An activity is thus a $\mathcal{PDG}$ built as the intersection of all of the constraints' $\mathcal{PDG}$s.

In order to define the process scheme, we shift to regular expressions, which are an equivalent way to describe regular grammars (and accepting FSAs as well). Regular expressions are closed to their operators, thus we are able to recursively define activities as the intersection of constraints on activities. The process scheme, in turn, is the intersection of constraints on activities.

We finally summarize the concepts introduced so far, from a hierarchical recursive point of view. Tasks are terminal characters, building blocks of constraints on tasks. Constraints are regular expressions, equivalent to regular grammars. Activities are regular expressions which are equivalent to the intersection of the constraints' regular grammars. Constraints can be formulated on top of activities, being regular expressions themselves. The process scheme is the intersection of constraints defined on top of activities.

Being the regular grammars expressible by means of regular expressions, we can consider each constraint as a regular expression (corresponding to an accepting FSA). Moving to this domain, we can compose more complex regular expressions as if each constraint was a building block and thus compose them.

Here, we adopt the Declare [10] taxonomy of constraints as the basic language for defining artful processes in a declarative way. But whereas in Declare constraints are translated into LTL formulas, we express each constraint through regular expressions, as shown in Table 1.

For sake of brevity, there we used the POSIX standard shortcuts. Therefore, in addition to the known Kleene star (*), concatenation and alternation (|) operators, we make use here of *(i)* the . and [^x] shortcuts for respectively matching any character in the alphabet, or any character but x, and *(ii)* the + and ? operators for respectively matching from one to many, or zero to one, occurrences of the preceding expression.

The $Existence(m, a)$ constraint imposes a to appear at least m times in the trace. The $Absence(n, a)$ constraint holds if a occurs at most $n - 1$ times in the trace. $Init(a)$ makes each trace start with a. $RespondedExistence(a, b)$ holds if, whenever a is read, b was already read or is going to be read (i.e., no matter if before or afterwards). Instead, $Reponse(a, b)$ enforces it by forcing a b to appear after a, if a was read. $Precedence(a, b)$ forces b to occur after a as well, but the condition to be verified is that b was read - namely, you can not have any b if you did not read an a before. $AlternateResponse(a, b)$ and $AlternatePrecedence(a, b)$ both strengthen respectively $Response(a, b)$ and $Precedence(a, b)$. The "alternation" is in that you can not have two a (b) in a row before b (after a). $ChainResponse(a, b)$ and $ChainPrecedence(a, b)$, in turn, specialize $AlternateResponse(a, b)$ and

Table 1. Semantics of Declare constraints as regular expressions

Constraint	Regular expression	Example
Existence constraints		
$Existence(m, a)$	`[^a]*(a[^a]*){m,}[^a]*`	bc**aa**c for $m = 2$
$Absence(n, a)$	`[^a]*(a[^a]*){0,n}[^a]*`	bc**aa**c for $n = 3$
$Init(a)$	`a.*`	**a**ccbbbaba
Relation constraints		
$RespondedExistence(a, b)$	`[^a]*((a.*b)|(b.*a))*[^a]*`	**b**c**a**accbbbaba
$Response(a, b)$	`[^a]*(a.*b)*[^a]*`	bc**a**acc**b**bb**ab**
$AlternateResponse(a, b)$	`[^a]*(a[^a]*b)*[^a]*`	bc**a**cc**b**bb**ab**
$ChainResponse(a, b)$	`[^a]*(ab[^a^b]*)*[^a]*`	bc**ab**bb**ab**
$Precedence(a, b)$	`[^b]*(a.*b)*[^b]*`	c**a**acc**b**bbaba
$AlternatePrecedence(a, b)$	`[^b]*(a[^b]*b)*[^b]*`	ca**a**cc**bab**a
$ChainPrecedence(a, b)$	`[^b]*(ab[^a^b]*)*[^b]*`	c**abab**a
$CoExistence(a, b)$	`[^a^b]*((a.*b)|(b.*a))*[^a^b]*`	**b**c**a**ccbbbaba
$Succession(a, b)$	`[^a^b]*(a.*b)*[^a^b]*`	c**a**acc**b**bbab
$AlternateSuccession(a, b)$	`[^a^b]*(a[^a^b]*b)*[^a^b]*`	c**a**cc**bab**
$ChainSuccession(a, b)$	`[^a^b]*(ab[^a^b]*)*[^a^b]*`	c**abab**
Negative relation constraints		
$NotChainSuccession(a, b)$	`[^a]*(a[^a^b])*[^a]*`	bca**a**cc**b**bbba
$NotSuccession(a, b)$	`[^a]*(a[^b]*)*[^a^b]*`	bc**a**acca
$NotCoExistence(a, b)$	`[^a^b]*((a[^b]*)|(b[^a]*))?`	caacca

Table 2. Additional constraints in MailOfMine

Constraint	Regular expression	Example
Existence constraints		
$Participation(a) \equiv Existence(1, a)$	`[^a]*(a[^a]*)+[^a]*`	bc**a**ac
$Unique(a) \equiv Absence(2, a)$	`[^a]*(a)?[^a]*`	bc**a**c
$End(a)$	`.*a`	bcaaccbbbab**a**

$AlternatePrecedence(a, b)$, both declaring that no other symbol can occur between a and b. The difference between the two is in that the former is verified for each occurrence of a (b must immediately follow), the latter for each occurrence of b (a must immediately precede). $CoExistence(a, b)$ holds if both $RespondedExistence(a, b)$ and $RespondedExistence(b, a)$ hold. $Succession(a, b)$ is valid if $Response(a, b)$ and $Precedence(a, b)$ are verified. The same holds with $AlternateSuccession(a, b)$, equivalent to the conjunction of $AlternateResponse(a, b)$ and $AlternatePrecedence(a, b)$, and with $ChainSuccession(a, b)$, w.r.t. $ChainResponse(a, b)$ and $ChainPrecedence(a, b)$. $NotChainSuccession(a, b)$ expresses the impossibility for b to occur immediately after a. $NotSuccession(a, b)$ generalizes the previous by imposing that, if a is read, no other b can be read until the end of the trace. $NotCoExistence(a, b)$ is even more restrictive: if a appears, not any b can be in the same trace.

In addition to the constraints above, we also consider the ones described in Table 2. Taking inspiration from the relational data model cardinality constraints, we call *(i)* $\mathit{Participation}(a)$ the $\mathit{Existence}(m, a)$ constraint for $m = 1$ and *(ii)* $\mathit{Uniqueness}(a)$ the $\mathit{Absence}(n, a)$ constraint for $n = 2$, since the former states that a must appear at least once in the trace, whereas the latter causes a to occur no more than once. $\mathit{End}(a)$ is the dual of $\mathit{Init}(a)$, in the sense that it constrain each string to end with a. Beware that $\mathit{End}(a)$ would be clueless in a LTL interpretation, since LTL is thought to express temporal logic formulae over infinite traces. On the contrary, it makes perfectly sense to have a concluding task for a finite process, expressed by means of a regular automaton.

We recall here that the graphical syntax proposed for these constraints is presented in Section 5.

2.3 An Example

Let us suppose to have an email archive, containing various process instances indicia, and to focus specifically on the planning of a new meeting for a research project. We suppose to execute the overall technique explained in Section 3, and we report the possible result, starting from the list of tasks in activities (Process Description 1).

Process Description 1. Activities and tasks list

```
Activity:   ⟨Flight⟩
  Task:     boFl ("bookFlight"): Productive
    Actors:   {You: Contributor, FlightCompany: Contributor}
    Duration: 2 hrs.
Activity:   ⟨Agenda⟩
  Task:     prAg ("proposeAgenda"): Productive
    Actors:   {You: Contributor, Community: Spectator}
    Duration: 4 dd.
  Task:     rqAg ("requestAgenda"): Clarifying
    Actors:   {Participant: Contributor, Community: Spectator}
    Duration: ⊥
  Task:     coAg ("commentAgenda"): Clarifying
    Actors:   {Participant: Contributor, Community: Spectator}
    Duration: ⊥
  Task:     cnAg ("confirmAgenda"): Productive
    Actors:   {You: Contributor, Community: Spectator}
    Duration: 2 dd.
Activity:   ⟨Accommodation⟩
  Task:     esHo ("establishHotel"): Productive
    Actors:   {Organizer: Contributor, Community: Spectator}
    Duration: ⊥
  Task:     boHo ("bookHotel"): Productive
    Actors:   {You: Contributor, Hotel: Contributor}
    Duration: 2 dd.
```

The "bookFlight" task is supposed to be retrieved by confirmation email messages received by the user when the booking is completed. The $\langle \mathit{Flight} \rangle$ activity is composed by this task only. $\langle \mathit{Agenda} \rangle$ is supposed to be slightly more complex instead. We suppose that a final agenda will be committed ("confirmAgenda") after that requests for a new proposal ("requestAgenda"), proposals themselves ("proposeAgenda") and comments ("commentAgenda") have been circulated in the group (*Community*, as identified in the Actors list). Finally, $\langle Accommodation \rangle$ is the activity related to the reservation of rooms, in a hotel located where the meeting will take place (or nearby). Beyond the confirmation email messages received by the user when the booking is completed (indicia for the "bookHotel" task), the "establishHotel" stems from the email messages informing about the hotel arranged by the meeting organizers.

The reader may notice that sometimes You appears among the Actors. We suppose it to be the special identifier adopted whenever the owner of the email archive is a Contributor. In case she is only a Spectator (see, e.g., "requestAgenda"), such an information is not reported, since it is redundant (if the owner were not among the recipients, there should have been no way to access email messages related to that task). The usage of $\perp$ as Duration stands for an unknown value: whenever there is only one indicium related to a task (i.e., only one email message proving its execution, for each activity indicium), no inference about Duration can be performed.

The aforementioned tasks and activities are bound to the following constraints. In order to express them, we use the base set introduced before in this Section, starting with the *existence* constraints of Process Description 2.

Process Description 2. Existence constraints on the example tasks and activities

Activity:	$\langle Flight \rangle$: [0, $*$]	
Task:	$boFl$: [1, $*$]	i.e., $\{Participation(boFl)\}$
Activity:	$\langle Agenda \rangle$: [1, 1]	i.e., $\{Participation(\langle Agenda \rangle), Uniqueness(\langle Agenda \rangle)\}$
Task:	$prAg$: [0, $*$]	
Task:	$rqAg$: [0, $*$]	
Task:	$coAg$: [0, $*$]	
	$Init(coAg)$	
Task:	$cnAg$: [1, 1]	i.e., $\{Participation(cnAg), Uniqueness(cnAg)\}$
Activity:	$\langle Accommodation \rangle$: [0, 1]	i.e., $\{Uniqueness(\langle Accommodation \rangle)\}$
Task:	$esHo$: [0, 1]	i.e., $\{Uniqueness(esHo)\}$
Task:	$boHo$: [1, 1]	i.e., $\{Participation(cnAg), Uniqueness(boHo))\}$

In Process Description 3 we report the relation constraints holding in this example process.

Process Description 3. Relation constraints on the example tasks and activities

Activity: $\langle Agenda \rangle$
$Response(rqAg,\ prAg)$
$RespondedExistence(coAg,\ prAg)$
$Succession(prAg,\ cnAg)$
$CoExistence(\langle Flight \rangle,\ \langle Accommodation \rangle)$
$RespondedExistence(\langle Agenda \rangle,\ \langle Accommodation \rangle)$

3 The MailOfMine Approach

The MailOfMine approach (and the tool we are currently developing) adopts a modular architecture, the components of which allow to incrementally refine the mining process, as in Figure 1. We describe it into three parts: *(i)* the preliminary steps, from the retrieval of email messages to the reconstruction of communication threads; *(ii)* the extraction of key parts, the activities and tasks indicia detection, and the tasks definition; *(iii)* the final steps, from the activities definition to the final mined process extraction.

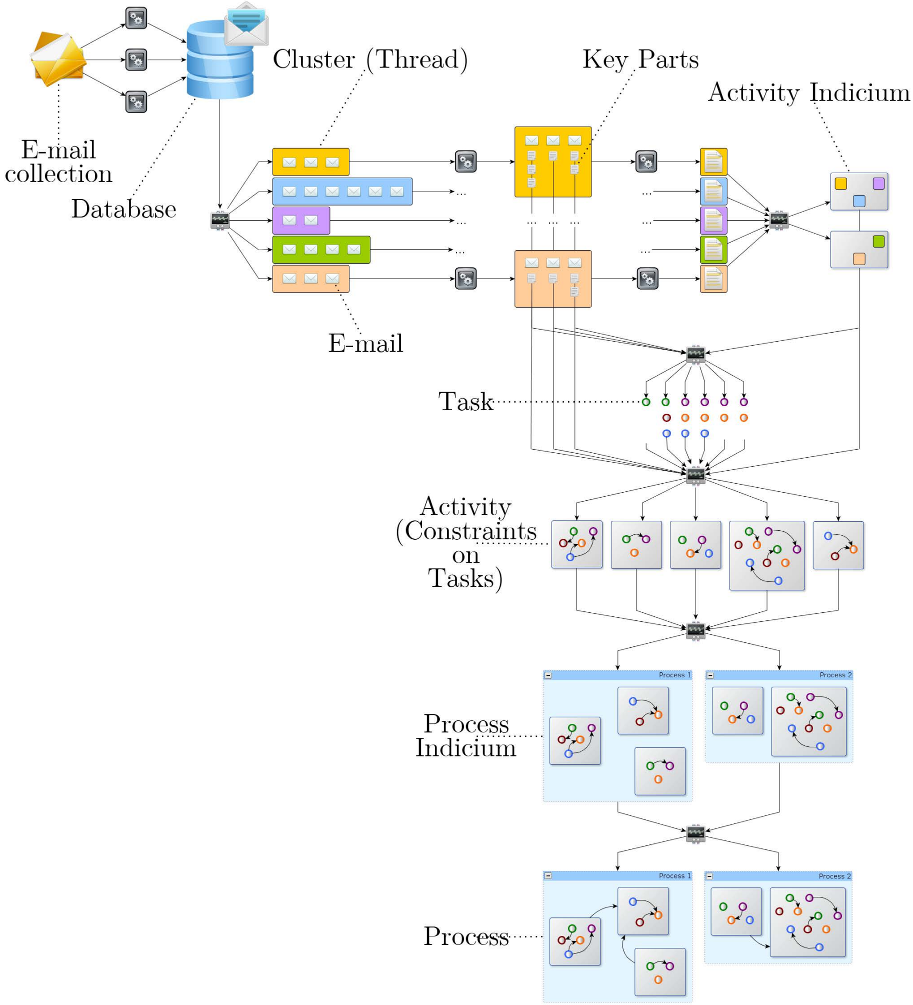


Fig. 1. The MailOfMine approach

Section 5 describes the design rationale for the graphical representation of the mined process.

3.1 Preliminary Steps and Email Clustering

First of all, we need to extract email messages out of the given archive(s). Archives are compliant to different standards, according to the email client in use, hence the component for reading the messages is intended to be plug-in based: each plug-in is a component able to access multiple standalone applications' archive formats (e.g., Mozilla Thunderbird, Microsoft Outlook, etc.) and/or online providers (e.g., Gmail, Yahoo! Mail, etc.). The outcome is the population of a database, on the basis of which all the subsequent steps are carried out.

The first of them is the clustering of retrieved messages into extended communication threads, i.e., flows of messages which are related to each other. The technique that is used to guess such a connection is based not only on the Subject field (e.g., looking at "Fwd:" or "Re:" prefixes) and the SMTP headers (e.g., reading the "In-Reply-To" field), but on the application of a more complex object matching decision method. Such indicators, though likely trustworthy, might be misleading: *(i)* in the everyday life, it is a common attitude to reply to old messages for talking about new topics, which may have nothing to do with the previous one; *(ii)* conversely, it may happen that a new message is related to a previous, though written and sent as if it were a new one. The technique is detailed further in Section 4.

Once the communication threads are recognized, we can assume them all as activity indicia candidates.

After the clustering phase, messages are analyzed in order to identify key parts. I.e., email messages are cleaned up from signatures and quotations citing some text already written in another message within the thread, by the usage of a combination of the techniques described in [11] and [12]. The key parts are the text remaining in bodies and subjects once such filtering operation is performed.

3.2 Identifying Activities

Once messages and key parts in threads are gathered, MailOfMine can build activity indicia as the concatenation of all the key parts (artificial text). Then, the clustering algorithm is used again, this time to identify the matches between activity indicia. E.g., let us suppose to have *(i)* a thread T_{del41} related to the writing of Deliverable 4.1 for a research project, *(ii)* another thread T_{del52} related to the writing of Deliverable 5.2 for the same research project, and, finally, *(iii)* a T_{air} dealing with an airplane flight booking, for the next review meeting. Thus, the algorithm is expected to cluster as follows: $\{T_{del41}, T_{del52}\}, \{T_{air}\}$. Its output represents the set of activities.

By taking into account the set of activities and the key parts (task indicia candidates), the clustering algorithm checks for matching key parts, identifying them as tasks. E.g., let us suppose to have *(i)* a key part $k_{airData}$ specifying the booked airplane data, *(ii)* another key part $k_{airBook}$ containing the confirmation of the booking, and, finally, *(iii)* a k_{del41}: "Please find attached Deliverable 4.1". Thus, the algorithm is expected to cluster as follows: $\{k_{airData}, k_{airBook}\}, \{k_{del41}\}$.

Hence, each set is a task indicium for one task. The analysis, though, does not terminate here. In order to understand whether the task under investigation is clarifying or productive, MailOfMine checks *(i)* if any document-oriented outcome in the original email messages was attached; *(ii)* if key parts contain Speech Acts [13], according to the technique proposed by [14]. If at least one of the listed conditions holds, it is productive. Otherwise, it is considered as clarifying. The detection of Speech Acts is supposed to be assisted by the experts, who are required to provide a dictionary of keywords of their domain field, as in [14]. For further information on the relation between Speech Acts and workflows, see [15]. The task name will be given as a rule of thumb by the conjunction of verbs and dictionary keywords composing the Speech Acts, although the user will be able to customize it at her wish.

3.3 Mining the Process

For each thread (activity indicium), tasks are put in sequences respecting the ordering of the task indicia (key parts with a Speech Act), given by the timestamp of the email they belong to. Tasks appear in many traces then, being the indicia potentially found in different threads. Thus, threads with email messages are turned into traces of enacted tasks. MailOfMine searches for execution constraints between tasks, stemmed from these traces, though an algorithm used in purpose, named MINERful (presented in [16] and detailed in [17]). The intersection of the mined constraints on tasks constitutes the $\mathcal{PDG}$ of activities.

Once activities and tasks are recognized, a supervised learning process takes place, in order to cluster activities into processes. This step cannot be fully performed by the system, since no linguistic connection could exist among related activities. E.g., the activity of drawing slides and the reservation of the airplane could sound completely apart, though those slides could be the material to present at a review meeting, hold in the city that the airplane was booked to reach.

The activity grammar would not be exposed to the user, of course. Instead, the threads are shown as witnesses for the activity; hence, the user associates part of them to different processes, and the learner associates all the related activities to processes. The choice of the name to be assigned to activities and processes is left to the user.

Once processes are identified, MailOfMine performs the second step for the construction of the $\mathcal{PDG}$, i.e., the mining of production rules among activities inside the same proces. For each activity indicium (thread), a timeframe is defined as the window between the timestamps of the first and the last email in the thread. Thus, traces are composed by ordering activity indicia belonging to the same process, on the basis of their timeframe. These traces are the input for MINERful to run again.

4 Email Clustering

In this section we describe how we tackle the problem of performing a Similarity Clustering (SC) of email messages. The purpose of such a clustering step is to identify groups of related email messages to be used for finding tasks that compose process activities. To face the SC problem, we exploit a previously proposed Object Matching (OM) algorithm [18], providing ad-hoc extensions for the task at hand. Here we briefly illustrate the rationale for adopting an OM algorithm to cluster email messages. From an abstract point of view, OM and SC share the common objective of classifying data-object pairs according to an hidden (i.e. not self-evident) property; for OM, the relevant hidden property to be discovered is if two objects *"do represent – or not – the same real-world entity"*, whereas, for SC, one needs to asses if two objects *"do belong – or not – to the same group"*. Moreover, in order to *infer* the class the pairs belong to, both OM and SC rely on pairwise distance (or, equivalently, similarity) measures. With respect to the choice of the specific OM algorithm [18], we point out that it relies on its fully automated nature and on its independence on the specific data object representation. Lastly, we remark that the statistical properties at the basis of the chosen algorithm are perfectly valid for the SC problem.

We decided to adopt an Object Matching technique, rather than classical Clustering, due to the fact that it extends the Record Linkage field (see Section 6). Coming from the database domain, RL produced dedicated frameworks and tools to cope with textual issues, over the years. For instance, one of their mandatory requirements was to be resilient to typos. Clustering techniques are focused on the similarity grouping of numerical entities, such as points or vectors in N-dimensional spaces, instead. Therefore, we exploit an algorithmic machinery developed to deal with text-related challenges.

We implemented our Similarity Clustering as a four steps procedure:

1. in the first step, we chose a representation format for email messages, in order to translate them into processable data objects;
2. the second step consisted of the definition of a specific distance metric to compare email objects;
3. as a third step, we ran a decision algorithm whose output was a set of email pairs declared as "matches", i.e. – in the present context – belonging to the same cluster;
4. the fourth and last step implied the application of a function performing a transitive closure among "match" pairs in order to build email clusters.

This procedure is detailed in the following together with some test results.

4.1 Email Object Representation and Distance Metric Definition

In order to compare email objects we first defined a representation format for each email message. Specifically, we considered each email as a record consisting of: *(i)* some header fields, *(ii)* the body and *(iii)* the names of attached files (if present).

We observe that, according to RFC 5322 and RFC 3864 (http://www.ietf.org/rfc.html), each message has exactly one header, which is structured into fields. Some of these fields are mandatory (such as the *message_id*), others are instead optional but indicated as common. Among such common header fields, we took into account the ones listed in the following; for each of them we also specify the chosen representation format:

- emailIdentifier: this is a string associated to each email that permits its unique identification.
- Sender: email address of the sender represented as a string.
- Receivers: email addresses represented as strings and concatenated into a unique string.
- Subject: represented as one single string.

Having as objective to build clusters consisting of email exchanged to accomplish a specific task, we found reasonable to merge Sender and Receivers fields into one single field SenderReceivers. This is because the clusters we want to determine are, in a sense, *invariant* with respect to the email exchange direction, as they are instead focused on the actions performed by means of such email exchanges.

As far as the body, we chose to consider it as a piece of free text and we represented it as one single string.

Finally, email attachments were taken into account by considering their names. More specifically, we represented attachments by a unique string, AttachmentsNames, resulting from the concatenation of the names of all the present attachment files [2].

On the basis of the representation choices described above, all the email fields but the body could be compared by means of traditional string metric distances (see [19] for a survey). In our experiments, we used the Levenshtein distance for these fields.

The body is an exception as it is a piece of free text rather than a (more or less) *structured* text field like the others[3]. The set of the bodies in an email collection is considered as a corpus of documents. As a first step, each body is regarded as a bag of words ("terms"). Then, we build a Term-Document matrix. To each term of a body a weight was assigned, based on the traditional TF-IDF metric [20]. Each body is in this way represented by a vector of a vector space and hence easily comparable to the other bodies. To such a scope, we use the cosine distance function. The distance so obtained will be referred as VSM distance (as it is based on the Vector Space Model [21]). TF-IDF was considered appropriate because of its ability to increase the importance of a term proportionally to the

[2] Given the disparate type of possible attachment files (e.g. video, images, etc.) a generalized approach that exploits attachment contents has not been considered as viable.

[3] Though the subject can contain more than one word, we decided to consider it too as a structured field. Indeed, since typically subjects are not long unstructured texts, text mining techniques are unnecessary, or even not suitable, for their comparison.

term frequency inside the document, while penalizing terms that are found to be very common in the corpus.

Our choices for email representation and the distance functions used for each field are summarized in Table 3, below.

Table 3. Email fields used for comparison and related distance functions

Field	**Distance**	**Field**	**Distance**
`Subject`	Levenshtein	`SenderReceivers`	Levenshtein
`AttachmentsNames`	Levenshtein	`Body`	VSM

4.2 Experimental Results

The testbed for our SC application consisted of two experiments. The first one worked on a small collection of 101 email messages, in order to be able to conduct a manual evaluation of the quality of the obtained clustering results. The second experiment worked instead on a larger collection of 1337 email messages, including the previous, in order to test the practical feasibility of our approach with a realistic mailbox dimension. The analyzed set of email messages represents the collection of messages exchanged during the development of a European research project that one of the authors was involved in. The smaller collection was the part of email messages sent or received in a given frame of the project lifetime.

In the first experiment, the 101 input email messages generated 5050 email-pairs. Our decision algorithm declared 98 pairs to belong to the same group. The transitive closure performed on the set of such 98 "matching" email-pairs determined 21 non overlapping and non trivial (that is containing at least 2 email messages) clusters: these were the final output produced by our SC algorithm. The output clusters size ranged from 2 to 11 email messages and the email messages involved in the clusters are found to be 68 out of 101.

To assess the quality of the obtained result, we manually analyzed the input email messages and the output clusters: we identified only 1 false positive cluster, whereas no false negatives were found. The false positive cluster involved two email messages, which happened to have the same sender and receivers, and almost completely overlapping short bodies. The very encouraging information is that no false negative was found, thus it seems there is no need to re-iterate it over times to recover from a possible information loss.

Out of the 893116 pairs generated by the 1337 input email messages of the second test, our decision algorithm declared 2974 email-pairs to belong to the same group. The transitive closure performed on the set of such 2974 "matching" email pairs determined 230 different clusters. The output clusters size ranged from 2 to 39 email with a mean of 4.6, and the email messages involved in the clusters were found to be 1060 out of the original 1337.

Given the size of our second experiment, building a "gold-standard" clustering solution by manual inspection was clearly unmanageable. Therefore, we were not able to evaluate the quality of our clustering results *directly*, that is in terms of false positives and false negatives. As an alternative, we rather tried to get a feeling of the goodness of our reconstructed email clusters as a mean for identifying tasks. To this end, we first asked the owner of the mailbox archive to provide us a small user-generated dictionary of relevant and common terms in the research-projects domain (like, e.g., "deliverable", "workpackage", "meeting", ...); the dictionary we received had 56 entries. Next, we treated the bodies of the email messages belonging to each identified cluster as a corpus of documents and assigned to each word inside such corpora its TF-IDF weight. Then we selected, for each cluster, the top-30 TF-IDF weighted words as representatives of the related e-mail conversation. We manually stemmed them to singular nouns and infinitive verbs (e.g., "deliverables" was turned into "deliverable" as "asks" into "ask"). Lastly, we applied an intersection operation between each of the 30-word-sets per cluster and the 56 entries of the user-generated dictionary. The outcome of such an exercise seems quite encouraging, as 214 of the 230 clusters (i.e., 93%) turned out to be represented by at least one of the domain-specific dictionary words.

5 Process Visualization

The literature dealing with the representation of processes typically aims at visualizing the process all at once, by means of diagrams that show the complete grid of interconnections among activities. Here we propose a change in the viewpoint. We want to model artful processes as a collection of constraints, through the declarative approach. Being highly flexible, this kind of representation does not necessarily impose a pre-defined strict order on activities, neither explicit nor implicit. For instance, one can state that a task a implies the execution of another task b afterwards (see Section 2.3), with no specification provided if a is not performed, meaning that b can be done or not during the process instance run. In other words, the process schema itself can change according to the things that may have happened before. This is why we do not consider as the best suitable solution adopting a static graph-based global representation alone, on one hand: a local view should work better in conjunction with it. On the other hand, no knowledge worker is expected to be able to read and understand the process by reading the list of regular-expression based constraints: a graphical representation, easy to understand at a first glimpse, must be used.

The process schema and the running processes are respectively modeled through *(i)* a set of diagrams, representing constraints on workflows (static view: Section 5.1) and *(ii)* an interactive evolutionary graphical representation for the visualization of running instances (dynamic view: Section 5.2). Furthermore, we propose two complementary views on constraints: *(i)* a local view focusing on one activity at a time and *(ii)* a global view providing a bird-eye sketch of the whole process schema.

As activities are basically collections of tasks (thus, they can be single tasks too), which have to be compliant with the same constraints as tasks, in the following we will consider tasks only, for sake of simplicity. The same statements remain valid for activities, though. For sake of simplicity and readability of figures, we adopt one-character identifiers for tasks, though they are going to adhere in practice to the naming rules described in Section 3.2, as the graphical representation of artful processes presented here is intended to be the core of the user interface for the visualization of the mined processes in MailOfMine.

5.1 Process Schema

The Local View. It is very hard to show a process schema all at once and keep it easily readable, due to the high flexibility of the declarative representation. Thus, given that the declarative approach is based on constraints, we collect all of those related to every single task, i.e., where the task (e.g., e) is either *(i)* directly implied (e.g., if d is done, then e must be done), or *(ii)* directly implying (e.g., if e is done, no matter when, f was done before or must be done in the future). The *directly* adverb is used due to the need not to make things too much complicated and to follow the idea of having a local view only. For instance, the process is such that if d is done, then e must be performed; moreover, the enactment of c implies that d can not be done further. If we look at the constraints directly affecting e, the latter rule is not taken into account. In fact, if c is not performed, nothing is imposed on d. This is an example providing a hint on the rationale: for sake of readability, we want to avoid the confusion coming from too many cross-implications to consider at a time.

The representation of relation constraints is based on three main degrees of freedom, namely *(i)* time, *(ii)* implication, *(iii)* repeatability. The time is considered here as a discrete ordered set of steps the tasks can take place in. We ideally consider each task as spending a single unit in this conception of time. The notion of implication is a based on two values (implying, implied). The repeatability is given by the specification of one among four values, standing for the number of times a task can be consequently fulfilled: *(i)* zero, one or more times; *(ii)* zero, or one time; *(iii)* exactly once; *(iv)* zero times.

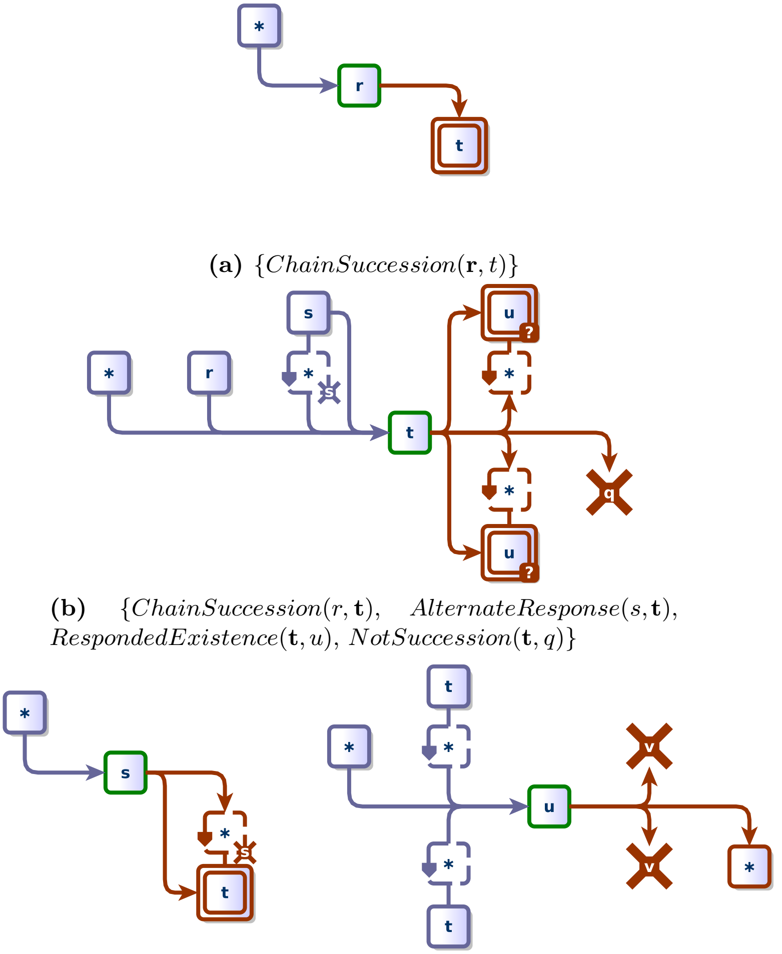
Our graphical notation represents time and implication as the coordinates of a bidimensional drawing, where time is on the ordinates. This ideal y axis divides the plane space into two separate regions: one for each value of the implication dimension (implying, implied), on the abscissae. The x axis divides the plane space into two regions: upwards, what can (or can not) happen *before* the task is executed, and, downwards, what can (or can not) happen after. On the origin of this chart, inspired to the cartesian coordinate system, we put the task under examination. The y axis is oriented towards the bottom, in order to follow the reading directionality. For the same reason, the implication relation order flows from the left to the right. Of course, the orientation of the axes can change according to the localization of the software running: e.g., users from Arabic countries might prefer a mirrored version, where the implied tasks are on the left, the implying on the right.

The repeatability is expressed by the thickness of the boundaries around the boxes representing tasks: dashed for tasks that can be done zero, one or more times, solid for zero or one times, double-line for exactly one time. The task box turns into a cross shape when repeatability is zero. The repeatability is referred to the quadrant the box appears in. For instance, u must appear once either before or after e took place. We recall here that the scope of repeatability, as all of the other degrees of freedom, is not extended to the whole process instance existence, but only for what concerns the time surrounding the single task under analysis.

For sake of readability, we do not explicitly mention every possible task the process can be composed of, on the graph. Instead, we render only such tasks that are interested in focused constraints, so to address the potential problem of too many nodes and arcs connecting one another in a scarcely readable spaghetti-like diagram. Though, visualizing the tasks involved in constraints only, might look like a way to force the actor to execute nothing else than the ones that are shown. On the contrary, declarative models allow to do more: roughly speaking, what is not mentioned, is possible. Thus, we make use of a wildcard ($*$) not intended as "every task" in the usual all-comprehensive form, but in the typical human conception: "any task", where it is understood that the other rules remain valid (e.g., if it is stated that q can not be executed after t, a $*$ after t means "any task, except q"). Examples of the diagrams are in Figure 2. The constraints depicted here are described in Section 2.

The graphical notation is enforced by arrows, easing the user to go across the flow of tasks, from the implying before to the implied afterwards. Colors are used for sake of readability and comprehensibility, as additional arrows making a loop on zero-one-more-repeatable tasks, though the idea is that such diagrams must be kept easy to be sketched by a pen, as well.

Figure 2a shows the only constraint pertaining r in an imaginary process, namely $ChainSuccession(r, t)$. It states that immediately after (i.e., below, on the diagram) the implying r (on the center) you must (double line) execute t, as an implication (on the right). Nothing is directly told about r as an implied task in a constraint: thus, a $*$ wildcard is put on the left of r. Then, Figure 2c focuses on s, which is the implying task for the $AlternateResponse(s, t)$ constraint: i.e., if s is executed, you can either perform t (see the direct arc in the fork on the right of s) or optionally (dashed line) perform any other task ($*$) but s (put inside the cross at the right bottom corner of the dashed box) until you do t. t must be executed in any case: this is why the line bounding the box is double. Figure 2b details the constraints that concern t. On the right side of the box with t, the $RespondedExistence(t, u)$ and $NotSuccession(t, q)$ are drawn. The former causes the arcs to fork both upwards and downwards, due to the fact that the $RespondedExistence$ constraints do not specify whether the implied task must be done before or after the implying. The question mark put on the right bottom corner is used to enforce this concept of optionality, together with the double line recalling that the execution of t is bound to the execution of u – i.e., no matter if beforehand or afterwards, u must be enacted. The $NotSuccession(t, q)$ is represented by the cross that the q identifier is inscribed in, meaning that q can not be executed after (below) t. On the left of t the reader can see the aforementioned constraints having it as the implied (see



(a) $\{\mathit{ChainSuccession}(\mathbf{r}, t)\}$

(b) $\{\mathit{ChainSuccession}(r, \mathbf{t})$, $\mathit{AlternateResponse}(s, \mathbf{t})$, $\mathit{RespondedExistence}(\mathbf{t}, u)$, $\mathit{NotSuccession}(\mathbf{t}, q)\}$

(c) $\{\mathit{AlternateResponse}(\mathbf{s}, t)\}$

(d) $\{\mathit{RespondedExistence}(t, \mathbf{u})$, $\mathit{NotCoExistence}(\mathbf{u}, v)\}$

Fig. 2. The MailOfMine local static constraint diagrams

Figures 2a and 2c). Finally, Figure 2d suggests that, if u is performed, neither before (above) nor afterwards (below) you are allowed to do v. Anything else is admitted ($*$). This is the sense of $\mathit{NotCoExistence}(u, v)$, as drawn on the right of u. On the left, the constraint which u was involved in as an implied task, that is $\mathit{RespondedExistence}(t, u)$, is depicted (see Figure 2b).

The local view can focus on a possible sub-trace of executed tasks, as in Figure 3. The meaning of symbols is the same as before, although here the focus is moved on what could have happened (or is allowed to happen) if t is performed after u, whereas diagrams in Figure 2 consider the enactment of an only task at a time.

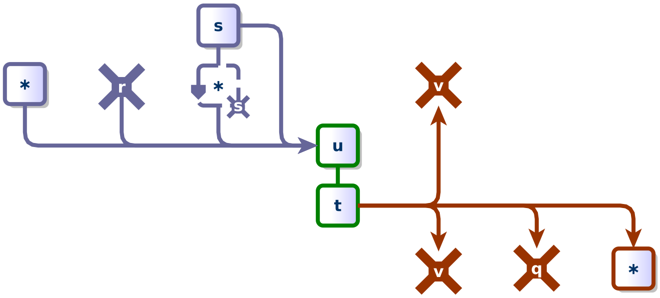
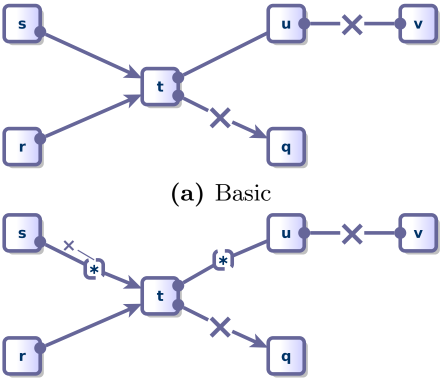


Fig. 3. The $< u, t >$ tasks subtrace constraints diagram

The Global View. The aim of the global view (Figure 4) is to show the relations between tasks, namely *(i)* whether the presence of one implies a further constraint (on the graph, a dot on the tail of an arrow, starting from the implying task and ending on the implied), *(ii)* which task must be performed after, between the implying and the implied, if known (on the graph, an arrow, put on the head or the tail), *(iii)* whether the presence of one implies the absence of another (a cross in the middle of the arrow), or not (no cross put upon). All of the previous information bits are independent of each other, hence all the possible combinations are allowed. This is the restricted basic graphical syntax used in Figure 4a. Indeed, it is not explicitly expressed how strong the constraint is (e.g., whether other tasks can be performed between the implying and the implied), in order to tidy the diagram up and provide a fast view of the overall process, without entering in details that are likely better explained through the local views: they can rely, in fact, on dimensions spread on axes the cartesian way, not as in graphs.



(a) Basic

(b) Extended

Fig. 4. The MailOfMine global static constraints diagram

Nonetheless, skilled users might want to have a complete vision of the constraints involved, even though it might result in a reduced readability, due to the unavoidable increase of graphical symbols to draw in the diagram. Thus, a richer graphical syntax is needed. Its design rationale is to extend the basic, though keeping coherence with *(i)* the visual language terms used and *(ii)* the graph structure. This allows the user to be required of a minimal cognitive effort in order to learn its semantics, on one hand, and lets her toggle between the basic and the extended view. Indeed, only arcs are loaded with new symbols, as depicted on Figure 4b: no additional shape nor any change in the graph topology are required.

Both diagrams in Figure 4 draw the same set of constraints that were locally represented by example in Figure 2.

The global view is inspired to the graphical syntax of [22], though its design focuses on the basic relations (before/after, implying/implied, existence/absence) between tasks in a binary constraint.

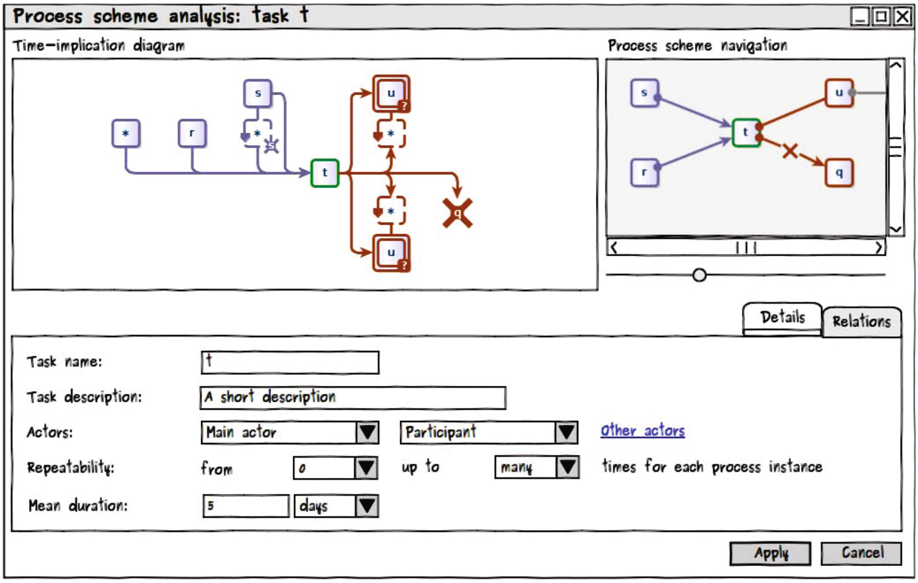
Coupling this diagram with the local view is useful for avoiding the misunderstanding that could arise by the usage of oriented graphs. Indeed, Finite State Automata, Petri Nets, State Transition Networks, UML Activity Diagrams, Flowcharts, and so forth, all share a common interpretation: roughly speaking, nodes are places to traverse one by one, following a path that respects the direction given by arrows along the arcs. Here, it is not the case: e.g., considering Figure 4a, one could intuitively suppose that, done s, the next task is t. It is not true: after s, r or u could be performed, even many times, and after some further passages finally t.

A GUI Sketch. Figure 5 draws a prototype of the window showing a local view, on the task t. The additional information regarding the cardinality of the task, as far as the actors involved and so forth, is located on the bottom of the window. The global view, put on the right, is used as a navigation tool on the process schema. Conversely, at any point in time it will be possible to activate the local view of a task selected on the global view screen, in order to freely switch from one to another.

5.2 Running Instances

A dynamic view is associated to the static process scheme, for the management of running instances. Such a view is designed to be interactive, i.e., to let the user play with the model, so to control the evolution of the running process. Moreover, she can better learn the constraints mechanism by looking at the process evolving. Indeed, it is based on the same visual notation provided for the visualization of constraints (see Figure 3), based in turn on local view diagrams. This choice is made in order to remark the user that global views do not explicitly express the evolution of the system over time, whereas local views do. Figure 6 depicts a sample evolution of a process instance.

From a starting step onwards, the user is asked to specify which the next task to perform is. At each step, the following tasks that can be enacted are shown,



Task name: t

Task description: A short description

Actors: Main actor ▼ Participant ▼ other actors

Repeatability: from 0 ▼ up to many ▼ times for each process instance

Mean duration: 5 days ▼

Fig. 5. The task details screen

by means of the same visual language used for static views. After one of them is fired, all the *possible* and *mandatory* following tasks are shown. And so forth. We recall here that the recognition of the possible initial tasks, as far as the evolution which follows, is a view on the current state of the FSA obtained as the intersection of all the FSA's expressing the constraints in the process scheme. Figure 7 is a prototype sketch.

6 Related Work

Text Mining, or *Knowledge Discovery from Text*, deals with the machine supported analysis of text: indeed, it refers generally to the process of extracting interesting information and knowledge from unstructured text. Though text mining covers many topics that are out of the scope of this paper (see [23,24] for comprehensive surveys), we list some techniques that are instead relevant to our work.

[25] proposes a method employing text mining techniques to analyze email messages collected at a customer center. Based on [26], this work shows how to cope with the information extraction in the context of email messages, for the construction of a key concept dictionary. Our aim is not restricted to the extraction of the key concept dictionary, but rather deals with the mining of activities performed on top of them; on the other side, in MailOfMine, we assume to rely on a user-provided dictionary of keywords, as described in Section 3.2. [14] introduces the usage of an ontology of email acts, i.e., Speech Acts [13], to classify messages.

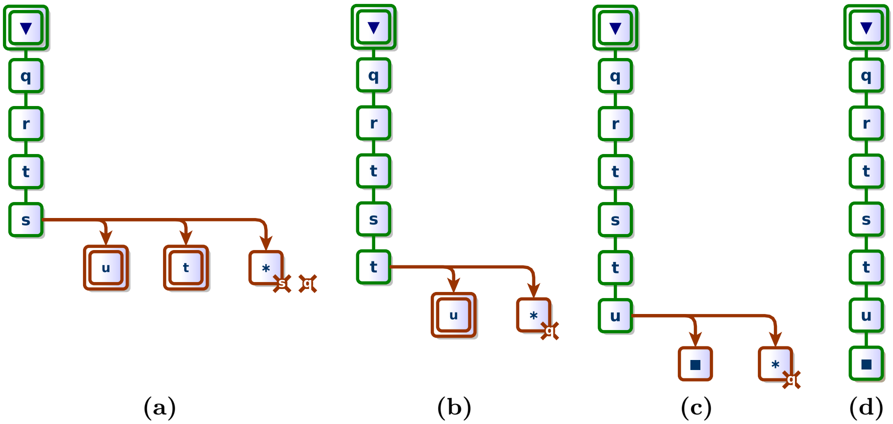
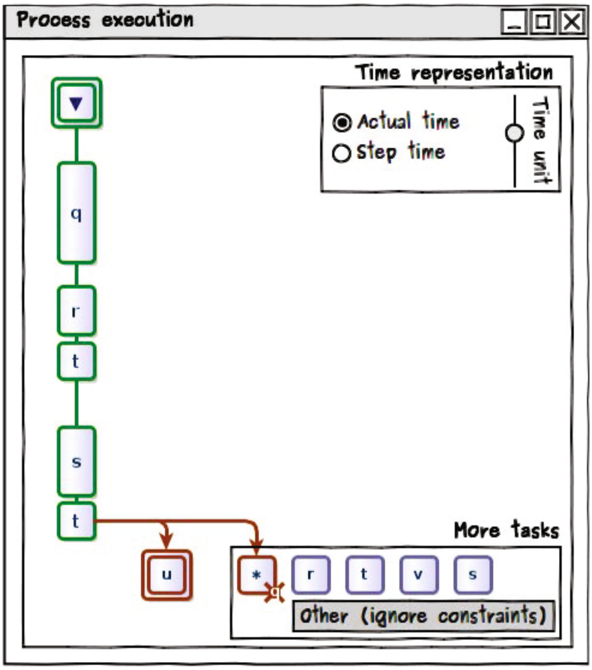


Fig. 6. The MailOfMineDynamic process view



● Actual time
○ Step time

Fig. 7. The process execution management window

Process Mining [27], a.k.a. *Workflow Mining* [28], is the set of techniques that allow the extraction of structured process descriptions, stemming from a set of recorded real executions. Such executions are intended to be stored in so called *event log*s, i.e., textual representations of a temporarily ordered linear sequence of tasks. There, each recorded *event* reports the execution of a *task* (i.e., a well-defined step in the workflow) in a *case* (i.e., a workflow instance). Events are always recorded sequentially, even though tasks could be executed in parallel: it is up to the algorithm to infer the actual structure of the workflow that they are traces of, identifying the causal dependencies between tasks (*condition*s). ProM [29] is one of the most used plug-in based software environment for implementing workflow mining techniques.

Most of the mainstream process mining tools model processes as Workflow Nets (WFNs – see [28]), explicitly designed to represent the control-flow dimension of a workflow. From [30] onwards, many techniques have been proposed, in order to address specific issues: pure algorithmic (e.g., α algorithm, drawn in [31] and its evolution α^{++} [32]), heuristic (e.g., [33]), genetic (e.g., [34]), etc. Indeed, heuristic and genetic algorithms have been introduced to cope with noise, which the pure algorithmic techniques were not able to manage. A very smart extension to the previous research work has been recently achieved by the two-steps algorithm proposed in [35].

EMailAnalyzer [36] is an integrated ProM plug-in for mining processes from email logs, that are XML files compatible with the ProM framework, built through the analysis of *(i)* senders and receivers, in order to guess the actors involved, and *(ii)* tags on email messages and subjects, for extracting the task. E-mail messages are extracted from an Outlook archive. Our approach shares similar ideas for the disambiguation of actors and sociograms, and aims at extending it by *(i)* building a plug-in based platform capable to retrieve email messages from multiple archives (not only Outlook), and *(ii)* extracting cases and relations among tasks from a more comprehensive analysis of email fields (headers, threads, body, attachments, etc.).

The need for flexibility in the definition of processes leads to an alternative to the classical “imperative”: the “declarative” approach. Rather than using a procedural language for expressing the allowed sequences of activities, it is based on the description of workflows through the usage of constraints: the idea is that every task can be performed, except what does not respect them. [22] shows how the declarative approach can help in obtaining a fair trade-off between flexibility in managing collaborative processes and support in controlling and assisting the enactment of workflows. Such constraints, in [37] (Chapter 6) are formulations of Linear Temporal Logic ([38] – Chapter 3). DecSerFlow [39] and ConDec [40] (now named Declare [41,42]) provide graphical representations for processes described through the declarative approach. Nonetheless, we believe that the declaration of collaborative workflows constraints can be expressed by means of regular expressions, rather than LTL formulae: regular expressions express finite languages (i.e., processes with finite traces, where the number of enacted tasks is limited). LTL formulae are thought to be used for verifying properties over semi-infinite runs instead. On the contrary, human processes have an end, other than a starting point.

The Declare framework [41] is a concrete implementation of a constraint-based process management system, supporting multiple declarative languages. Declare Miner is the plug-in for mining declarative processes from within the ProM framework. [43] described the usage of Inductive Logic Programming Techniques to mine models expressed as a $\mathcal{S}$CIFF ([44]) theory, finally translated to the ConDec notation.

7 Conclusions

We are currently in the process of realizing the various techniques into a working prototype and then validating it over a large email messages collection.

Up to now, once analyzed in depth the current literature in order to figure out how to extend or specialize them to our purpose (Section 6), we defined a modular architecture for the realization of the approach (Section 3). We established the theoretical basis for our work, in terms of modeling entities for the expression of declarative workflows (Section 2) and graphical notations for their visualization (Section 5). We programmed the preliminary module for storing the email-related information in a database, on top of `eml` raw email messages, retrieved via IMAP or out of text mail archives. We built a fair reliable algorithm for clustering mail conversations into threads (Section 4) and a mining algorithm for discovering declarative models of processes out of traces [17], too. We are currently working on the rest of the modules, currently stub-based.

We would like to note that this work aims at addressing the open research challenge of dealing with mixed logs, i.e., logs in which traces from multiple processes are present. Most of existing workflow mining approaches suppose to treat logs in which only traces of the same process are present; conversely, an email collection can be abstracted as a log containing traces from different processes.

Future work includes the extension of the approach to multiple users' email messages collections, and a more extensive validation. Moreover, we plan to deal with privacy concerns that naturally arise when mining over personal email messages. Indeed, it is an open issue how to perform private object matching, and the case of email messages is particularly challenging due to the already limited amount of fields that can be used to perform the matching task.

We want to apply the MailOfMine methodology on the field of collaborative activities of Open Source software development, reported by mailing lists [45].

References

1. Warren, P., Kings, N., Thurlow, I., Davies, J., Buerger, T., Simperl, E., Ruiz, C., Gomez-Perez, J.M., Ermolayev, V., Ghani, R., Tilly, M., Bösser, T., Imtiaz, A.: Improving knowledge worker productivity - the Active integrated approach. BT Technologiy Journal 26(2), 165–176 (2009)

2. Catarci, T., Dix, A., Katifori, A., Lepouras, G., Poggi, A.: Task-Centred Information Management. In: Thanos, C., Borri, F., Candela, L. (eds.) Digital Libraries: R&D. LNCS, vol. 4877, pp. 197–206. Springer, Heidelberg (2007)
3. Innocenti, P., Ross, S., Maceciuvite, E., Wilson, T., Ludwig, J., Pempe, W.: Assessing digital preservation frameworks: the approach of the SHAMAN project. In: Chbeir, R., Badr, Y., Kapetanios, E., Traina, A.J.M. (eds.) MEDES, pp. 412–416. ACM (2009)
4. Heutelbeck, D.: Preservation of enterprise engineering processes by social collaboration software. Personal Communication (2011)
5. Smart Vortex Consortium: Smart Vortex – Management and analysis of massive data streams to support large-scale collaborative engineering projects. FP7 IP Project
6. De Giacomo, G., Mecella, M.: On the representation of constraints for declarative models. Private communication (April 2010)
7. Chomsky, N., Miller, G.A.: Finite state languages. Information and Control 1(2), 91–112 (1958), The equivalence between regular grammars and FSAs is demonstrated here
8. Rabin, M.O., Scott, D.: Finite automata and their decision problems. IBM J. Res. Dev. 3, 114–125 (1959); A comprehensive study on ASFs: it contains the demonstration of the equivalence between ASFDs and ASFNDs
9. Gisburg, S., Rose, G.F.: Preservation of languages by transducers. Information and Control 9(2), 153–176 (1966); Many closure properties about regular languages are demonstrated here
10. Maggi, F.M., Mooij, A.J., van der Aalst, W.M.P.: User-guided discovery of declarative process models. In: CIDM, pp. 192–199. IEEE (2011)
11. de Carvalho, V.R., Cohen, W.W.: Learning to extract signature and reply lines from email. In: CEAS (2004)
12. Myers, E.W.: An O(ND) difference algorithm and its variations. Algorithmica 1(2), 251–266 (1986)
13. Searle, J.: A Taxonomy of Illocutionary Acts, pp. 334–369. University of Minnesota Press, Minneapolis (1975)
14. Cohen, W.W., Carvalho, V.R., Mitchell, T.M.: Learning to classify email into “speech acts”. In: EMNLP, pp. 309–316. ACL (2004)
15. Weigand, H., de Moor, A.: Workflow analysis with communication norms. Data Knowl. Eng. 47(3), 349–369 (2003)
16. Di Ciccio, C., Mecella, M.: Mining Constraints for Artful Processes. In: Abramowicz, W., Kriksciuniene, D., Sakalauskas, V. (eds.) BIS 2012. LNBIP, vol. 117, pp. 11–23. Springer, Heidelberg (2012)
17. Di Ciccio, C., Mecella, M.: MINERful, a mining algorithm for declarative process constraints in MailOfMine. Technical report, Dipartimento di Ingegneria Informatica, Automatica e Gestionale “Antonio Ruberti” – SAPIENZA, Università di Roma (2012)
18. Zardetto, D., Scannapieco, M., Catarci, T.: Effective automated object matching. In: Li, F., Moro, M.M., Ghandeharizadeh, S., Haritsa, J.R., Weikum, G., Carey, M.J., Casati, F., Chang, E.Y., Manolescu, I., Mehrotra, S., Dayal, U., Tsotras, V.J. (eds.) ICDE, pp. 757–768. IEEE (2010)
19. Elmagarmid, A.K., Ipeirotis, P.G., Verykios, V.S.: Duplicate record detection: A survey. IEEE Transactions on Knowledge and Data Engineering 19, 1–16 (2007)
20. Baeza-Yates, R.A., Ribeiro-Neto, B.A.: Modern Information Retrieval. ACM Press/Addison-Wesley (1999)

21. Salton, G.: Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer. Addison-Wesley (1989)
22. van der Aalst, W.M.P., Pesic, M., Schonenberg, H.: Declarative workflows: Balancing between flexibility and support. Computer Science - R&D 23(2), 99–113 (2009)
23. Hotho, A., Nürnberger, A., Paaß, G.: A brief survey of text mining. LDV Forum - GLDV Journal for Computational Linguistics and Language Technology 20(1), 19–62 (2005)
24. Berry, M.W., Castellanos, M. (eds.): Survey of Text Mining II: Clustering, Classification, and Retrieval. Springer (September 2007)
25. Sakurai, S., Suyama, A.: An e-mail analysis method based on text mining techniques. Appl. Soft Comput. 6(1), 62–71 (2005)
26. Sakurai, S., Ichimura, Y., Suyama, A., Orihara, R.: Acquisition of a knowledge dictionary for a text mining system using an inductive learning method. In: Proceedings of IJCAI 2001 Workshop on Text Learning: Beyond Supervision, pp. 45–52 (2001)
27. van der Aalst, W.M.P.: Process Mining: Discovery, Conformance and Enhancement of Business Processes. Springer (2011)
28. van der Aalst, W.M.P.: The application of petri nets to workflow management. Journal of Circuits, Systems, and Computers 8(1), 21–66 (1998)
29. van der Aalst, W.M.P., van Dongen, B.F., Günther, C.W., Rozinat, A., Verbeek, E., Weijters, T.: Prom: The process mining toolkit. In: de Medeiros, A.K.A., Weber, B. (eds.) BPM (Demos). CEUR Workshop Proceedings, vol. 489 (2009), CEUR-WS.org
30. Agrawal, R., Gunopulos, D., Leymann, F.: Mining Process Models from Workflow Logs. In: Schek, H.-J., Saltor, F., Ramos, I., Alonso, G. (eds.) EDBT 1998. LNCS, vol. 1377, pp. 469–483. Springer, Heidelberg (1998)
31. van der Aalst, W.M.P., Weijters, T., Maruster, L.: Workflow mining: Discovering process models from event logs. IEEE Trans. Knowl. Data Eng. 16(9), 1128–1142 (2004)
32. Wen, L., van der Aalst, W.M.P., Wang, J., Sun, J.: Mining process models with non-free-choice constructs. Data Min. Knowl. Discov. 15(2), 145–180 (2007)
33. Weijters, A., van der Aalst, W.: Rediscovering workflow models from event-based data using little thumb. Integrated Computer-Aided Engineering 10, 2003 (2001)
34. Medeiros, A.K., Weijters, A.J., Aalst, W.M.: Genetic process mining: an experimental evaluation. Data Min. Knowl. Discov. 14(2), 245–304 (2007)
35. van der Aalst, W., Rubin, V., Verbeek, H., van Dongen, B., Kindler, E., Gnther, C.: Process mining: a two-step approach to balance between underfitting and overfitting. Software and Systems Modeling 9, 87–111 (2010), doi:10.1007/s10270-008-0106-z
36. van der Aalst, W.M.P., Nikolov, A.: Mining e-mail messages: Uncovering interaction patterns and processes using e-mail logs. IJIIT 4(3), 27–45 (2008)
37. ter Hofstede, A.M., van der Aalst, W.M.P., Adamns, M., Russell, N. (eds.): Modern Business Process Automation: YAWL and its Support Environment. Springer (2010)
38. Clarke, E.M., Grumberg, O., Peled, D.: Model Checking. MIT Press (2001)
39. van der Aalst, W.M.P., Pesic, M.: DecSerFlow: Towards a Truly Declarative Service Flow Language. In: Bravetti, M., Núñez, M., Zavattaro, G. (eds.) WS-FM 2006. LNCS, vol. 4184, pp. 1–23. Springer, Heidelberg (2006)
40. Pesic, M., van der Aalst, W.M.P.: A declarative approach for flexible business processes management. In: Eder, J., Dustdar, S. (eds.) BPM Workshops 2006. LNCS, vol. 4103, pp. 169–180. Springer, Heidelberg (2006)

41. Pesic, M., Schonenberg, H., van der Aalst, W.M.P.: Declare: Full support for loosely-structured processes. In: EDOC, pp. 287–300. IEEE Computer Society (2007)
42. Pesic, M., Schonenberg, M.H., Sidorova, N., van der Aalst, W.M.P.: Constraint-Based Workflow Models: Change Made Easy. In: Meersman, R., Tari, Z. (eds.) OTM 2007, Part I. LNCS, vol. 4803, pp. 77–94. Springer, Heidelberg (2007)
43. Chesani, F., Lamma, E., Mello, P., Montali, M., Riguzzi, F., Storari, S.: Exploiting inductive logic programming techniques for declarative process mining. T. Petri Nets and Other Models of Concurrency 2, 278–295 (2009)
44. Alberti, M., Chesani, F., Gavanelli, M., Lamma, E., Mello, P., Torroni, P.: Verifiable agent interaction in abductive logic programming: The sciff framework. ACM Trans. Comput. Log. 9(4) (2008)
45. Morera-Mesa, A., Di Ciccio, C.: On the mining of processes out of open source development mailing lists. Private Communication (June 2011)

BowlognaBench—Benchmarking RDF Analytics

Gianluca Demartini[1], Iliya Enchev[1], Marcin Wylot[1],
Joël Gapany[2], and Philippe Cudré-Mauroux[1]

[1] eXascale Infolab
[2] Faculty of Humanities
University of Fribourg, Switzerland
{firstname.lastname}@unifr.ch

Abstract. The proliferation of semantic data on the Web requires RDF database systems to constantly improve their scalability and efficiency. At the same time, users are increasingly interested in investigating large collections of online data by performing complex analytic queries (e.g.,"how did university student performance evolve over the last 5 years?"). This paper introduces a novel benchmark for evaluating and comparing the efficiency of Semantic Web data management systems on analytic queries. Our benchmark models a real-world setting derived from the Bologna process and offers a broad set of queries reflecting a large panel of concrete, data-intensive user needs.

1 Introduction

The amount of semantic data available on the Web is rapidly increasing due to successful initiatives such as the Linked Open Data movement[1]. At the same time, semantic knowledge bases are being developed and improved in order to face the pressing demand of storing and querying large scale datasets. As examples, the semantic search engine Sindice[2] indexes up to 200M RDF documents, while very large datasets containing billions of RDF triples[3] are increasingly common. The bigger the datasets grow in size, the more interesting it becomes to perform complex analysis on RDF data, in order for example to identify patterns or discover data correlations. This is somehow similar to OLAP (On-Line Analytical Processing) transactions on large databases where complex read-mostly queries are run over big datasets.

Modern business processes generate much data which is often stored for further analysis or even for legal purposes. Such data can be seen an important asset for a company willing to improve their processes by means of data analysis. One of the current open challenges that limits such analysis is the poor performance of data management platforms. Specifically, for the case of graph data (of which business process data is a good example) much improvement can be obtained.

[1] http://linkeddata.org/

[2] http://sindice.com/

[3] http://km.aifb.kit.edu/projects/btc-2010/

K. Aberer, E. Damiani, and T. Dillon (Eds.): SIMPDA 2011, LNBIP 116, pp. 82–102, 2012.

In order to foster such efficiency and scalability advances that address the need for business process analysis, in this paper[4], we propose a benchmark for evaluating and comparing the efficiency of knowledge management systems focusing in particular on *analytic queries* over RDF data. Other benchmarks for RDF storage systems have been proposed already—the most popular one being LUBM [9], which provides an ontology describing universities together with an automatic instance generator and a set of 14 simple queries. Extensions of such benchmarks have also been proposed. For example, Minack *et al.* [11] created a benchmark for full-text search over semantic repositories. Further RDF benchmarks include a benchmark over full-text data (see, for instance, [3,13]), a benchmark based on scientific publications (that is, DBLP) [14], and a benchmark based on a Web data exploration application [1]. Recently, the DBpedia SPARQL Benchmark [12] has also been proposed: it consists of a RDF store benchmark which focuses on evaluating system performance over real queries and real data. It contains queries mined from real logs of the DBPedia system[5] and resembles original DBPedia data.

Why is there a need for a new RDF benchmark? The simple answer is that none of the aforementioned benchmarks focuses on complex analytic queries. Analytic operations have become a major concern of any modern company, as they are used to measure and improve the performance of many complex operations (from data warehousing to reporting or data mining). Traditional relational systems were designed and optimized for transactional queries mostly, and cannot handle analytic queries efficiently. This recently led to many advances in data management technologies with the emergence of companies like Teradata[6] or, more recently, Vertica[7], and the definition of new benchmarks, such as TPC-H[8] (which is the standard benchmark for data analytics today) or the recent benchmark we suggested for mixed OLAP/OLTP settings [8]. In the university context we can expect relatively low volumes of *insert* and *update* queries (as compared, for example, to an e-commerce scenario). For this reason, we thought that it would be interesting to focus on analytic and trend queries rather than supporting high volume of transactions. To the best of our knowledge, BowlognaBench is the first RDF benchmark focusing on this topic.

Beyond its focus on data analytics, we feel that BowlognaBench is based on a more realistic ontology than previous attempts. Duan *et al.* [6] recently proposed a comparison between available RDF benchmarks and real datasets such as DBpedia [2] and YAGO [15]. Their study showed that existing benchmarks have little in common with real world datasets. This finding confirms our motivation for creating a novel benchmark for RDF that better reflects user information needs. LUBM, the most popular RDF storage benchmark, does not directly reflect a real university setting, while the ontology we propose models an academic

[4] This paper is an extended version of [4].

[5] `http://dbpedia.org/`

[6] `http://www.teradata.com/`

[7] `http://www.vertica.com/`

[8] `http://www.tpc.org/tpch/`

setting as prescribed by the Bologna process. The second drawback of LUBM is that its set of queries tries to cover possible failures in knowledge management systems and advanced (but rather rare) inference cases, rather than model real user information needs. We develop a set of queries of real interest to the academic domain (e.g., "what is the percentage of students who continue their university studies after the Bachelor?"). Third, the LUBM benchmark does not involve the temporal dimension, which is a very important aspect of analytic queries; in our benchmark, we propose to evaluate queries asking for specific time or time intervals over the data (e.g., "how did student performance evolve over the last 5 years"?), which is a popular type of query that need to be supported by current systems (see for example [10]).

In summary, BowlognaBench provides a framework for evaluating the performance of RDF systems on a real-world context derived from the Bologna process; it strains systems using both analytic and temporal queries; and it models real academic information needs.

The rest of the paper is structured as follows. In Section 2 we briefly describe the Bologna reform of the European university studies. Then, in Section 3 we present the lexicon of terms relevant to the Bologna process that has been defined in order to avoid ambiguities and misunderstandings. Next, in Section 4 we introduce the Bowlogna Ontology that has been developed to model the university setting after the Bologna reform. Together with the set of queries defined on top of the Bowlogna Ontology in Section 5, we propose the Bowlogna Benchmark to evaluate and compare efficiency of knowledge bases for analytic queries. In Section 6 we present an experimental comparison of different systems over the proposed benchmark. Finally, we conclude the paper in Section 7.

2 Information Systems in the Bologna Process

The Bologna Process is the set of reforms into which European higher education institutions entered from June 1999 onward, after the ministries of education of 29 European countries convened in Bologna and laid new foundations for the future European higher education. The Bologna Declaration defines a general framework in which signatory countries commit themselves to develop their higher education system. As such, it also leaves a certain margin for interpretation, which has allowed local appreciation and various implementations of the process. As a consequence, during the last decade, a specific form a relationship between universities has unquestionably developed, according to which both cooperation and competition in teaching and research are equally fostered.

In this context, the need to make available (for example, in RDF form) comprehensive and transferable datasets about current students and course offerings is increasing dramatically, as a key dimension of quality processes. In this competitive environment also, the capacity to attract new students, especially at the master level, is generally regarded as a factor of success. It is therefore not surprising that universities invest considerable efforts into the development of new information infrastructures, and into increasingly sophisticated analytic tool to help them monitor and assess their performance.

3 The Bologna Linguistic Lexicon

The Bologna Process initiated a radical change within higher education institutions. This change encompasses the creation of new administrative procedures, inclusive quality procedures, in the every day life of the universities. It also gave rise to the emergence of new concepts for the description of curricula. In this respect, we recently published [7] in the Bologna Handbook the basic set of entities that correspond to the new system of studies and that seem to be rather stable across time and institutions. In this paper, we described how the compilation of a lexicon of about 60 definitions contributed to the implementation of the Bologna Process in our institution. This booklet has been spread among academic and administrative staff, in order to create a common understanding of Bologna. It is worth noting that the defined entities are concepts, rather than single words, and that they were selected and defined by professional terminologist who adopted a rigorous, text-based approach to study the Bologna Process.

4 The Bowlogna Ontology

The formal lexicon described above paved the way for an ontology-based computerization of the Bologna process. Based on our understanding of the Bologna and on the Bologna lexicon, we developed an OWL ontology describing all the relevant concepts and their relations[9]. Figure 1 shows some of the classes in the Bowlogna ontology together with their relations. The ontology contains 29 top level classes (67 in total) describing concepts like students, professors, student evaluations (i.e., exams where students get a grade from a lecturer in the context of a teaching unit), teaching units (i.e., courses given by lecturers and attended by students), ECTS credits, as well as formal documents such as, for instance, study programs, certificates, or grade transcripts. Additionally, all definitions from the lexicon, both in German and French, are included in the ontology.

One of the key classes in this ontology is “Evaluation” in which a Student, a Professor and a Teaching Unit are involved. This class models the event when a student, after attending a course given by a professor (i.e., a teaching unit) is evaluated and given a grade. Properties of teaching units include the semester and the language in which the units are given, as well as the number of corresponding ECTS credits earned by students who successfully follow the course. The class Student also has a number of properties including the name of the student, his enrollment or graduation date for Bachelor and Maser degrees. The same student instance can be, over time, registered for different academic degrees (e.g., first a Bachelor and then a Master). The class “ECTS credit” is linked to all classes directly or indirectly related to the measurement of students advancement: for example, teaching units and study programs. The class Semester, with start and end dates, enables queries on the temporal dimensions (e.g., the number of students enrolled at the university during a certain semester).

[9] For a comprehensive description of the Bowlogna ontology see [5].

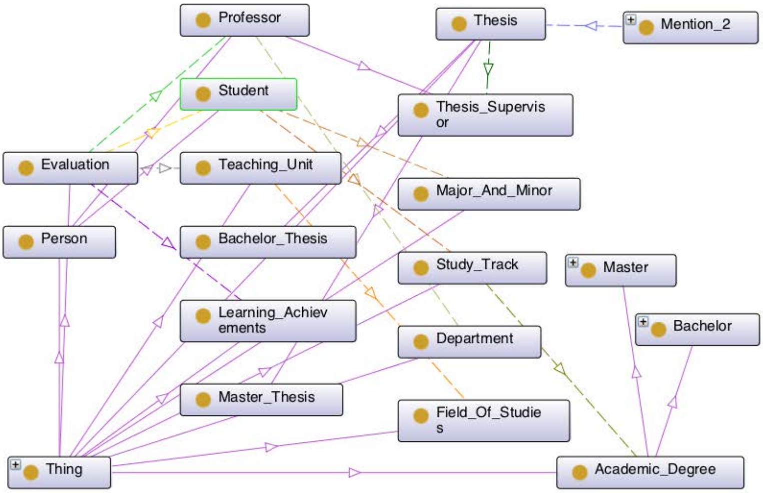


Fig. 1. The Bowlogna ontology: key classes and relations

4.1 Public and Private Ontology Parts

The Bowlogna ontology can be divided in two important parts according to the type of information stored. Some information are public and can be shared with other universities as well as with the general public. Examples of such public information include the departments, the teaching units together with information about their ECTS credits and teaching language. The second part of the ontology consists of information which should not be publicly available, such as grades given to students. In real instantiations of the ontology, the private part will contain many more instances as compared to the public part. It is also clear that aggregations over private information might often be of general interest (e.g., the number of students currently enrolled at the university), and can be safely shared.

4.2 Additional Testing for Scalability: Multiple Universities

The ontology we described so far is suitable to describe a single university setting. As the benchmark we are proposing aims at testing system performance as the amount of stored data increases, we can imagine, for example, to test systems on different repositories containing an increasing number of students or departments. Additionally, we can imagine that different universities may adopt the same ontology for describing their data. In such settings, we can even imagine exchange students following teaching units in different universities. In this case, we can scale the size of the dataset and evaluate system performance on the private part of a single a university as well as on the public part of several universities.

5 Queries

In this section, we describe the various queries we defined in order to evaluate the performance of RDF data management systems systems in an analytic setting. The proposed set of queries aims at covering a broad range of requests, including Average, Max, Min, GroupBy, Top-K, Rank, and Path queries. In the following we describe the queries using natural language. The complete set of SPARQL queries is available in the Appendix as well as on the benchmark website.

5.1 Count Queries

This subset of queries requires systems to count the number of instances having specific characteristics. Some of them only need to access the public information, and thus can be scaled to multiple universities, while others access private information such as the result of student evaluations.

`Query 1. What is the percentage of master theses that received a mention?`
For this query the system has to select the successful master theses and to check how many of them are related to a Mention object. In general, such information is publicly available, thus systems only need to access the public part of the ontology.

`Query 2. What is the percentage of students who continue their university studies beyond the Bachelor?`
This query needs to check, for every student who obtained a Bachelor degree, whether he/she is registered to any Master program, count the number of such students, and finally divide it by the total number of students who registered for a Bachelor degree. In order to run this query the system only has to access the public part of the ontology (provided that registration information is increasingly shared across universities).

`Query 3. What is the current number of ECTS credits` *`Student0`* `has acquired?`
This is a more transactional query focusing on a specific instance (i.e., *Student0*), which requires to check for each evaluation taken by this student whether it was successful or not. For all such cases, the system must then sum the number of ECTS credits assigned to the related teaching unit. For this query, systems have to access the private part of the ontology (i.e., grades of evaluations).

5.2 Selection Queries

This group aims at retrieving a large set of instances from the knowledge base. The queries may be simple in terms of operators but it requires knowledge bases to handle large amount of data.

`Query 4. Return all students whose surname starts with ‘A’.`
This query returns a portion of all the student instances stored in the knowledge base. This type of queries may be useful when creating catalogs of students where, for each letter of the alphabet, there is a page containing all students whose surname starts with the same letter. As student names should not be released to the general public, in order to run this query systems needs to be able to access the private part of the dataset.

5.3 Molecule Queries

This type of queries aims at retrieving all the information surrounding a specific instance in the graph. To answer such queries, systems have to be efficient in retrieving all triples in the vicinity of a given node in the RDF graph, which represents a complex operation (with series of *joins*) for many transactional systems.

`Query 5. Return all information about` *`Student0`* `within a scope of two.`
This query returns all triples connected directly or indirectly to *Student0*, that is, a subgraph centered on *Student0* and containing all the triples directly attached to the student, plus the triples attached to those triples. As private student information are also requested, this query needs to access the private as well as the public part of the ontology.

5.4 Max and Min Queries

These queries require to find some specific instances which, among others of the same type, have the highest/lowest value for the requested characteristic.

`Query 6. Which teaching unit has the lowest success rate?`
This query requires to compute the success rate, that is, the number of successful student evaluations divided by the total number of student evaluations done for that teaching unit, and find the minimum. It needs to access private information of the ontology.

`Query 7. Who is the professor who supervised most theses?`
This query requires, after counting the number of theses grouped by professors, to compute the maximum and return the professor. For this query, the functionality of subclass inference needs to be additionally supported by systems. This is the case as matching instances are of type Thesis_Supervisor which is a subclass of Professor. Thus, systems have to infer that such instances are of type Professor to correctly answer the query. The information required to answer this query is typically publicly available.

5.5 Ranking and TopK Queries

For such queries, the systems have to either rank all instances of a certain class or to find the top k elements of a ranked list.

`Query 8. Who are the top 5 performing students in` *`StudyTrack0`* `and semester` *`Semester0`*`?`

This is a *top-k* query that asks the system to rank students for a specific Study Track and Semester after computing the average grade obtained in all the evaluations they have performed so far. This requires to access student grades, which is private information.

5.6 Temporal Queries

This set of queries aims at checking system support and efficiency of temporal queries, that is, requests that involve a filter of instances based on a specific date or time interval.

`Query 9. What is the average completion time of Bachelor studies for each Study Track?`
This query requires the repository to compute the completion time for each student of the university, average them and group the results by study track. Information about registration and graduation date may be private or public depending on the university.

`Query 10. How did university student performance evolve over the last 3 semesters?`
This query requires the system to compute the average grade of all student evaluations grouped by semester. Evaluation results required to answer this query are private information.

5.7 Path Queries

This type of query wants to test the ability of systems to exploit several nodes in the ontology joined through RDF predicates.

`Query 11. What are the names of students who took an exam with a professor of` *`Department0`*`?`
This is a path query in the sense that it requires to join triples over several different predicates (namely, Student:hasFamilyName, Student:hasFirstName, Evaluation:performedByStudent, Evaluation:evaluatedByProfessor, Professor:isAffiliatedWithDepartment, Department:hasName). As student names should not be released to the general public, in order to run this query systems needs to be able to access the private part of the dataset.

5.8 Multiple University Queries

As the goal of building an RDF benchmark is to test for scalability of systems as the size of the dataset grows, we can imagine to run certain queries over several university repositories and aggregate the results to provide a final answer to the user. In this setting, the amount of data that has to be managed can increase a lot as compared to a single university, but, on the other hand, initial queries may be run in parallel and results aggregated later on.

Query 12. In which university can I attend *Teaching_Unit0* in English?
In case different universities share their ontology, it is possible to run queries over different instances of the same schema. In this case, the system has to check the language of the instance *Teaching_Unit0* in all accessible universities. Information about teaching units is assumed to be public for all universities.

Query 13. How did the number of newly registered students to each University evolve over the last 5 years?
For this query, systems need to find the number of students enrolled at each university for each semester. This computation requires to check the registration and the possible graduation dates for each student and compare them with the semester start and end dates. We consider that information about registration and graduation date are shared across collaborating universities.

5.9 Query Classification

Table 1 presents a classification of the proposed queries based on several dimensions. First, we classify queries accordingly to the fact that they need to access the public or the private part of the ontology (see Section 4.1). Then, we use the dimensions that were introduced in [9], that is, Input Size, Selectivity, and Complexity. Input size measures the number of instances involved in the query: we classify a query as having a large input size if it involves more than 5% of instances, and small otherwise. Selectivity measures the amount of instances that match the query: we classify a query as having high selectivity if less than 10% of instances match the query, and low otherwise. Complexity measures the amount of classes and properties involved in the query: queries are classified as having high or low complexity accordingly to the RDF schema we have defined.

Table 1. Classification of queries according to their need to access private and public data, input size, selectivity, and complexity.

	Count			Selection	Molecule	MaxMin		TopK	Temp		Path	MultiUniv	
Query	1	2	3	4	5	6	7	8	9	10	11	12	13
Public	x	x					x	x	x		x	x	x
Private			x	x	x	x		x	x	x	x		x
Input Size	Small	Large	Small	Large	Small	Large	Small	Large	Large	Large	Large	Large	Large
Selectivity	High	Low	Low	High	Low	Low	Low	High	Low	Low	Low	Low	Low
Complexity	Low	Low	Low	Low	High	High	Low	High	Low	High	High	Low	High

As we can see the majority of queries have a low selectivity which reflects our intent of performing analytic queries, that is, queries for which a lot of data is retrieved and aggregated. For the same reason, most of the queries have a large input. Finally, queries are equally divided in high and low complexities.

6 Comparing RDF Systems Using BowlognaBench

In this section we report the results of an experiment which aims at comparing the performance of different knowledge base platforms on the analytic queries of BowlognaBench. First, we describe all the additional elements composing the proposed benchmark, namely the instance generator and the BowlognaBench website.

BowlognaBench Instance Generator. The artificial instance generator provided with BowlognaBench is a Java program that takes as input the number of departments, the number of fields per department, and the number of semesters to scale over the time dimension as well. It generates a populated Bowlogna ontology by producing RDF files (one per department) in NTriple format. In order to generate large datasets, it is recommended that at least 2G of memory are assigned to the Java process.

Artificial data may not well reflect real data distribution [6]. For this reason, we analyzed a sample of real academic data provided by the University of Fribourg administration. The observed properties are used to add additional parameters to the BowlognaBench artificial instance generator. In the following we report some observations made over the real data we had access to.

The data describes professors, departments, teaching units, and fields of study over 34 semesters at the University of Fribourg. Figure 2 shows the distribution of professors with respect to the number of teaching units related to them. As expected this follows a power law distribution where we can see that many professors teach only one teaching unit while few professors teach many units. We manually inspected special cases such as, for example, the professor with most teaching units and observed that she is an University administration employee assigned to 643 foreign language courses. Strictly speaking, this is not an error in the data but rather an imprecise link as the relation between that person and the language courses is not of the same type as for the others. Such cases are typical in real data but not so easy to find in artificially generated data.

Another aspect of real data is *incompleteness*. While it is easy to generate complete artificial data, systems running over real data need to face the problem of missing values. In the available real data we observed that 0.8% of the teaching units have a ‘null’ value for the ‘number of ECTS credits’ field which is not supposed to be left empty. Another example of data incompleteness is shown in Figure 3 where the out-degree of the Teaching Unit class is displayed. We can see that 17 teaching units have out-degree of 1 which is wrong as at least ‘bb:belongsToFieldOfStudies’ and ‘bb:isForSemester’ should be defined for a teaching unit. Otherwise, the data distribution looks as expected.

BowlognaBench Website. The benchmark we produced and used for the following experiments—that is, the Bowlogna ontology together with the set of queries and the Bowlogna artificial instance generator—is available for download at: `http://diuf.unifr.ch/xi/bowlognabench/`. The instance generator includes various options to scale and change the statistical distribution of the

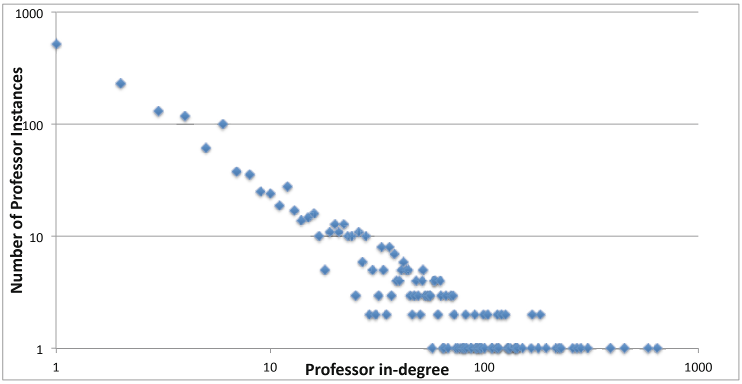


Fig. 2. In-degree of Professor instances (i.e., Teaching Units related to a Professor)

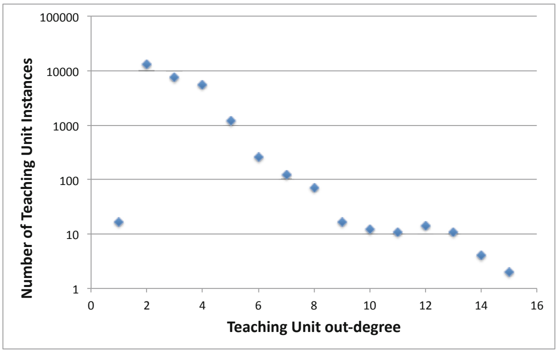


Fig. 3. Out-degree of Teaching Unit instances (i.e., relations with other entities)

data. On the BowlognaBench website we also allow third-parties to publish their own test configurations and performance results.

6.1 Experimental Setting

We compared the following RDF systems: 4Store 4s-query revision v1.1.3, RDF-3X 0.3.5, Virtuoso Open-Source Edition 6.1.3, and our own dipLODocus[RDF] system [16]. In order to run such comparison we use two different datasets generated with the BowlognaBench Instance Generator for 1 and 10 departments, 4 fields per department and 15 semesters. The datasets contain 1.2 and 12 million RDF triples for 273MB and 2.7GB respectively. We run the 13 queries of BowlognaBench to compare the query execution time for four different RDF systems. Reported execution times are obtain by averaging over 10 executions of the same query over a warm-cache system. Aggregation (COUNT, SUM, AVG, etc.) and operations such as percentage or ratio have not been performed as not supported by all system: instead, all data needed to compute such aggregates is retrieved. We also compare the load time and the size of the created indexes. Experiments were run on a machine with Intel Core i7-2600 CPU @ 3.40GHz, 8GB RAM, Ubuntu 11.10.

RDF systems. In the following we provide some details about the different systems we compared over BowlognaBench:

RDF 3X[10] creates various heavily compressed indices starting from a huge triple-table. Compression is done by means of dictionary encoding and byte-wise compression. Indices are based on the six possible permutations of RDF triple elements, and aggregate indices storing only two out of the three elements. The RDF-3X query executor optimizes join orderings and determine the cheapest query plan by means of a dedicated cost-model.

4Store[11] from Garlik is a recent parallel RDF database distributing triples using a round-robin approach. It stores data in quadruple-tables.

Virtuoso[12] is an object-relational database system using bitmap indices to optimize the storage and processing of RDF data.

dipLODocus[RDF][13] is a hybrid RDF system combining subgraph storage (where joins are materialized) and column-store technologies (where values are compactly stored in vectors) to answer both triple pattern queries and analytic queries very efficiently.

6.2 Experimental Results

Table 2 shows the index size and the load time for the two datasets and four RDF systems. We can see that Virtuoso takes more time to load and index the dataset but the size of the indices scales better than for other systems. The faster system is 4Store which also has the biggest indices. RDF 3X, Virtuoso, and dipLODocus achieve good compression with Virtuoso showing the best scaling.

Table 2. Load time and index size for four RDF systems and two datasets

	RDF 3X	4Store	Virtuoso	dipLODocus
Load Time 1 dept. (s)	11.94	6.25	31.71	18.35
Load Time 10 dept. (s)	139.55	69.65	363.24	526.65
Indices size 1 dept. (MB)	60	192	108	92
Indices size 10 dept. (MB)	618	1752	616	920

Figure 4 and 5 report the experimental results for the datasets consisting of 1 and 10 departments respectively. The values indicate query execution times for each query of the BowlognaBench benchmark[14].

[14] Query 4 could not be run on RDF 3X and dipLODocus as they do not provide full SPARQL support (i.e., for this query no support for pattern matching is provided).

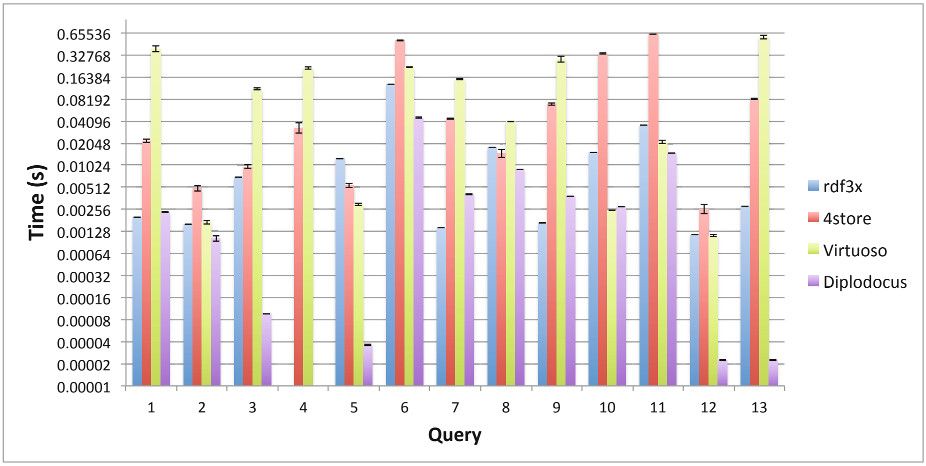


Fig. 4. Query execution time for BowlognaBench queries over the single department dataset

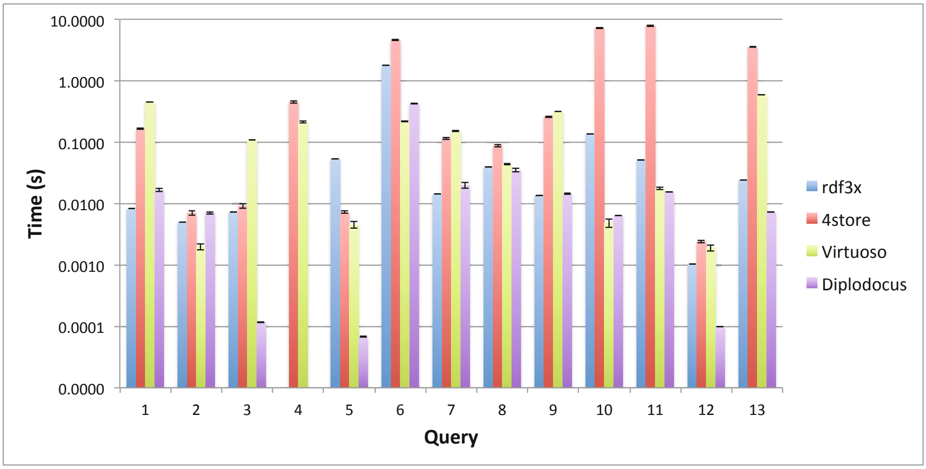


Fig. 5. Query execution time for BowlognaBench queries over the 10 departments dataset

As we can see, query execution time for the BowlognaBench analytic queries strongly vary for different systems (note that Figure 4 and 5 show query execution time on a logarithmic scale). For the smaller of the two datasets (i.e., data for a single department) the dipLODocus system is faster than others for 9 out of 13 queries. Specifically, shortest query executions can be observed for queries 12 and 13. The slowest is the path query which involves several joins. The RDF 3X system obtained the best performance on 3 queries (i.e., query 1, 7, and 9).

We can see that query 8 (i.e., top K) is not easy to be efficiently answered for all systems. Query 3 and 11 are also challenging for most systems because of the several joins involved.

For the bigger dataset (i.e., data for 10 departments) we can observe the slower performances of 4Store for 7 out of 13 queries as compared with the other three systems: for some queries (e.g., 11) the execution times took up to 8 seconds. One difference that we can observe as compared with the smaller dataset is the good result of Virtuoso: it performed faster than other systems on 4 out of 13 queries. The dipLODocus system performed best on 6 queries with dramatic improvements for some queries (e.g., 5 the molecule query) showing execution times orders of magnitude smaller than other systems. In general, we can again observe that query 8 and 11 are difficult to answer efficiently because of several joins.

7 Conclusions

As RDF continues to grow in popularity, being able to efficiently execute analytic, data-intensive operations over large RDF graphs is becoming increasingly important in the business process analysis field as well as in others. In this paper, we proposed a benchmark for evaluating and comparing the performance and scalability of knowledge base systems on complex analytic queries. After describing the European academic landscape as outlined by the Bologna reform, we introduced BowlognaBench, a benchmark for RDF systems consisting of an ontology derived from a systematic study of the official Bologna documents, a set of complex queries derived from concrete user information needs, and an automatic instance generator that allows to populate the ontology with a given number of instances and relations. Finally, we compared the query execution time of different RDF systems over the proposed benchmark showing common limitations and bottlenecks of such systems as, for example, queries involving several joins. As for former benchmarks, we hope that this project will stimulate a healthy competition among RDF data management systems as well as foster the emergence of new RDF back-ends geared towards complex, data-intensive Semantic Web applications.

Acknowledgment. This work is supported (in part) by the Swiss National Science Foundation under grant number PP00P2_128459.

References

1. Abadi, D.J., Marcus, A., Madden, S.R., Hollenbach, K.: Scalable semantic web data management using vertical partitioning. In: Proceedings of the 33rd International Conference on Very Large Data Bases, VLDB 2007, pp. 411–422. VLDB Endowment (2007)

2. Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., Ives, Z.G.: DBpedia: A Nucleus for a Web of Open Data. In: Aberer, K., Choi, K.-S., Noy, N., Allemang, D., Lee, K.-I., Nixon, L.J.B., Golbeck, J., Mika, P., Maynard, D., Mizoguchi, R., Schreiber, G., Cudré-Mauroux, P. (eds.) ASWC 2007 and ISWC 2007. LNCS, vol. 4825, pp. 722–735. Springer, Heidelberg (2007)
3. Bizer, C., Schultz, A.: Benchmarking the performance of storage systems that expose SPARQL endpoints. In: Proceedings of the ISWC Workshop on Scalable Semantic Web Knowledgebase systems (2008)
4. Demartini, G., Enchev, I., Gapany, J., Cudré-Mauroux, P.: BowlognaBench—Benchmarking RDF Analytics. In: Aberer, K., Damiani, E., Dillon, T. (eds.) SIMPDA 2011. LNBIP, vol. 116, pp. 82–102. Springer, Heidelberg (2012)
5. Demartini, G., Enchev, I., Gapany, J., Cudré-Mauroux, P.: The Bowlogna Ontology: Fostering Open Curricula and Agile Knowledge Bases for Europe's Higher Education Landscape. In: Semantic Web - Interoperability, Usability, Applicability (2012)
6. Duan, S., Kementsietsidis, A., Srinivas, K., Udrea, O.: Apples and Oranges: A Comparison of RDF Benchmarks and Real RDF Datasets. In: SIGMOD (2011)
7. Gapany, J., Vergauwen, G.: Curricular design and computerisation: are information systems useful in curricular reorganisation? European University Association: the Bologna Handbook, 10 (C 3.9-3) (2010)
8. Grund, M., Krüger, J., Plattner, H., Zeier, A., Cudre-Mauroux, P., Madden, S.: Hyrise: a main memory hybrid storage engine. Proc. VLDB Endow. 4, 105–116 (2010)
9. Guo, Y., Pan, Z., Heflin, J.: LUBM: A benchmark for OWL knowledge base systems. Web Semantics: Science, Services and Agents on the World Wide Web 3(2-3), 158–182 (2005)
10. Hoffart, J., Suchanek, F.M., Berberich, K., Lewis-Kelham, E., de Melo, G., Weikum, G.: YAGO2: exploring and querying world knowledge in time, space, context, and many languages. In: Srinivasan, S., Ramamritham, K., Kumar, A., Ravindra, M.P., Bertino, E., Kumar, R. (eds.) WWW (Companion Volume), pp. 229–232. ACM (2011)
11. Minack, E., Siberski, W., Nejdl, W.: Benchmarking Fulltext Search Performance of RDF Stores. In: Aroyo, L., Traverso, P., Ciravegna, F., Cimiano, P., Heath, T., Hyvönen, E., Mizoguchi, R., Oren, E., Sabou, M., Simperl, E. (eds.) ESWC 2009. LNCS, vol. 5554, pp. 81–95. Springer, Heidelberg (2009)
12. Morsey, M., Lehmann, J., Auer, S., Ngonga Ngomo, A.-C.: DBpedia SPARQL Benchmark – Performance Assessment with Real Queries on Real Data. In: Aroyo, L., Welty, C., Alani, H., Taylor, J., Bernstein, A., Kagal, L., Noy, N., Blomqvist, E. (eds.) ISWC 2011, Part I. LNCS, vol. 7031, pp. 454–469. Springer, Heidelberg (2011)
13. Schmidt, M., Hornung, T., Küchlin, N., Lausen, G., Pinkel, C.: An Experimental Comparison of RDF Data Management Approaches in a SPARQL Benchmark Scenario. In: Sheth, A.P., Staab, S., Dean, M., Paolucci, M., Maynard, D., Finin, T., Thirunarayan, K. (eds.) ISWC 2008. LNCS, vol. 5318, pp. 82–97. Springer, Heidelberg (2008)
14. Schmidt, M., Hornung, T., Lausen, G., Pinkel, C.: SPˆ 2Bench: A SPARQL Performance Benchmark. In: IEEE 25th International Conference on Data Engineering, ICDE 2009, pp. 222–233. IEEE (2009)

15. Suchanek, F.M., Kasneci, G., Weikum, G.: Yago: a core of semantic knowledge. In: Proceedings of the 16th International Conference on World Wide Web, WWW 2007, pp. 697–706. ACM, New York (2007)
16. Wylot, M., Pont, J., Wisniewski, M., Cudré-Mauroux, P.: dipLODocus[RDF] - Short and Long-Tail RDF Analytics for Massive Webs of Data. In: Aroyo, L., Welty, C., Alani, H., Taylor, J., Bernstein, A., Kagal, L., Noy, N., Blomqvist, E. (eds.) ISWC 2011, Part I. LNCS, vol. 7031, pp. 778–793. Springer, Heidelberg (2011)

Appendix A: SPARQL Queries

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX bb: <http://diuf.unifr.ch/main/xi/BowlognaBench/2011/05/bb#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
```

Query 1. What is the percentage of master theses that received a mention?

```
SELECT COUNT(?X) as MasterTheses
WHERE {
?X rdf:type bb:Master_Thesis }

SELECT COUNT(?X) as MasterThesesWithMention
WHERE {
?X rdf:type bb:Master_Thesis .
?Y rdf:type bb:Mention .
?Y bb:mentionGivenTo ?X }
```

Query 2. What is the percentage of students who continue their university studies beyond the Bachelor?

```
SELECT COUNT(?X) as studentsWithBachelorDegree
WHERE{
?X rdf:type bb:Student .
?X bb:endsBachelorStudiesOn ?end^^xsd:date .
?end < "currentDate"^^xsd:date }

SELECT COUNT(?Y) as studentsEnrolledAtMaster
WHERE{
?Y rdf:type bb:Student .
?Y bb:enrolledForMasterStudiesOn ?start^^xsd:date .
?start < "currentDate"^^xsd:date }
```

Query 3. What is the current number of ECTS credits *Student0* has acquired?

```
SELECT SUM(?credits) as ECTSCredits
WHERE{
?X rdf:type bb:Student .
?Y rdf:type bb:Evaluation .
?Z rdf:type bb:Teaching_Unit .
?X bb:hasFamilyName "Student0" .
?X bb:hasFirstName "Student0" .
?Y bb:performedByStudent ?X .
?Y bb:evaluatesTeachingUnit ?Z .
?Y bb:hasMark ?M .
?M >= "C" .
?Z bb:hasNumberOfECTS ?credits }
```

Query 4. Return all students whose surname starts with ‘A’.

```
SELECT ?S
WHERE{
?S rdf:type bb:Student .
?S bb:hasFamilyName ?ln
FILTER (regex(?ln, "^A")) }
```

Query 5. Return all information about *Student0* within a scope of two.

```
SELECT ?P1, ?O1, ?P2, ?O2
WHERE{
?S rdf:type bb:Student .
?S bb:hasFamilyName "Student0" .
?S bb:hasFirstName "Student0" .
?S ?P1 ?O1 .
?O1 ?P2 ?O2 }

SELECT ?S1, ?P1, ?S2, ?P2
WHERE{
?O rdf:type bb:Student .
?O bb:hasFamilyName "Student0" .
?O bb:hasFirstName "Student0"
?S1 ?P1 ?O
?S2 ?P2 ?S1 }
```

Query 6. Which teaching unit has the lowest success rate?

```
SELECT ?X, count(?E) as FailuresForTeachingUnit
WHERE{
?X rdf:type bb:Teaching_Unit .
?E rdf:type bb:Evaluation .
?E bb:evaluatesTeachingUnit ?X .
?E bb:hasMark ?Y .
?Y < "C" }
GROUP BY ?X

SELECT ?X, count(?E) as EvaluationsForTeachingUnit
WHERE{
?X rdf:type bb:Teaching_Unit .
?E rdf:type bb:Evaluation .
?E bb:evaluatesTeachingUnit ?X }
GROUP BY ?X
```

Query 7. Who is the professor who supervised most theses?

```
SELECT ?X, COUNT(?Y) as numOfThesisSupervised
WHERE{
?X rdf:type bb:Professor .
?Y rdf:type bb:Thesis .
?Y bb:supervisedBy ?X }
ORDER BY numOfThesisSupervised
LIMIT 1
```

Query 8. Who are the top 5 performing students in *StudyTrack0* and semester *Semester0*?

```
SELECT ?X, AVG(?M) as ?avgMark
WHERE{
?X rdf:type bb:Student .
?Y rdf:type bb:Evaluation .
?T rdf:type bb:Study_Track .
?S rdf:type bb:Semester .
?T bb:hasName "StudyTrack0" .
?S bb:hasName "Semester0" .
?X bb:isInStudyTrack ?T .
?Y bb:isForSemester ?S .
?Y bb:performedByStudent ?X .
?Y hasMark ?M }
GROUP BY ?X
ORDER BY ?avgMark
LIMIT 5
```

Query 9. What is the average completion time of Bachelor studies for each Study Track?

```
SELECT ?T, AVG(?end-?start) as CompletionTime
WHERE{
?T rdf:type bb:Study_Track .
?X rdf:type bb:Student .
?X bb:enrolledForBachelorStudiesOn ?start^^xsd:date .
?X bb:endsBachelorStudiesOn ?end^^xsd:date .
?X bb:isInStudyTrack ?T }
GROUP BY ?T
```

Query 10. How did university student performance evolve over the last 3 semesters? *Repeat for all semesters*

```
SELECT AVG(?mark) as AverageSemester1
WHERE{
?X rdf:type bb:Evaluation .
?Y rdf:type bb:Teaching_Unit .
?Z rdf:type bb:Semester .
?X bb:evaluatesTeachingUnit ?Y .
?X bb:isForSemester ?Z .
?Z bb:hasName "Semester1" .
?X bb:hasMark ?mark }
```

Query 11. What are the names of students who took an exam with a professor of *Department0*?

```
SELECT ?FamName, ?FirstName
WHERE{
?X rdf:type bb:Student .
?Y rdf:type bb:Professor .
?Z rdf:type bb:Evaluation .
?D rdf:type bb:Department .
?Z bb:performedByStudent ?X .
?Z bb:evaluatedByProfessor ?Y .
?Y bb:isAffiliatedWithDepartment ?D .
?X bb:hasFamilyName ?FamName .
?X bb:hasFirstName ?FirstName .
?D bb:hasName "Department0" }
```

Query 12. In which university can I attend *Teaching_Unit0* in English?
Repeat for all universities.

```
SELECT ?X
WHERE{
    ?X rdf:type bb:Teaching_Unit .
    ?X bb:isTaughtInLanguage "EN" .
    ?X bb:hasName "TeachingUnit0" }
```

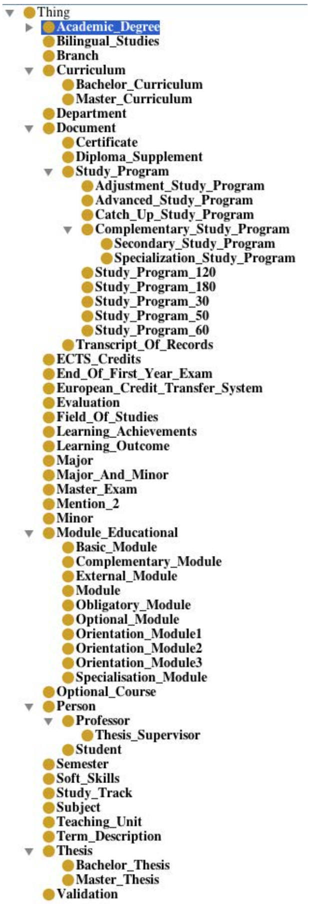
Query 13. How did the number of newly registered students to each University evolve over the last 5 years?
Repeat for all semesters and all universities.

```
SELECT COUNT(?X) as BachelorStudentsSemester0
WHERE{
?X rdf:type bb:Student .
?S rdf:type bb:Semester .
?S bb:hasName "Semester0" .
?S bb:beginsOnDate ?start^^xsd:date .
?S bb:endsOnDate ?end^^xsd:date .
?X bb:enrolledForBachelorStudiesOn ?Y^^xsd:date .
?Y <= ?end .
?Y >= ?start }

SELECT COUNT(?X) as MasterStudentsSemester0
WHERE{
?X rdf:type bb:Student .
?S rdf:type bb:Semester .
?S bb:hasName "Semester0" .
?S bb:beginsOnDate ?start^^xsd:date .
?S bb:endsOnDate ?end^^xsd:date .
?X bb:enrolledForMasterStudiesOn ?Y^^xsd:date .
?Y <= ?end .
?Y >= ?start }
```

Appendix B: The Bowlogna Ontology Classes

Figure 6 presents all the 67 classes defined in the Bowlogna ontology.



- Thing
  - Academic_Degree
  - Bilingual_Studies
  - Branch
  - Curriculum
    - Bachelor_Curriculum
    - Master_Curriculum
  - Department
  - Document
    - Certificate
    - Diploma_Supplement
    - Study_Program
      - Adjustment_Study_Program
      - Advanced_Study_Program
      - Catch_Up_Study_Program
      - Complementary_Study_Program
        - Secondary_Study_Program
        - Specialization_Study_Program
      - Study_Program_120
      - Study_Program_180
      - Study_Program_30
      - Study_Program_50
      - Study_Program_60
    - Transcript_Of_Records
  - ECTS_Credits
  - End_Of_First_Year_Exam
  - European_Credit_Transfer_System
  - Evaluation
  - Field_Of_Studies
  - Learning_Achievements
  - Learning_Outcome
  - Major
  - Major_And_Minor
  - Master_Exam
  - Mention_2
  - Minor
  - Module_Educational
    - Basic_Module
    - Complementary_Module
    - External_Module
    - Module
    - Obligatory_Module
    - Optional_Module
    - Orientation_Module1
    - Orientation_Module2
    - Orientation_Module3
    - Specialisation_Module
  - Optional_Course
  - Person
    - Professor
      - Thesis_Supervisor
    - Student
  - Semester
  - Soft_Skills
  - Study_Track
  - Subject
  - Teaching_Unit
  - Term_Description
  - Thesis
    - Bachelor_Thesis
    - Master_Thesis
  - Validation

Fig. 6. The class hierarchy of the Bowlogna ontology

Towards Evaluating an Ontology-Based Data Matching Strategy for Retrieval and Recommendation of Security Annotations for Business Process Models

Ioana Ciuciu, Yan Tang, and Robert Meersman

Semantics Technology and Applications Research Laboratory, Vrije Universiteit Brussel,
B-1050 Brussels, Belgium
{iciuciu,yan.tang,meersman}@vub.ac.be

Abstract. In the Trusted Architecture for Securely Shared Services (TAS^3) EC FP7 project we have developed a method to provide semantic support to the process modeler during the design of secure business process models. Its supporting tool, called Knowledge Annotator (KA), is using ontology-based data matching algorithms and strategy in order to infer the recommendations the best fitted to the user design intent, from a dedicated knowledge base. The paper illustrates how the strategy is used to perform the similarity (matching) check in order to retrieve the best design recommendation. We select the security and privacy domain for trust policy specification for the concept illustration. Finally, the paper discusses the evaluation of the results using the Ontology-based Data Matching Framework evaluation benchmark.

Keywords: ontology-based data matching, ontology, semantic annotation, knowledge retrieval, security constraints, security policy, business process model design, evaluation methodology.

1 Introduction and Motivation

The Knowledge Annotator (KA) tool has been designed to support process modelers in designing security-annotated business process models (BPM). This work has been done in the context of the EC FP7 Trusted Architecture for Securely Shared Services[1] (TAS^3) project, whose aim is to provide a next generation trust and security architecture for the exchange and processing of sensitive personal data. The TAS^3 architecture meets the requirements of complex and highly versatile business processes while enabling the dynamic, user-centric management of policies.

One of the challenges is to offer a secure business processes framework for sharing, accessing, and using personal data processing services in federated environments.

In order to make business processes secure, the business process model is annotated with security constraints which apply to authentication, authorization, audit logging, and other security issues. The business process management system (BPMS)

[1] http://www.tas3.eu/

K. Aberer, E. Damiani, and T. Dillon (Eds.): SIMPDA 2011, LNBIP 116, pp. 103–119, 2012.

transforms the security annotations into descriptive security policies or triggers process model extensions. It finally executes secure business processes by dedicated system components (e.g. these components allocate actors to activities, enforce data-specific authorizations, or trigger security-specific user involvements). This infrastructure guarantees that business processes are performed according to the annotated security constraints. In order to ensure interoperability between the different actors and components of the system, we provide a security ontology which explicitly documents the relationship between the core security concepts. One goal is to annotate all security-relevant business process specifications with a common, agreed upon conceptualization (ontology).

The KA tool ensures the correct specification of the security annotations, by supporting the process modeler with syntactically correct security concepts and with annotation recommendations. The recommendations are obtained by matching the knowledge stored in a knowledge base and an ontology of security constraints against the process modeler's request, specifying his design intent.

The paper focuses on the evaluation of the matching process for the specification of security annotations for security (trust) policies, applied to business process models. The evaluation is done with the Ontology-based Data Matching Framework (ODMF [1]) evaluation benchmark.

The rest of the paper is organized as follows: Section 2 provides background information on the ontology-based data matching strategy. Section 3 proposes an approach to applying ontology-based data matching for knowledge retrieval for annotating secure BPM. Section 4 presents the matching results and their interpretation. An evaluation methodology and the evaluation results are shown in Section 5. The related work of the paper is discussed in Section 6. We conclude the presented work and propose our future work in Section 7.

The main contribution of the paper therefore consists of (1) mining the user knowledge using natural language and an ontology-based data matching strategy; (2) retrieving similar patterns from the Knowledge and Ontology Base and proposing them to the user; and (3) evaluating the results using a generic evaluation benchmark.

2 Background

In this study we applied ontology-based data matching algorithms and a strategy in order to compute the similarity between the user request and the security-related knowledge represented by security constraints. The Ontology-based Data Matching Framework (ODMF) has been introduced in the EC FP6 Prolix[2] project for the purpose of competency matching. Ontology-based data matching (ODM [1,2]) is a new discipline of data matching and is ontology-based. The goal of ODM is to find the similarities between two data sets, each of which are annotated with one ontology and corresponds to one part of the ontology. There is one ontology in the particular problem, represented as a graph. This approach brings a new precision level thanks to the community (indirect) support.

[2] http://www.prolixproject.org/

The selected strategy for this research is the Controlled Fully Automated Ontology Based Data Matching Strategy (C-FOAM). C-FOAM is a hybrid strategy of ODMF, combining (1) string matching algorithms; (2) lexical matching algorithms and (3) at least one graph-based matching algorithm. C-FOAM is described in the following paragraphs.

C-FOAM

The C-FOAM strategy starts with combining two character strings representing context-object pairs. If the two character strings of the contexts are the same, data objects that belong to the same object type will be compared. To resolve the actual data objects stored in the ontology, the combination of both the context term and the object term is used.

In case the data object is denoted by several terms a lexical lookup is done taking synonyms into account. If a given object term could not be found in the ontology and lexicon, the best fitting data object is returned from the ontology using fuzzy matching based on string similarity (e.g. using JaroWinklerTFIDF algorithm [3,4]). The similarities between the given data object and the most similar data object in the ontology should be above a certain threshold, which is set in the application configuration of our tool. If the data object is found based on fuzzy matching then a penalty percentage will be used on the confidence value for the matching score.

C-FOAM is based on two main modules: (1) the *Interpreter* module and (2) the *Comparator* module, as shown in Fig. 1.

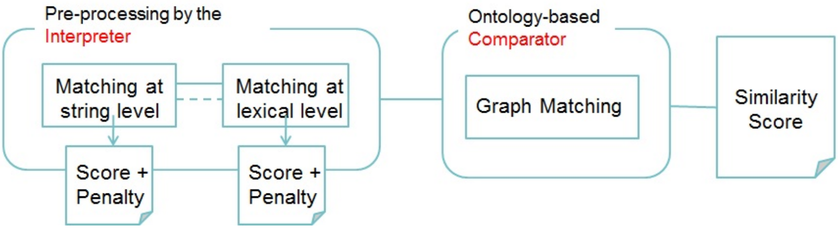


Fig. 1. C-FOAM model for ontology-based data matching

The *Interpreter* module makes use of the lexical dictionary, WordNet[3], the domain ontology and string matching algorithms to interpret end users' input. Given a term that denotes either (a) a concept in the domain ontology, or (b) an instance in the ontology, the interpreter will return the correct concept(s) defined in the ontology or lexical dictionary, and an annotation set of the concept.

The *Comparator* computes the similarity between two found data objects annotated with binary facts from the ontology base. A graph based algorithm or a combination of different graph-based algorithms (e.g. OntoGram, LeMaSt developed within ODMF) is used by the comparator to find the similarities between the two annotation sets. In case more than one graph algorithm is used, a positive percentage must be specified for each of them in the configuration file.

[3] http://wordnet.princeton.edu/

The *ontology* is modeled following the Developing Ontology Grounded Methodology and Applications (DOGMA [5]) framework. In DOGMA, the ontology is two-layered in order to make the reuse of facts easier. It is separated into 1) a *lexon* base layer (binary facts represented in semi-natural language) and 2) a *commitment* layer (defining constraints on the committing lexons). An example of lexon is $< \gamma, Security\ Annotation, has, is\ of, Parameter\ >$ which represents a fact that "within the context identified by γ, a Security Annotation has a Parameter and a Parameter is of a Security Annotation". A commitment on this lexon can be, i.e. each Security Annotation has at least one Parameter, which is a mandatory constraint.

The security annotations built on top of the BPM are semantically annotated with lexons from a dedicated security constraints ontology. The binary facts are disambiguated via a context handler.

3 Approach

The knowledge annotator is designed as a user-friendly system, intended to assist the process modeler during the specification of the security-specific constraints and to learn from the process modeler by using a dedicated knowledge base. This is realized by capturing the process modelers' modeling intentions via a user-friendly interface (UI) and by presenting him/her with recommendations (as shown in Fig. 2). The recommendations are determined by an ontology-based data matching operation between the user input, the security constraints ontology, user-defined dictionary, Synsets from lexical databases (e.g. WordNet) and the collected security annotations retrieved from the knowledge base. This operation and the evaluation of the outcome represent the focus of this paper and will be detailed in the following sections.

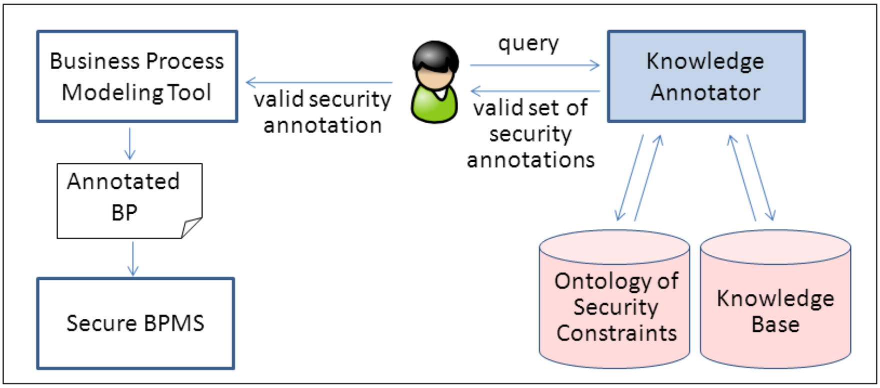

Business Process Modeling Tool
valid security annotation
query
valid set of security annotations
Knowledge Annotator
Annotated BP
Secure BPMS
Ontology of Security Constraints
Knowledge Base

Fig. 2. User-system interactions for the annotation of security constraints

3.1 The Knowledge Annotator. Recommender

The KA encapsulates several functions in a web service, which are supported by six architectural components: (1) the capturer – for capturing the user design intent;

(2) the annotator – for annotating objects with concepts from the security ontology; (3) the indexer – for indexing elements for efficient retrieval; (4) the retriever – for retrieving information; (5) the comparator – for comparing the user input with the knowledge base; and (6) the presenter – for presenting the user with recommendations and for user query specification. We refer to [6] for details regarding the overall architecture and functionality of the KA.

In order to understand the basic data element used by the KA to retrieve recommendations, we first give the definition of a security annotation.

Security Annotation. A security annotation is a text annotation attached to a BPMN element. The syntax of a security annotation is specified by an annotation term, followed by a list of parameters (mandatory or optional) with their corresponding values:

```
<<AnnotationTerm: list(parameter = value)>>.
```

Currently, our security language supports 'auditing', 'authentication', 'authorization', 'data and message flow security', 'delegation', and 'user interactions' [7].

The concept of security annotation, represented in DOGMA and modeled with Collibra Business Semantics Glossary[4], is illustrated in Fig. 3. A security annotation, specified as above, is applied to one or more BPMN elements.

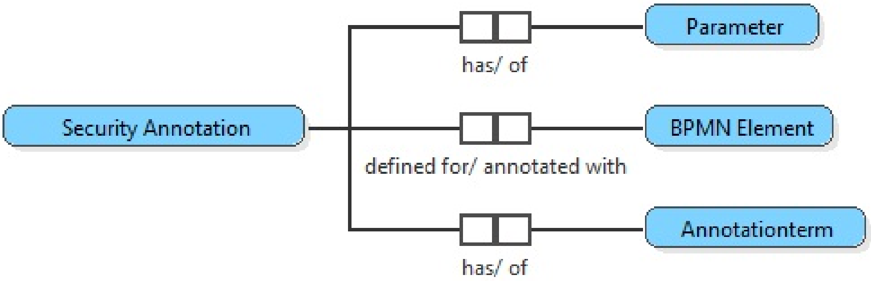


Fig. 3. Representation of the security annotation concept (Collibra Business Semantics Glossary screenshot)

Basic Data Object of the KA. According to the above definition, the basic data object used by the knowledge annotator is represented as a Security Annotation Term – Element – Parameter –Value (STEPV) object. The STEPV object encapsulates the five entities needed to completely define a security annotation: the security constraint, the BPMN element being annotated, the parameters and their corresponding values. In case the value of the parameters is ignored, the object will be referred to as STEP object.

[4] http://www.collibra.com/products-and-solutions/products/business-semantics-glossary

When the process modeler wants to define a security annotation, he fills in the fields corresponding to his design intent and knowledge (STEPV elements). This represents the input to the web service call. The system captures the input and analyzes it against the ontology base and the knowledge base in order to make the best fitted recommendations to the process modeler. The STEPV elements returned are considered valid annotations according to the constraints defined in the commitment layer of the ontology.

This study focuses on components (4) and (5) of the KA:

(4) *The retriever* component retrieves similar fragments from the knowledge base (e.g., all existing security annotations which share at least one common element with the input object). The similarity measure can be defined according to the user needs. For example, the user could only be interested in STEPV objects with a particular value for the 'name' parameter. The knowledge base contains semantic annotation instances (STEPV-Annotated objects) of the security constraints.

(5) *The comparator* component performs a matching operation in order to compare the process modeler's demand (input object) with the resulted elements retrieved from the knowledge base in the previous step.

3.2 C-FOAM for Recommendations

The advanced C-FOAM matching strategy was applied in order to match the user defined search patterns with the knowledge existing in the data repositories of the system (the security constraints ontology, the security upper common ontology, the knowledge base, WordNet Synsets, user-defined dictionaries, etc.). As previously explained, C-FOAM makes use of string matching, lexical matching and graph-based matching to calculate the similarity score. The following paragraphs illustrate the three matching techniques performed by C-FOAM when retrieving the recommended security annotations.

String Matching. Various string matching algorithms (SecondString and ODMF-specific), are used to calculate the similarity of two objects belonging to the same super-ordinate concept based on the string similarity of their description. For example, two security annotations descriptions 'Set Trust Policy name' and 'Set Trust Policy type' can be compared using different string matching algorithms. An ODMF-specific string matching algorithm based on fuzzy matching, ODMF.JaroWinklerTFIDF, can be used to compensate for the user typing errors (e.g. when the user types in 'Trsut' the algorithm can be used to find 'Trust').

Matching at string level is easy to implement and does not require a complex knowledge resource. A natural language description of the security annotation and/or its composing elements is sufficient.

The Interpreter component in C-FOAM uses string matching, in particular the JaroWinklerTFIDF algorithm.

Lexical Matching. The lexical matching algorithms calculate the similarity of two objects that belong to the same super-ordinate concept based on the semantic similarity of their descriptions. The object descriptions should be terminologically annotated. For example, the security annotation description 'Set Trust Policy name' could be annotated with the terms 'set', 'trust', 'policy', and 'name'. This set of terms may be then compared to a second set of terms to calculate the similarity between the two semantic descriptions, using the Jaccard similarity coefficient:

$$\text{Jaccard}(\text{Set1}, \text{Set2}) = |\text{Set1} \cap \text{Set2}|/|\text{Set1} \cup \text{Set2}|\,. \tag{1}$$

In addition to plain string matching techniques, linguistic information is used to improve the matching. Two techniques are used for improving the matching:

(1) Tokenization and lemmatization. *Tokenization* is the process of identifying the tokens, i.e. words and punctuation symbols, in a character string. *Lemmatization* is the process of determining the lemma of a given word. A lemma is the base form of a word as it appears in the index of a dictionary or a terminological database.

(2) An ontologically structured terminological database. In the form of a categorization framework, such a database is used in order to take into account synonyms and/or translation equivalents of a given term. Hypernyms or hyponyms may be used to take into account more generic or more specific terms of a given term. C-FOAM makes use of the concepts, concept relations, and terms based on WordNet and automatically tokenizes and lemmatizes the description of the security annotations.

Examples of lexical matching based on synonyms and user dictionaries are illustrated below in Table 1:

Table 1. Lexical synonyms defined for concepts in the ontology

Synonyms	**Concept in the ontology**
'user' 'user interaction' 'interaction'	'requestToUser'
'select service' 'choice of service' 'choice'	'service selection'
'task' 'subprocess' 'flow element'	'activity'
'trigger' 'message'	'event'

Graph Matching. The graph matching algorithms are used to calculate the similarity between two objects that represent two sub-graphs of the same (ontology) graph. The similarity score can be used also to find related objects for a given object, as it is the case in this paper. The similarity of two objects is calculated based on their classification, properties and semantic relations. For example, the comparator module of the KA computes the similarity between two security annotations. In the same way, using classification information of objects, the relations between objects and the properties of objects, it is possible to find related security annotations for a given security annotation. For example, if the user wants to find relevant BPMN elements applying to the security annotation type 'Delegation'.

The graph is a semantic graph (ontology), in which concepts in the ontology are represented as vertices and semantic relations between concepts are represented as arcs. The arcs are bi-directed and correspond to the role and co-role in a binary fact type (lexon).

For this technique, a domain ontology and an application ontology must be available. In our case, the domain ontology is the security concepts ontology and the application ontology is the ontology of security constraints. Both ontologies act as the model for the rule-based reasoning which applies forward chaining to infer new knowledge, based on the existing knowledge expressed in the knowledge base.

For the purpose of secure business process models design, we take as reference of comparison the security annotation (the STEPV object), with its corresponding components, as described above. In order to find a correct security annotation, the system compares all the elements specified by the user (completely or partially) for the desired security annotation with the existing elements in the ontology and the knowledge base and retrieves a set of related security annotations, ranked according to the calculated similarity score. An example will be given in the next section.

Lexon Matching Strategy (LeMaSt) is applied by the Comparator module of C-FOAM in order to compute the similarity between the user input and the knowledge in the knowledge base and in the ontology. LeMaSt calculates the similarity of two security annotations based on the semantic similarity of their descriptions represented as a set of lexons. Therefore, this technique demands that the two object descriptions are annotated with lexons (elementary facts that are accepted to be true within the context of that object). For example, the security annotation description << Set Trust Policy name="$name" type="$type" insertplace(*) ="$insertplace" role(*)="$role" >> can be annotated with the lexons illustrated in Table 2:

Table 2. Security annotation 'SetTrustPolicy' modeled with DOGMA

Context	Head	Role	Co-role	Tail
SetTrustPolicy	SetTrustPolicy	has_parameter	parameter_of	name
SetTrustPolicy	SetTrustPolicy	has_parameter	parameter_of	type
SetTrustPolicy	SetTrustPolicy	has_optional_parameter	optional_parameter_of	insertplace
SetTrustPolicy	SetTrustPolicy	has_optional_parameter	optional_parameter_of	role

To calculate the similarity of the two security annotations (STEPV objects) that are annotated with lexons, we calculate the similarity of their corresponding lexon sets. For this purpose we extended the Jaccard similarity coefficient (see Equation 1) to account for partial overlap of two lexons:

$$\begin{gathered} \mathrm{C}=\sum\nolimits_{i=1}^{n} x_i/n, \quad 0 \le x_i \le 1 \\ S=\sum\nolimits_{i=1}^{n} C_i \times S_i, \quad 0 \le S_i \le 1 \\ \sum\nolimits_{i=1}^{n} C_i = 1\,. \end{gathered} \tag{2}$$

The Jaccard similarity scores are used to calculate the contribution score C using Equation 2. The final score S is the average scores of the lexons. For each lexon, x_i is a contribution score depending on the matching items. For example, two lexons that have the same head, role, co-role and tail term contribute 100% to the result (x_i=1). If the lexons have the same head and tail term but different role and co-role they contribute for 50% (x_i=0.5). If the lexons have the same head or tail term and the same role and co-role they contribute for 50% (x_i=0.5). If the lexons only have the same head or tail term they contribute for 25% (x_i=0.25).

4 Results and Analysis

An annotation scenario was created in order to analyze the behavior of the KA tool. The scenario consists of three user requests (queries) composed of different elements of security annotations: security annotation name, annotation term, parameter name and BPMN element. Depending on the accuracy of the user input, the matching is performed at string, lexical or graph level. The test data (user input, KA recommendations, matching scores and justification) is given in Table 3.

Let us take the ‘Set trust policy’ annotation to illustrate the matching process. Via the UI, the process modeler can specify a desired STEPV object by indicating the parts that are known to them. For this specific example, the user input concerns the Security Annotation and the BPMNElement fields. The STEPV element is therefore specified only by two fields out of five. The value parameter is excluded from this example, for simplification. In this case, we are dealing with STEP objects. The system interprets the user input by performing matching operations at different levels and infers the correct most similar annotation. The user input and the system result are illustrated in Fig. 4.

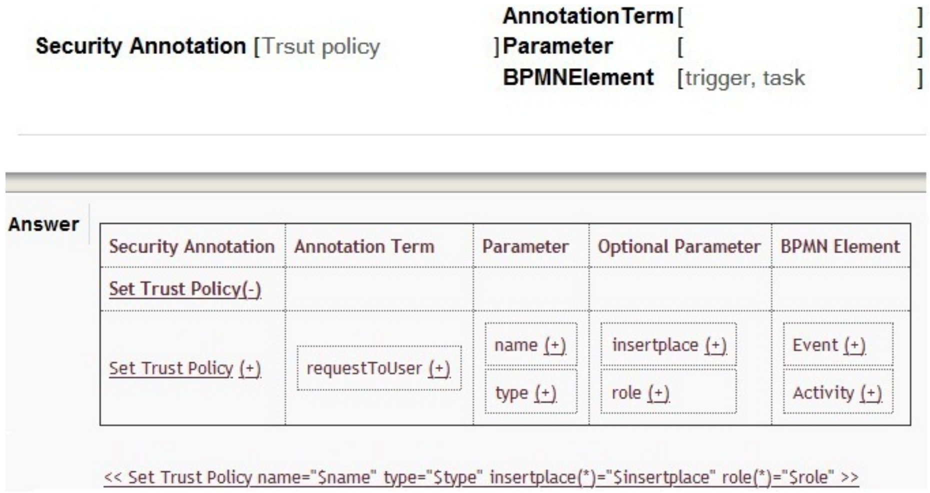
Let us now analyze how these results were inferred by the system. The process starts with the user specifying fields of the STEP object. He indicates ‘Trsut policy’ and ‘trigger, task’. These values are resolved by performing matching at different levels. In case of ‘Trsut policy’, C-FOAM performs a matching operation at string level by applying JaroWinkler and finds the correct string ‘Set Trust Policy’ in the ontology. In

Table 3. The results of the annotation scenario

User Input				**KA Recommendation SA<<AT: list (param = value)>>**	**Score**	**Remarks**
SA	**AT**	**P**	**BPMNE**			
'Trsut policy' (user input text)	-	-	Trigger, task	Set Trust Policy <<request ToUser name="$name" type="$type" insertplace(*)="$insertplace" role(*)="$role">>	0.88	[TYPO+ SYNONYM+ ONTOLOGY] 'Set Trust Policy' 'Activity' Lexon term String matching Lexical matching Graph matching
-	Interaction	-	Task	User Consent <<requestToUser type="Consent" insertplace(*)="$insertplace" target="$target" display="$display" role(*)="$role">> User Assignment <<Assignment type="user" user="$user">>	0.52 0.13	[SYNONYM+ ONTOLOGY] 'Request to User' 'Activity' Lexon term Lexical matching Graph matching
'Selecr service' (user input text)	-	-	Trigger	Service Selection <<requestToUser type="SelectService" insertplace(*)="$insertplace" display(*)="$display" role(*)="$role">> Select Data Policy <<requestToUser type="SelectDataPolicy" role(*)="$role" target="$target" insertplace(*)="$insertplace" display="$display">> Select Trust Policy <<requestToUser type="SetTrustPolicy" policy="$policy" role(*)="$role" insertplace(*)="$insertplace">>	0.8 0.5 0.45	[TYPO+ SYNONYM+ ONTOLOGY] 'Service selection' 'Activity' Lexon term String matching Lexical matching Graph matching

case of ‘trigger’ and ‘task’, they are matched to the correct concepts ‘event’ and ‘activity’ respectively in the ontology, by performing matching at lexical level, using WordNet and a user-defined dictionary (see Table 1) to resolve the synonymy.

Once the correct concepts in the ontology are found together with their corresponding annotation sets (lexons), C-FOAM starts performing a matching step at graph (ontology) level, by applying LeMaSt. LeMaSt compares the annotation set corresponding to the user input with the ontology of security constraints and infers the most similar annotation, which in this case is the following:



Security Annotation [Trsut policy]

AnnotationTerm [ ]

Parameter [ ]

BPMNElement [trigger, task]

Answer

Security Annotation	Annotation Term	Parameter	Optional Parameter	BPMN Element
Set Trust Policy(-)				
Set Trust Policy (+)	requestToUser (+)	name (+) type (+)	insertplace (+) role (+)	Event (+) Activity (+)

<< Set Trust Policy name="$name" type="$type" insertplace(*)="$insertplace" role(*)="$role" >>

Fig. 4. User query for retrieving the ‘Select Trust Policy’ security annotation and the result

<< Set Trust Policy name="$name" type="$type" insertplace(*)="$insertplace" role(*)="$role" >>.

This annotation is presented to the user as a recommendation. The user can further refine his search by modifying fields of the retrieved STEP object.

5 Evaluation Results

The Knowledge Annotator tool is evaluated based on a generic evaluation method [1] which adapts the principles in the methodologies for program evaluation [8] and purpose-oriented evaluation [9].

According to the basic principle of the program evaluation methodologies, the evaluation methodology needs to help the KA tool to enhance its functions and/or services and also help to ensure that it is delivering the correct list of recommended annotations.

According to the principles in the purpose-oriented evaluation methodologies:

- The evaluation process must be able to determine what information exists in the process of the KA tool, which is important so that engineers can analyze the processing information;

- The evaluation process must test and collect continuous feedback in order to revise the process;
- The evaluation must have a precondition analysis and a post-condition analysis of the evaluated system;
- End users must be able to judge the outcome of a system based on the evaluation methodology.

Accordingly, we have developed an evaluation method that can be used to evaluate any of the matching strategies in ODMF. Its process is described in Fig. 5.

Design a generic use case step. The step of designing a generic use case is the preparation step. In this step the problem is scoped and clear requirements are initialized for a viable use case. Terms and tests are gathered and analyzed from the test beds' materials. The output of this step is a report describing a generic use case.

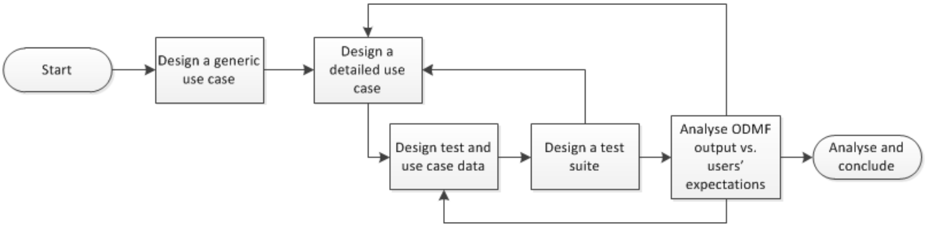

Start
Design a generic use case
Design a detailed use case
Design test and use case data
Design a test suite
Analyse ODMF output vs. users' expectations
Analyse and conclude

Fig. 5. A generic ODMF matching strategy evaluation method

Design a detailed use case step. Here the problem is specified. The design terms from the previous step are designed by specifying types of information used by the KA tool (e.g. process information). We also analyze preconditions and post-conditions of the use case. The output of this step is a report containing a detailed use case.

Design test and test and evaluation data step. In this step we design the test data that are used by the KA tool (not by end users). The output of this step is a report containing a list of test and evaluation data.

Design test suite. In this step a knowledge engineer designs a user test suite, the data of which need to be provided by an expert. The test suite is designed based on the results from the first two steps.

Analyze ODMF output vs. users' (experts') expectations step. The output of this step is a report of comparison (KA tool's similarity scores vs. expert expected similarity scores).

Analyze and conclude step. This step is to analyze the comparison report that is produced in the previous step and draw valuable conclusions. The output of this step is a report of comparison analysis and conclusion.

We focus on the last two steps in order to evaluate the results of the KA against experts' expectations. For this, we use the satisfactory rate. The satisfactory rate is calculated based on the similarity scores generated by the KA and the experts' expected relevance levels in the test suite. The relevance levels provided by the experts need to be correctly interpreted in order to calculate the satisfactory rate. The average score provided by the KA is 0.5, the maximum score is 1 and the minimum score is 0. Therefore, the scale of the similarity scores is [0, 0.2]. We equally split is as shown below:

- Relevance level 5: similarity score > 0.8;
- Relevance level 4: similarity score > 0.6 and <= 0.8;
- Relevance level 3: similarity score > 0.4 and <= 0.6;
- Relevance level 2: similarity score > 0.2 and <= 0.4;
- Relevance level 1: similarity score <= 0.2.

If a similarity score falls in the range, then we say that the similarity score is "completely satisfied". If it does not fall in the range, then we need to calculate the bias. The bias of the evaluation set is calculated as the minimum value of low boundary bias and high boundary bias, which are calculated as shown below:

Pseudo-code for Computing the Bias of the Evaluation Set

```
IF (Similarity Score < Low Boundary)
      THEN Low Boundary Bias = Low Boundary - Similarity Score
      ELSE Low Boundary Bias = Similarity Score - Low Boundary
IF (Similarity Score < High Boundary)
      THEN High Boundary Bias = High Boundary - Similarity Score
      ELSE High Boundary Bias = Similarity Score - High Boundary.
```

For instance, if the similarity score for relevance level 4 is 0.61, then the low boundary bias is 0.61 – 0.6 = 0.01and the high boundary bias is 0.8 – 0.61 = 0.19. The bias is 0.01 (the smallest value in {0.01, 0.19}).

If the bias is less than 0.2 (one interval), then we say that this similarity score is "satisfied". If it is more than 0.2 and less than 0.4 (two intervals), then we say it is "not really satisfied". All the remaining scores are considered "completely unsatisfied".

For the particular case of this paper, the results of the comparison between the KA similarity scores and the experts' expected scores (see Fig. 6) are as follows: out of six recommended annotations, one similarity score completely satisfies the experts' expectations; four similarity scores satisfy the experts' expectations; only one score does not really satisfy the experts' expectation; there are zero completely unsatisfied similarity scores.

The satisfactory rate is 83%: 16% completely satisfied + 67% satisfied.

The results of the evaluation are recorded in Table 4.

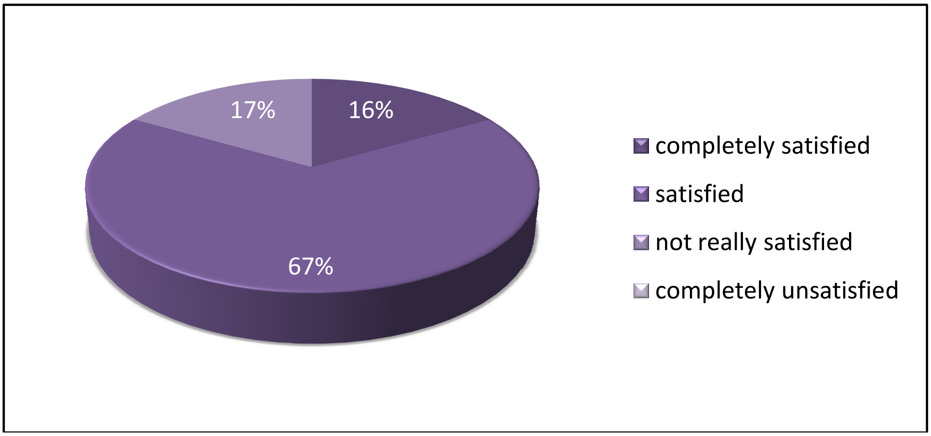


Fig. 6. C-FOAM similarity scores vs. expert expected scores

Table 4. Conclusion of evaluating the Knowledge Annotator

Knowledge Annotator using C-FOAM
1. Difficulty of managing the required knowledge resource.
Usage level basic to professional. Knowledge engineers need to know how to configure the penalty scores and understand the meaning of these scores.
2. Difficulty of using the matching strategy
C-FOAM is the most difficult of all the ODMF strategies because it is the composition of matching algorithms and/or other matching strategies (e.g. LeMaSt & JaroWinkler).
3. Satisfactory rate
Satisfactory rate is 83% (completely satisfied and satisfied, see Fig. 6).
What affects the matching score
Any factors that affect the selected algorithms are counted as the factors that affect the similarity scores. In addition, the two penalty values are the factors that affect the final similarity scores.
Advantage
The advantages of C-FOAM contain the advantages of all the combined algorithms and strategies.

6 Related Work

Many ontology matching approaches exist in ontology engineering (OE) [10]. Several EU projects, such as Knowledge Web [11] or OpenKnowledge [12] invested effort into the creation of algorithms and tools for ontology matching/integration.

Our approach is different from ontology matching or ontology integration and can be considered as a subdomain of data matching, based on ontology. In our problem setting, there exists only one ontology. ODMF finds similarities between two data sets, each of which corresponds to one part of the ontology. Classical methods, such as using linguistic methods for concept searching, using WordNet as the external dictionary and applying graph matching principles are included in our work. ODMF builds upon these techniques while designing and implementing an innovative generic matching framework.

Regarding the recommender systems used in business process modeling, our approach uses ODMF for security annotations retrieval expressed in natural language in order to support the business modeler. This is one of the contributions of this paper.

Betz [13] proposes an approach for the automatic user support based on an auto completion mechanism during the modeling process (where business processes are represented as Petri Nets). Born [14] presents a tool for the user-friendly integration of domain ontology information in the process modeling, through match matching and filtering techniques. A similar approach based on linguistic analysis of process element labels and of the concept names is presented in [15] in order to support process modelers with annotation suggestions.

In this paper, we have illustrated how to use the generic evaluation method [1], which adapts the principles in the methodologies for program evaluation [8] and purpose-oriented evaluation [9], to evaluate the matching result. We can as well use other classic evaluation methods, such as Turing test, for the evaluation. We are interested in the further investigation on how we can extend our evaluation method by involving different evaluators and how to take the evaluation result for adjusting the strategy.

The approach presented in this paper contributes to the state of the art by focusing on capturing and evaluating the user knowledge. Regarding the evaluation methodology and the delivery of security annotations suggestions, they are based on the ontology based data matching methodology, which contributes from the knowledge of the community (e.g. members of multiple organizations sharing a common goal).

7 Conclusion and Future Work

This paper is focusing on the matching strategies used by an annotation system while retrieving recommendations to assist the business process modeler into designing secure business processes. The matching strategy is ontology-based and belongs to ODMF methodology for ontology-based data matching. The applied strategy benefits from all its composing algorithms applied at string, lexical and graph level in an iterative refinement process and from the user-system interactions.

The similarity score between the user input and the knowledge stored in the ontology and knowledge bases is used by the system in order to infer the best fitted security annotations recommendations. An evaluation methodology has been developed to evaluate the (ontology-based) matching strategy used. The evaluation methodology shows a satisfactory rate of 83% when comparing the results delivered by the annotator with the experts' expectations.

The knowledge (i.e. security annotations) is modeled using the DOGMA ontology, which has the advantage of being grounded in natural language.

A future work is to consider various statistics during the recommendation which capture the user's characteristics and annotation behavior (e.g. to infer the user context and knowledge from his inputs). The enrichment of the knowledge base and a mechanism for checking the correctness of the specified parameter values is work in progress. The outcome of the annotator and the quality of the recommendations are dependent on the knowledge base updates and on the domain expert responsible for managing the knowledge base.

Acknowledgments. This research is supported by the EC FP7 TAS^3 (Trusted Architecture for Securely Shared Services) project. The authors would like to thank all TAS^3 project partners for their contribution to the research.

References

1. Tang, Y., Meersman, R., Ciuciu, I.G., Leenarts, E., Pudney, K.: Towards Evaluating Ontology Based Data Matching Strategies. In: Proceedings of 4th IEEE Research Challenges in Information Science (RCIS 2010), Nice, France, pp. 137–146 (2010) ISBN: 978-1-4244-4839-5
2. De Baer, P., Tang, Y., Meersman, R.: An Ontology-Based Data Matching Framework: Use Case Competency-Based HRM. In: Proceedings of the 4th Int. OntoContent 2009 Workshop, OTM, Portugal. LNCS, pp. 514–523 (2009)
3. Jaro, M.A.: Probabilistic linkage of large public health data files (disc: P. 687–689). Statistics in Medicine 14, 491–498 (1995)
4. Cohen, W.W., Ravikumar, P.: Secondstring: An open source java toolkit of approximate string-matching techniques (2003), http://secondstring.sourceforge.net
5. Spyns, P., Tang, Y., Meersman, R.: An ontology engineering methodology for DOGMA. Journal of Applied Ontology 3(1-2), 13–39 (2008)
6. Ciuciu, I., Zhao, G., Mülle, J., von Stackelberg, S., Vasquez, C., Haberecht, T., Meersman, R., Böhm, K.: Semantic Support for Security-Annotated Business Process Models. In: Halpin, T., Nurcan, S., Krogstie, J., Soffer, P., Proper, E., Schmidt, R., Bider, I. (eds.) BPMDS 2011 and EMMSAD 2011. LNBIP, vol. 81, pp. 284–298. Springer, Heidelberg (2011)
7. Mülle, J., von Stackelberg, S., Böhm, K.: A Security Language for BPMN Process Models. Technical Report. Karlsruhe Institute of Technology (KIT), no. 2011-09, Karlsruhe (2011)
8. Patton, M.Q.: Qualitative Research and Evaluation Methods, 3rd edn. Sage Publications, Inc., London (2002) ISBN: 0-7619-1971-6
9. Bhola, H.S.: Evaluating "Literacy for development" projects, programs and campaigns: Evaluation planning, design and implementation, and utilization of evaluation results. UNESCO Institute for Education; German Foundation for International Development, Hamburg, Germany (1990)
10. Jain, P., Hitzler, P., Sheth, A.P., Verma, K., Yeh, P.Z.: Ontology Alignment for Linked Open Data. In: Patel-Schneider, P.F., Pan, Y., Hitzler, P., Mika, P., Zhang, L., Pan, J.Z., Horrocks, I., Glimm, B. (eds.) ISWC 2010, Part I. LNCS, vol. 6496, pp. 402–417. Springer, Heidelberg (2010)
11. Knowledge Web FP6-507482, http://knowledgeweb.semanticweb.org

12. OpenKnowledge FP6-027253, http://www.iiia.csic.es/en/project/open-knowledge
13. Betz, S., Klink, S., Koschmider, A., Oberweis, A.: Automatic User Support for Business Process Modeling. In: Proceedings of the Workshop on Semantics for Business Process Management at the 3rd European Semantic Web Conference 2006, Budya, Montenegro, pp. 1–12 (2006)
14. Born, M., Dorr, F., Weber, I.: User-friendly Semantic Annotation in Business Process Modeling. In: Proceedings of the Workshop on Human-friendly Service Description, Discovery and Matchmaking (2007)
15. Di Francescomarino, C., Tonella, P.: Supporting Ontology-Based Semantic Annotation of Business Processes with Automated Suggestions. In: Halpin, T., Krogstie, J., Nurcan, S., Proper, E., Schmidt, R., Soffer, P., Ukor, R. (eds.) BPMDS 2009. LNBIP, vol. 29, pp. 211–223. Springer, Heidelberg (2009)

FP-Growth in Discovery of Customer Patterns

Jerzy Korczak[1] and Piotr Skrzypczak[2]

[1] Wrocław University of Economics, Poland
[2] Delikatesy Alma, Wrocław, Poland
53-345 ul. Komandorska 118/120, Wrocław, Poland
jerzy.korczak@ue.wroc.pl, piotrek.skrzypczak@gmail.com

Abstract. The paper describes a knowledge discovery platform and a novel process for finding association rules based on the algorithm FP-Growth and its variants. Built software solution has been optimized in terms of memory usage and computation time as well as the impact of all modifications made to the whole process of rules discovery The process of rule discovery is illustrated on a real database containing transactions of customers of the e-shop Delicatessen Alma24.

Keywords: Mining of association rules, analysis of customers' transactions, improvement in FP-Growth performance.

1 Introduction

One of the most popular and widely used methods of finding customers' shopping patterns are the methods based on algorithms of association rules. Over several years, a number of search algorithms have been developed for association rules in data sets [Hand, 2005; Kotsiantis, 2006; Morzy, 2010; Pasztyła, 2010]. Many of them have evolved, some proved to be less useful, because of their performance on large data sets and too great demands on available memory. One interesting proposal is the FP-Growth (frequent pattern growth) algorithm developed by J. Han, H. Pei and Y. Yin [Han, 2000; Han 2004]. FP-Growth uses an extended prefix-tree structure, called FP-tree, to store the customer transactions in a compressed form. This algorithm is fast and scalable. The publication of J. Han showed that FP-Growth performance surpasses other popular methods of searching for association rules, such as Apriori or Tree Projection algorithms. The papers of [Zaki, 1997], and [Borgelt, 2005], showed that this algorithm has better performance than Eclat and Relim.

The popularity and effectiveness of the FP-Growth algorithm was appreciated in many studies, in which to improve its efficiency many changes have been proposed to the original algorithm [Gyorödi, 2003; Racz, 2004; Zaki 1997]. These changes are mainly related to accelerating the construction of the FP-tree and its reduction of computing time as well as memory complexity.

The first modification was proposed by C. Gyorödi [Gyorödi, 2003]. It addressed two problems in the FP-Growth algorithm, namely, that the resulting FP-tree is not unique to the same "logical" database, and that in order to create the FP-tree two complete scans of the database are required. The developed algorithm DynFP-Growth

K. Aberer, E. Damiani, and T. Dillon (Eds.): SIMPDA 2011, LNBIP 116, pp. 120–133, 2012.

solved the first problem by introducing the lexicographical order of support, thus ensuring the uniqueness of the FP-trees for different but "logically equivalent" databases. In order to solve the second problem, the algorithm changes dynamically the order of elements of the FP-tree by performing the "promotion" (offset) to the higher-order one of the smallest items detected. An important feature of this solution is that it is not necessary to rebuild the FP-tree when the database is updated.

The way to reduce the size of the tree is implemented in the algorithm of FP-Bonsai [Gyorödi, 2003]. The FP-tree is pruned using the data reduction technique ExAnte [Bonchi, 2003]. The originality of this solution is based on the rejection at the first scan of items whose support is less than the required minimum value. In addition, after the first scan and creation of a table header, the data set is sorted by the value of support and re-entries when the support less than the assumed minimum value are rejected. Thanks to these modifications the size of the FP-tree is reduced several times. The pruned FP-tree called FP-Bonsai improves the efficiency of the algorithm.

The latter essential modification refers to the time and memory complexity of the FP-Growth algorithm. The NONORDFP algorithm [Hand, 2005] modifies the structure of the FP-tree, which is thus more compact and does not need to be rebuilt for each conditional step. The new FP-tree representation in the memory assures faster search, faster allocation, and possibly better projection.

The paper will present results of pilot studies of consumer behavior of one of Wroclaw's Alma Delicatessen stores, based on sales from August and September 2009 to September 2009 [Skrzypczak, 2010], and the studies on a new project carried out on the data from August 2009 to January 2010. In the project, the MySQL database system, the authors' software DM Cafe, and RapidMiner package (http://www.rapidminer.com/) have been used to implement the FP-Growth algorithm. The main aim of the platform's development was to assist decision makers in search of interesting and nontrivial association rules that can allow better understanding of customers' profiles and their preferences, and improve sales performance.

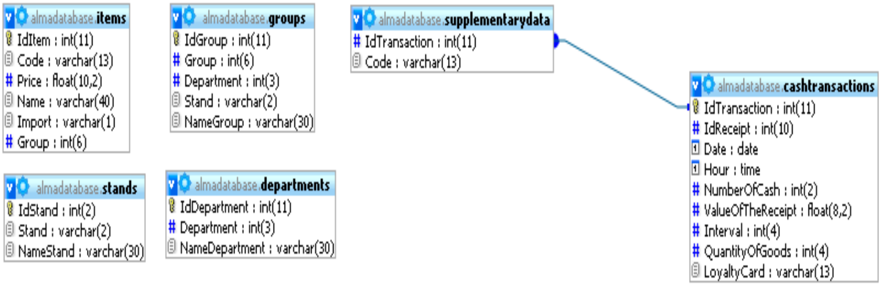
The article is divided into four sections. The next section describes the algorithm of FP-Growth and characteristics of its main parameters. In the third section, a database containing transaction data and information on commodities is presented. The fourth section describes the process of extracting association rules using RapidMiner [Bereta, 2010]. A modified process of rule discovery, implemented in [Skrzypczak, 2010], is detailed, as well as its impact on system performance.

2 Database of Customer Transactions and Mining Algorithm

The aim of the project was to design a platform to integrate the existing transactional system with new functionalities of rule extraction. In the experimental platform three functional components have been identified, namely, a transactional database with the store commodities tables, software for exchanging information between the stock information system and cash management, and an application to mine association rules.

The database, managed by the MySQL server, consists of the following tables: Items, Groups, Departments, Stands, Cash transactions, and Supplementary Data (Fig. 1).

To exchange information between systems and the database, DM Cafe is applied. It is a program to write data to the stock information system and the cash management whose data are used in the study. Data for the test are read directly from the database using the appropriate SQL query in RapidMiner.



almadatabase.items
IdItem : int(11)
Code : varchar(13)
Price : float(10,2)
Name : varchar(40)
Import : varchar(1)
Group : int(6)

almadatabase.groups
IdGroup : int(11)
Group : int(6)
Department : int(3)
Stand : varchar(2)
NameGroup : varchar(30)

almadatabase.supplementarydata
IdTransaction : int(11)
Code : varchar(13)

almadatabase.cashtransactions
IdTransaction : int(11)
IdReceipt : int(10)
Date : date
Hour : time
NumberOfCash : int(2)
ValueOfTheReceipt : float(8,2)
Interval : int(4)
QuantityOfGoods : int(4)
LoyaltyCard : varchar(13)

almadatabase.stands
IdStand : int(2)
Stand : varchar(2)
NameStand : varchar(30)

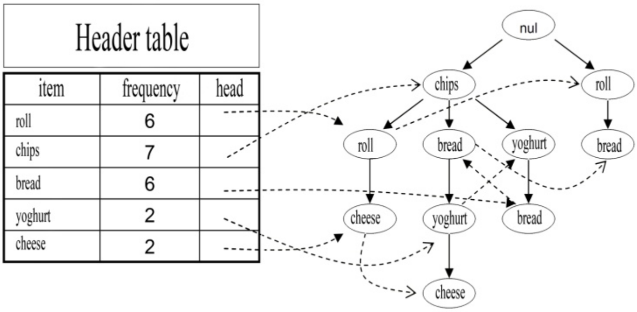
almadatabase.departments
IdDepartment : int(11)
Department : int(3)
NameDepartment : varchar(30)

Fig. 1. Database schema

Generally, a database contains a set of customers transactions $T_1,\ldots, T_n$, where each transaction T_i describes a set of items bought by a customer. In the case study the database contains over 470 thousand records in six tables. Most of the data are collected in the table Supplementary Data; there are codes assigned to the transaction IDs. There are over 370 thousand records for the entire store, and over 25 thousand Alma24 records (Alma24 is an online store). The table Items stores information about more than 63 thousand of Alma Delicatessen commodity codes. During the evaluation period, the point of sales registered over 39 thousand transactions, of which more than 1,000 belonged to Alma24.

To discover customer basket patterns, the FP-Growth algorithm has been used [Han, 2000]. The algorithm is looking for the complete set of frequent patterns understood as patterns with the occurrence frequency no less than a predefined by managers minimum support ratio. Among these frequent patterns, the confident ones are selected to create classification rules. These rules are used to predict items to be bought and class labels for future transactions.

The algorithm contains two basic steps: compression of the data set in a form of FP-tree and mining of association rules from FP-tree. The FP-tree is built using two passes over the database. In the first pass, the algorithm searches the database for all frequent 1-item and then removes infrequent items from the transaction T_i. As a result, a modified set of transactions $T^* = T_{1^*},\ldots, T_{m^*}$ is created consisting of only frequent 1-item sets. Then a set of transactions is sorted in descending order according to the support ratio of each transaction and transformed into a compact tree structure called a FP-tree. The FP-tree is a rooted acyclic graph with non-labeled vertices. The root graph has a label 'null', the remaining graph vertices, both internal nodes and leaves, represent 1-item sets. Each graph node, except the root, is linked to the label that represents a 1-item set and the counter of transactions, representing the number of transactions supporting a given set (Fig. 2). In the second scan of the transaction database the FP-tree is constructed and then can be used for mining frequent customer basket patterns. It is important to note that the FP-tree efficiently compresses the database and avoids costly and repeated data scans as Apriori type algorithms. More information about the theoretical foundations of the FP-tree construction can be found in [Han, 2000; Han, 2004].



Header table

item	frequency	head
roll	6	
chips	7	
bread	6	
yoghurt	2	
cheese	2	

Fig. 2. Example of the header table and the corresponding FP-Tree

To explore information stored in an FP-tree and extract the complete set of frequent patterns, the algorithm FP-Growth, has been applied [Han, 2004]. The FP-Growth starts to mine the frequent patterns 1-itemset and progressively grows each such itemset by mining its conditional pattern-base. A conditional pattern-base is a set of patterns that co-occur with a particular node in a given path. All the computed frequent patterns related with node a_i creates a small FP-tree, called a_i-conditional FP-tree and denoted as "*FP-tree*/a_i". The processes of construction of conditional pattern-bases and conditional FP-trees are carried out by pattern growth recursively.

The pseudo-code of the algorithm FP-Growth is given below:

```
Procedure FP-Growth(Tree, α)
// α is an itemset in transactional database
// β is an itemset in α's conditional pattern-base
{
if Tree contains a single prefix path                // Mining single prefix-path FP-tree
   then {
         let P be the single prefix-path part of Tree;
         let Q be the multipath part with the top branching node replaced by a null root;
         for each combination (denoted as β) of the nodes in the path P do
            generate pattern β ∪ α with support = minimum support of nodes in β;
         let freq_pattern_set(P) be the set of patterns so generated; }
   else let Q be Tree;
for each item a_i in Q do {                          // Mining multipath FP-tree
    generate pattern β = a_i ∪ α with support = a_i .support;
    construct β's conditional pattern-base and then β's conditional FP-tree Treeβ ;
    if Treeβ = ∅
       then call FP-Growth(Treeβ, β);
     let freq pattern set(Q) be the set of patterns so generated; }
return(freq_pattern_set(P) ∪ freq_pattern_set(Q) ∪ (freq_pattern_set(P) × freq_pattern_set(Q)))
```

The algorithm has two initial parameters: *Tree = FP-tree*, and *α = null*. If the *FP-tree* has only a single path *P*, then for each combination of *β* vertex path *P* is created a set of $\beta \cup \alpha$ with the support equal to the minimum support of items belonging to the set *β*. If the *FP-tree* contains more than one path, then for each element belonging to the array, α_i *Tree* header is created with a set of $\beta = \alpha_i \cup \alpha$ supporting corresponding elements α_i. Next is generated a conditional pattern base of *β* and conditional *FP-tree* pattern of *β*, denoted *Treeβ*. After this step, it is verified whether *Treeβ* is empty or not. If it is empty, the algorithm is ended, otherwise the procedure FP-Growth is restarted with parameters of *Tree* = *Treeβ*, and $\alpha = \beta$. The last line of the procedure returns the three sets the generated frequent patterns from *P, Q*, and $P \times Q$.

3 Process of Association Rules Discovery

The rule extraction process depends not only on the size of the database, but also on two very important measures of rule interestingness: support and confidence ratios. A support ratio defines the frequency of a given combination of items in the database, and a confidence ratio that reflects the likelihood that a particular rule appears. The threshold values of the support and the confidence are set by the managers to indicate which pattern or group of items can be considered as a frequent pattern or frequent itemset. Depending on these values, a different number of frequent sets (by increasing or decreasing the value of the support ratio) and association rules (by changing the value of the confidence ratio) may be generated.

In our application the customer transactions are read by RapidMiner using SQL queries. The transactions are then transformed into a matrix and passed to the FP-Growth algorithm. The matrix contains the items, and 0 and 1 (0 means no item occurrence of the transaction, 1 - the item was in the shopping cart) is needed to establish the structure of the FP-Tree. Figure 3 shows the process created to discover rules in the transactional data of the online store Alma24.

The approach to rule discovery presented in this article can be applied to any transactional database system. Just a suitable SQL query has to be used that returns two columns: transaction ID and the name of the item, group, department, etc. By default, the process finds all association rules for given support and confidence ratios. In addition, the association rules can be found for specific items, group or department; for example, if a customer shopping cart contains tomato, what else is there? The system is able to answer the specific question asked by the stand manager or market data analyst.

After preprocessing and searching frequent itemsets, association rules are created. When the process finishes, the results are visualized.

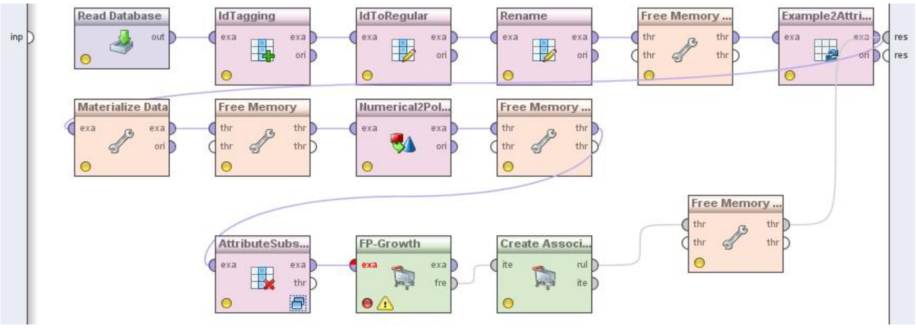
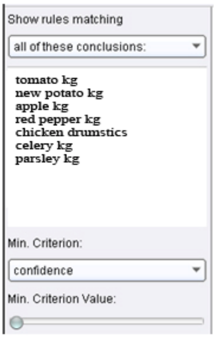


Fig. 3. Schema of the rule discovery process



Show rules matching

all of these conclusions:

tomato kg
new potato kg
apple kg
red pepper kg
chicken drumstics
celery kg
parsley kg

Min. Criterion:

confidence

Min. Criterion Value:

Fig. 4. Changing the display mode

The obtained association rules can be presented in a table; if required, they may be sorted. In addition, the display mode can be changed using the sliders and drop-down list located on the left side of the results (Fig. 4). It is also possible to visualize the rules as a graph (Graph View option), as well as text, similar to those written to the file by the operator Write as Text. In the paper, the rules are presented in the tables ordered by Confidence level. If needed, part or all of the rules can be exported to various external formats, such as a PDF, XML.

The data to be analyzed represented the Alma24 customer purchases from August to September 2009. Each customer transaction contained information on purchased items, ie., commodity code, quantity, price, value of purchases, information about the possession of Connaisseur Club card (sort of loyalty card), and also the mode of payment (cash, card, gift certificate, bank transfer). The study was conducted at the level of the item code for the customers whose shopping cart value was greater than 200 zł (ca 50 euro). In general, a shopping cart with the value greater than 200 zł is supposed to contain more than 2-3 items, so in consequence, the program can be able to discover association rules that are more interesting and useful for managers.

The minimum support value was set at 2%, while the minimum confidence was set at 80%. The minimum values of the parameters were defined in cooperation with the Sales Department of Alma24. The results are shown in part in Table 1. Thirty one rules were found respecting 2% minimum support. Among the interesting association rules can be pointed out those with a probability of 100% (confidence = 1), notably :

"At least 2% of the customers of tomatoes and a quarter chicken always buy the new potatoes",

"At least 2% of the customers of tomatoes per kg and chopped Cirio tomatoes in pieces always buy new potatoes"

"At least 2% of the customers of tomatoes per kg and Lubella brand wheat flour wheat always buy new potatoes"
"At least 2% of the customers of red pepper per kg and Hajnowka butter always buy tomatoes per kg"

Table 1. Subset of discovered rules

Premisses	**Conclusion**	**Confidence**
tomato kg, quarter chicken	new potato kg	1
tomato kg, Cirio chopped tomatoes 400g	new potato kg	1
tomato kg, Lubella wheat flour 1kg	new potato kg	1
red pepper kg, Hajnowka butter 200g extra	tomato kg	1
white grape kg, watermelon kg	tomato kg	1
banana chiquita kg, white grape kg	new potato kg	1
new potato kg, UHT milk 3.2% 0.5l	chicken drumsticks	1
chicken drumsticks, UHT milk 3.2% 0.5l	new potato kg	1
apple kg, red onion kg	red pepper kg	1
tomato kg, banana chiquita kg, white grape kg	new potato kg	1
tomato kg, Piatnica cottage cheese	new potato kg	0.889
apple kg, cheese gouda	tomato kg	0.889
tomato kg, celery kg	parsley kg	0.889
tomato kg, parsley kg	celery kg	0.889
chicken breast, Piatnica cottage cheese	new potato kg	0.889
bunch chives, red onion kg	red pepper kg	0.889
new potato kg, Cirio chopped tomatoes 400g	tomato kg	0.875
tomato kg, peaches kg	new potato kg	0.875
new potato kg, peaches kg	tomato kg	0.875
dark rye bread, Piatnica cottage cheese	tomato kg	0.875
dark rye bread, Piatnica cottage cheese	new potato kg	0.875
UHT milk 3.2% 1l , Piatnica cottage cheese	new potato kg	0.875
banana chiquita kg, chicken drumstick	apple kg	0.875
tomato kg, red pepper kg, white grape kg	new potato kg	0.875
new potato kg, red pepper kg, white grape kg	tomato kg	0.875
red pepper kg, parsley bunch	tomato kg	0.846
red pepper kg, Danone yogurt 135g natural	tomato kg	0.818
red pepper kg, white grape kg	tomato kg	0.800
red pepper kg, white grape kg	young potato kg	0.800
wholemeal bread, ground cucumber	red pepper kg	0.800
banana chiquita kg, red onion kg	red pepper kg	0.800

The majority of the resulting association rules refer to best-selling products and are generally known to the sales department of the Alma Delicatessen. Some interesting rules were discovered concerning the fruit and vegetable stand; however it should be pointed out that they were related to the period of August-September. During this period, products such as tomatoes, potatoes, watermelon, and parsley were very cheap and were often found in shopping baskets. Puzzling is the rule "*At least 2% of the customers of tomatoes per kg and chopped Cirio tomatoes always buy new potatoes*", because the chopped tomatoes should rather be related to pasta, chicken and herbs, the ingredients used to cook spaghetti.

The second study was carried out on transactions from August 2009 to January 2010. Due to the fact that the transactions involved a longer period than previously, the value of the minimum support is set to 2%, so as to be able to find association rules useful for the sales department. Table 2 shows the results of the experiment for customers with shopping cart value more than 200 zł.

Table 2. The association rules relating to shopping cart value more than 200 zł

The number of item sets: 503	**The number of association rules: 7**	
Min support: 2%	**Min confidence: 80%**	
Rules		**Confidence**
carrots kg, celery kg	parsley kg	0,882
mandarin kg, parsley kg	carrots kg	0,871
onion kg, celery kg	parsley kg	0,861
tomato kg, celery kg	parsley kg	0,846
lemon kg, parsley kg	carrots kg	0,811
banana chiquita kg, celery kg	parsley kg	0,806
onion kg, parsley kg	carrots kg	0,8

More specific analysis might be carried out. For instance, from the standpoint of sales, interesting rules might be also retrieved in more narrow itemsets containing items from specific groups or departments. Such analysis might be more useful for managers because of binding association rules with some specific goods. Managers are also interested to filter out the transactions with best-selling products. Some of these cases will be illustrated further in this section. However, in our experiments most of this pre-processing resulted in the rejection of items and generating known and useless association rules.

To carry out the process of discovering association rules for specific items, a predefined SQL query has been created. It consists of two parts:

- query asking the names of items that will create a set of rules to look for;
- subqueries returning the transaction IDs that include the items of interest.

This query composition has been used in all three experiments presented below.

Experiment 1. Market managers are also interested in the discovery of association rules related to particular items. To illustrate the problem, two queries were asked:

1) What items except pastas are in the shopping cart above 200 zł?
2) What items besides milk are in the shopping cart above 200 zł?

The minimum support ratio was set up after consultation with the managers equal to 4%, and the minimum confidence ratio of 70%.

As a result of the first question, 33 rules were discovered; the most interesting are presented in Table 3.

Table 3. The association rules relating to shopping cart value more than 200 zł contain any pasta

The number of itemsets: 197	**The number of association rules: 33**	
Min support: 4%	**Min confidence: 70%**	
Rules		**Confidence**
new potato kg, chicken legs	gouda cheese	1
cheese Gouda, chicken legs	new potato kg	1
cheese Gouda, chicken legs	new potato kg	1
chicken wings	new potato kg	0.889
water nes.wat. Naleczowianka 1.50l n	chicken fillet	0.875
new potato kg, white grapes kg	tomato kg	0.875
red pepper kg, banana chiquita kg	tomato kg	0.875
red pepper kg, parslay bunch p.	tomato kg	0.875
flour lubella 1kg Poznańska pszenna	new potato kg	0.857
butter Hajnówka 200g extra	cheese „gazda z dziurami”	0.857
new potato kg, cheese „Gazda z dziurami”	tomato kg	0.857
red pepper kg, white grapes kg	tomato kg	0.857
banan chiquita kg, carrots kg	tomato kg	0.857

Among the association rules obtained there are many items from the fruit-vegetable stand. Some of these rules can be interpreted using one's cooking experience, and say that the customers of this target group buy these items to cook a particular dish, for instance, spaghetti (if pasta, red pepper and parsley, then tomato) or potatoes au gratin (if pasta, gouda cheese, and chicken sticks, then potato).

Experiment 2. In the second experiment the fruits and vegetables have been removed from the itemsets. If not, something very similar to the previous rules would be generated. But we wanted to find a unique relationship between the shopping cart in which there were milk and other products.

The discovery process drew 59 rules. Most of the rules describe a combination of milk, butter, gouda cheese, cream and chicken legs (or chicken wings). Sample rules are shown in Table 4.

Table 4. The association rules related to shopping cart value more than 200 zł contain any milk

The number of item-sets: 412	**The number of association rules: 59**	
Min support: 4%	**Min confidence: 70%**	
Rules		**Confidence**
butter kerrygold 200g irish	gouda cheese	1
butter kerrygold 200g irish	sour cream piątnica 18% 200ml	1
butter kerrygold 200g irish	chicken wing	1
butter kerrygold 200g irish	gouda cheese, sour cream piątnica 18% 200ml	1
sour cream Piątnica 18% 200ml, butter kerrygold irish 200g	Gouda cheese	1
butter kerrygold irish 200g	Gouda cheese , chicken wing	1
chicken wing, butter kerrygold irish 200g	Gouda cheese	1
butter kerrygold irish 200g	sour cream Piątnica 18% 200ml, chicken wing	1

It can be stated that customers belonging to this group prepare a dish usually made up of these items. Among this group are also those with children; this is due to the following rules:

“4% of customers who had in the basket milk and semolina porridge ml., always buy the nectar Bobo Frut 300ml apple-raspberry-cherry "

"4% of customers who bought milk and semolina porridge 190g, organic vanilla, always buy the yogurt with apples or black berries"

"4% of customers who bought milk 190g and dessert nutricia biscuits bobovita banana, always bought porridge nutricia vita 190g and creamy bobo" „

"4% of customers who bought milk and chocolate 100g kraft nussbeisser alpen 100g, bought chicken fillet in 77.8% of cases".

Further analysis of the rules for chocolate and chicken would give a precise explanation of the relationship between these items. Among the rules was found also this quite obvious association::

"4% of customers who bought milk and coffee Jacobs Cronat gold200g, bought sugar in 83.3% of cases."

Experiment 3. The third experiment focused on multi-level frequent pattern mining, based on item hierarchies in Alma. The FP-Growth was applied several times reusing the existing tree structure to discover multi-level association rules. The algorithm examined the tree structure in a bottom-up manner, it means starting at the leaves and proceeds all the way up until the root of the tree collecting information about item names, groups of items and related frequencies. Rules have been generated respecting the support and confidence ratios.

The table 5 illustrates some of the interesting rules.

Table 5. Some interesting association rules for a group of customers with shopping carts above 200 zł containing any butter or chicken

Min support: 4%	**Min confidence: 70%**	
Rules		**Confidence**
butter, pasta	garlic	1
butter, cheese, eggs	toilet paper	1
butter, toilet paper, eggs	cheese	1
butter, cheese, toilet paper	eggs	1
chicken, toilet paper	milk 2%	0.800
chicken, margarine	sparkling water	0.800
chicken, bread	garbage bags	0.800
chicken, canned tomatoes	cheese	0.800

From this experiment the following rules are discovered:

- 4% of the customer group with a basket above 200 zł buy items that are needed to prepare spaghetti (rule: if someone bought butter and pasta, it always bought the garlic, and if someone bought the chicken and tomatoes , 80% of cases, also bought the cheese);
- 4% of customers in this target group have in their shopping carts butter, cheese, eggs and toilet paper
- 4% of customers order items that are not logically related to each other, eg.

  If the shopping cart contained a chicken and toilet paper, in 80% of cases there were 2% milk

  If the shopping cart contained a chicken and bread, and in 80% of cases were also garbage bags.

The rules of Table 5 can be used directly to increase sales in the online shop Alma24. One way would be a campaign to promote butter or chicken at reduced prices, ei. in the form of a discount coupon that can be sent to the customers of Alma24. At the same time, in order to increase the shop's income, every customer who buys the items using the coupon would receive an additional offer immediately after the purchase of items from the association rules at reduced prices, eg. for 90% of the normal price. This is a typical use of the method of up-selling, which is very effective at increasing the income from a single transaction.

4 Performance Analysis

Initially, the pilot study of the process was developed and evaluated. The transactional data were collected from CSV files, which resulted in many sub-processes that complicated the task of of rule discovery. t. Several experiments demonstrated that the process excessively uses memory and is very time consuming [Skrzypczak, 2010].

The low performance of the prototype has lead to redesign the whole process of rule discovery. The data were imported using SQL queries directly into Rapid Miner, transformed in the matrix from which frequent itemsets were searched, and, finally, association rules were discovered.

In the new solution, the demand for memory was decreased respectively from 1000 MB to 800 MB. The reduction of memory, however, was not significant for a smaller data set. More advantages generated the operators Materialize Data (writes data from memory) and Free Memory (clears working memory) that considerably decreased usage of memory. With these modifications the process runs almost six times faster than the pilot version. The computing time was diminished (respectively 16 s. for data used in the article, and 87 s. for the pilot study).

Figure 5 shows the effect of the new solution on the memory size and the duration of the current process.

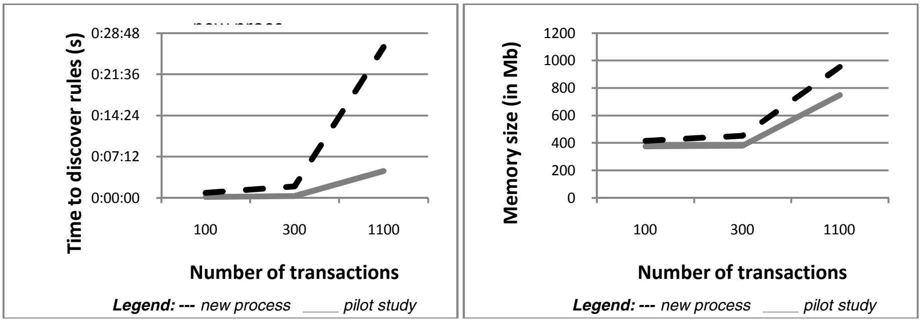


Fig. 5. Performance of rule discovery system depending on the number of transactions

Thanks to these improvements the whole process is much efficient, despite the use of the same algorithm for finding frequent items and construction rules. The new process also consumes less memory, making it possible to conduct experiments on a much larger data sets with an identical hardware configuration.

5 Conclusions

The article has presented the process of extracting association rules in customers' transactions of the Internet Delicatessen Alma24. Initially, the process of rule discovery was developed and used in pilot studies. After conducting several experiments, it turned out that the process is inefficient and uses large amounts of memory. Therefore the process was redesigned.

The new solution demonstrated that the design of rule discovery process has an impact not only on the required amount of memory, but also on the computing time to obtain the final result. Thanks to this solution the whole process is much shorter, despite the use of the same algorithm for finding frequent items and construction rules. The FP-Growth is fast and scalable avoiding the costly process of candidate generation and testing used by Apriori algorithm

Of course, it is important to choose efficient data mining algorithms. However, one has to take into account the process of data cleaning, consolidation, and transformation of data into appropriate form.

The implemented solution helped the Alma managers to make better profitable sale decisions by discovering the buying habits of their customers. The rules allowed to focus on items and group items that are most likely to buy by customers. In general, the rules have been used to improve the shopping environment, customize marketing efforts and provided location-aware recommendations to customers. From the viewpoint of internet shop, the obtained knowledge was used to improve Website services, advertisements and increase volume of sales.

Acknowledgements. The authors thank the Board of Alma Delicatessen in Wroclaw for access and the use of corporate data for the article.

References

1. Bereta, M.: Data Mining z wykorzystaniem programu RapidMiner, czerwiec, (2010), http://michalbereta.pl/dydaktyka/ZSI/Lab%20Data%20Mining%201.pdf, czerwiec (2010), http://michalbereta.pl/dydaktyka/ZSI/Lab%20Data%20Mining%202.pdf
2. Bonchi, F., Giannotti, F., Mazzanti, A., Pedreschi, D.: ExAnte: Anticipated Data Reduction in Constrained Pattern Mining. In: Lavrač, N., Gamberger, D., Todorovski, L., Blockeel, H. (eds.) PKDD 2003. LNCS (LNAI), vol. 2838, pp. 59–70. Springer, Heidelberg (2003)
3. Borgelt, C.: Keeping Things Simple: Finding Frequent Item Sets by Recursive Elimination. In: Borgelt, C. (ed.) Workshop Open Source Data Mining Software (OSDM 2005), Chicago, pp. 66–70. ACM Press (2005)
4. Gyorödi, C., Gyorödi, R., Cofeey, T., Holban, S.: Mining Association Rules using Dynamic FP-trees. In: Proceedings of The Irish Signal and Systems Conference, pp. 76–82. University of Limerick (2003)
5. Han, J., Pei, J., Yin, Y.: Mining Frequent Patterns without Candidature Generation. In: Proc. of the 200 ACM SIGMOD Int. Conf. on Management of Data, Dallas, pp. 1–12 (2000)
6. Han, J., Yin, Y., Mao, R.: Mining Frequent Patterns without Candidat Generation: A Frequent-Pattern Tree Approach. In: Data Mining and Knowledge Discovery, vol. 8, pp. 53–82. Kluwer Academic Publ. (2004)
7. Hand, D., Mannila, H., Smyth, P.: Eksploracja danych. Wydawnictwo Naukowo-Techniczne WNT, Warszawa (2005)
8. Kotsiantis, S., Kanellopoulos, D.: Association Rules Mining: A Recent Overview. Proc. GESTS Internat. Trans. on Computer Science and Eng. 32(1), 71–82 (2006)
9. Morzy, T.: Eksploracja danych (2010), http://www.portalwiedzy.pan.pl/images/stories/pliki/publikacje/nauka/2007/03/N_307_06_Morzy.pdf
10. Pasztyła, A.: Analiza koszykowa danych transakcyjnych – cele i metody (2010), http://www.statsoft.pl/pdf/artykuly/basket.pdf
11. Rácz, B.: NONORDFP: An FP-Growth Variation without Rebuilding the FP-Tree. In: 2nd Int'l Workshop on Frequent Itemset Mining Implementations FIMI (2004)

12. RapidMiner 4.3. User Guide. Operator Reference. Developer Tutorial (2010), http://docs.huihoo.com/rapidminer/rapidminer-4.3-tutorial.pdf
13. Skrzypczak, P.: Modelowanie wzorców zachowań klientów Delikatesów Alma przy wykorzystaniu reguł asocjacyjnych, Master Thesis, Uniwersytet Ekonomiczny, Wrocław (2010)
14. Zaki, M., Parthasarathy, S., Ogihara, M., Li, W.: New Algorithms for Fast Discovery of Association Rules. In: Proc. 3rd Int. Conf. on Knowledge Discovery and Data Mining (KDD 1997), pp. 283–296. AAAI Press (1997)

Case Study in Process Mining in a Multinational Enterprise

Paul Taylor[1], Marcello Leida[2], and Basim Majeed[2]

[1] BT Innovate & Design, Adastral Park, Martlesham Heath, Ipswich, UK
[2] EBTIC (Etisalat BT Innovation Center), Khalifa University, P.O. Box 127788, Abu Dhabi, U.A.E.

Abstract. Process mining has become an active area of research and while there are numerous papers on approaches to process mining there are fewer detailing its application to real industrial scenarios and its applicability in these spaces. In this paper we introduce the approach to process mining used in a number of multinational enterprises and then reflect upon the issues that have been encountered during our ongoing work. In our opinion these issues are a clear example of the challenges that need to be addressed during business process discovery from heterogeneous data.

Keywords: Process Mining, Data Driven Process Discovery, Industrial Application, Case Study, Process Improvement.

1 Introduction

With the industrial revolution, the introduction of mass production forced a drastic shift from artisanal crafting of products, usually performed by one person, to the large scale production of goods, where each person was responsible for a single step in a process typically orchestrated by someone that was not involved in the execution. One of the first persons to define this series of steps as a formal process was Adam Smith [1], where he describes the series of steps required for the production of a pin.

With increasing demand for goods, the increase in competition and the need for a reduction in costs required the reengineering of the existing processes. In most major companies this was already being considered even before it was formally defined during the 1990s by Hammer [2] and Davenport [3]. They defined a procedural approach for improvement of business processes to ensure correctness and to improve effectiveness, efficiency, and compliance with statutes and protocols.

In the present day business processes are well understood and formally defined with standardized representations and associated reengineering procedures. Business analysts exploit these formal representations by defining performance measures and quality constraints to analyse and improve specific aspects of the enterprise. Such business improvement activity is based on the process captured

K. Aberer, E. Damiani, and T. Dillon (Eds.): SIMPDA 2011, LNBIP 116, pp. 134–153, 2012.

and defined using a formal language. However with the increase of the complexity of the formal languages used to model processes we witness also an increase in the distance between the formal process model and the process that is actually being executed [4,5].

Large enterprises are required to actively respond to market demands and market evolution, therefore it is necessary to ensure control of crucial activities and to facilitate the reengineering of the processes running across the enterprise. This activity is usually performed by defining strategic requirements over the process (i.e. reduced execution time, minimising the amount of repeated work, removing loops, and so on). This set of requirements is relatively easy to capture and monitor in a fully automated process execution environment, however the same cannot be said for complex processes running across multiple departments, or for processes involving human intervention.

Broadly speaking every activity in an Enterprise (or an SME) can be seen as a step in a more comprehensive and complex process. Most enterprises capture the trace of the execution of activities within a process in several different ways and for several different reasons. Most of the information systems keep track of the activities being executed within them: Enterprise Resource Planning (ERP) systems, Business-to-Business (B2B), Customer Relationship Management (CRM) systems are each able to generate activity logs. Others are explicitly captured by Workflow Management Systems (WMS) which log the beginning and the conclusion of activities. Some activities such as email exchanges are generated just to have a tangible record of an agreement established by word of mouth. Others are formal legal documents that establish some kind of relationship between the parties involved. Each of these pieces of information contains knowledge about the enterprise, which can be used in several different analyses. One way to exploit this information is to reconstruct the process executing in the enterprise; this allows the behaviour of those various processes and the actors involved to be monitored in order to identify reasons for bottlenecks, incorrect executions, rewinds, loops and other issues preventing the process from matching the desired strategic requirements. However the process of extracting measurable evidence from the enterprise knowledge base is a non-trivial activity, which requires understanding and analysis of the activities and their interactions to transform them in measurable models.

In this paper we present a series of case studies directly from our experience with the analysis of real process execution data collected from several different systems, each of which uses a different technique to capture data. We believe that the cases presented in this paper will help make the research community more aware of the issues that could arise when dealing with enterprise-scale data and influence the future research directions in the area of business process analysis. This paper is divided as follows: Section 2 presents some background information about the tool we used to perform the business process analysis described in the paper. This section will provide the knowledge required for a better understanding of the use cases presented. In Section 3 we discuss existing work detailing other real-cases of analysis of large-scale processes. In Section 4 we present a set of relevant case studies regarding the application of business process mining techniques in a multinational enterprise, which is the focus of this work. We conclude the paper by bringing together some final considerations in Section 5.

2 The Aperture Process Mining Tool

As introduced in the previous section, one of the major issues in process analysis is to capture and interpret data correctly. There are languages such as BPMN [6] and BPEL [7] that are used to explicitly define a process and capture execution information for fully automated systems (e.g. web service orchestration). However in most cases integration of BPEL engines in existing and on-going processes is a non-trivial undertaking and often is an effort that enterprises prefer to avoid unless a minimum Return of Investment (ROI) is guaranteed. This effort might also be undesirable where the process is not formally captured and it exists only in an idealized sense in the mind of the people involved; this is particularly the case for many SMEs. Therefore there are many situations where a process model is of very limited use or is simply not available; in which case the process model needs to be inferred from the information created during process execution.

To respond to this specific requirement we are using an internally developed process mining tool known as **Aperture**. It has been designed to provide an analysis that is focused not upon process mining itself but upon process analysis and business improvement. One of the design goals of Aperture is that it should be usable by those without experience or expertise in process mining, and should fit into the toolkits used by those people in the target enterprises. Aperture is effective with process data describing either sequential or parallel behaviour and requires a minimum of data to be effective (currently a process identifier for each process instance, a task identifier for each activity, the start time of the activity and the end time of each activity).

The important goals for the process mining algorithm used in Aperture are *predictability*, *plausibility*, and *traceability*. Predictability refers to the ability of the tool to produce the same model given the same input data, plausibility means that the generated models should be viewed with confidence by any person knowledgeable about the process, and traceability means the ability to trace the components of the generated model back to the source data. Traceability is very important in practice as it is quite regularly needed to convince skeptics of process mining within the business and operational units whose initial reaction is often to reject the mined model as being unrealistic, or just plain wrong. However the ability to trace each part back to the source builds confidence and leads to increased buy-in from the users.

Aperture is able to reconstruct the process model form data stored across heterogeneous sources of information, providing the users with a unified framework to analyse processes and tasks that are executed across the enterprise, that otherwise would be extremely difficult to monitor and improve.

Trained users can perform analysis across several business domains without knowing domain specific aspects. The more the data that is available, the greater the accuracy and flexibility of analysis that can be performed.

We will now briefly introduce the data model at the base of Aperture tool; this will help to fully understand the case studies presented in the next sections. The main elements of the back end data model are the *Process Instance* and the *Task Instance*:

- each *Process Instance* describes one job or order or process execution (service fulfillment or fault repair processes for example);
- each *Process Instance* consists of a number of *Task Instances* linked together by a temporal relations (activity `A` is executed before activity `B`).

The minimum information required to create a process model to be used in Aperture is a process instance with a minimum of one task instance. The minimum amount of information to create a task instance is the start time and end time of each task instance. Process instance start time and end time can be derived from the starting time of the first task and the ending time of the last task if they are not explicitly provided.

To convert the raw data from source systems and make it available to the users for analysis, the tool follows a computer assisted multi-phase approach (Figure 1). The first phase is to acquire workflow data from the process under study. The identification of appropriate data and providing a source data set is performed manually. Acquiring the data is often challenging as it might come from multiple independent systems which may not have corresponding linking keys. Data acquisition is an extremely delicate phase since if there is misalignment with the process identification the resulting model will be of no use. The second phase is in many ways the most important: a competent computer programmer will implement a data importer that will extract and transform each process instance from the data acquired in the first phase into Aperture's internal format. This phase is supported by the process mining algorithm of Aperture that will be used to generate the structure of the process instance from each group of tasks.

The algorithm used by Aperture to connect tasks into a process instance is based upon minimum-spanning-tree algorithms [8]. We attempt to minimise the time spent in between tasks, thus satisfying the predictability, and traceability goals. To ensure that the models are plausible, domain knowledge can be incorporated into the importer thus placing appropriate constraints on the mining algorithm. An example of where this ability has been used is in a system handling complex orders which may have more than one sub-order, to achieve the plausibility goal it was necessary to ensure that each sub-order was connected independently to avoid cross-contamination and loss of conceptual clarity.

In the third and final phase the data is made available for interrogation via the provided user interface, which is a browser based application (Figure 2) written using the Google Web Toolkit (GWT)[1]. This allows the user to view the data from the overall process model (Figure 3) right down to the level of the individual

[1] `http://code.google.com/webtoolkit/`

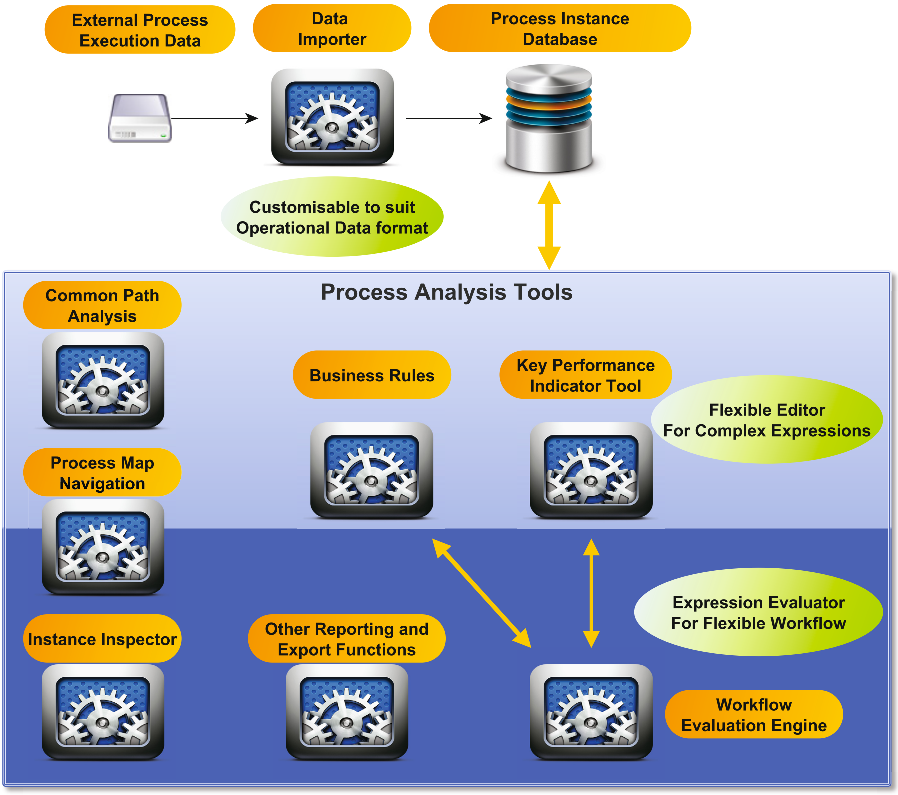

External Process Execution Data
Data Importer
Process Instance Database
Customisable to suit Operational Data format
Process Analysis Tools
Common Path Analysis
Business Rules
Key Performance Indicator Tool
Flexible Editor For Complex Expressions
Process Map Navigation
Expression Evaluator For Flexible Workflow
Instance Inspector
Other Reporting and Export Functions
Workflow Evaluation Engine

Fig. 1. The architecture of Aperture

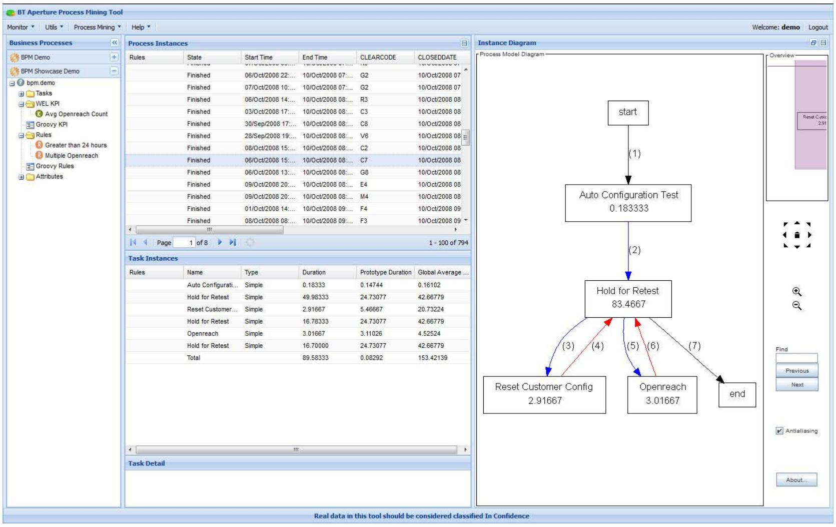
process instances. In addition to the minimal information required, the rest of the source data is imported and attached to the processes as process and task attributes. These attributes can be used by the user to filter sub-groups of data, and to run calculations against the dataset without requiring a re-importing of the data which speeds up comparative work considerably.

This approach differs significantly from that followed by other process mining tools (such as ProM [9] or BPM|one from Pallas Athena[2]) in that we do not process multiple process traces directly into a single model, rather we consider each process instance separately. The system then takes care of generating the compound process models and any associated analytics.

2.1 Analysis

The first and most intuitive of the analysis options offered by Aperture is the creation of the compound process model, as in the example in Figure 3. This will

[2] http://www.pallas-athena.com



Process Instances

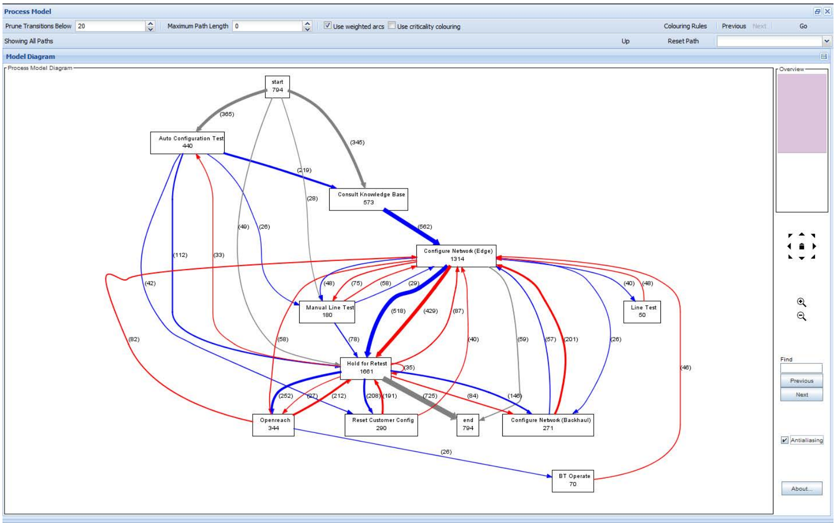
Rules	State	Start Time	End Time	CLEARCODE	CLOSEDDATE
	Finished	06/Oct/2008 22:...	10/Oct/2008 07:...	G2	10/Oct/2008 07
	Finished	07/Oct/2008 10:...	10/Oct/2008 07:...	G2	10/Oct/2008 07
	Finished	06/Oct/2008 14:...	10/Oct/2008 08:...	R3	10/Oct/2008 08
	Finished	03/Oct/2008 17:...	10/Oct/2008 08:...	C3	10/Oct/2008 08
	Finished	30/Sep/2008 17:...	10/Oct/2008 08:...	C8	10/Oct/2008 08
	Finished	28/Sep/2008 19:...	10/Oct/2008 08:...	V6	10/Oct/2008 08
	Finished	08/Oct/2008 15:...	10/Oct/2008 08:...	C2	10/Oct/2008 08
	Finished	06/Oct/2008 15:...	10/Oct/2008 08:...	C7	10/Oct/2008 08
	Finished	06/Oct/2008 13:...	10/Oct/2008 08:...	G8	10/Oct/2008 08
	Finished	09/Oct/2008 20:...	10/Oct/2008 08:...	E4	10/Oct/2008 08
	Finished	09/Oct/2008 20:...	10/Oct/2008 08:...	M4	10/Oct/2008 08
	Finished	01/Oct/2008 14:...	10/Oct/2008 09:...	F4	10/Oct/2008 09
	Finished	08/Oct/2008 08:...	10/Oct/2008 09:...	F3	10/Oct/2008 09

Page 1 of 8 1 - 100 of 794

Task Instances

Rules	Name	Type	Duration	Prototype Duration	Global Average ...
	Auto Configurati...	Simple	0.18333	0.14744	0.16102
	Hold for Retest	Simple	49.98333	24.73077	42.66779
	Reset Customer...	Simple	2.91667	5.46667	20.73224
	Hold for Retest	Simple	16.78333	24.73077	42.66779
	Openreach	Simple	3.01667	3.11026	4.52524
	Hold for Retest	Simple	16.70000	24.73077	42.66779
	Total		89.58333	0.08292	153.42139

Fig. 2. A Screenshot of Aperture's main screen



Prune Transitions Below 20 Maximum Path Length 0 ☑ Use weighted arcs □ Use criticality colouring

Showing All Paths

Fig. 3. A Screenshot of Aperture's process model view

combine the mined models for each process instance into a single graph showing the overall shape of the process. Weighting of the arcs of the graph is used to indicate frequency allowing users to see at a glance where the process is running *hottest*. In addition, the tool allows users to visually distinguish in the graph between the first time a task is executed in a process and where it is repeated, the first occurrence is connected using a blue (or solid) arc and the second using a red (or dashed arc). In a compound diagram this shows clearly where most of the repetition (or *rework*) is occurring and therefore where efficiency savings could potentially be made.

A set of more complex analysis tools can be used starting from the process model graph: drill down facilities are provided on top of the model which allow the user to see, for example, if a particular problem transition is only associated with a particular group of process instances.

The second of the analysis options offered by Aperture is the extraction of the common paths through the process which are known as *prototypes*. Each prototype is a distinct concrete ordering of tasks (either sequential or parallel) as executed in the source system(s); in Aperture the process instances that have the same concrete sequence are considered to be in the same prototype. These prototypes can then be examined for frequency, and for the commonality of their attributes, allowing correlations between attributes of the process, and the execution path to be discovered.

Furthermore the tool provides ad-hoc querying for Key Performance Indicators (KPIs) using either a custom workflow oriented language called Workflow Execution Language (WEL) [10] or more complex measures defined through the Groovy[3] scripting language, interacting directly with the Aperture data model. In conjunction with the ability of the tool to rapidly extract subsets of data this allows the user to rapidly get an insight into the relative performance of those subsets in terms that are familiar from strategic reporting (once the formula for the KPIs has been entered into the tool).

3 Related Work

Existing literature describing systems able to infer the process model from arbitrary process execution data is not extensive, furthermore we do not intend to focus on the differences between Aperture and report systems based on top of WMS, ERP, CMS because they are out of the scope of this paper.

In the set of available works, we identify the ProM framework [9] as the pioneering and leading research project for business process mining from execution logs and a system closer to our approach; however the definition of a new process mining tool is not the reason for writing this paper. We introduced our system in Section 2 for the sake of completeness and clarity, but the reasons that led us to collect a set of case studies of real applications of a process mining tool are led by the experience our group gained in the deployment of Aperture tool in enterprise-scale environments, during the last few years.

[3] http://groovy.codehaus.org/

The list of papers describing case studies of the application of process discovery techniques in enterprise or government-scale level is also not extensive as we have only been able to identify few published realistic pieces of work.

In [11] the authors describe the use of the ProM system for the analysis of the Dutch National Public Works Department. The paper described how the logs, generated by a WMS were analysed and how the ProM tool was used to extract the process model, the social network of the organization and the process logic based on decision trees. However the step of importing the data into ProM is not described in detail: the authors mention that they converted the original log to MXML[4] [12] format used in ProM. But the conversion process is not reported and if any issue arose during the import of the data, there is no record of it in the paper.

In [13] an application of the ProM framework in the Health sector shows how the tool is used to extract the process model from event logs. In this particular case an application of the clustering technique is used to extract clusters of similar process models in order to filter wrong models or models representing outliers. The raw data used contains information about 627 gynecological oncology patients, collected by the billing system of the Academic Medical Center (AMC) hospital in Amsterdam which is converted into MXML for use with ProM. In this case the authors mention a problem with the billing system caused by the fact that the timestamps associated to the events refers only to the day the event happened. The authors mention in the paper that this situation had led to wrong ordering of events happening in the same day. Moreover the data import task has been preceded by a preprocessing of the MXML document to aggregate events by department, this way the resulting process model was simpler to understand and analyse. However in the rest of the paper the authors do not mention if and how the problem influenced the resulting analysis.

The case studies reported in the literature focus mainly on the process mining aspect with little or no mention of the issues relating to the data collection process. The set of examples in this paper will also address the possible problems and issues that need to be considered before importing the data into a process mining tool. This will provide valuable experience to process mining practitioners as they tackle new data sets by helping to avoid common mistakes, leading to improved results.

4 Case Study Issues

This main section of the paper, discusses the issues and common patterns encountered while process mining across a variety of processes in a number of diverse multinational organisations. The most significant challenges are to understand and reconcile data models from systems which can be very old, and to acquire appropriate process data from these systems when the data that they record was defined in another era for other less rigorous approaches.

[4] http://www.processmining.org/logs/mxml

These cases do not form an exhaustive list of all issues that we have encountered, but those presented have occurred with sufficient frequency that we are confident that they are likely to be present in most enterprises to a greater or lesser degrees.

4.1 Variation in Standardized Processes

The first use case to present is the analysis of how a process that is standardized across the entire enterprise is executed with different performance for reasons that cannot be inferred from the process execution data directly. The process we are focusing on is a service provisioning process of a multinational enterprise.

In the provision of services globally one approach to ensure the performance of provisioning tasks would be to use a centrally defined process model that is well supported and well understood by all the departments that execute it worldwide. One of the processes we had the opportunity to analyse with the tool we developed was of this nature. The process improvement team examining this process requested the analysis of the process execution data in order to extract the process workflow.

The goal of the analysis process was to identify areas for performance improvement in the global process model which could then be tested and deployed around the world.

Once the process model was extracted from the process execution data provided by the enterprise, it was clear that what was understood to be a single process had a large amount of unexpected variation. As-designed executions were only a small percentage of the overall executions of the processes analyzed; all the remaining process instances deviated significantly from the original process.

Even though the original process model was lost in a plethora of different executions, our goal was to identify areas of inefficiency such and looping, repeating tasks and what is known as *rewind* or *rework*: the situation where the execution has to be returned to an earlier part of the process because work there has been left undone. Unfortunately the size and complexity of the generated model were such that it was not possible to identify any such issues immediately.

This being the case, we carried out a process mining exercise in order to identify which criteria was best able to group the process executions into models based on the issues we have previously discussed: such as bottlenecks, repeated work and so on. The separation of the various process models would allow the process improvement teams to conduct a more detailed analysis and interviews with the people involved in the execution of the process to uncover the underlying reasons behind the variation, so that it could be addressed.

For example, if the process model for a particular product type, say product A showed that in almost all executions the credit checking task was performed multiple times while the process model for every other product showed only a single execution, then the process improvement team could investigate that specific task to try to uncover the reason behind the repeat for product A. This shows our general approach of using process mining as a tool to inform, rather than a tool to indicate potential solutions directly.

From the analysis performed using Aperture it was possible to highlight that the most striking of the variations in the set of performance metrics defined to evaluate the process executions, was the difference between the processes executed in each of the order management teams around the world. Therefore the criterion that was used to separate the process model was the geographical location of the various order management teams.

In order to confirm this assumption, we used our process analysis tool to generate models of the process as it was followed in each of the separate order management teams. Once each of these models had been generated we proceeded to visually identify the areas of largest variation, together with the analysis of the performance indexes, and the areas exhibiting the undesirable properties we have previously discussed. Examples of two of the models created using the geographical location as clustering criteria can be seen in Figure 4 and Figure 5.

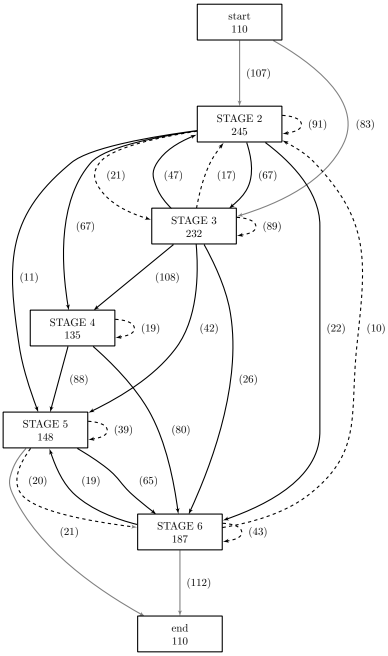


Fig. 4. Process Executed in France

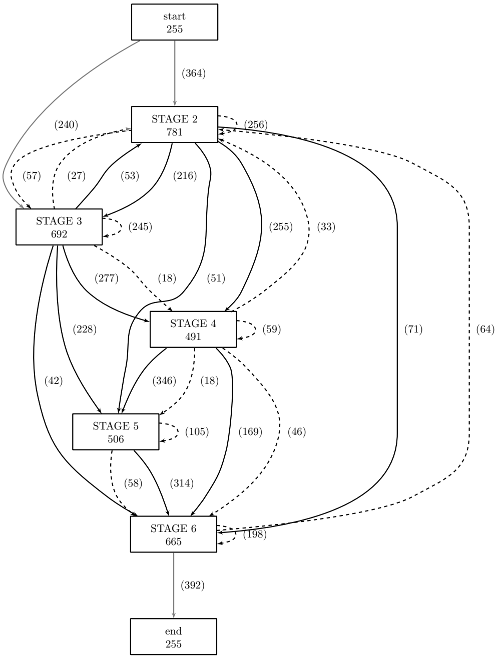


Fig. 5. Process Executed in Germany

As it is possible to notice from Figure 4 and Figure 5 there is a marked contrast between the processes executed in France and in Germany. To reiterate: the process model that should be executed both in France and Germany should be the same in this case.

For the period that these models are referring to, the evaluation of the performances of the processes was markedly better in France than Germany. This gives a significant hint that we could improve the performance of the German team if best practice from the more efficient French implementation is adopted. The outcome from this analysis has been to gather representatives of the two regions and examine some meaningful orders identified by our process analysis tool and discuss the rationale behind the decisions made during execution to identify the root cause of the differences and the deviation from the standard process.

The complete analysis indicated that while variation between the regions was common; the most conformant process with the standard process model and also the one with the minimum of process issues was the process model followed by the French team, while the process showing the most frequent process issue was the one followed by the German team.

Since this process is standardised the process improvement teams started to look at potential reasons for the differences that were not captured by process execution data, this may have included variations in the process inputs or setup (e.g. exact product configurations, equipment requirements, expertise of the employee performing the installation), insufficient or incomplete documentation of the original process model or simply the way the local process was interpreted and executed.

The process improvement teams took this information away to one of the regular meetings of representatives from the regional teams so they could present our findings and try to gain some insight from the operational level as to the reasons for the features identified. Discussions with the process team following this meeting revealed that the process models, and diagrams we produced had been invaluable in drawing out additional information in the discussions.

4.2 Documentation Out of Synchronization with Reality

This case study highlights the importance of collecting the process execution data in a way that is aligned with the process model that was defined and put in place. This example is useful to point out the fact that extracting the process model from execution data is a not straightforward task.

Process improvement projects that do not capture process execution data for analysis by a process mining tool normally perform analysis on the designed process model when looking for areas to improve. This is a problem when the designed process is significantly different to the process that is being executed. In our experience we have observed many situations where this is the case and this happens for a number of very different reasons, for example:

- when a change in the system that executes the process, forces a change to the sequence of constituent tasks;
- when an exceptional circumstance arises (natural disaster, strike action, etc.) leading to temporary process changes which are unintentionally adopted permanently as staff assimilate the temporary process;
- when an upstream process drastically changes its performance, leading to a change in the downstream process execution, e.g. change of the task execution order;

– when a process execution includes constraints from local conditions that are not captured in the process definition;
– when a process work-around that is discovered by operators becomes the local standard as it is perceived to be more efficient than the designed process.

However from our experience we identified one example that stands out as being a recurrent issue for process mining tools and practitioners to be aware of.

This issue is the misalignment between the level the process designers are working at and the level that the process executors are working at; including the systems that are being used to manage the workflow though the execution phase. To be clear, this is not an issue with systems, or with the data itself, or data collection step; it is a process design issue.

First, let us introduce queues. A queue is simply a collection of orders that are waiting for action. Each queue in the workflow system is assigned to a particular team for them to take work from, this means that there is an approximate 1 : 1 mapping between queue and team. In situations where each team has only a single task to do in the process, data on when a process entered a queue and exited a queue can be used as an approximation of the time that team spent working on that task.

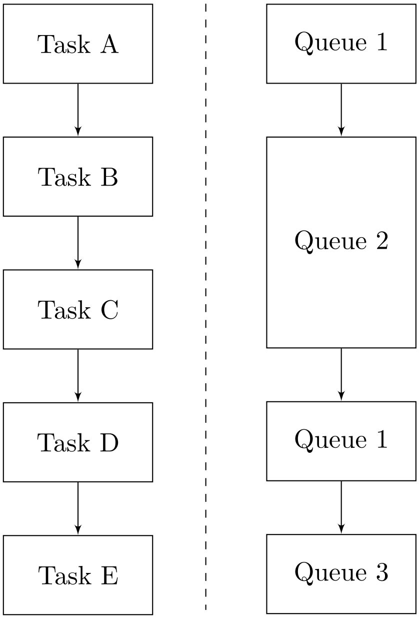


Fig. 6. An example showing how a process instance may look if data about tasks, or queues is stored

For example, imagine we have `Task A` to `Task E` where `Task A` and `Task D` are serviced via `Queue 1`, `Task B` and `Task C` by `Queue 2`, and `task E` by

`Queue 3`. Figure 6 shows the difference between the process models according to tasks (left) and queues (right). It is possible to notice in this simple example how information is lost, making it extremely difficult to interpret the results in terms of analysis of the process under investigation.

We have now seen that the queue abstraction can mask data about the process from being recorded in the logs of the workflow system, and this is a widespread issue particularly in older workflow systems. If the process designers take this into account when designing the process and ensure that there is a 1 : 1 mapping between task and queue, then the process can be recovered with a reasonable degree of accuracy, however if the process designers ignore this abstraction then the process designs will not map onto the data that is provided by the workflow system, meaning that the process model discovered by process mining cannot match the process as designed.

One concrete circumstance in which we have often observed this pattern is where a product has many specific configurations that the customer might choose from. In this situation the teams creating the workflow will create product specific queues for each team e.g. Order Management (Product X), Field (Product X), since there is just one product. The problem arises when the process designers, who are a separate team, then treat each specific configuration (erroneously) as its own specific product, so if there are five configurations, the workflow team will encode a single process into the queues in the workflow system, but the process designers will provide five separate process designs for the actual users of the system to utilise.

This is a major problem for process mining because the generated models will be using the queue abstraction from the workflow system and that will not correspond to the process model that is provided to the process analysis teams. This means that the analysts may propose/implement solutions that will not be possible to implement or that will not generate the expected outcome. The easiest way to mitigate this problem is to maintain a mapping between the queues and the tasks performed in those queues and use the mapping to convert process execution data with workflows aligned with the conceptual model.

However in practice this mapping either does not exist or it can get quickly outdated for the reasons we described previously. Moreover organisational changes and previous process improvement efforts may have caused multiple queues to perform the same task, or a single queue to perform multiple disparate activities. An example mapping for the process in Figure 6 is given in Table 1.

Table 1. An example mapping of tasks to queues

Task	Queue
Task A	Queue 1
Task B	Queue 2
Task C	Queue 2
Task D	Queue 1
Task E	Queue 3

In addition to having to maintain this mapping through business as usual process improvement and process change work, a difficulty is introduced if there is reorganisation within the business. Let us imagine that we have 2 products in this workflow system, and each of these products has 5 different configurations that have different process designs in the way described above (so 1 workflow system, 10 process designs). Since each of these designs requires that an engineer is dispatched, the team responsible for dispatching is assigned a common queue called FIELD. There is a simple mapping between the dispatch task in the process design and the field queue so there is no issue. Now the business is reorganised, and it is mandated that the field teams for each of the 2 products are to be managed separately, this means that we have to create a new queue and change the assignment of each queue to the appropriate new team, however we would also have to update 10 separate mapping documents (one for each of the process designs) to take this change into account. In a real system containing hundreds of products, and potentially thousands of configurations maintaining an accurate mapping between queues and tasks is a practical impossibility.

The lesson we have taken from this is that it is extremely important to have an accurate and consistent mechanism to translate between the different abstractions used (where they must be used) otherwise process analysis will be performed on incorrect models and the resulting decisions based on faulty assumptions. The ideal solution is that the components of the workflow system match the components of the design so that no mapping is required and there are no conflicting abstractions.

4.3 Deficiencies in Workflow Systems/Recorded Data

It is extremely important to have correct and consistent information available for process mining to derive accurate models and measures for the process under study, and often such data is not available. In most cases this is due to a deficiency in workflow management systems that cause misleading or incorrect data to be provided for process mining. This may not be due to a faulty workflow management system, as much as an issue with the way it is being used and how it records data.

We came across this issue during the analysis of a process based on the grouping of tasks into stages of execution. These were then organised in the process flow in a way that a stage was able to start when all of the tasks in the previous stage were completed. If the workflow system is able to record the times these tasks are actually performed (i.e. started and concluded) then no issue exists since we can analyse the workflow in the standard way.

However if the workflow system, like in the case we analysed, is able to collect only the time the task becomes ready for execution, then the data generated is misleading as it looks as though the various tasks in a stage are all running concurrently. This situation is exacerbated if some of the tasks have a long execution time, or a long lead time (such as tasks awaiting equipment, or an appointment), this means that a task to confirm the delivery of an item which takes only few minutes of effort, could show in the data collected by the workflow system as having a duration of several weeks.

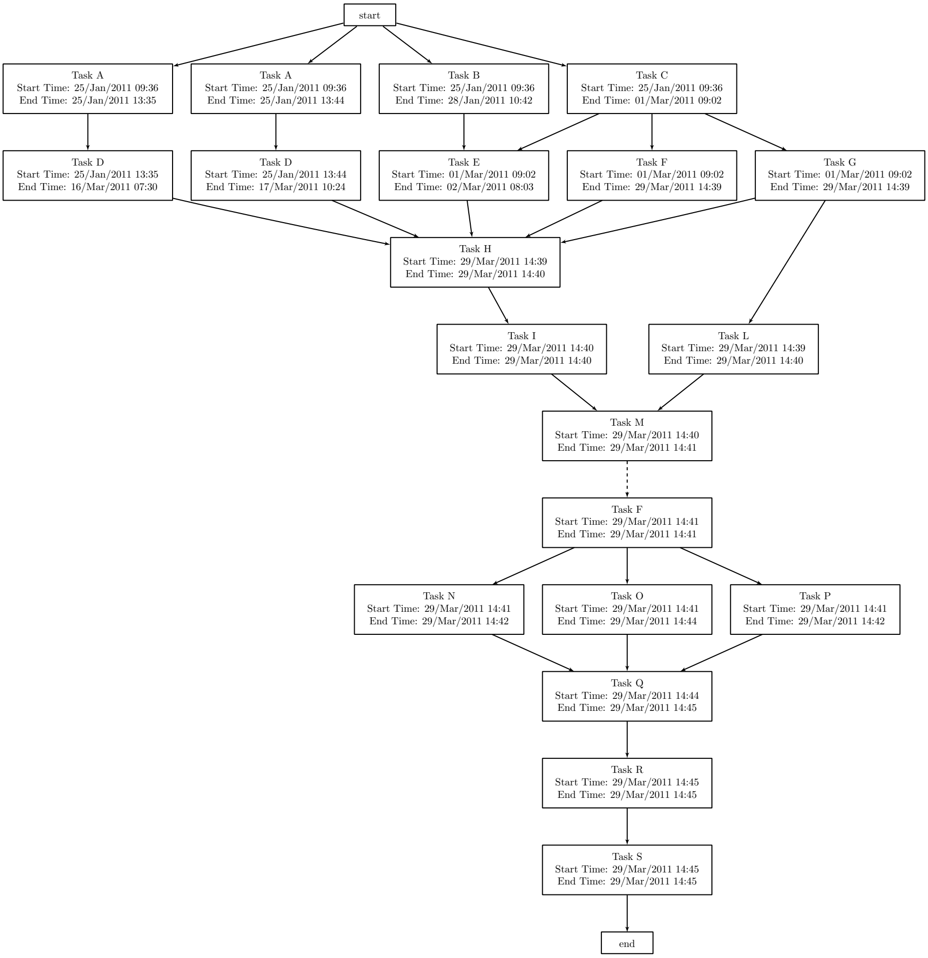

start
Task A
Start Time: 25/Jan/2011 09:36
End Time: 25/Jan/2011 13:35
Task A
Start Time: 25/Jan/2011 09:36
End Time: 25/Jan/2011 13:44
Task B
Start Time: 25/Jan/2011 09:36
End Time: 28/Jan/2011 10:42
Task C
Start Time: 25/Jan/2011 09:36
End Time: 01/Mar/2011 09:02
Task D
Start Time: 25/Jan/2011 13:35
End Time: 16/Mar/2011 07:30
Task D
Start Time: 25/Jan/2011 13:44
End Time: 17/Mar/2011 10:24
Task E
Start Time: 01/Mar/2011 09:02
End Time: 02/Mar/2011 08:03
Task F
Start Time: 01/Mar/2011 09:02
End Time: 29/Mar/2011 14:39
Task G
Start Time: 01/Mar/2011 09:02
End Time: 29/Mar/2011 14:39
Task H
Start Time: 29/Mar/2011 14:39
End Time: 29/Mar/2011 14:40
Task I
Start Time: 29/Mar/2011 14:40
End Time: 29/Mar/2011 14:40
Task L
Start Time: 29/Mar/2011 14:39
End Time: 29/Mar/2011 14:40
Task M
Start Time: 29/Mar/2011 14:40
End Time: 29/Mar/2011 14:41
Task F
Start Time: 29/Mar/2011 14:41
End Time: 29/Mar/2011 14:41
Task N
Start Time: 29/Mar/2011 14:41
End Time: 29/Mar/2011 14:42
Task O
Start Time: 29/Mar/2011 14:41
End Time: 29/Mar/2011 14:44
Task P
Start Time: 29/Mar/2011 14:41
End Time: 29/Mar/2011 14:42
Task Q
Start Time: 29/Mar/2011 14:44
End Time: 29/Mar/2011 14:45
Task R
Start Time: 29/Mar/2011 14:45
End Time: 29/Mar/2011 14:45
Task S
Start Time: 29/Mar/2011 14:45
End Time: 29/Mar/2011 14:45
end

Fig. 7. Diagram showing pattern caused by staged executions

Figure 7 shows an example of such a workflow: as it is possible to see there is an initial stage composed of four tasks which are immediately opened for execution with identical start times which refers to the start time of the stage, the end times of these tasks however show a more linear sequence, despite this, because the start times are all equal it is not possible to extract a sequential flow of the execution. In the later part of the diagram you can visually identify an example of this where the start of the final stage where a group of 3 tasks are started when Task F is finished, each of the 3 tasks has an identical start time because they all become available when the stage is opened.

Ideally the workflow system could be changed to record the real starting times of the tasks so a more accurate duration could be derived. Unfortunately such a change is often impractical in large entrenched or legacy systems. There are then two alternatives available; either treat the data as accurate, leading to the situation in Figure 7, or we could force the tasks into a sequential sequence, this however is prone to errors and therefore misleading results. Therefore the evaluation of Key Performance Indicators (KPIs) or decisions taken on this process model needs to be carefully evaluated before they are put in place, since the KPIs may be measuring a situation that is not reflecting reality.

4.4 Inadequate Data Quality

The fourth case study that we identified is related to the quality of the data acquired by the enterprise. It is important to recall, in order to fully understand this case, that the process analysis tool we developed and applied in the use cases described in this paper, works mainly with the start and end times of the task executions, and most of the cycle time analysis and performance metrics are derived from these measures.

In this case, and many others that we have investigated, the data being available for analysis contained a number of issues such as the start and end times resulting in negative process durations, missing execution times, invalid dates, and ambiguous or misreported task identifiers. When issues like this are found it is important to investigate if the data collection process put in place by the enterprise has been properly defined or not.

In order to identify the reason for these errors it is possible to use external information from users, or other data sources to mitigate them. Such corrective information can be incorporated into the Aperture importer, which transforms the raw data to be analysed into an appropriate format for the tool. However, as for the previous case, it is important to ensure that the users are aware of the compromises made when they are evaluating the results of the process analytics. In addition, it would also be advisable to validate the measures obtained from the tool against metrics that are calculated outside it to give meta-information that can be used as a quality measure for the data. This is important in order to ensure that the business management is not being misled through low quality data.

Table 2. An example of some source data with quality issues

ID	Task Name	Start Time	End Time	Order Status
1	Task A	2010-01-01 12:00	2010-01-01 12:00	CLOSED
2	Task B	2010-01-01 12:00	2009-01-12 12:00	CLOSED
3	Equipment ordered by Sandra for delivery on Thursday	2010-01-02 12:00	2010-01-03 12:00	CLOSED
4	Task D	2010-01-03 12:00		CLOSED
5	Task E		2010-02-04 12:00	CLOSED

An example of data provided in this situation is shown in Table 2 which is based on a real data set we have processed. Each row in the table shows a sample of an encountered data quality issue:

Row 1. This row shows information that at first sight seems to be correct. However it is important to analyse if the start time and end time captured by the system are correct. The resulting task might cause a problem for the evaluation of some metrics since the duration is zero.

Row 2. This row shows a task where the end time occurs before the start time. This could have been caused by many reasons: for example, because a user edited task data by adding a new note to this task after it was finished, or a metadata update was performed on it after the task execution was concluded. Another example would be where the data collection system, for some task types, swapped the start and end times. Using Aperture it is straightforward to identify this type of error since the duration of the task will be negative.

Row 3. This row shows an example where the task name has been replaced when the user performing the task has added a note or other information to the process or task. In such situations, depending on the workflow engine it may or may not be possible to determine the original task name.

Row 4. This row shows an open task called `Task D` as part of a completed process. The problem is that from the data, it appears as though the task has never finished, since the end time is missing. This can be mitigated if we assume the task was immediately completed, but it would be equally valid to assume this task ran until the recorded completion time for the entire process. In order to associate the most suitable information with the end time value, knowledge of the underlying process and workflow engine is required.

Row 5. This row shows a case which is similar to the previous one, where the `Task E` is in a closed state, with related end time, but according to the data the task was never started. The same assumptions made for the previous row are also valid here.

If these issues are ignored or go undetected it is easy to see that they could lead to errors being introduced into the process models and resulting analyses.

The approach that we followed to tackle these issues is to liaise with the business improvement team that commissioned the analysis, to identify a suitable solution; however this may not be always possible so a fallback position would be to remove the entire job from consideration to ensure it does not pollute the model. In cases where this approach would be too drastic, then the problematic task could be removed however this is a riskier strategy since it could introduce false dependencies between the remaining tasks

4.5 Undocumented Process Evolution

In the last case study we want to present in this paper, we highlight an example of evolution in the process model that was not driven by a decision in the process design team. In this case we have to deal with the variation from the original model that was due to a temporary change in the workflow caused by the ongoing replacement of machinery.

In this case we noticed that even when the machinery was restored to service some process executions were still following the temporary pattern. In the analysis of the overall process model it was possible to notice by examining the process model before and after the restoration that two process models existed, without management being aware of the second process.

Another process we had the chance to analyse was a process model that was not fully constrained. Due to several reasons (change of the employees, delayed training and so on) the process being executed was the process that was understood by the employee, which was in many cases different from the designed process. For example, tasks in the process were sometimes skipped because they were considered unnecessary by a process actor. Moreover in many cases the performance was better in the modified process than the normal one, but there were also many cases where the process execution flow was generating many rewinds, and other repetition.

The reason was that the task that was skipped was useful in very specific cases and in such cases the process would go into an uncontrolled situation, causing loops and rewinds, since the required task had not been executed. In case of an expensive and long process this is not an insignificant issue.

In these cases our tool allowed the extraction and documentation of these deviations from the standard process model and was also able to evaluate the impact of such deviations in the enterprise. The evaluation of the impact was an important step to enforce the designed process model, since some teams were claiming better performance without taking into account the drastic performance loss in the other executions, causing overall process performances to be worse. In addition, this behaviour can lead to many outlier execution instances, which in some cases are extremely risky.

5 Conclusion

The area of process mining is not simply concerned with finding and visualising the best model that fits the work execution data. In our opinion, a process mining system must have a rich set of tools that allow the practitioners to carry out a variety of performance measurements and other analytics. In addition, process mining techniques that focus on the dominant execution paths and ignore outlier instances lead to the reason behind the outliers staying undiscovered and thus capable of appearing again in the future causing persistent performance issues. In this paper we presented a set of case studies that we consider interesting for the community of professionals involved in the practice of business process management. We intentionally avoided considering situations where the original process design was not optimal because this is an issue of process modelling rather than analysis; instead we preferred to focus on the issues arising from the system used to capture the data and how this data is interpreted in the extraction of the process model.

The most important issue that we have found in our work is that **data quality is paramount**: techniques for knowledge extraction such as process mining can only be mislead by incidences of erroneous data. This is, of course, the well known garbage-in-garbage-out phenomenon; however we have identified some data quality issues that are particularly troubling for process mining (Sections 4.3 & 4.4).

The other major conclusion from our work is that having access to someone with process knowledge is extremely important for the accurate consumption and evaluation of the data. For example, throughout this paper we have highlighted repeated task execution as a process problem to be identified and avoided, however in some processes such repetition could be beneficial or even necessary. Consider a fault resolution process where the fault is handed to a different business unit in some circumstances, there it would be helpful to test the problem before it is referred to avoid unnecessary referrals and also helpful to test after the fault is received back to ensure the fault is actually resolved. This highlights the necessity for analysis of the results of process mining to be carried out by people who have, or who have ready access to knowledge about the process under examination. If this information is not available, suboptimal decisions can be made both in preparing the data for mining and in the interpretation of the results.

As a result of all the data issues that were encountered during these case studies, it has become clear that the data import process, i.e. transforming the raw systems data into a viable process model, needs to be enhanced by adding domain knowledge that constrains the algorithms in order to first highlight any inconsistencies and also resolve the issues that are discovered by bringing into action a set of business rules that act on the data accordingly. In fact, one of the major benefits that our partners in the business teams have found in going through the process mining exercise is to discover many data quality issues of which they were not aware previously, and that affected the existing reporting process by giving erroneous results to many of the performance measurements.

References

1. Smith, A.: An Inquiry into the Nature and Causes of the Wealth of Nations, 5th edn., republished from: Edwin cannan's annotated edition. Methuen & Co., Ltd. (1904)
2. Hammer, M.: Reengineering work: don't automate, obliterate. Harvard Business Review 68(4), 104–112 (1990)
3. Davenport, T., Short, J.: The new industrial engineering: Information technology and business process redesign. Sloan Management Review, 11–27 (summer 1990)
4. Browning, T.R.: On the alignment of the purposes and views of process models in project management. Journal of Operations Management (November 2009)
5. Cardoso, J., Aalst, W., Bussler, C., Sheth, A., Sandkuhl, K.: Inter-Enterprise System and Application Integration: A Reality Check. In: Filipe, J., Cordeiro, J., Cardoso, J., Aalst, W., Mylopoulos, J., Rosemann, M., Shaw, M.J., Szyperski, C. (eds.) Enterprise Information Systems. LNBIP, vol. 12, pp. 3–15. Springer, Heidelberg (2009)

6. OMG: Business process model and notation (bpmn) version 2.0 (January 2011)
7. OASIS: Web services business process execution language version 2.0 (April 2007)
8. Eisner, J.: State-of-the-art algorithms for minimum spanning trees - a tutorial discussion (1997)
9. van der Aalst, W.M.P., Weijters, T., Maruster, L.: Workflow mining: Discovering process models from event logs. IEEE Transactions on Knowledge and Data Engineering 16(9), 1128–1142 (2004)
10. Majeed, B.: Us patent number 2011/0093308 a1: Process monitoring system (2011)
11. van der Aalst, W.M.P., Reijers, H.A., Weijters, A.J.M.M., van Dongen, B.F., Alves de Medeiros, A., Song, M., Verbeek, H.M.W.: Business process mining: An industrial application. Inf. Syst. 32(5), 713–732 (2007)
12. van der Aalst, W.M.P., van Dongen, B.F., Herbst, J., Maruster, L., Schimm, G., Weijters, A.J.M.M.: Workflow mining: a survey of issues and approaches. Data Knowl. Eng. 47, 237–267 (2003)
13. Mans, R., Schonenberg, M., Song, M., Aalst, W., Bakker, P.: Application of process mining in healthcare–a case study in a dutch hospital. In: Biomedical Engineering Systems and Technologies, pp. 425–438 (2009)

Discovering Workflow Changes with Time-Based Trace Clustering

Rafael Accorsi and Thomas Stocker

Albert-Ludwigs-Universität Freiburg, Germany
{accorsi,stocker}@iig.uni-freiburg.de

Abstract. This paper proposes a trace clustering approach to support process discovery of configurable, evolving process models. The clustering approach allows auditors to distinguish between different process variants within a timeframe, thereby visualizing the process evolution. The main insight to cluster entries is the "distance" between activities, i.e. the number of steps between an activity pair. By observing non-transient modifications on the distance, changes in the original process shape can be inferred and the entries clustered accordingly. The paper presents the corresponding algorithms and exemplifies its usage in a running example.

1 Introduction

Process discovery aims to reconstruct process models from execution logs. Specifically: given a log, each of the cases (i.e. execution traces) is analyzed, eventually producing a (Petri net) model of the process. The reconstructed model provides a basis for other process mining techniques, e.g. conformance checking and case prediction. Such techniques are not only useful for process designers, but essential tools for auditors analyzing process-aware information systems (PAIS).

However, process discovery techniques have problems with regard to the quality of produced models. Especially the emerging trend of PAIS to allow for configurable process models whose structure changes along time poses a challenge for their precision. Roughly speaking, process discovery consolidates all the different process instances into a single model which produces a coarse view of the underlying process.

Trace clustering techniques act as a preprocessing step for process mining [7,8,12], thereby allowing for a fine-grained set of models. The idea is to group traces according to different characteristics and, subsequently, mine a particular set of clusters. While clustering allows for the selective reconstruction of traces, it still fails to mine the complete "history" (i.e. *evolution provenance*) of a business processes, identifying their diverse "tenancies" and how they differ.

This paper presents an approach for time-oriented log-clustering that is able to directly reflect the dynamics of a process's structure. The goal is to cluster the cases belonging to one process structure, while different clusters mean that the process structure varies. The variation happens, e.g., when activities are *added* to (or *deleted* from) the process model or when the *order* of activities is

K. Aberer, E. Damiani, and T. Dillon (Eds.): SIMPDA 2011, LNBIP 116, pp. 154–168, 2012.

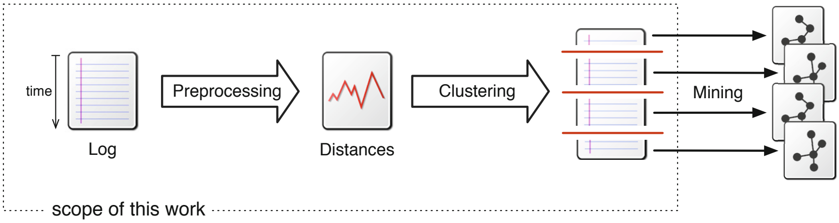


Fig. 1. Approach overview

changed. Thinking of process logs as sequential records of triggered activities, these variations are reflected in the log as modifications on the "distance" (i.e. number of entries) between pairs of activities. For instance, if an activity C is inserted between the activities A and B, then the distance between them increments by 1. Our approach clusters traces according to the time point where process variants have been introduced. In doing so, we obtain a chronological ordering of process tenancies which allows an auditor to appreciate the evolution of the original process.

Fig. 1 depicts our approach consisting of two steps. *Firstly*, the distance between the activities is measured. *Secondly*, the traces with similar "distance" behavior are clustered, so that process discovery techniques can be applied. Identified clusters can then serve as input for arbitrary discovery algorithms to construct process models for further analysis, e.g. [2,3].

Altogether, this paper provides the following contributions:

- It introduces the "activity distance" as a basis for trace clustering (Sec. 2).
- It provides a clustering method and the corresponding algorithms (Sec. 3).
- It reports on tests employing the approach and discusses the results (Sec. 4).

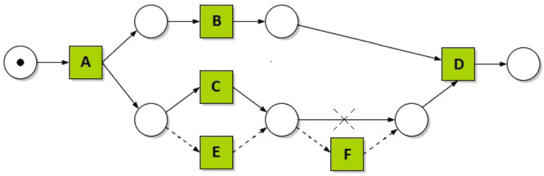
The overall goal is to devise techniques powerful and scalable enough to cope with industry-size data. The current implementation of the technique is a standalone application. In future, the technique is going to be added to ProM [14] and be part of the Security Workflow Analysis Toolkit (SWAT) [4], thereby making it possible for people to experiment with it.

2 Extracting Activity Distances

This section introduces the shape of logs taken into account, introduces the concept of "distance", and defines algorithms to extract the distances.

2.1 Logs and Distances

We assume that log entries have a timestamp and are chronologically ordered. Fig. 2 shows a workflow as a Petri net with a corresponding log. Adding the



Trace	Activities
1	A B C D
2	A C B D
3	A C B D
4	A B C D
5	A B C F D
6	A E F B D
7	A B E F D

Fig. 2. Workflow net and corresponding log

dashed edges leads to another version of this workflow with the two additional activities E and F. Traces (also called cases) related to the new workflow structure appear in the log below the dashed line. Petri nets are often used as a meta-model to describe workflow models. They consist of transitions (rectangles) that stand for workflow activities and places (circles) that can contain tokens used to model the control flow. The execution of activities is modeled in terms of "firing" transitions. Transitions fire if they are "enabled" which means that there is at least one token in each input place. Upon firing, they take one token from each input place and put one token in each output place. The net in Fig. 2 contains a single enabled transition which is A. See [11] for details on Petri nets.

Process logs and traces are defined as follows:

Definition 1. *Let A be a set of activities of a process. $\sigma \in A^*$ is a log trace and $L = \{\sigma_1, ..., \sigma_n\} \in \mathcal{P}(A^*)$ is a process log. The order of traces is given by the starting time of the corresponding instance and reflected on the indexes σ_i.* ⊣

Loops in process models can cause activity names to occur several times within the same trace. For the sake of simplicity, we do not consider loops. The distance between any two process activities within a trace is defined as the number of intermediate activities. Considering the first trace of the log in Fig. 2, the distance between A and D is 2. Formally distance is defined as follows:

Definition 2. *Let $\sigma = \{a_1, ..., a_n\}$ be a trace containing activities a_i. The* ***distance*** *between any two subsequent trace-activities $a_i, a_j \in \sigma$, $j>i$ is defined as $d_\sigma(a_i, a_j) := j - i - 1$.* ⊣

If two activities happen sequentially within a process, their distance remains constant over time, otherwise the distance varies in fixed boundaries conditioned by a minimum and maximum distance. Consider the activity pair (A, D) in Fig. 2: $d(A, D) = 2$ holds within the first 4 traces, whereas the distance $d(B, D)$ ranges in the interval $[0; 1]$.

Structural changes in the process cause interval variations for activity-pair distances. Fig. 3 shows the distance progress for the activity pair (B, D) in Fig. 2. Within the first four traces the distance remains inside the interval $[0, 1]$, but with trace 5 this interval is enlarged to $[0, 2]$. Depending on the number

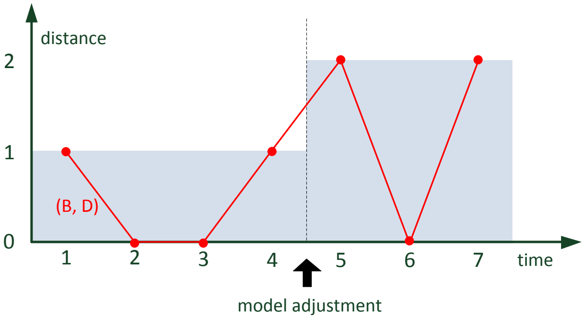


Fig. 3. Distance graph of the activity pair (B,D)

Trace	Activities	Distances						
		(A,B)	(A,C)	(A,D),	(B,C)	(B,D)	(C,B)	(C,D)
1	A B C D	0	1	2	0	1	-	0
2	A C B D	1	0	2	-	0	0	1
3	A C B D	1	0	2	-	0	0	1
4	A B C D	0	1	2	0	1	-	0
	intervals	[0;1]	[0;1]	2	0	1	0	[0;1]

Fig. 4. Distance matrix for the upper part of the log in Fig. 2

and position of inserted or removed workflow activities and links between them, distance intervals are enlarged, reduced or moved (same value range, but different starting point, i.e. $[1,3] \rightarrow [2,4]$).

2.2 Extracting Activity Distances

Algorithm 1 constructs a distance matrix containing distances of activity-pairs as a preprocessing step for the clustering procedure presented in Sec. 3. For every pair of subsequent activities, this matrix contains their distance within every trace of the log. Fig. 4 shows the distance matrix for the traces 1–4 in Fig. 2.

Definition 3. *Let $L = \{\sigma_1, ..., \sigma_n\}$ be a log and A the set of activities in L. A distance matrix M_L holds for every $a', a'' \in A$ a distance list $D_{a',a''} := [d_{\sigma_1(a',a'')}, ..., d_{\sigma_n(a',a'')}]$. Distances for a specific index $k \in [1;n]$ are represented by $D_{a',a''}[k] := d_{\sigma_k(a',a'')}$. The number of observations within a set $D'_{a',a''} \subseteq D_{a',a''}$ is defined as: $obs(D'_{a',a''}) := |\{d_{\sigma(a',a'')} \in D' \mid d_{\sigma(a',a'')} \neq null\}|$.* ⊣

Inserted “null” values (denoted $-$) stand for the fact that not all the traces contain all possible activitiy-pairs. This ensures that the distance lists have equal sizes, which is required by the clustering algorithm. Alg. 1 shows how the distance matrix is built.

Algorithm 1. Build distance matrix

```
procedure BuildDistanceMatrix(workflow log $L = \{\sigma_1, \sigma_2, ...\}$)
    for all $k = 1, |L|$ do
        for $i = 1, |\sigma_k| - 1$ do
            for $j = i, |\sigma_k|$ do
                $D_{\sigma_k[i],\sigma_k[j]}[k] \leftarrow (j - i - 1)$
            end for
        end for
        insert null for all other activity-pairs.
    end for
end procedure
```

3 Clustering Log Traces

Our trace clustering approach is interactive and consists of two phases. The first phase determines the *cluster cuts* for every observed activity pair (a', a''), i.e. the time-points (trace numbers) at which distance interval changes of (a', a'') suggest to end one cluster and to begin a new one. The second phase merges these individual results by counting the number of activity pairs at time point i that would cut the log at that position. As a result, one can browse through a chart showing these counts and, starting with a rather low window size, check how changing this parameter affects the accumulated cluster cuts.

3.1 Determining Cluster Cuts

Sequentially processing traces according to their timestamp (in this case, trace index), the clustering algorithm uses samples to determine the typical workflow behavior in terms of boundaries for the minimum and maximum observed value of activity distances. Such a behavior is determined on the basis of a parameter w (window size) that defines the minimum number of traces used as "training" data (sample size) and the minimum number of traces grouped as a cluster. Note that such a step is required since "change" always relates to typical behavior that has to be defined or measured first. The initial sample contains the first w traces of the log. As long as following traces show typical behavior, they are clustered together, otherwise a new sample of size w is created and used as a basis to determine the "new" typical behavior. The following defines these notions:

Definition 4. *Let $W_{s,t} = \{\sigma_s, ..., \sigma_t\} \subseteq L$ be a sample of a log $L = \{\sigma_1, ..., \sigma_n\}$ with size $|W_{s,t}| \geq w$. The minimum and maximum distances of any two activities a', a'' within W are defined as $min_{a',a''}(W_{s,t}) := min(d_{\sigma_s}(a', a''), ..., d_{\sigma_t}(a', a''))$ and $max_{a',a''}(W_{s,t}) := max(d_{\sigma_s}(a', a''), ..., d_{\sigma_t}(a', a''))$. The corresponding distance interval is defined as $[min_{a',a'}(W_{s,t}); max_{a',a'}(W_{s,t})]$. The projection of a distance list on a sample is defined as: $D_{a',a''}|W_{s,t} := \{d_{\sigma_s(a',a'')}, ..., d_{\sigma_t(a',a'')}\} \subseteq D_{a',a''}$.* ⊣

Definition 5. *The support of a distance γ between (a', a'') within a window W is defined as: $supp_{a',a''}(W_{s,t}, \gamma) := |\{d_\sigma(a', a'') \in W \mid d_\sigma(a', a'') = \gamma\}|$.* ⊣

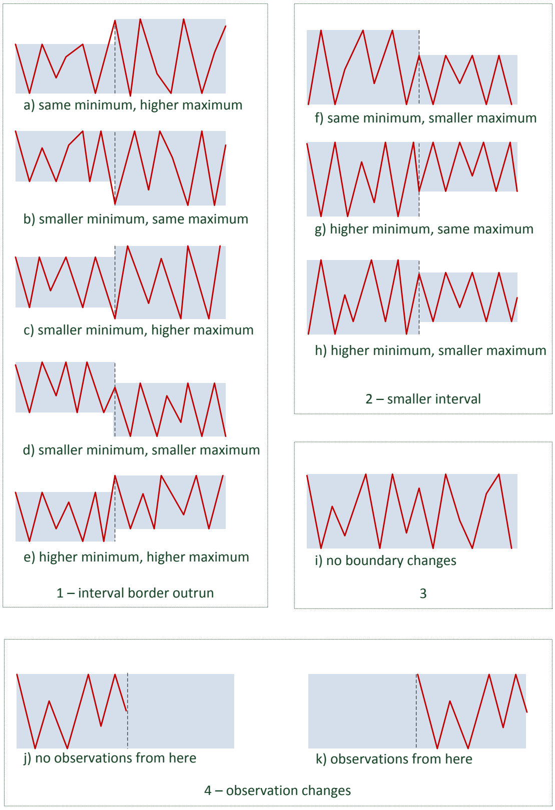

a) same minimum, higher maximum
b) smaller minimum, same maximum
c) smaller minimum, higher maximum
d) smaller minimum, smaller maximum
e) higher minimum, higher maximum
1 – interval border outrun
f) same minimum, smaller maximum
g) higher minimum, same maximum
h) higher minimum, smaller maximum
2 – smaller interval
i) no boundary changes
3
j) no observations from here
k) observations from here
4 – observation changes

Fig. 5. Possible changes for distance intervals

Considering a sample $W_{s,i-1}$ that holds all traces accumulated so far, for every (a', a'') of the next trace σ_i it is checked if the actual distance value of (a', a'') changes the corresponding distance interval in $D_{a',a''}|W_{s,i-1}$. To distinguish between momentary and persistent changes, the clustering method uses a w-size lookahead $W_{i,i+w-1}$. Fig 5 shows the possible cases of distance interval changes. In detail there are four different types of interval changes that introduce new clusters:

1. **(C1) Interval border outrun**. The actual distance of (a', a'') *outruns* its typical distance interval within the sample (cases in Group 1 in Fig. 5). To check the persistency of this change, the support of the new distance $d_{\sigma_i}(a', a'')$ within the lookahead is checked, using a threshold τ.
   case condition: $d_{\sigma_i}(a', a'') > max(W_{s,i-1}) \vee d_{\sigma_i}(a', a'') < min(W_{s,i-1})$.
   persistency condition: $supp_{a',a''}(W_{i,i+w-1}, \tau) \geq \tau$.
2. **(C2) Smaller interval**. The actual distance of (a', a'') *remains inside* the typical interval but persistently reduces the size of the corresponding distance interval (cases in Group 2 in Fig. 5).

case condition: $d_{\sigma_i}(a',a'') \geq min_{a',a''}(W_{s,i-1}) \wedge d_{\sigma_i}(a',a'') \leq max_{a',a''}(W_{s,i-1})$

persistency condition: $(min_{a',a''}(W_{i,i+w-1}) > min_{a',a''}(W_{s,i-1}) \wedge max_{a',a''}(W_{i,i+w-1}) \leq max_{a',a''}(W_{s,i-1})) \vee (max_{a',a''}(W_{i,i+w-1}) < max_{a',a''}(W_{s,i-1}) \wedge min_{a',a''}(W_{i,i+w-1}) \geq min_{a',a''}(W_{s,i-1}))$

3. **(C3) Distance observations from here**. While there are no distance observations for (a',a'') within $W_{s,i-1}$, the actual trace σ_i provides a distance value (case j in Group 4 in Fig. 5).
   case condition: $obs(D_{a',a''}|W_{s,i-1}) = 0 \wedge d_{\sigma_i}(a',a'') \neq \textbf{null}$
4. **(C4) No distance observations from here**. While there were distance observations for (a',a'') in $W_{s,i-1}$, the actual trace σ_i contains a phase where no distance values for (a',a'') are observed (Case k in Group 4 in Fig. 5).
   case condition: $obs(D_{a',a''}|W_{s,i-1}) \neq 0 \wedge obs(D_{a',a''}|W_{i,i+w-1}) = 0$

As long as the distances of observed traces do not introduce new distance intervals, they are put in the same cluster. Once a new interval is detected, the typical behavior is calculated again by choosing a new sample on the basis of the next w traces. This approach allows the identification of modifications and the time point from which they hold, thereby allowing time-based clustering. If distance values were observed both for (a',a'') and (a'',a') within the actual cluster $W_{s,i-1}$, the activities a' and a'' are likely to be in parallel. Since there typically are lots of interval changes within the same process model for parallel activities, distance interval changes for such activity pairs are not considered as indicators for workflow changes.

3.2 Consolidating Cluster Cuts

After determining the cluster cuts for every observed activity pair individually, the suggested cut positions are combined by counting the number of activity pairs for every time point (trace number) that have a cluster cut at this position and visualized as cluster cut graphs. With the help of a prototypical implementation it is possible to generate such graphs for different window sizes on the fly which gives auditors the chance to identify promising log regions for change point detection in an interactive and intuitive manner.

Besides high peaks – which clearly indicate a workflow change –, regions with an accumulation of smaller peaks are interesting, too. Not in every case a process change affects a large number of activity pairs at once, but rather progressively over a set of traces. For the same reason, peaks or accumulations occur slightly shifted with respect to the actual changing point, especially in case of minor process changes and when "old" traces are still executable within the new model.

Besides the peaks indicating structural process changes, there may be other smaller peaks. They occur if within the same model a set of paths is preferred for a while and can be interpreted as "intra-model changes". Depending on the chosen window size and structure of considered workflow models the number of peaks strongly varies. The smaller the window size, the more changes are

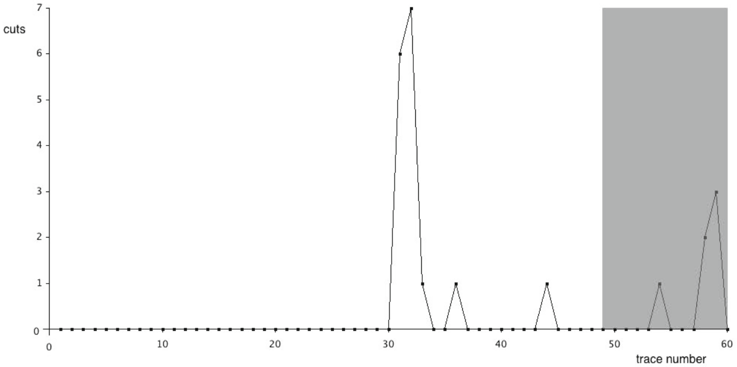


Fig. 6. Combined cluster cuts for an example log of the nets in Fig. 2

detected. Considering a smaller number of traces limits the typical behavior which causes the probability of observing unseen/unknown behavior to grow. An interesting characteristic of cluster cut graphs is, that in most cases the peaks for "real" changes seem to be more stable with varying window sizes.

Fig. 6 shows the resulting graph after applying Algorithm 2 to a log containing 30 randomly generated traces for the first net followed by 30 traces of the second net in Fig. 2. Using a window size of 11, the cluster cut graph shows some high peaks around trace 30 which means that a high number of activity pairs experienced “change” in this region. This is in fact the region where the underlying model for the generated log traces changed. From the point on where the lookahead contains less traces than the window size, more changes are detected because the persistency of changes is based on a smaller number of traces than it is the case before. Simply combining the last w traces to one cluster or unifying them with the last detected cluster is a rather strict approach. We decided keep the cut information and mark the “critical region” (gray area within the graph) to give auditors the chance to decide on its usefulness. Alg. 2 provides the method to determine cluster cuts for particular activity pairs.

4 Testing the Clustering Approach

The clustering method is tested using the non-trivial process shown in Fig. 7. It contains 15 different activities that are composed using parallel and conditional branches. Its size and structural complexity is not exceeded by the vast majority of industrial processes. In order to show the detection rate of workflow changes, a set of process variants was generated, based on the change patterns by Weber et al.[17]. Using process simulation techniques, test logs containing traces for each variant are generated, serving as input for the change point detection method.

Algorithm 2. Determine cluster cuts for specific activity pair

```
function GETCUTS(log $L$, distance matrix $D$, window size $w$, activity pair $(a', a'')$)
    $window, lookahead, C \leftarrow \emptyset$
    for $index = 0, |L| - 1$ do
        if $|window| < w$ then
            $window \leftarrow window \cup \sigma_{index}$
            $lookahead \leftarrow lookahead \cup \sigma_{index+w}$
        else
            $actualValue \leftarrow D_{a',a''}[index]$
            if $actualValue \neq$ null but it always is in $window$ then
                NEWCLUSTER(index-1)
            else if $actualValue =$ null but there are non-null values in $window$ then
                NEWCLUSTER(index-1)
            else
                $violated \leftarrow actualValue < min_{a',a''}(interval) \vee$
                            $actualValue > max_{a',a''}(interval)$
                $conform \leftarrow actualValue \geq min_{a',a''}(interval) \wedge$
                            $actualValue \leq max_{a',a''}(interval)$
                $parallelPair \leftarrow obs(D_{a'',a'}|window) = 0$
                if $violated \wedge supp_{a',a''}(lookahead, actualValue) \geq \tau \wedge \neg parallelPair$ then
                    NEWCLUSTER(index-1)
                else if $conform$ then
                    $higherMin \leftarrow min_{a',a''}(lookahead) > min_{a',a''}(interval) \wedge$
                                $max_{a',a''}(lookahead) \leq max_{a',a''}(interval)$
                    $lowerMax \leftarrow max_{a',a''}(lookahead) < max_{a',a''}(interval) \wedge$
                                $min_{a',a''}(lookahead) \geq min_{a',a''}(interval)$
                    if $(higherMin \vee lowerMax) \wedge \neg parallelPair$ then
                        NEWCLUSTER(index-1)
                    end if
                end if
            end if
        end if
    end for
    return C
end function
procedure NEWCLUSTER(endIndex)
    $C = C \cup endIndex$
    $interval, lookahead \leftarrow \emptyset$
end procedure
```

Instead of focusing on simple operations (add/remove activities), we consider more complex operations to allow for a more intuitive understanding and interpretation of the change types that can be successfully detected. See [17] for an in-depth description of the considered change patterns. In this paper, the following set of change patterns to derive process variants is considered:

- **(C1) Serial Insert:** Insert a process fragment between two directly succeeding workflow activities.

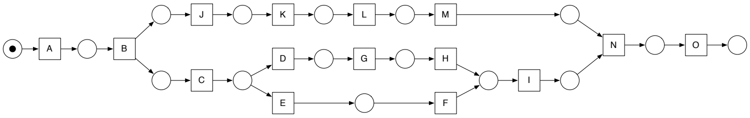
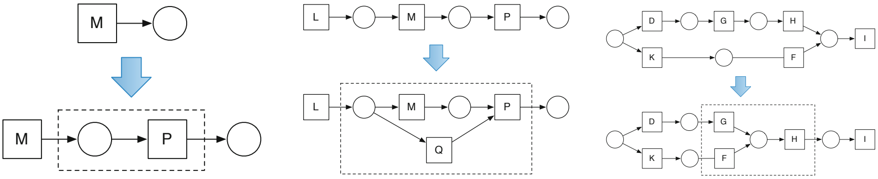


Fig. 7. Process for generating process variants and trace clustering



(a) $M_0 \xrightarrow{(C1)} M_1$ (b) $M_1 \xrightarrow{(C2)} M_2$ (c) $M_4 \xrightarrow{(C5)} M_5$

Fig. 8. Examples of workflow changes by applying change patterns

- **(C2) Conditional Insert:** Insert a process fragment between two activity sets with an additional condition.
- **(C3) Parallel Insert:** Insert a process fragment between two activity sets with an additional condition.
- **(C4) Delete:** Remove a process fragment.
- **(C5) Move:** Shift a process fragment to another position.
- **(C6) Swap:** Swap a process fragment with another one.
- **(C7) Replace:** Replace a process fragment with another one.
- **(C8) Extract Sub Process:** Extract an existing process fragment and replace it by a "reference activity".
- **(C9) Parallelize:** Parallelize previously sequential process fragments.
- **(C10) Inline Sub Process:** Replace a "reference activity" by the referenced process fragment.
- **(C11) Embed Fragment in Conditional Branch:** Bind the execution of an existing process fragment to a condition.
- **(C12) Embed Process Fragment in Loop:** Add a loop construct surrounding an existing process fragment.

Successively applying these patterns to the original process M_0 in Fig. 7 results in a set of 13 models $M_0, ..., M_{12}$. Each model M_i was created by applying pattern **(Ci)** to the previous model M_{i-1}. Due to space limitations the process variants are not explicitly depicted in this paper, but Fig. 8 exemplary shows how some of the variants are formed.

All process variants have been modeled with PNML and simulated with a proprietary simulation module of SWAT [4]. In total, we create a process log with 13.000 traces. The flog in Fig. 9 contains different numbers of traces for different process variants successively.

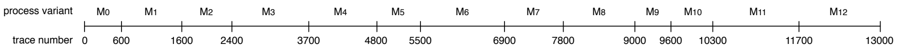


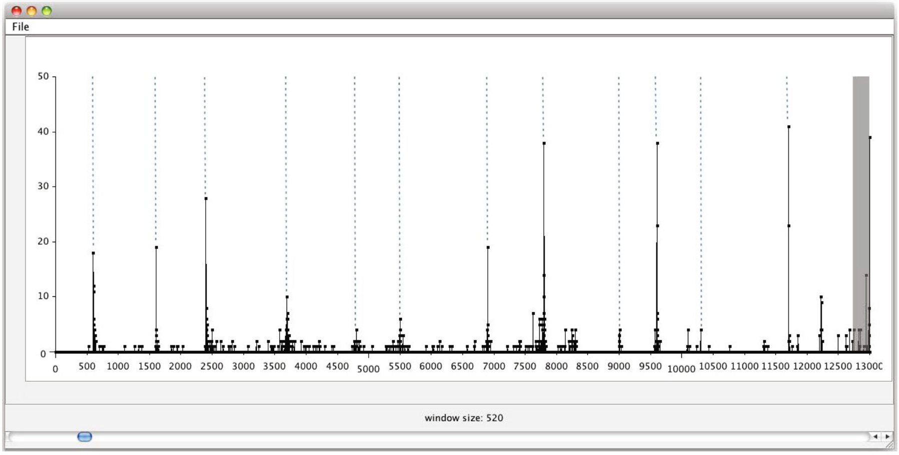
Fig. 9. Log with different trace numbers for each process variant

To evaluate the effectiveness of our clustering method we first check if it successfully detects the change points between the different process variants. Fig. 10 shows the resulting cluster cut graph for the log in Fig. 9 in the clustering-application prototype. With the chosen window sizes, most of the change points are clearly identifiable by considering the highest peaks. It should be mentioned that using other window sizes does not significantly change the shape of the resulting graphs but simply results in a higher or lower number of additional peaks around change points.

However there are peaks which are not caused by model changes and/or cannot be directly obtained by graph analysis because they do not produce noticeable peaks, e.g. the model M_{10} and M_{11}. Taking a deeper look at this specific change reveals that the only difference between the two models is that an execution condition is added to a mandatory process fragment of M_{10} which additionally allows traces where the fragment does not occur. In contrast to the removal of a process fragment which is obtainable by the absence of distance observations for affected activity pairs in following traces, in this case the detection fails because the fragment may or may not occur and so there are no activity pairs whose distance is persistently altered by the change.

Another change that is not clearly identifiable is from M_4 to M_5 (see Fig. 8(c)) by moving the last activity (H) out of a conditional branch the corresponding join. Since changes between parallel activity pairs are omitted by the clustering method and only 2 sequential activities are affected by this change (namely E and F) the low peak at trace 4.800 is hardly surprising. The more locally constrained changes are and the less activities are affected, the harder their detection. This also applies to the change from M_5 to M_6 which is caused by swapping the nearby activities N and P. Although in this case the change is still identifiable within the graph in Fig. 10. Still, the cluster cut graphs provide valuable information for the identification of process changes which shows the appropriateness of monitoring activity pair distances for this purpose.

Change Localization. Besides the detection of change points, another goal of time-based trace clustering is *change localization* [6]. For this, the regions of change in the process models are of interest. In our approach, the region of change is obtained by considering the corresponding activity pairs for the peaks in the cluster cut graph. Zooming in on trace 600 in Fig. 10, e.g., reveals two high peaks, one in trace 601 with 18 related activity pairs and another in trace 603 with 11 related pairs. The resultant set of activity pairs for these two peaks is depicted in Table 1 and shows that most activity pairs refer to the activities $\{P, N, O\}$. Considering the change from Model M_0 to model M_1 in Fig. 8(a), this set contains the newly inserted activity P being the most referenced activity (it is contained in 15 out of 29 activity pairs), plus its successors.



window size: 520

Fig. 10. Resulting cluster cut graph for the log in Fig. 9

Table 1. Activity pairs for the peaks around trace 600 in Fig. 10

group 1	group 2	group 3
(A,N) (A,O) (A,P)	(P,N) (P,O)	(C,P)
(B,N) (B,O) (B,P)		(D,P)
(E,N) (E,O) (E,P)		(G,P)
(F,N) (F,O) (F,P)		(H,P)
(I,N) (I,O) (I,P)		(L,P)
(J,N) (J,O) (J,P)		(M,P)
(K,N) (K,O) (K,P)		

5 Discussion

While our experiments show that time-based clustering is feasible for different types of changes, some general observations wrt the window-based apply.

There are various parameters that influence change point detection; the window size seems to be the most important one. Depending on this parameter, changes may become undiscoverable. This can either happen when the window size is too small which results in a higher number of peaks and "real" changes become indistinguishable from momentary changes (*diffusion*), or when changes are *masked* due to the inability of detecting clusters whose sizes are smaller than the window size itself (this is an inherent disadvantage of window-basing). Instead of relying on an optimal window size, our approach solves this problem by allowing users to identify changes by analyzing the variance of cluster cut graphs with respect to different window sizes. To the best of our knowledge, this is the first approach allowing such kind of interactive change point detection.

Besides the window size, the variance of cluster sizes also plays a role. If the number of traces relating to different process variants significantly differ, there is no window size that can uncover all change points without masking or diffusing others. In case of similar trace numbers, the precision of change point detection is proportional to the average number of traces relating to the process variants. The more traces can be taken into account on determining the typical process behavior, the more precisely the process characteristics can be captured.

Another insight is that the difficulty to to detect changes strongly depends on the number of affected activities. For most patterns, structural characteristics of the original and the resulting process have influence on the ability to detect change points. Due to the observations we did so far, the number of activities that are affected by a change and the structural complexity of the underlying process (degree of parallelity) seem to exert the highest influence. However there are change patterns, that are generally hard to detect independent of these structural characteristics such as **(C11)**.

6 Related Work

Trace clustering approaches aim to group process traces according to some criteria. One of the first attempts in this area is multi-phase process mining [15,16]. Initially generating a process model for each individual process instance/case, these instance graphs are stepwise aggregated based on their similarity to introduce more general models. Greco et al. [8] propose a clustering technique in form of a stepwise refinement of a set of workflow schemas modeling specific usage scenarios. Starting with a single workflow schema describing the behavior of the overall process, in each refinement step the model with the lowest "soundness value" is selected for being refined. While "completeness" measures the covering rate of a model with respect to a process log, "soundness" is able to decide on over-generalization by measuring the percentage of enactments of the mined model that find some correspondence in the log [8].

Instead of iteratively checking the appropriateness of mined models to decide on further clustering, some approaches operate at trace level and focus on trace-similarity. Utilizing *k-means* for clustering, Song et al. [12] use features to characterize sets of traces. Trace-similarity is measured by distances of feature-vectors (i.e. the number of activities of a trace). Bose et al. [6] map the problem of multiple sequence alignment in bioinformatics on the trace-clustering problem to identify groups of similar traces. In [10] Lakshmanan et al. utilize spectral graph analysis to compute the difference of cluster-graphs to identify changes. Obtained models reflect different process characteristics but are loosely coupled with each other, hence failing to show the change of process models along time.

Recent mining approaches consider change in workflows [9]. In [10] Lakshmanan et al. utilize spectral graph analysis to compute the difference of cluster-graphs to identify changes. However by using fixed cluster sizes, this approach is rather imprecise. Workflow dynamics in the sense of context changes can be characterized as *concept drift*, which is a well-studied paradigm in the data mining area. Bose et al. [5] provide methods to handle concept drift in process mining by showing that workflow changes are indirectly reflected in workflow logs and change point detection is feasible by examining activity relations.

In contrast to [5], our approach does not consider ordering relations (follows/precedes) but variations on the distance of activity pairs within a trace (the number of intermediate activities) as structural property. Sequentially processing traces according to time, traces whose structure differs from the typical trace structure so far are treated as indicators for new clusters. Proceeding this way ensures that a log file is partitioned into chronologically subsequent clusters, whose sizes depend on the frequency of changes in process structure. However, we do not identify the kind of change, but focus on the identification of time points of permanent process changes.

7 Summary and Further Work

This paper presents a trace clustering approach to support process discovery of configurable, evolving process models. The clustering approach allows users to distinguish between different process variants within a timeframe, thereby visualizing the process evolution. The main insight to cluster entries is the "distance" between activities, i.e. the number of steps between an activity pair. The evaluation shows that the approach is promising. Still, there are several issues that remain to be addressed. The following presents the most important issues:

1. **Loop support.** The clustering method at this point does not support workflow models containing loops which definitely is its biggest disadvantage as it limits its applicability. We are confident that an enhancement in this direction is straightforward and can be done without negatively influencing the effectiveness of the method.
2. **Identification of change patterns.** Although our experiments convey an impression of the identifiability of change patterns, a more systematic approach is needed to provide a well-founded basis for their effective detection.
3. **Feature set.** So far our approach solely relies on the activity pair distance metric, but incorporating additional metrics may enable more precise change point detection. For change patterns like **(C11)** where the identification of change points is not straightforward, additionally considering the length of log traces of subsequent samples may be useful.
4. **Change localization.** As mentioned before, the proposed clustering method provides sufficient information for the localization of change within workflow models. Here, automatic mechanisms for the extraction of change regions from a set of affected activity pairs are envisioned.

References

1. Accorsi, R., Stocker, T.: On the exploitation of process mining for security audits: The conformance checking case. In: ACM Symposium on Applied Computing, pp. 1709–1716. ACM (2012)
2. Accorsi, R., Wonnemann, C.: Auditing Workflow Executions against Dataflow Policies. In: Abramowicz, W., Tolksdorf, R. (eds.) BIS 2010. LNBIP, vol. 47, pp. 207–217. Springer, Heidelberg (2010)

3. Accorsi, R., Wonnemann, C.: Strong non-leak guarantees for workflow models. In: ACM Symposium on Applied Computing, pp. 308–314. ACM (2011)
4. Accorsi, R., Wonnemann, C., Dochow, S.: SWAT: A security workflow toolkit for reliably secure process-aware information systems. In: Conference on Availability, Reliability and Security, pp. 692–697. IEEE (2011)
5. Bose, R.P.J.C., van der Aalst, W.M.P., Žliobaitė, I.e., Pechenizkiy, M.: Handling Concept Drift in Process Mining. In: Mouratidis, H., Rolland, C. (eds.) CAiSE 2011. LNCS, vol. 6741, pp. 391–405. Springer, Heidelberg (2011)
6. Jagadeesh Chandra Bose, R.P., van der Aalst, W.: Trace Alignment in Process Mining: Opportunities for Process Diagnostics. In: Hull, R., Mendling, J., Tai, S. (eds.) BPM 2010. LNCS, vol. 6336, pp. 227–242. Springer, Heidelberg (2010)
7. Alves de Medeiros, A.K., Guzzo, A., Greco, G., van der Aalst, W.M.P., Weijters, A.J.M.M.T., van Dongen, B.F., Saccà, D.: Process Mining Based on Clustering: A Quest for Precision. In: ter Hofstede, A.H.M., Benatallah, B., Paik, H.-Y. (eds.) BPM Workshops 2007. LNCS, vol. 4928, pp. 17–29. Springer, Heidelberg (2008)
8. Greco, G., Guzzo, A., Pontieri, L., Saccà, D.: Discovering expressive process models by clustering log traces. IEEE Transactions on Knowledge and Data Engineering 18(8), 1010–1027 (2006)
9. Günther, C., Rinderle-Ma, S., Reichert, M., van der Aalst, W.M.P., Recker, J.: Using process mining to learn from process changes in evolutionary systems. Int. J. Business Process Integration and Management 1, 111 (2007)
10. Lakshmanan, G., Keyser, P., Duan, S.: Detecting changes in a semi-structured business process through spectral graph analysis. In: Workshops of the Conference on Data Engineering, pp. 255–260. IEEE (2011)
11. Murata, T.: Petri nets: Properties, analysis and applications. Proceedings of the IEEE 77(4), 541–580 (1989)
12. Song, M., Günther, C.W., van der Aalst, W.M.P.: Trace Clustering in Process Mining. In: Ardagna, D., Mecella, M., Yang, J. (eds.) Business Process Management Workshops. LNBIP, vol. 17, pp. 109–120. Springer, Heidelberg (2009)
13. van der Aalst, W.M.P.: Process Mining – Discovery, Conformance and Enhancement of Business Processes. Springer (2011)
14. van Dongen, B.F., Alves de Medeiros, A.K., Verbeek, H.M.W(E.), Weijters, A.J.M.M., van der Aalst, W.M.P.: The ProM Framework: A New Era in Process Mining Tool Support. In: Ciardo, G., Darondeau, P. (eds.) ICATPN 2005. LNCS, vol. 3536, pp. 444–454. Springer, Heidelberg (2005)
15. van Dongen, B.F., van der Aalst, W.M.P.: Multi-phase Process Mining: Building Instance Graphs. In: Atzeni, P., Chu, W., Lu, H., Zhou, S., Ling, T.-W. (eds.) ER 2004. LNCS, vol. 3288, pp. 362–376. Springer, Heidelberg (2004)
16. van Dongen, B., van der Aalst, W.M.P.: Multi-phase process mining: Aggregating instance graphs into EPCs and Petri nets. In: PNCWB 2005 Workshop, pp. 35–58 (2005)
17. Weber, B., Rinderle, S., Reichert, M.: Identifying and evaluating change patterns and change support features in process-aware information systems. Technical Report. University of Twente, Enschede, The Netherlands (2007)

Striving for Object-Aware Process Support: How Existing Approaches Fit Together

Vera Künzle and Manfred Reichert

Institute of Databases and Information Systems, Ulm University, Germany
{vera.kuenzle,manfred.reichert}@uni-ulm.de

Abstract. Many limitations of contemporary process management systems (PrMS) can be traced back to the missing integration of processes and data. A unified understanding of the inherent relationships existing between processes and data, however, is still missing. In the PHILharmonicFlows project we figured out that process support often requires *object-awareness*. This means, data must be manageable in terms of object types comprising object attributes and relations to other object types. In this paper, we systematically introduce the fundamental characteristics of object-aware processes. Further, we elaborate existing approaches recognizing the need for a tighter integration of processes and data along these characteristics. This way, we show the high relevance of the identified characteristics and confirm that their support is needed in many application domains.

Keywords: Process-aware Information Systems, Object-aware Process Management, Data-driven Process Execution.

1 Introduction

Despite the widespread adoption of existing process management systems (PrMS) there exist numerous processes which are currently not adequately supported by these PrMS. In this context, it has been confirmed by different authors that many limitations of contemporary PrMS can be traced back to the missing integration of processes and data [1,2,3,4,5,6,7,8]. However, a unified understanding of the inherent relationships existing between processes and data is still missing.

In the PHILharmonicFlows[1] project, we analyzed various processes from different domains which require a tight data integration [9,10,11]. We figured out that the support of many of these processes requires *object-awareness*; i.e., these processes focus on the processing of business data represented through *business objects*. The latter comprise a set of *object attributes* and are *inter-related*. In this context, business processes coordinate the processing of business objects among different users enabling them to cooperate and communicate with each other.

Existing PrMS, however, focus on business functions and their control flow, whereas business objects are ”unknown” to them. Most PrMS only cover simple

[1] Process, Humans and Information Linkage for harmonic Business Flows.

K. Aberer, E. Damiani, and T. Dillon (Eds.): SIMPDA 2011, LNBIP 116, pp. 169–188, 2012.

data elements, which are needed for control flow routing and for supplying input parameters of activities. Business objects, in turn, are usually stored in external databases and are outside the control of the PrMS. For this reason, existing PrMS are unable to adequately support *object-aware processes* [12].

In this paper, we introduce the main characteristics of object-aware processes which we gathered in several case studies [9,10] (see [13] for details about the research methodology we applied). Following this, we evaluate to what extent existing data-aware or data-driven process support paradigms support these characteristics. Overall, this evaluation reveals three major results: First, the characteristics we identified for object-aware processes are of high relevance. Second, object-aware process support is needed in many domains. Third, a comprehensive framework for object-aware process management is still missing.

The paper is structured as follows. In Section 2 we elaborate the role of business objects in the context of process management in detail and introduce fundamental characteristics of object-aware processes along a running example. Following this, we evaluate existing approaches against these characteristics in Section 3. Section 4 discusses the outcomes of this evaluation. Finally, Section 5 introduces a comprehensive framework for object-aware process management. We close with a summary and outlook in Section 6.

2 Object-Aware Process Support

We first discuss fundamental characteristics of object-aware processes which constitute aggregations of more detailed property lists. The latter rely on an extensive analysis of processes currently not adequately supported by PrMS [9,10,12,11]. To ensure that the processes we analyzed are not "self-made" examples, but constitute real-world processes of high practical relevance, we particularly considered processes as implemented in existing business applications. In addition, we rely on extensive practical experiences gathered when developing contemporary business applications; i.e., we have deep insights into their application code and process logic. In order to justify our findings, we complemented our process analyses by an extensive literature study to ensure both importance and completeness. With the latter we want to ensure that we have not excluded important properties already identified by other researchers. However, in this study we did not consider properties in respect to process change and process evolution. Instead, our focus was on process modeling, execution and monitoring. In order to emphasize the relevance of the identified characreristics and their inter-relations we contrast them with the different application examples considered by existing approaches (cf. 4) As illustrated in Fig. 1, we discuss the characteristics along a (simplified) real-world scenario for recruiting people as known from human resource management.

Example 1 (Recruitment Example). *In the context of recruitment,* `applicants` *may apply for* `job vacancies` *via an Internet online form. Once an* `application` *has been submitted, the responsible* `personnel officer` *in the human resource department is notified. The overall process goal is to decide which* *applicant shall get the job. If an application is ineligible the applicant is immediately rejected. Otherwise, personnel officers may request internal reviews for each applicant. In this context, the concrete number of reviews may differ from application to application. Corresponding review forms have to be filled by employees from functional divisions. They make a proposal on how to proceed; i.e., they indicate whether the applicant shall be invited for an interview or be rejected. In the former case an additional appraisal is needed. After the employee has filled the review form she submits it back to the personnel officer. In the meanwhile, additional applications might have arrived; i.e., reviews relating to the same or to different applications may be requested or submitted at different points in time. The processing of the application, however, proceeds while corresponding reviews are created; e.g., the personnel officer may check the CV and study the cover letter of the application. Based on the incoming reviews he makes his decision on the application or initiates further steps (e.g., interviews or additional reviews). Finally, he does not have to wait for the arrival of all reviews; e.g., if a particular employee suggests hiring the applicant he can immediately follow this recommendation.*

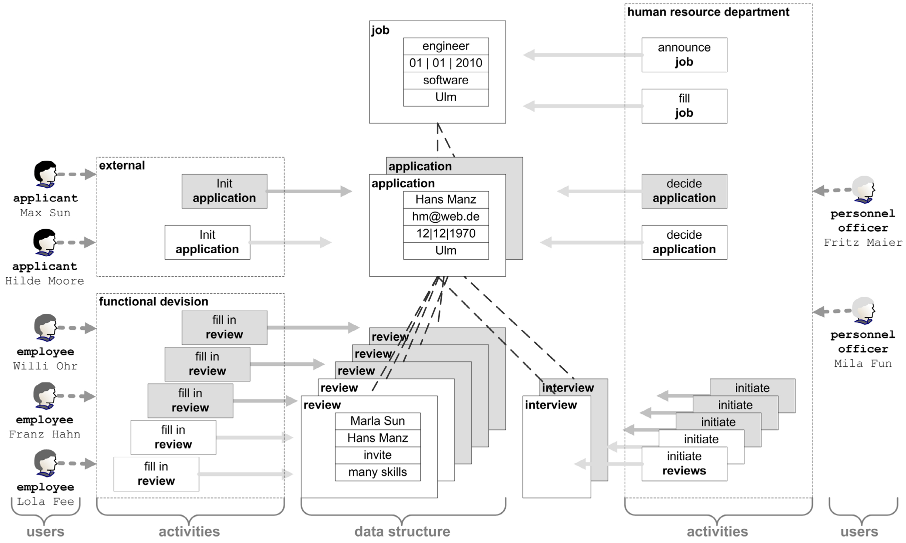


Fig. 1. Example of a recruitment process from the human resource domain

Basically, data must be manageable in terms of *object types* comprising *object attributes* and *relations* to other object types (cf. Fig. 2a). At run-time, the different object types comprise a varying number of inter-related object instances, whereby the concrete instance number should be restrictable by lower and upper cardinality bounds (cf. Fig. 2b). For each application, for example, at least one and at most five reviews must be initiated. While for one application two reviews are are available, another one may comprise three reviews (cf. Fig. 1).

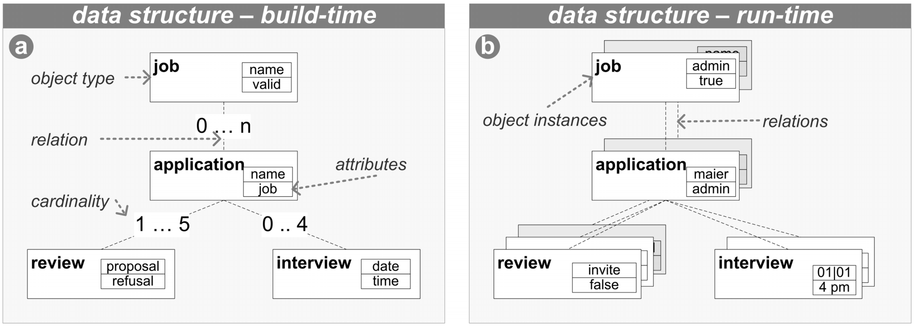


Fig. 2. Data structure at build-time and at run-time

In accordance to data modeling, the modeling and execution of processes can be based on two levels of granularity: *object behavior* and *object interactions*.

2.1 Object Behavior

To cover the processing of individual object instances, the first level of process granularity concerns *object behavior*. More precisely, for each object type a separate process definition should be provided (cf. Fig. 3a), which can be used for coordinating the processing of an individual object instance among different users. In addition, it should be possible to determine in which order and by whom the attributes of a particular object instance have to be (mandatorily) written, and what valid attribute values are. At run-time, the creation of an object instance is directly coupled with the creation of its corresponding process instance. In this context, it is important to ensure that mandatorily required data is provided during process execution. For this reason, object behavior should be defined in terms of *data conditions* rather than based on black-box activities.

Example 2 (Object behavior). *For requesting a* review *the responsible* personnel officer *has to mandatorily provide values for object attributes* return date *and* questionnaire*. Following this, the* employee *being responsible for the* review *has to mandatorily assign a value to object attribute* proposal*.*

2.2 Object Interactions

Since related object instances may be created or deleted at arbitrary points in time, a complex data structure emerges, which dynamically evolves depending on the types and number of created object instances. In addition, individual object instances (of the same type) may be in different processing states at a certain point in time.

Taking the behavior of individual object instances into account, we obtain a *complex process structure* in correspondence to the given data structure

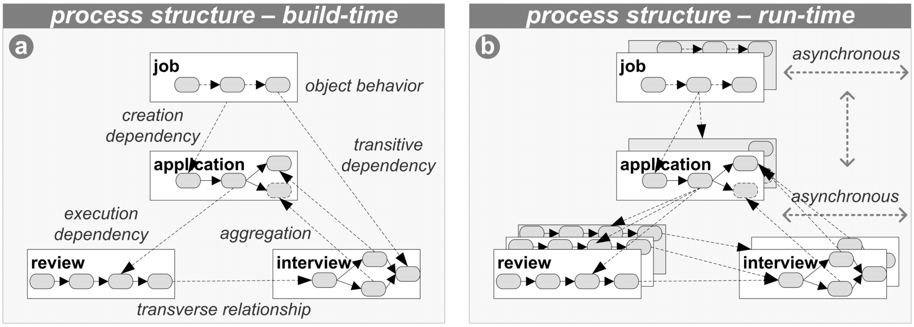


Fig. 3. Process structure at build-time and at run-time

(cf. Fig. 3a). In this context, the second level of process granularity comprises the *interactions* that take place between different object instances. More precisely, it must be possible to execute individual process instances (of which each corresponds to a particular object instance) in a loosely coupled manner, i.e., concurrently to each other and synchronizing their execution where needed. First, it should be possible to make the creation of a particular object instance dependent on the progress of related object instances (*creation dependency*). Second, several object instances of the same object type may be related to one and the same object instance. Hence, it should be possible to aggregate information; amongst others, this requires the aggregation of attribute values from related object instances (*aggregation*). Third, the executions of different process instances may be mutually dependent; i.e., whether or not an object instance can be further processed may depend on the processing progress of other object instances (*execution dependency*). In this context, interactions must also consider *transitive dependencies* (e.g., reviews depend on the respective job offer) as well as *transverse* ones (e.g., the creation of an interview may depend on the proposal made in a review) between object instances (cf. Fig. 3).

Example 3 (Object interactions). *A* `personnel officer` *must not initiate any* `review` *as long as the corresponding* `application` *has not been finally submitted by the* `applicant` *(creation dependency). Further, individual* `review` *process instances are executed concurrently to each other as well as to the* `application` *process instances; e.g., the* `personnel officer` *may read and change the* `application` *while the* `reviews` *are processed. Further,* `reviews` *belonging to a particular* `application` *can be initiated and submitted at different points in time. Besides this, a* `personnel officer` *should be able to access information about submitted* `reviews` *(aggregative information); i.e., if an* `employee` *submits her* `review` *recommending to invite the* `applicant` *for an* `interview`*, the* `personnel officer` *needs this information immediately. Opposed to this, when proposing rejection of the* `applicant`*, the* `personnel officer` *should only be informed after all initiated* `reviews` *have been submitted. Finally, if the* `personnel officer` *decides to hire one of the* `applicants`*, all others must be rejected (execution dependency). These* *dependencies do not necessarily coincide with the object relations. As example consider reviews and interviews corresponding to the same application; i.e., an interview may only be conducted if an employee proposes to invite the applicant during the execution of a review process instance.*

2.3 Data-Driven Execution

In order to proceed with the processing of a particular object instance, usually, in a given state certain *attribute values are mandatorily required.* Thus, object attribute values reflect the progress of the corresponding process instance. In particular, the activation of an activity does not directly depend on the completion of other activities, but on the values set for object attributes. More precisely, *mandatory activities* enforce the setting of certain object attribute values in order to progress with the process. If required data is already available, however, mandatory activities can be *automatically skipped* when being activated. In principle, it should be possible to *set respective attributes also up front*; i.e., before the mandatory activity normally writing this attribute becomes activated. However, users should be allowed to *re-execute a particular activity*, even if all mandatory object attributes have been already set. For this purpose, data-driven execution must be combined with *explicit user commitments* (i.e., activity-centred aspects). Finally, the execution of a mandatory activity may also depend on available attribute values of related object instances. Thus, coordination of process instances must be supported in a data-driven way as well.

Example 4 (Data-driven execution). *During a review request the personnel officer must mandatorily set a return date. If a value for the latter is available, a mandatory activity for filling in the review form is assigned to the responsible employee. Here, in turn, a value for attribute proposal is mandatorily required. However, even if the personnel officer has not completed his review request yet (i.e., no value for attribute return data is available), the employee may optionally edit certain attributes of the review (e.g., the proposal). If a value of attribute proposal is already available when the personnel officer finishes the request, the mandatory activity for providing the review is automatically skipped. Opposed to this, an employee may change his proposal arbitrarily often until he explicitly agrees to submit the review to the personnel officer. Finally, the personnel officer makes his decision (e.g., whether to reject or to accept the applicant) based on the incoming reviews.*

2.4 Variable Activity Granularity

For creating object instances and changing object attribute values, *form-based activities* are required. Respective user forms comprise *input fields* (e.g., textfields or checkboxes) for writing and *data fields* for reading selected attributes of object instances. In this context, however, different users may prefer different work practices. In particular, using *instance-specific activities* (cf. Fig. 4a), all input fields and data fields refer to attributes of one particular object instance, whereas *context-sensitive activities* (cf. Fig. 4b) comprise fields referring to different, but related object instances (of potentially different type). When initiating a review, for example, it is additionally possible to edit the attribute values of the corresponding application. Finally, *batch activities* involve several object instances of the same type (cf. Fig. 4c). Here, the values of the different input fields are assigned to all involved object instances in one go. This enables a personnel officer, for example, to reject a number of application in one go. Depending on their preference, users should be able to freely choose the most suitable activity type for achieving a particular goal. In addition to form-based activities, it must be possible to integrate *black-box activities*. The latter enable complex computations as well as the integration of advanced functionalities (e.g., provided by web services).

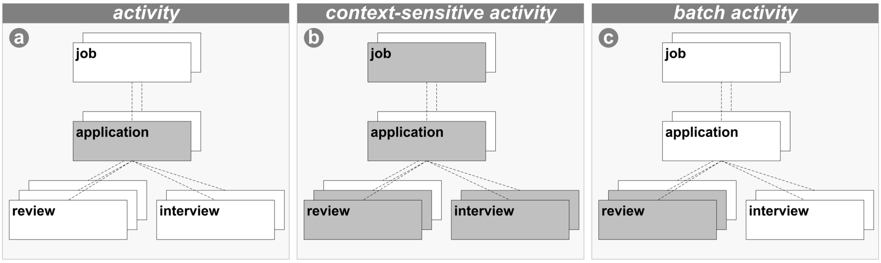


Fig. 4. Different kinds of activities

Moreover, whether certain object attributes are mandatory when processing a particular activity might depend on other object attribute values as well; i.e., when filling a form certain attributes might become mandatory on-the-fly. Such *control flows being specific to a particular form* should be also considered.

Example 5 (Activity Execution). *When an* `employee` *fills in a* `review`*, additional information about the corresponding* `application` *should be provided; i.e., attributes belonging to the* `application` *for which the* `review` *is requested. For filling in the* `review` *form, a value for attribute* `proposal` *has to be assigned. If the* `employee` *proposes to invite the* `applicant`*, additional object attributes will become mandatory; e.g., then he has to set attribute* `appraisal` *as well. This is not required if he assigns value reject to attribute* `proposal`*. Further, when a* `personnel officer` *edits an* `application`*, all corresponding* `reviews` *should be visible. Finally, as soon as an* `applicant` *is hired for a* `job`*, for all other* `applications` *value reject should be assignable to attribute* `decision` *by filling one form.*

2.5 Integrated Access

To proceed with the control flow, *mandatory activities* must be executed by responsible users in order to provide required attribute values. Other attribute values, however, may be optionally set. Moreover, users who are usually not involved in process execution should be allowed to optionally execute selected activities. In addition to a *process-oriented view* (e.g. worklists), a *data-oriented view* should be provided enabling users to access and manage data at any point in time. For this purpose, we need to define permissions for creating and deleting object instances as well as for reading/writing their attributes. However, attribute changes contradicting to specified object behavior should be prevented. Which attributes may be (mandatorily or optionally) written or read by a particular form-based activity not only depends on the user invoking this activity, but also on the progress of the corresponding process instances. While certain users must execute an activity mandatorily in the context of a particular object instance, others might be authorized to optionally execute this activity; i.e., *mandatory and optional permissions* should be distinguishable. Moreover, for object-aware processes, the selection of potential actors should not only depend on the activity itself, but also on the object instances processed by this activity. In this context, it is important to take the relationships between users and object instances into account.

Example 6 (Integrated Access). *A `personnel officer` may only decide on `applications` for which the name of the `applicants` starts with a letter between 'A' and 'L', while another `officer` may decide on `applicants` whose name starts with a letter between 'M' und 'Z'. An `employee` must mandatorily write attribute `proposal` when filling in a `review`. However, her `manager` may optionally set this attribute as well. The mandatory activity for filling the `review` form, in turn, should be only assigned to the `employee`. After submitting her `review`, the `employee` still may change her `comment`. In this context, it must be ensured that the `employee` can only access `reviews` she submitted before. However, attribute `proposal`, in turn, must not be changed anymore. The `personnel officer` might have already performed the proposed action.*

3 Existing Approaches

Many existing approaches have already confirmed the high relevance of a tighter integration of processes and data. In this section, we evaluate selected approaches along the main characteristics of object-awareness: object behavior, object interactions, data-driven process execution, variable activity granularity, and integrated access to data.

3.1 Case Handling

Case Handling [1] is a *data-driven process support paradigm* in which activities are explicitly represented through user *forms* comprising a number of input fields (i.e., text fields, combo boxes, check boxes). The latter refer to *atomic data elements* which are either defined as *mandatory*, *restricted* or *free*. An activity is considered as being completed if all mandatory data elements have an assigned value. Beside defining who shall work on an activity, Case Handling also allows specifying who may redo or skip it; for these uses separate roles exist.

Object Behavior. Unfortunately, Case Handling does not provide explicit support for complex objects and relations between them. However, a "case" can be considered in tight accordance with an "object"; i.e., the data elements can be considered as object attributes. This enables the definition of object behavior specifying in which order and by whom object attributes shall be written. Since it is possible to assign a corresponding value constraint to each input field, valid attribute settings can be enforced. In addition, data constraints can be assigned to transitions connecting individual activities (i.e., user forms). Altogether, processes are defined in terms of data conditions.

Data-Driven Execution. An activity is considered as completed if a value for all mandatory data elements is available. Since different forms may comprise the same data element, it is possible to provide required attribute values up-front; i.e., before they are mandatory for one activity. This way, activities can be automatically skipped if mandatorily required data is already available. Despite the introduction of the redo-role, re-executing activities may be often not possible. In particular, if all data elements being mandatory for the current and for the subsequent activity are available, both activities are automatically marked as executed. In this case, a user may only re-execute the activity if he additionally owns the redo-role for both the current and the subsequent activity. In particular, it is not supported that users re-execute a certain activity as long as they have not explicitly confirmed its completion.

Object Interactions. In Case Handling, it is possible to involve related cases by using *sub-plans*. The latter must be instantiated at specific points during the execution of the higher-level case (i.e., creation dependency). Further, these sub-plans can be categorized as dynamic. This enables the instantiation of a variable number of instances at run-time. In this context, cardinality constraints are considered by defining a minimal and maximal number of instances which should be completed at run-time. However, it is not possible to execute sub-plans asynchronously to the higher-level one. Thus, execution dependencies cannot be defined. Aggregations, in turn, are possible. For this purpose, the higher-level case may include an array whose elements are mapped to data elements of the respective sub-plan. Finally, since sub-plans require a strong hierarchical collocation of related cases, the consideration of arbitrary relationships between cases is not supported. It is not possible, for example, to define a dependency between reviews and interviews (cf. Fig. 1); i.e., it is only allowed to initiate an interview if at least one review proposes to invite the applicant.

Variable Activity Granularity. Using Case Handling, all forms must be predefined at build-time. This means, it is not possible to automatically generate user forms based on the current processing state und the user executing the activity. However, when creating forms, it is possible to invoke data elements corresponding to the higher- and lower-level plans as well. Thus, in addition to instance-specific forms, context-specific ones can be provided. Batch activities, in turn, are not support. Context-specific activities contain input fields corresponding to selected object instances. In this context, it is a cumbersome and desperate task to provide all conceivable granularities of forms.

Integrated Access. In principle, each involved user may execute the currently activated activity. In this context, he can read all data elements and additionally write all data elements which are not categorized as mandatory or restricted. However, it is not possible to assign different permissions to optionally read and write data elements for different user (roles). Moreover, it is not possible to make these permissions dependable on the process state. Opposed to free data elements, mandatory and restricted ones can only be written by users owning the execute-role. This prevents changes of data element values contradicting process execution. However, it is not possible to define one and the same activity as optional for one user while being mandatory for another one.

3.2 Proclets

A proclet [2,14] is an *object-specific process* which is modeled using a Petri net. The latter consists of nodes which comprise places (e.g. circles) and transitions (e.g., rectangles). Places and transitions are connected with arcs. In this context, it is only allowed to connect different node types; i.e., places with transitions or transitions with places. Transitions represent activities. Places, in turn, have assigned tokens representing the current state of the Petri net. In particular, a transition (i.e., an activity) can be activated if each input place contains at least one token. During its activation, one token from each input place is removed and to each output place, in turn, one token is assigned. The proclet framework enables the communication between different proclets based on *messages*. The latter are exchanged through *ports* which are connected by transitions. Each sent or received message is stored in the *knowledge base* of the respective proclet. To discover related proclets for communicating with them, a naming service is called which manages all existing proclets based on their unique proclet ID. Activities have assigned pre- and post-conditions using information from the knowledge base. At run-time, an activity becomes enabled if its corresponding transition is activated (i.e., each input place contains at least one token), the pre-condition evaluates to true, and all input ports contain a message.

Object Behavior. Although proclets are object-specific processes, they do not support the definition of object behavior as required for the support of object-aware processes. In particular, these processes are defined in an activity-centred way and not in terms of data conditions. This way, it is neither possible to determine the order in which object attributes should be written nor to define what valid attribute settings are.

Data-Driven Execution. Since the proclet framework is based on an activity-centered paradigm, data-driven activation of activities is not possible. However, each activity can be associated with a *pre- and post-condition* defined on basis of the information from the knowledge base (i.e., exchanged messages). At run-time, an activity becomes enabled if its incoming transition is activated, the pre-condition evaluates to true, and required messages are available. This way, data-driven coordination of proclet instances becomes possible.

Object Interactions. At run-time, for each proclet type a dynamic number of instances can be handled. In addition, it is possible to send messages to multiple proclets; e.g., to all instances of a certain proclet type. This way, a complex process structure evolves in which the individual proclet instances can be executed asynchronously to each other. In this context, for each port a cardinality constraint can be defined which determines the number of recipients. Creation and execution dependencies as well as aggregations can be defined based on pre-conditions for activities. In this context, however, it is not possible to handle transitive or transverse relationships between proclet instances. This means, it is not possible to synchronize a job offer instance with the set of corresponding reviews directly. Such dependencies must be specified using (several) intermediate dependencies. More precisely, reviews must be synchronized with their corresponding applications and applications with the job offer to which they refer. Otherwise, it is not possible to ensure that only the reviews required for the applications belonging to the respective job offer are considered; i.e., reviews for applications referring to other job offers are then invoked as well.

Variable Activitiy Granularity. All activities are defined in the context of one proclet type: This way, instance-specific activities are supported while context-specific ones are not explicitly considered (i.e., it is not possible to access data elements from lower- or higher-level proclet instances). Regarding the latter, information from other proclet instances can be transferred using messages. However, there is no interrelationship between activity execution and the content of messages. Batch-oriented activities, in turn, are partially supported by enabling multiple instantiation of related proclet instances. The execution of a set of instance-specific activities (which are represent as transitions of individual proclet instances) in one go, however, is not possible. Since activities are treated as black-boxes, internal process logic is not supported.

Integrated Access. Integrated access to application data is taken into account.

3.3 Business Artifacts

The business artifacts framework [15,3] is a process design methodology which focuses on data objects (i.e., business artifatcs) rather than on activities. A business artifact holds all business information relevant about itself. This includes *atomic and structured attributes* as well as all *related artifacts*. In addition, a *life-cycle* is defined for each business artifact which is specified using a *finite-state* *machine* capturing the main *processing stages* and the *transitions* between them. Transitions can be associated with *conditions*. The latter can be defined in terms of attribute values or relationships to other business artifacts. Attribute values are assigned during the execution of services. In particular, *services* are executed to move business artifacts through their lifecycles. Services as associated with *pre- and post-conditions* for their execution (i.e., available attribute values, stages). In addition, *associations* specify how services are associated with artifacts. They are defined using ECA-rules (i.e., event, condition, action). *Events* may comprise currently assigned attribute values, a reached state, a launched or completed service, an incoming message, or a performer request. Conditions, in turn, are defined using first-order logic (e.g., SQL statements). Finally, *actions* represent service invocations or the activation of a subsequent artifact stage. Artifacts (including their informational structure and lifecycles), services and ECA-rules only constitute a logical representation of business processes and business data. In particular, there exists *no well-defined operational semantics* for the direct execution of the defined models. Instead, the definitions are mapped to an activity-centred flow diagram (i.e. for optimization) and are then (manually) implemented. This results in hard-coded process logic.

Object Behavior. Since transitions are associated with (data-) conditions, it is possible to synchronize process state and data state. However, it is a cumbersome task to ensure that pre- and post-conditions assigned to services are consistent with ECA-rules and the conditions defined for the transitions between the stages an artifact moves through.

Object Interactions. Within the data structure of an artifact, related artifacts are defined as well. Their corresponding lifecycles, however, are treated independently (i.e., within their own artifact model). Consequently, the arising data structure is distributed among several data models and overlapping; i.e., some artifacts are defined in the context of several other artifacts. This makes it a hard job to comprehend the emerging process structure and to keep it consistent. In addition to object behavior support, ECA-rules can be used for coordinating artifact lifecycles as well (i.e., by using quantifiers). Consequently, there is no clear separation between object behavior and object interactions. Moreover, services may change attribute values of related business artifacts. In this context, it is a cumbersome task to take care of the lifecycles of related artifacts. Finally, aggregations are not taken into account and transitive and transverse relationships between business artifacts are not considered.

Data-Driven Execution. It is possible to activate a service based on data conditions. However, since the invocation of a service depends on pre-conditions, it is not possible to dynamically skip services if their post-condition has already been fulfilled. Moreover, if certain pre-conditions cannot be met during run-time, process execution will be blocked.

Variable Activitiy Granularity. Each service requires its own implementation. For that reason, the granularity of each service is fixed at build-time; i.e. form-based activities are not explicitly supported. In addition, control flows specific to a particular form (i.e., some attributes values become mandatory on-the-fly) are not considered.

Integrated Access. Optional activities can be enforced by using ECA-rules as well; e.g., without specifying an event or a condition. However, the framework focuses on process-relevant activities. For this reason, it is not possible for users to distinguish optional and mandatory activities in their worklist. For process authorization (i.e,, service execution) users are directly assigned to the service they should execute. In this context, it is neither possible to consider the business artifact properties nor the relation of a user to the respective artifacts. Data authorization (i.e., permissions for creating, deleting and changing of business artifacts), in turn, is not explicitly considered. However, services are annotated with conditional effects (i.e., for ensuring specific attribute values to be set). This way, mandatorily required attribute values can be distinguished from optional ones. However, it depends on the respective service implementation how these assumptions are ensured at run-time. Attribute values violating a constraint are marked as invalidated. These attribute values should then be changed in a subsequent service invocation.

3.4 Data-Driven Coordination

The data-driven coordination framework [4,16] enables the coordination of individual processes based on *objects* and *object relations*. Objects are defined in terms of *states* and *(internal) transitions* between them. The latter are assigned with processes which must be executed to reach the subsequent state. According to the relations between objects, so called *external transitions* connect states belonging to different objects with each other. This enables the coordination of the individual object lifecycles. More precisely, whether or not a certain state of an object can be activated may depend on the currently activated states of other objects. In addition to correctness constraints (i.e., for ensuring the proper termination of the arising process structure), the approach also comprises well-defined operational semantics for the automatic enactment of object lifecycles and process structures.

Object Behavior. Although the behavior of objects is explicitly defined, object attributes and their respective values are not taken into account. For this reason, it is not possible to determine mandatorily required data and to define what valid attribute settings are. Consequently, processes are further defined in terms of black-box activities rather than based on data conditions. Thus, it is not possible to ensure that the actual processing state is in accordance with corresponding attribute values.

Object Interactions. It is possible to asynchronously execute individual object-related processes and to synchronize them. In particular, creation as well as execution dependencies can be defined by using external transitions. The latter, however, can only be specified along relations between objects which are directly represented within the corresponding data structure. Transitive or transverse relations, in turn, are not supported. Although it is possible to create a variable number of instances at run-time, aggregations are not supported.

Data-Driven Execution. Processes themselves are still activity-driven; i.e., the activation of a subsequent state depends on the completion of a process associated with the incoming state transition. Thus, it is not possible to execute processes or respective activities up front or to dynamically skip them. Opposed to this, process synchronization follows a data-driven approach.

Finally, since object attributes are out of scope and process execution is based on black-box activities, neither a **variable granularity of activities** nor **integrated access** to application data is provided.

3.5 Product-Based Workflow Support

The product-based workflow design or support approach [5,17,6] uses a so-called *product data model* which specifies required data elements to assemble a particular product. This product data model is described by a tree [5] and consists of *atomic data elements* and *operations*. Data elements are depicted as circles. Operations, in turn, are represented by arcs with small cycles having zero or more input data elements and exactly one output data element. An operation is executable if values for all input data elements are available. In addition, conditions on the value of the input data elements may restrict the execution of the operation. If the condition is not satisfied, the operation is not executable, even if values for all of its input data elements are available. The product is considered as being fully processed as soon as a value for the top element of the product data model (i.e., the root element) is available. Several operations can have the same output element while having a different set of input elements. Such a situation represents alternative ways to produce a value for that output element. For selecting the best execution alternative, operations are associated with different attributes describing characteristics of the operations like execution cost, processing time, and failure probability. This enables the consideration of different execution priorities during process enactment. To achieve a corresponding process model, the product data model can be manually translated [5]. Since this is time-consuming and error-prone, different algorithms are provided to automatically generate a process model on the basis of a given product data model [17]. Opposed to the consideration of an explicit process model, however, the approach also provides the direct execution of the product data model [17,6]. For this purpose, end users are supported with recommendations about which operation should be executed next. Such recommendations take care of the required performance criterion and calculate how the various alternative executions differ from each other with respect to that performance criterion. This way, the need to translate a product data model into a process model disappears.

Object Behavior. Using atomic data elements, it is possible to specify which data is mandatorily required during process execution. In addition, it is possible to determine the order in which data elements have to be written, and to specify what valid attribute settings are.

Object Interactions. Since each instance is executed in isolation, it is not possible to coordinate them.

Integrated Access. Access to data is only possible during the execution of operations specified within the product data model; i.e., users are not allowed to access and manage data at any point in time. Different ways for reaching a process goal are definable using alternative execution paths. If one path has been selected, however, it is no longer possible to additionally execute the other one. Thus, it is not possible to differentiate between optional and mandatory operations.

Data-Driven Execution. The direct execution of the product data model enables data-driven process execution. In particular, a subsequent operation can be executed if for all required input data elements a corresponding value is available and the eventually associated data condition evaluates to true. Since activity activation depends on pre-conditions, however, it is not possible to automatically skip an activity if the required output data element has been already available. In addition, re-execution of activities is not possible.

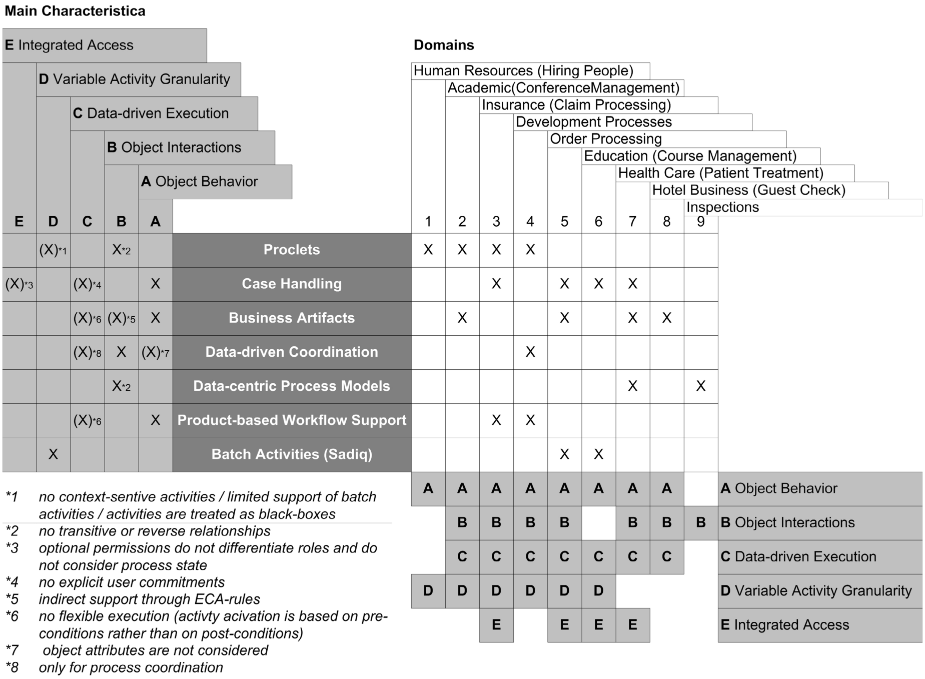
Variable Activity Granularity. Each operation requires a specific implementation and therefore constitutes a black-box activity. As a consequence, all operations have a fixed granularity and control-flow support during the execution of activities is not provided.

3.6 Further Approaches

Similar to the Proclets [2] framework, the Object-centric Business Process Modeling framework [7,8] enables the coordination of object-specific processes along their corresponding object relations. However, processes are defined in an activity-centred way; i.e., in terms of black-box activities. For coordination, however, cardinality constraints as well as creation and execution dependencies are taken into account. Finally, [18] proposes to group and ungroup related activities within user worklists. This way, batch activities can be supported.

4 Comparing the Different Approaches

As illustrated in Fig. 5, each characteristic is addressed by at least one existing approach. Although the mentioned approaches have limitations (see footnotes in Fig. 5), they can be considered as pioneer work towards object-aware process support. However, none of them covers all characteristics in a comprehensive and integrated way. Also note that Fig. 5 does not make a difference between process modeling and process execution. Though some approaches (e.g., the business artifacts framework [3]) provide rich capabilities for process modeling, they do not cover run-time issues (or at least do not treat them explicitly).



Main Characteristica

E Integrated Access
D Variable Activity Granularity
C Data-driven Execution
B Object Interactions
A Object Behavior

Domains

1 Human Resources (Hiring People)
2 Academic(ConferenceManagement)
3 Insurance (Claim Processing)
4 Development Processes
5 Order Processing
6 Education (Course Management)
7 Health Care (Patient Treatment)
8 Hotel Business (Guest Check)
9 Inspections

E	D	C	B	A		1	2	3	4	5	6	7	8	9	
	(X)*1		X*2		**Proclets**	X	X	X	X						
(X)*3		(X)*4		X	**Case Handling**			X		X	X	X			
		(X)*6	(X)*5	X	**Business Artifacts**		X			X		X	X		
		(X)*8	X	(X)*7	**Data-driven Coordination**				X						
			X*2		**Data-centric Process Models**							X		X	
		(X)*6		X	**Product-based Workflow Support**			X	X						
	X				**Batch Activities (Sadiq)**					X	X				
						A	**A**	**A**	**A**	**A**	**A**	**A**	**A**		**A** Object Behavior
							B	**B**	**B**	**B**		**B**	**B**	**B**	**B** Object Interactions
							C	**C**	**C**	**C**	**C**	**C**	**C**		**C** Data-driven Execution
						D	**D**	**D**	**D**	**D**	**D**				**D** Variable Activity Granularity
								E		**E**	**E**	**E**			**E** Integrated Access

**1 no context-sentive activities / limited support of batch activities / activities are treated as black-boxes*
**2 no transitive or reverse relationships*
**3 optional permissions do not differentiate roles and do not consider process state*
**4 no explicit user commitments*
**5 indirect support through ECA-rules*
**6 no flexible execution (activty acivation is based on pre-conditions rather than on post-conditions)*
**7 object attributes are not considered*
**8 only for process coordination*

Fig. 5. Evaluation of existing approaches

In order to emphasize the high practical impact of object-aware process support, we contrast the characteristics with the different application examples considered by existing approaches (cf. Fig. 5). In particular, existing approaches partially consider similar scenarios, while addressing different characteristics (cf. the grey boxes on the bottom of Fig. 5). For example, order processing was taken as illustrating scenario by Case Handling [1], Batch Activities [18], and Business Artifacts [3]. Case Handling addresses the need for enabling object behavior, data-driven execution, and integrated access. Business Artifacts, in turn, consider data-driven execution, object behavior and object interactions. Finally, [18] describes the need for executing several activities in one go (i.e., the execution of batch-activities). Consequently, this indicates that integrated support of all these characteristics is urgently needed to adequately cope with *order processes.* Altogether, this comparison demonstrates two things: First, the characteristics are related to each other. Second, broad support for them is required by a variety of processes from different application domains.

5 An Integrated Framework for Object-Aware Processes

Altogether, we believe that object-aware process management will provide an important contribution towards the realization of flexible process management technology in which daily work can be done in a more natural way.

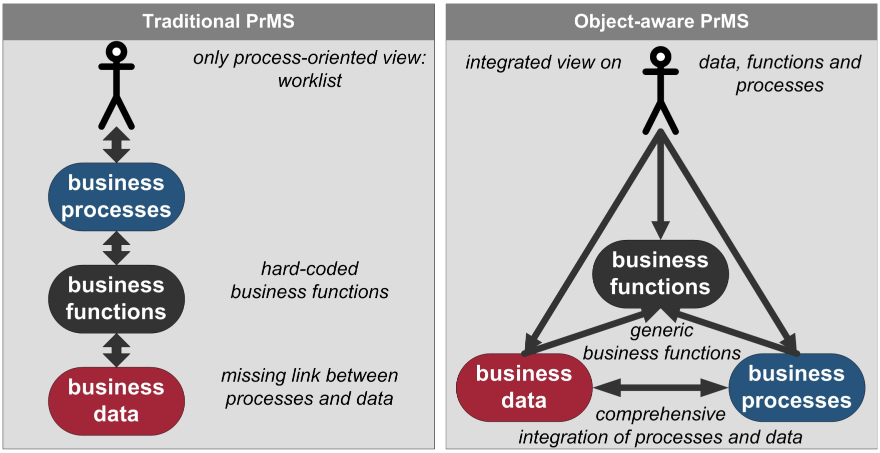

Traditional PrMS
only process-oriented view: worklist
business processes
business functions
hard-coded business functions
business data
missing link between processes and data
Object-aware PrMS
integrated view on
data, functions and processes
business functions
generic business functions
business data
business processes
comprehensive integration of processes and data

Fig. 6. Object-aware Process Management

In particular, as illustrated in Fig. 6, a comprehensive integration of processes and data entails three major benefits:

1. Flexible execution of unstructured, knowledge-intensive processes.
2. Integrated view on processes, data, and functions to users.
3. Generic business functions: automatically generated form-based activities.

In the PHILharmonicFlows project we target at a *comprehensive framework for object-aware process management* enabling the introduced characteristics (cf. Fig. 7). PHILharmonicFlows enforces a *well-defined modeling methodology* governing the definition of processes at different levels of granularity and being based on a *well-defined formal semantics*. Due to lack of space, this paper focuses on existing approaches already addressing particular characteristics of object-aware process support. Hence, we omit details about the different modeling components and formal issues (e.g., ensuring soundness). The same applies in respect to the formal semantics process execution is based on (see [11,19] for details).

The framework differentiates between *micro and macro processes* in order to capture both *object behavior* and *object interactions*. As a prerequisite, object types and their relations need to be captured in a data model. For each object type a corresponding *micro process type* needs to be defined. In this context, our approach applies the well established concept of modeling object behavior in terms of states and state transitions [3,4]. Opposed to existing approaches, however, PHILharmonicFlows enables a *mapping between attribute values and objects* *states* and therefore ensures compliance between them. Finally, the presented execution paradigm combines *data-driven process execution* with *activity-oriented aspects*. Optional access to data, in turn, is enabled asynchronously to process execution and is based on permissions for creating and deleting object instances as well as for reading/writing their attributes. For this, PHILharmonicFlows maintains a *comprehensive authorization table* taking the current progress of the corresponding micro process instance into account. In accordance to the relations between the invoked object instances, the corresponding micro process instances additionally form a *complex process structure*. By using *macro processes*, however, we *hide this complexity* from modelers as well as from end-users to a large degree.

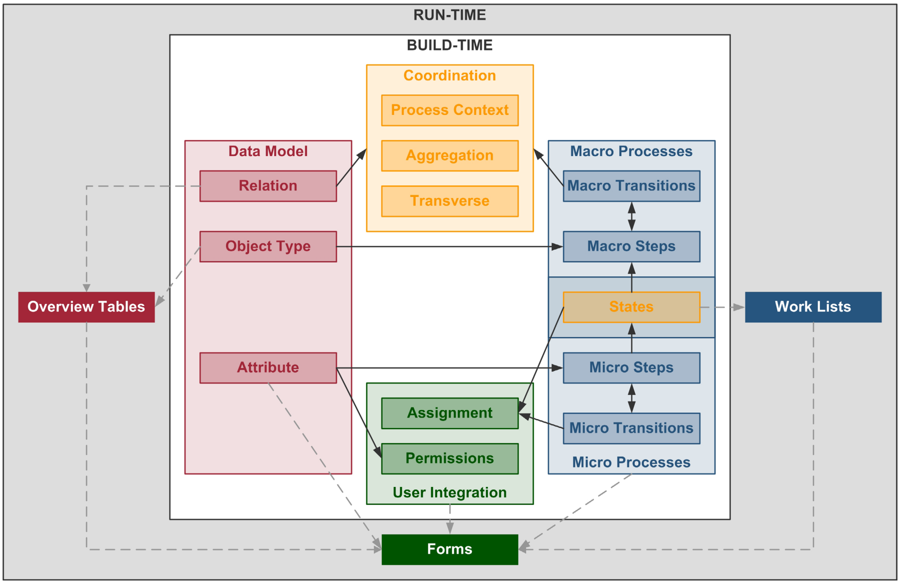


Fig. 7. Overview about the PHILharmonicFlows Framework

6 Summary and Outlook

In this paper we made several contributions. First, we systematically introduced the main characteristics of object-aware processes. Second, we elaborated pioneering work recognizing the need for a tighter integration of processes and data along these characteristics. Overall, the conducted evaluation has confirmed the high relevance of the characteristics and that their support is needed in many application domains. However, as discussed, a comprehensive framework for object-aware process management is still missing. PHILharmonicFlows offers a comprehensive solution framework to adequatlely support object-aware processes and to automatically generate most end-user components at run-time. In [11] we have already introduced the basic components of PHILharmonicFlows as well as their complex interdependencies. In addition, details on micro process support and the automatic generation of form-based activities can be found in [19]. Finally, a tighter integration of processes and data implicates further challenges in respect to the integration of users [10]. These issues are considered in PHILharmonicFlows as well. To evaluate our framework, we are currently developing a proof-of-concept prototype for the modeling as well as run-time support of object-aware processes. In this context, additional techinques enabling improved system performance and scalability are introduced.

References

1. van der Aalst, W.M.P., Weske, M., Grünbauer, D.: Case Handling: A new Paradigm for Business Process Support. DKE 53(2), 129–162 (2005)
2. van der Aalst, W.M.P., Barthelmess, P., Ellis, C.A., Wainer, J.: Workflow Modeling using Proclets. In: Scheuermann, P., Etzion, O. (eds.) CoopIS 2000. LNCS, vol. 1901, pp. 198–209. Springer, Heidelberg (2000)
3. Bhattacharya, K., Hull, R., Su, J.: In: A Data-Centric Design Methodology for Business Processes, pp. 503–531. IGI Global (2009)
4. Müller, D., Reichert, M., Herbst, J.: Data-Driven Modeling and Coordination of Large Process Structures. In: Meersman, R., Tari, Z. (eds.) OTM 2007, Part I. LNCS, vol. 4803, pp. 131–149. Springer, Heidelberg (2007)
5. Reijers, H.A., Liman, S., van der Aalst, W.M.P.: Product-Based Workflow Design. Management Information Systems 20(1), 229–262 (2003)
6. Vanderfeesten, I., Reijers, H.A., van der Aalst, W.M.P.: Product-based Workflow Support. Information Systems 36(2), 517–535 (2011)
7. Redding, G.M., Dumas, M., ter Hofstede, A.H.M.: Transforming Object-oriented Models to Process-oriented Models. In: ter Hofstede, A.H.M., Benatallah, B., Paik, H.-Y. (eds.) BPM Workshops 2007. LNCS, vol. 4928, pp. 132–143. Springer, Heidelberg (2008)
8. Redding, G.M., Dumas, M., ter Hofstede, A.H.M., Iordachescu, A.: A flexible, object-centric approach for business process modelling. In: Service Oriented Computing and Applications, pp. 1–11 (2009)
9. Künzle, V., Reichert, M.: Towards Object-Aware Process Management Systems: Issues, Challenges, Benefits. In: Halpin, T., Krogstie, J., Nurcan, S., Proper, E., Schmidt, R., Soffer, P., Ukor, R. (eds.) BPMDS 2009. LNBIP, vol. 29, pp. 197–210. Springer, Heidelberg (2009)
10. Künzle, V., Reichert, M.: Integrating Users in Object-Aware Process Management Systems: Issues and Challenges. In: Rinderle-Ma, S., Sadiq, S., Leymann, F. (eds.) BPM 2009. LNBIP, vol. 43, pp. 29–41. Springer, Heidelberg (2010)
11. Künzle, V., Reichert, M.: PHILharmonicFlows: Towards a Framework for Object-aware Process Management. Journal of Software Maintenance and Evolution: Research and Practice 23(4), 205–244 (2011)
12. Künzle, V., Weber, B., Reichert, M.: Object-aware Business Processes: Fundamental Requirements and their Support in Existing Approaches. International Journal of Information System Modeling and Design (IJISMD) 2(2), 19–46 (2011)
13. Künzle, V., Reichert, M.: PHILharmonicFlows: Research and Design Methodology. Technical Report UIB-2011-05. University of Ulm, Ulm, Germany (May 2011)

14. van der Aalst, W.M.P., Mans, R.S., Russell, N.C.: Workflow Support Using Proclets: Divide, Interact and Conquer. Bulletin of the IEEE Computer Society Technical Committee on Data Engineering 32(3), 16–22 (2009)
15. Liu, R., Bhattacharya, K., Wu, F.Y.: Modeling Business Contexture and Behavior Using Business Artifacts. In: Krogstie, J., Opdahl, A.L., Sindre, G. (eds.) CAiSE 2007 and WES 2007. LNCS, vol. 4495, pp. 324–339. Springer, Heidelberg (2007)
16. Müller, D., Reichert, M., Herbst, J.: A New Paradigm for the Enactment and Dynamic Adaptation of Data-Driven Process Structures. In: Bellahsène, Z., Léonard, M. (eds.) CAiSE 2008. LNCS, vol. 5074, pp. 48–63. Springer, Heidelberg (2008)
17. Vanderfeesten, I.: Product-Based Design and Support of Workflow Processes. Phd thesis. Eindhoven University of Technology (2009)
18. Sadiq, S.W., Orlowska, M.E., Sadiq, W., Schulz, K.: When workflows will not deliver: The case of contradicting work practice. In: Proc. BIS 2005 (2005)
19. Künzle, V., Reichert, M.: A Modeling Paradigm for Integrating Processes and Data at the Micro Level. In: Halpin, T., Nurcan, S., Krogstie, J., Soffer, P., Proper, E., Schmidt, R., Bider, I. (eds.) BPMDS 2011 and EMMSAD 2011. LNBIP, vol. 81, pp. 201–215. Springer, Heidelberg (2011)

Scrum Conceptualization Using K-CRIO Ontology

Yishuai Lin, Vincent Hilaire, Nicolas Gaud, and Abderrafiaa Koukam

Laboratoire Systèmes et Transports (SeT)
Université de Technologie de Belfort-Montbéliard (UTBM)
90010 Belfort Cedex, France
vincent.hilaire@utbm.fr
www.multiagent.fr

Abstract. Many enterprises are on the path towards the automation of their business processes. To be fully efficient this kind of automation needs to rely on rich models of business processes. This paper presents an ontology, named K-CRIO, that allows the description of a specific kind of business processes: those that are dedicated to the design of a product. This ontology draws from organizational theories. The contribution of this paper is twofold. On the one hand, it described the key concepts and relationships of the k-CRIO ontology and on the other hand it illustrates this ontology by taking the example of the Scrum development process.

1 Introduction

Many enterprises are on the path towards the automation of their business processes. To be fully efficient this kind of automation needs to rely on rich models of business processes. This paper presents an ontological approach that allows the description of a specific kind of business processes: those that are dedicated to the design of a product. This ontology draws from organizational theories.

As stated in [3], an ontology of organizations "is the first, fundamental and ineliminable pillar on which to build a precise and rigorous enterprise modeling". Obviously, the field of ontologies for organizations is very broad, among the works in this domain one can cite [3,8]. As the term organization covers a wide range of entities we have chosen to reduce our field of investigation. In our case the targeted organizations are those composed of human actors involved in the design of a product. These human actors follows a previously defined design process to frame their design activities. Even if this statement reduce our filed of investigation it is general enough to cover the description of many enterprises.

This ontology is the first building block of a software environment, based on muti-agent systems, that will support enterprises engaged in products design during their day-to-day works. Example of such support can be file sharing, enhanced communications, wikis, etc. The goal of this ontology is used to support knowledge management within the described organizations. More specifically, the ontology will provide means for annotating resources, reasoning, monitoring design processes, enabling searches and to proactively propose tips and proper

K. Aberer, E. Damiani, and T. Dillon (Eds.): SIMPDA 2011, LNBIP 116, pp. 189–211, 2012.

contents. These resources and contents are those handled by the multi-agent system.

The presented approach is based upon the use of an existing organizational metamodel, namely CRIO [5], already used for the description of Multi-Agent Systems (MAS) organizations. In our case, the concepts of this metamodel are used to model human activities. The concepts and relationships of this metamodel are represented by the K-CRIO ontology.

The chosen ontology language is OWL [12] as it is widely used and allows the representation of the underlying concepts such as human interactions by using the OWL-WS sub-ontology [2].

This paper is organized as follows: Section 2 introduces the background of our work. Section 3 details the application of K-CRIO to the SCRUM methodology. Section 4 reviews and discusses related works. Eventually, Section 5 concludes and describes some future work directions.

2 Background

2.1 K-CRIO

In the following, we will try to single out which are the main concepts of an ontology that represent organizations. The K-CRIO ontology is presented in Figure 4.

The targeted organizations are those dedicated to product design. We have then defined a concept named DesignObject (*owl:class*) which is the root of all possible products that an organization can produce.

An organization can be seen as a set of interacting entities: sub-organizations or roles, which are regulated by social rules and norms.

- With respect to the ontology, an organization may be seen as a concept connected to other concepts by various kinds relationships, such as *hierarchical* relations between organizations and sub-organizations, or *composed of* relation between an organization and its roles.
- With respect to the human processes in enterprises, an organization may be considered as a collective global system able to achieve particular goals through its collaborative members.

Using OWL, the concept of organization may be specified as following: Organization is an *owl:class* which may be linked to sub-organizations with "isSubOrganizationOf" (*owl:ObjectProperty*)and "includes" (*owl:ObjectProperty*) a collection of Roles (an *owl:class*), with "provided" (*owl:ObjectProperty*) Capacity (as well as an *owl:class*) and "hascontext" (*owl:ObjectProperty*) Ontology (an *owl:class*). The latter class defines a specific context for the concerned organization. Additionally, organizations are linked by "isThePlaceOf" (*owl:ObjectProperty*) to Interaction (*owl:class*) which conceptualize interactions occuring in the organization context. It may be expressed as detailed in Figure 1.

In the rest of this paper, we denote these definition with logic language:

Organization(X) means that X is an Organization.

DesignObject(X) means that X is a DesignObject.

Role(X) means that X is a Role.

Capacity(X) means that X is a Capacity.

Ontology(X) means that X is an Ontology.

Interaction (X) means that X is an Interaction.

isSubOrganizationOf (X,Y) means that X is a sub-Organization of Y and {*X: Organization(X), Y:Organization(Y)*}.

includes (X,Y) means that X includes Y and {*X: Organization(X), Y: Role(Y)*}.

provided (X,Y) means that X includes Y and {*X: Organization(X), Y: Capacity(Y)*}.

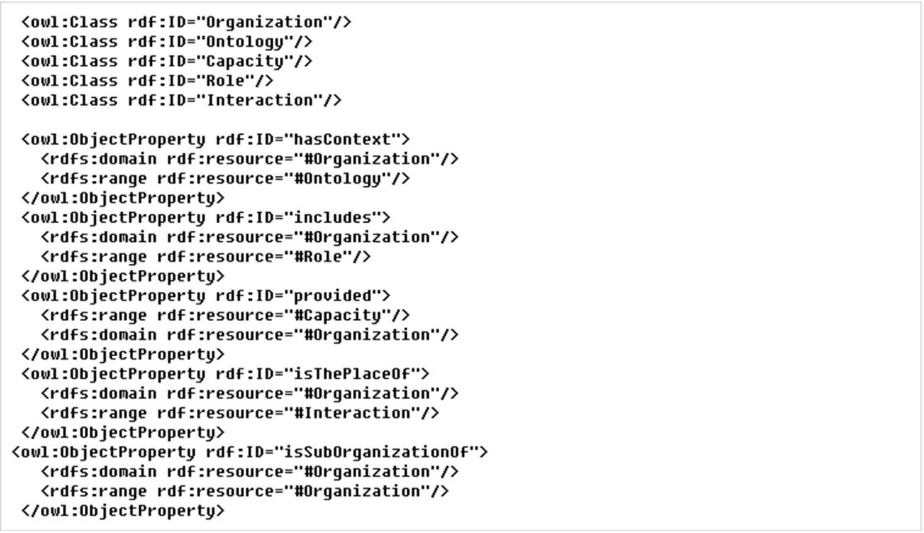
hasContext (X,Y) means that X hasContext Y and {*X: Organization(X), Y: Ontology(Y)*}.

isThePlaceOf (X,Y) means that X is the place of Y happening and {*X: Organization(X), Y: Interaction(Y)*}.

We also define some necessary restrictions between Organization and associated classes, which are represented by qualified cardinality restrictions in OWL (cardinality constraints including *owl:cardinality, owl:minCardinality, owl:maxCardinality* [16]) and axiomatized by logic language.

Specially, one Organization may have an unspecified number of sub-organizations (zero or more). Moreover, the *ObjectProperty: isSubOrganizationOf* is transitive objectProperty. For example, if we consider a university as one Organization, different departments may be seen as its sub-organizations. In the other side, one department is usually composed of diverse majors, which may be seen as sub-organizations of this department. Because of transitivity, these majors are also the sub-Organizations of the university. Expressed by owl:Restriction, the sub-Organization "isSubOrganizationOf" its Organization with minCardinality 0 (*owl:restriction*). The logic expression of transitivity of *ObjectProperty: isSubOrganizationOf* as:

$\mathit{isSubOrganizationOf}\,(O_1, O_2),\ \mathit{isSubOrganizationOf}\,(O_2, O_3) \rightarrow \mathit{isSubOrganizationOf}\,(O_1, O_3)$



```
 <owl:Class rdf:ID="Organization"/>
 <owl:Class rdf:ID="Ontology"/>
 <owl:Class rdf:ID="Capacity"/>
 <owl:Class rdf:ID="Role"/>
 <owl:Class rdf:ID="Interaction"/>

 <owl:ObjectProperty rdf:ID="hasContext">
   <rdfs:domain rdf:resource="#Organization"/>
   <rdfs:range rdf:resource="#Ontology"/>
 </owl:ObjectProperty>
 <owl:ObjectProperty rdf:ID="includes">
   <rdfs:domain rdf:resource="#Organization"/>
   <rdfs:range rdf:resource="#Role"/>
 </owl:ObjectProperty>
 <owl:ObjectProperty rdf:ID="provided">
   <rdfs:range rdf:resource="#Capacity"/>
   <rdfs:domain rdf:resource="#Organization"/>
 </owl:ObjectProperty>
 <owl:ObjectProperty rdf:ID="isThePlaceOf">
   <rdfs:domain rdf:resource="#Organization"/>
   <rdfs:range rdf:resource="#Interaction"/>
 </owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="isSubOrganizationOf">
   <rdfs:domain rdf:resource="#Organization"/>
   <rdfs:range rdf:resource="#Organization"/>
 </owl:ObjectProperty>
```

Fig. 1. Organization in K-CRIO

Moreover, one Organization must include one Role at least (inversely, one Role must be included by one Organization at least) and in which one or more interactions must be occurred. That means, describing with owl:Restriction, Organization "includes" Role with minCardinality 1 (*owl:restriction*) and "isThePlaceOf" interaction happening with minCardinality 1 (*owl:restriction*). The logic axiomatization expression as:

$$\forall\ O\ Organization\ (O) \rightarrow \exists\ R\ \{\ Role\ (R) \bigwedge includes\ (O,R)\ \}$$

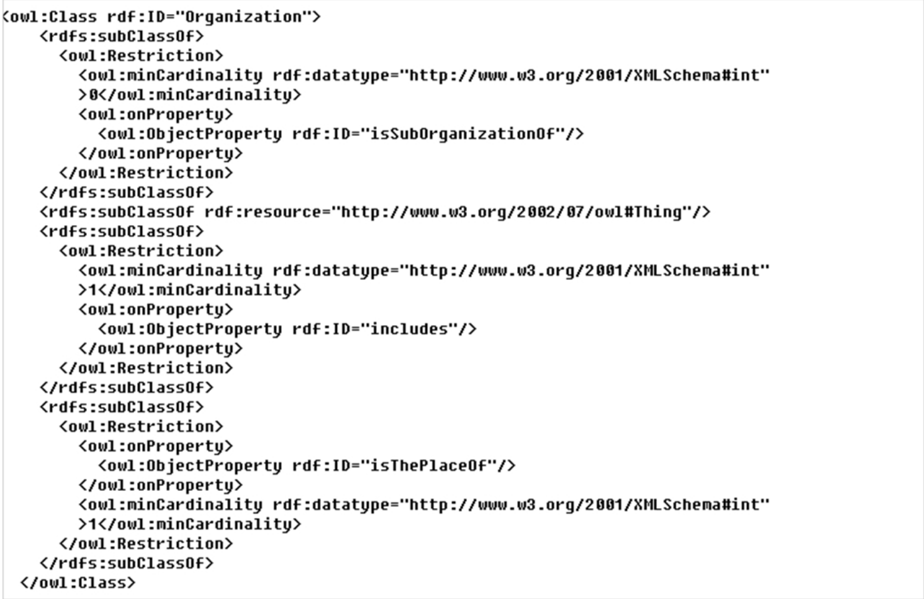
$$\forall\ R\ Role(R) \rightarrow \exists\ O\ \{\ Organization\ (O) \bigwedge includes\ (O,R)\ \}$$

$$\forall\ O\ Organization\ (O) \rightarrow \exists\ I\ \{\ Interaction\ (I) \bigwedge isThePlaceOf\ (O,I)\ \}$$

It is detailed in the Figure 2 that these property restrictions used the xsd namespace declaration made in the header element to refer to the XML Schema document.

A role may identify a person, an activity or a service which is a necessary part to achieve social objectives (goals of its organization). In order to fulfill this common target, each role may have specific individual capacities. An Organization (an *owl:class*) "includes" (*owl:ObjectProperty*) Roles (an *owl:class*) which may "required" (*owl:ObjectProperty*) Capacity (an *owl:class*):

required (X,Y) means X required Y and $\{X\text{: } Role(X),\ Y\text{: } Capacity(Y)\}$.



```
<owl:Class rdf:ID="Organization">
   <rdfs:subClassOf>
     <owl:Restriction>
       <owl:minCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
       >0</owl:minCardinality>
       <owl:onProperty>
         <owl:ObjectProperty rdf:ID="isSubOrganizationOf"/>
       </owl:onProperty>
     </owl:Restriction>
   </rdfs:subClassOf>
   <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
   <rdfs:subClassOf>
     <owl:Restriction>
       <owl:minCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
       >1</owl:minCardinality>
       <owl:onProperty>
         <owl:ObjectProperty rdf:ID="includes"/>
       </owl:onProperty>
     </owl:Restriction>
   </rdfs:subClassOf>
   <rdfs:subClassOf>
     <owl:Restriction>
       <owl:onProperty>
         <owl:ObjectProperty rdf:ID="isThePlaceOf"/>
       </owl:onProperty>
       <owl:minCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
       >1</owl:minCardinality>
     </owl:Restriction>
   </rdfs:subClassOf>
 </owl:Class>
```

Fig. 2. Restrictions of Organization in K-CRIO

Moreover, Interaction (*owl:class*) occurring in Organization "hasParticipants" (*owl:ObjectProperty*) that are Roles.

hasParticipants (X, Y) means the at *X* hasParticipants *Y* happening and {*X: Interaction(X), Y: Role(Y)*}.

A capacity is a know-how and ability, which may be considered as an interface between the role and the role-players. Capacities are required by the role and provided by the organization to define the behaviors of other roles. With the respect of K-CRIO, beside the relationship with Organization and Role described above, Capacity is represented by an *owl:class*. This class is related to some *ContextOntologyElement* (which are parts of the Organization Ontology defining it context) divided in two sets "input" and "output" (both are *owl:ObjectProperty*). The Capacity class is also related to properties which are conceptualized by the Predicate concept. A predicate is an assertion on a property of some *ContextOntologyElement*. There are two types of relationships between Capacity and Predicate. The first type is named "*requires*" and represents the properties that are required by the capacity. The second type is named "*ensures*" and represents the effects of the capacity.

Interactions are mandatory for modeling human processes. The aim of an interaction in this context is to achieve a goal or to contribute to a goal of one organization. Interactions may be divided into *Casual Interaction* and *Formalized Interaction*. The former ones depict interactions between roles which are not defined initially in the model and that can take place in a non determined fashion. Inversely, formalized interactions identify how different roles belonging to the same organization can interact with each other so as to achieve the common goal of the organization and in the meanwhile produce *DesignObjects* of any sort. Interactions may then be seen as a description of possible patterns of actions and exchanges between the participant roles that is very similar to a workflow. In order to conceptualize *FormalizedInteraction*, we have chosen to reuse and improve an existing ontology, OWL-WS, dedicated to the description of workflows [2] and inspired from OWL-S [11]. OWL-WS is based on the assumption that a workflow is a kind of complex service and therefore it can be represented in OWL-WS as a full OWL-S Service. This service can be a simple one or composed of simpler services using the OWL-S control constructs such as: *Sequence*, *RepeatUntil*, *Split*, etc.

A process may be an atomic process which is a description of one (possibly complex) message and returning one (possibly complex) message in response. A process may also be a composite which means that it can be divided into atomic processes describing a behavior (or a set of behaviors) sending and receiving a series of messages. If we consider the formalized interaction as a composite process owning an overall effect, the the entire process must be performed in order to achieve that objective. Moreover, the definition of formalized interaction express the different ways of executing this interaction and correlative results. Let's take the example of a selling service, represented by one organization named selling service including two roles: Client and Bank Service. The formalized interaction details how to perform this trade between these two roles (exchanging messages).

- The initial state is when the Client inputs the Credit Card Number and code,which is a message sent to Bank Service.
- When Bank Service receives a message from the Client role, there should be a control accompanying different situations as
  - Both datum are right, Bank Service returns confirm message to Client;
  - One datum is invalid, Bank Service returns error message to Client, simultaneously, the state of Client returns to original state (waiting for an input of the Credit Card Number and Password and waiting new answer of Bank Service);
- This sequence is repeated until the Client receives the confirm message from Bank Service, he can then send Verify Paying message to Bank Service;
- Bank Service validate the charge following the message Verify Paying from Client.

From the point of view of K-CRIO, *CasualInteraction* and *FormalizedInteraction* are represented by two *owl:class* both subclasses of *Interaction* (an *owl:class*), which are related by "hasParticipants" (*owl:ObjectProperty*) to Roles:

CasualInteraction (X) means X is CasualInteraction.

FormalizedInteraction (X) means X is FormalizedInteraction.

Interaction (X) $\equiv$ *CasualInteraction (X)* $\bigcup$ *FormalizedInteraction (X).*

The *FormalizedInteraction* class is related by the "*produces*" (*owl:ObjectProperty*) relationships to the "*DesignObject*" (an *owl:class*) concept:

produces (X,Y) means *X* produces *Y* and {*X: FormalizedInteraction(X), Y: DesignObject(Y)*}.

Moreover, concerning the state of each formalized interaction, we defined FormalizedInteraction is in (an ObjectProperty) some State (an owl:class), which has three sub-classes: NotStart, Doing, Done. In certain situations, a formalized interaction has its Pre-Interaction (hasPre-Interaction is an objectProperty, the domain and range of which are both FormalizedInteraction):

at (X,Y) means *X* is at *Y* state and {*X: FormalizedInteraction(X), Y: State(Y)*}.

hasPre-Interaction (X,Y) means *X* has pre-interaction *Y* and {*X: FormalizedInteraction(X), Y: FormalizedInteraction(Y)*}.

In order to test whether the interaction is executed following the schedule or not, we defined an owl:class: Time which is related with FormalizedInteraction by the objectProperty: follows. Specially, Time has four sub-classes: BeginningTime, EndTime, RealBeginningTime, RealEndTime, in which, BeginningTime and EndTime expresses the planning schedule and RealBeginningTime and RealEndTime expresses the real schedule. Comparing the subtraction of two types of beginning time and end time, we could express whether the process is earlier or later than the schedule actually:

For *Follows(FormalizedInteraction, BeginningTime)* $\bigwedge$ *(FormalizedInteraction, EndTime)*$\bigwedge$*(FormalizedInteraction, RealBeginningTime)*$\bigwedge$*(FormalizedInteraction, RealEndTime)*:

(EndTime - BeginningTime) > *(RealEndTime - RealBeginningTime)* means the FormalizedInteraction is earlier than the schedule.

(RealEndTime - RealBeginningTime) > *(EndTime - BeginningTime)* means the FormalizedInteraction is later than the schedule.

In summary, the whole K-CRIO Ontology is presented in Figures 3 and 4.

2.2 Introduction to Scrum

Scrum is an iterative, incremental framework for projects and products or application developments and is often seen as an agile software development process. Scrum structures development in cycle of work called Sprints (or iterations). These iterations last no more than one month each, and take place one after

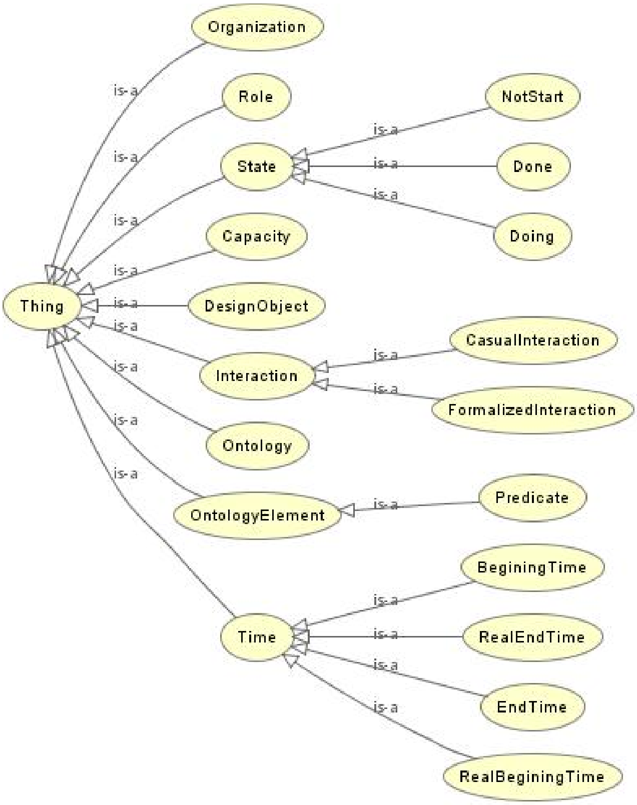


Fig. 3. K-CRIO taxonomy

the other without pause. The people involved in Scrum could be divided into two groups. The first group denotes the Scrum team and has a figurative name : ”Pig”, where are three core roles: The Product Owner, The Scrum Master and The Team. All these roles are committed to the project in the Scrum process, who are the ones producing the product (objective of the project). The other group delegates Ancillary people, relatively, which is named ”Chicken”. Precisely, the ancillary people in Scrum are those with no formal role and infrequent involvements in the Scrum process and must nonetheless be taken into account, such as some Stakeholders, Customers, Vendors or Managers (that represents a kind of people who will set up the environment for product development).

The Scrum process could be presented by the Figure 5 from [7]. In this figure, we could see that during the first step, the Product Owner needs to communicate with all the Stakeholders in order to transform each User Story into one item or one feature in the Product Backlog, which is the basic document containing prioritized descriptions which will be referred by the Scrum Team during the Scrum Process. In a second step, before the beginning of each Sprint, a cross-functional team selects items (customer requirements) from the prioritized list

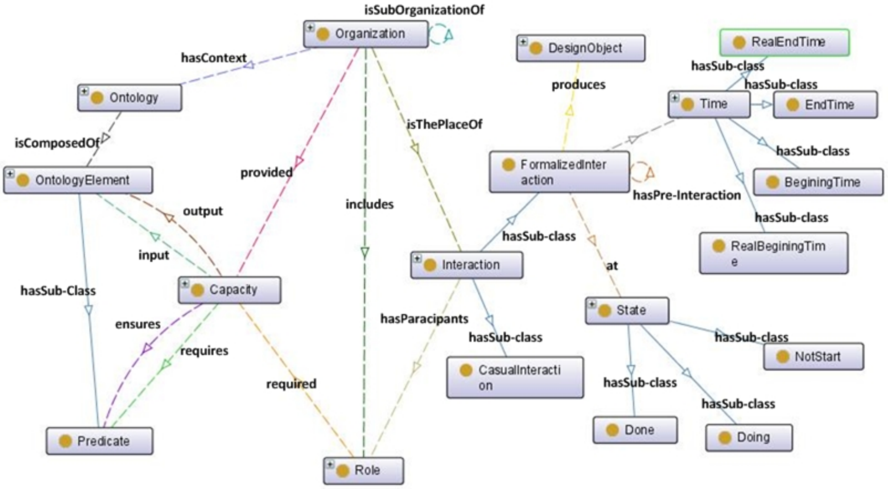


Fig. 4. K-CRIO Ontology

(Product Backlog) [7]. This phase occurs in the Sprint Planning Meeting. The team should commit to complete these items by the end of the Sprint, which may be called task and represented in the Sprint Backlog. More informations about Scrum could be found in [7,17].

3 Scrum Represented by K-CRIO

In this section, we will use the K-CRIO Ontology [10] to describe human roles and interactions during the Scrum process, containing actors, their capacities and how these roles interacts with others in each phase during a Scrum process. Based on general understand of Scrum above, we could see that the whole Scrum contains diverse kinds of people working together with a specific interactive mode in order to achieve a common goal, such as delivering a product or exploring a project. Following the definition of K-CRIO, Scrum may be seen as one Organization in K-CRIO, in which there are smaller groups also seen as Organizations (sub-organization of Scrum), like the groups ”Pig” and ”Chicken”, or some explicit people seen as Role, like Scrum Master, Product Owner,etc. The expression of definition with logic language as:

Scrum*: Organization (Scrum)* $\bigwedge$ *isSubOrganization (Pig, Scrum)* $\bigwedge$ *isSubOrganization (Chicken, Scrum)*

- In the *Organization: Scrum*, the group ”Pig” (Scrum Team) aims at releasing the product following the requirement of customers and vendors through

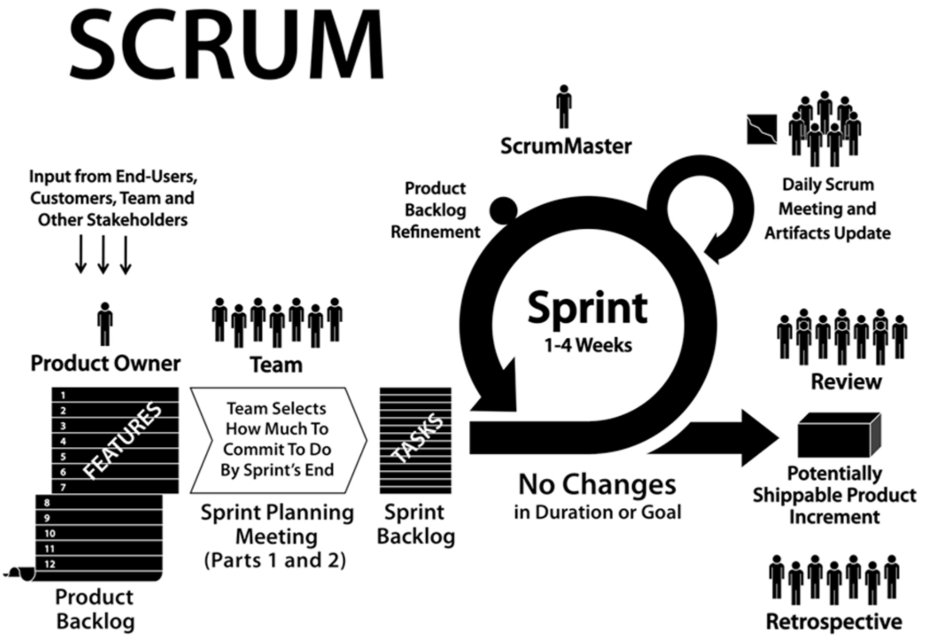

SCRUM
ScrumMaster
Input from End-Users, Customers, Team and Other Stakeholders
Product Backlog Refinement
Daily Scrum Meeting and Artifacts Update
Sprint
1-4 Weeks
Product Owner
Team
Review
FEATURES
1
2
3
4
5
6
7
8
9
10
11
12
Team Selects How Much To Commit To Do By Sprint's End
TASKS
No Changes
in Duration or Goal
Potentially Shippable Product Increment
Sprint Planning Meeting (Parts 1 and 2)
Sprint Backlog
Product Backlog
Retrospective

Fig. 5. The process of Scrum

its members contributions. Therefore, the group "Pig" (Scrum Team) may be seen as the *Organization: "Pig"*(*Organization: Scrum Team*) in K-CRIO, which is sub-Organization of *Organization: Scrum*. As three significant members of "Pig"(Scrum Team), the Product Owner (PO), the Scrum Master (SM) and the Developing Team (DT) fulfill the objective of *Organization: "Pig"* through accomplishing their jobs. Therefore, we could define "Pig" as:

Pig(ScrumTeam): *Organization (Pig)* $\bigwedge$ *isSubOrganization (Pig, Scrum)* $\bigwedge$ *isSubOrganization (DT, Pig)* $\bigwedge$ *includes (Pig, PO)* $\bigwedge$ *includes (Pig, SM)* $\bigwedge$ provided (Pig, Capacities for PO) $\bigwedge$ provided (Pig, Capacities for SM)

Because of the transitivity of isSubOrganization, there is:

isSubOrganization (Pig, Scrum), isSubOrganization (DT, Pig) $\rightarrow$ *isSubOrganization (DT, Scrum)*

Considering about the two Roles of the Pig Group, Product Owner represents the voice of the customer and writes customer-centric items (typically user stories), prioritizes them, and adds them to the product backlog, which is accountable for ensuring that the Team delivers value to the business:

PO: *Role (PO)* $\bigwedge$ *includes (Pig, PO)* $\bigwedge$ *provided (Pig, Capacities for PO)*

As an enforcer of rules, Scrum Master is accountable for removing impediments to the ability of the team to deliver the sprint goal/deliverables and ensures that the Scrum process is used as intended:

SM: *Role (SM)* $\bigwedge$ *includes (Pig, SM)* $\bigwedge$ *provided (Pig, Capacities for SM)*

Differently, as one of necessary members in the group "Pig", Developing Team is typically made up of 5 to 9 developers with cross-functional skills who do the actual work (analyze, design, develop, test, technical communication, document, etc.) for releasing the product. For this reason, we may see Product Owner and Scrum Master as two Roles included in the *Organization: "Pig"* and the Developing Team is one Organization which is sub-Organization of *Organization: "Pig"*. Additionally, the Developer in Developing Team may be seen as Role in K-CRIO, which is included in *Organization: Developing Team.*(Sometimes, Product Owner and Scrum Master may be as members in the Developing Team [1,7], that means *Organization: Developing Team* may include *Role: Product Owner* and *Role: Scrum Master* in certain situation. However, generally these two Roles are out of Developing Team.):

DT: *Organization (DT)* $\bigwedge$ *isSubOrganization (DT, Pig)* $\bigwedge$ *includes (DT, Developer)* $\bigwedge$ *provided (DT, Capacities for Developer)*

Developer: *Role (Developer)* $\bigwedge$ *includes (DT, Developer)* $\bigwedge$ *required (Developer, Capacities for Developer)*

Following the analysis above, *Organization: "Pig"* and *Organization: Developing Team* provide the following capacities required by the roles composing it.

- *Organization: "Pig"* provides:
  - ∗ *Capacity: Representing interests of stakeholder*, *Capacity: Writing User Stories*, *Capacity: Prioritizing Users Stories*, *Capacity: Maintaining the Product Backlog*, etc. (required by *Role: Product Owner*);
  - ∗ *Capacity: Protecting and keeping team focused on its tasks*, *Capacity: Enforcing rules,Removing impediments*, *Capacity: Ensuring Scrum Process used as intended*,etc. (required by *Role: Scrum Master*);
- *Organization: Developing Team* provides: *Capacity: Cross-functional developing skills*, *Capacity: Delivering the Product* (required by *Role: Developer*).

– Similarly, in the *Organization: Scrum*, the group "Chicken" (Ancillary People) is a group of people including Stakeholder and Manager. This group may be seen as the *Organization: "Chicken"* (*Organization: Ancillary People*) in K-CRIO, which is sub-Organization of *Organization: Scrum*. Especially, the Stakeholder represents people for whom the project will produce the agreed-upon benefits, which justify its production, such as the Customer and the Vendor. Furthermore, Manager is the people who will set up the environment for product development. Consequently, *Organization: "Chicken"* includes one *Role:Manager* and has one sub-Organization *Organization: Stakeholder*, which includes *Role: Customer* and *Role: Vendor*:

***Chicken**: Organization (Chicken)* $\bigwedge$ *includes (Chicken, Manager)* $\bigwedge$ *isSubOrganization (Stakeholder, Chicken)* $\bigwedge$ *provided (Chicken, Capacities for Manager)*

***Stakeholder**: Organization (Stakeholder)* $\bigwedge$ *isSubOrganizationOf (Stakeholder, Chicken) includes (Stakeholder, Customer)* $\bigwedge$ *includes (Stakeholder, Vendor)* $\bigwedge$ *provided (Stakeholder, Capacities for Customer)* $\bigwedge$ *provided (Stakeholder, Capacities for Vendor)*

Moreover, we could reason:

isSubOrganization (Chicken, Scrum), isSubOrganization (Stakeholder, Chicken) $\rightarrow$ *isSubOrganization (Stakeholder, Scrum)*

Finally, we denote various capacities supplied to corresponding roles.

- *Organization: "Chicken"* provides *Capacity: Setting up environment for product development*, which is required by *Role: Manager*.
- *Organization: "Stakeholder"* provides *Capacity: Enabling project* required by both *Role: Customer* and *Role: Vendor*.

Following K-CRIO definitions, the above description may be expressed as presented in Figure 6. Organization, Role and Capacity are represented in purple, black and green respectively. Additionally, as an example describing how to define these elements by OWL, Figure 7 expresses *Role: Scrum Master* by OWL.

So far we have declared relevant Organizations, Roles and Capacities in Scrum. In order to describe the whole Scrum process, we will state the interactions occurring during the different Scrum phases. Interactions in K-CRIO may be considered as Casual Interaction or Formalized Interaction separately. Chat is a kind of Casual Interaction between different persons (Roles), other examples may be Exchanging Mail or Joining Conference Meeting, etc. It is Formalized Interactions of Scrum that produces relative *DesignObjects*. In the process of Scrum, we maybe define the following objects as *DesignObject*: *Product, Product Backlog, a release of its sprint, Sprint Backlog, Updated Burndown Chart* and represent Formalized Interaction by an OWL-WS process.

As sketched by Figure 8, the whole Scrum process may be seen as a Composite Process in order to produce the *DesignObject: Product*, which is composed by the Control Construct *Sequence*: a Composite Process: *Articulate Product Vision* (also called *The Pregame Phase*), a Composite Process: *The Game Phase* and a Composite Process: *The Postgame Phase* [17]. In all the figures describing Processes, a single rectangle means an Atomic Process and a double rectangle means a Composite Process.

Precisely, *Articulate Product Vision* may be seen as a Composite Process with the Control Construct *Sequence* (showing in the Figure 9): an Atomic Process:

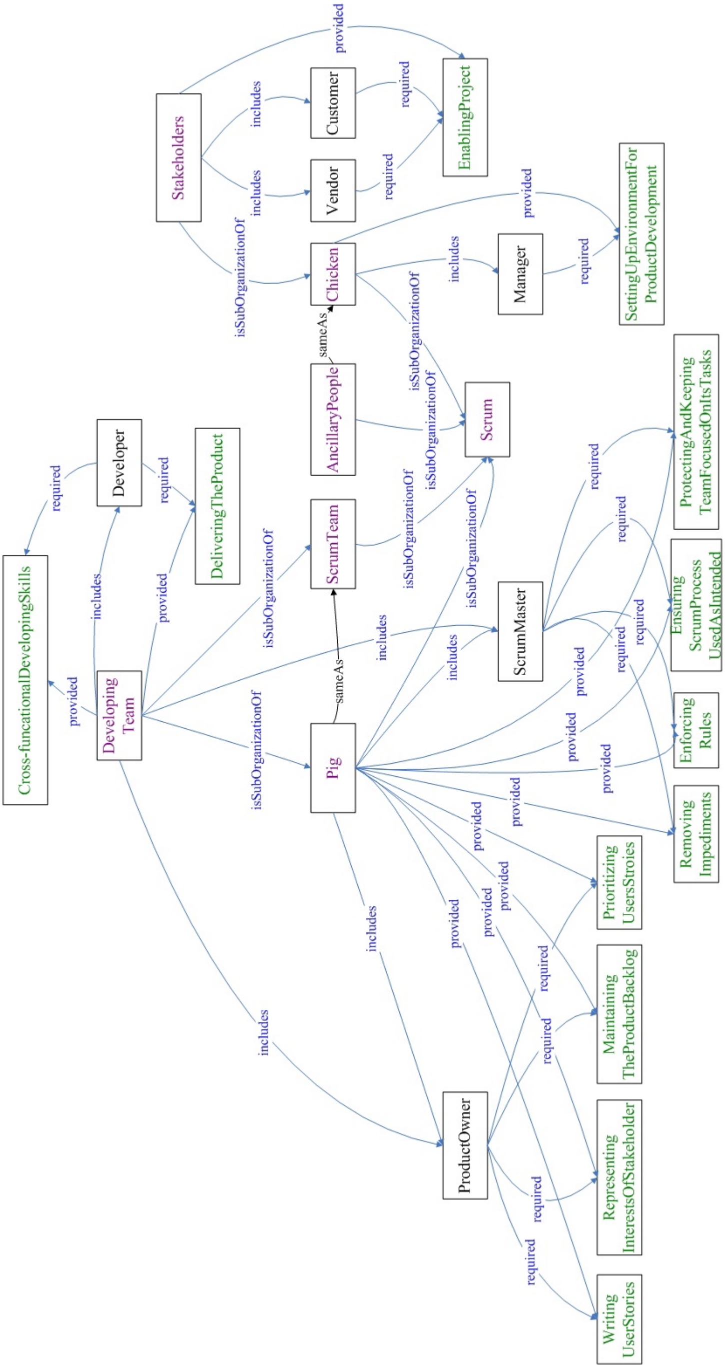
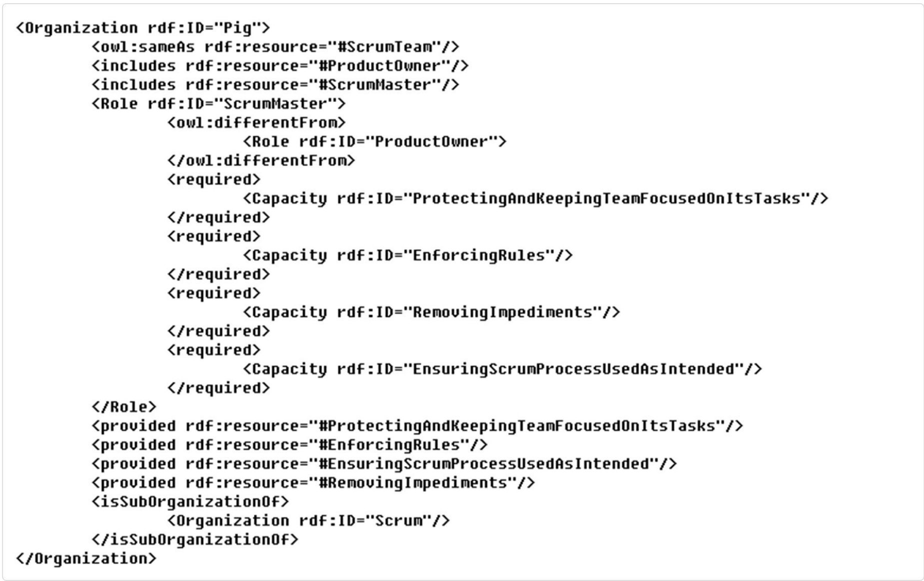


Fig. 6. Scrum with K-CRIO



```
<Organization rdf:ID="Pig">
        <owl:sameAs rdf:resource="#ScrumTeam"/>
        <includes rdf:resource="#ProductOwner"/>
        <includes rdf:resource="#ScrumMaster"/>
        <Role rdf:ID="ScrumMaster">
                <owl:differentFrom>
                        <Role rdf:ID="ProductOwner">
                </owl:differentFrom>
                <required>
                        <Capacity rdf:ID="ProtectingAndKeepingTeamFocusedOnItsTasks"/>
                </required>
                <required>
                        <Capacity rdf:ID="EnforcingRules"/>
                </required>
                <required>
                        <Capacity rdf:ID="RemovingImpediments"/>
                </required>
                <required>
                        <Capacity rdf:ID="EnsuringScrumProcessUsedAsIntended"/>
                </required>
        </Role>
        <provided rdf:resource="#ProtectingAndKeepingTeamFocusedOnItsTasks"/>
        <provided rdf:resource="#EnforcingRules"/>
        <provided rdf:resource="#EnsuringScrumProcessUsedAsIntended"/>
        <provided rdf:resource="#RemovingImpediments"/>
        <isSubOrganizationOf>
                <Organization rdf:ID="Scrum"/>
        </isSubOrganizationOf>
</Organization>
```

Fig. 7. *Role: Scrum Master* in OWL

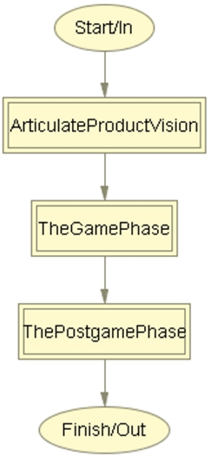


Fig. 8. A Composite Process: *Scrum Process*

Get Requirement from Stakeholder, in which the Product Owner communicates with the stakeholder of the product for collecting User Stories and an Atomic Process: *Make Product Backlog* in order to output the *DesignObject: Product Backlog*.

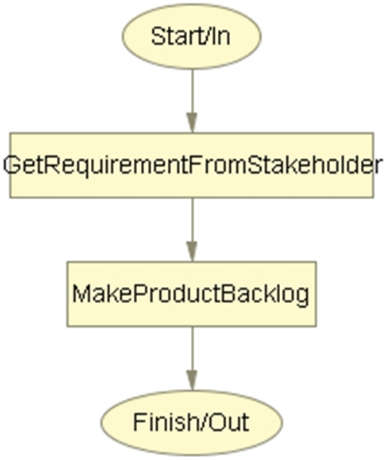


Fig. 9. A Composite Process: *Articulate Production Vision*

The Figure 10 details *The Game Phase*, which may be seen as a composite process with the Control Construct *Repeat-While* for producing the *DesignObject: Product*. The Condition *Has Next Sprint* controls the loop and is initialized to true. While *Has Next Sprint* is true, the processes: *Sprint Planning Meeting*, *Sprint*, *Sprint Review Meeting* and *Sprint Retrospective Meeting* are executed orderly by the Control Construct *Sequence* and return a new value of *Has Next Sprint*. If *Has Next Sprint* is false, the Composite Process: *The Game Phase* is finished.

Sprint Planning Meeting may be seen as a composite process aiming at producing the *DesignObject: Sprint Backlog* as the Figure 11, which is composed of two sequential Atomic Processes: *Sprint Planning Meeting_PartOne* which is for discussing what items of product are chosen for being realized in this Sprint, and *Sprint Planning Meeting_PartTwo* which is focusing on how to finish these tasks.

Sprint may be seen as a composite process with the Control Construct *Repeat-While* as the Figure 12. Every sprint is producing the *DesignObject: a release of its sprint*. The Condition *Sprint Is Not Finished* controls the loop and is initialized to true. While *Sprint Is Not Finished* is true, the Atomic Process: *Daily Scrum* (producing *DesignObject: Updated Burndown Chart*) is executed and return a new value of *Sprint Is Not Finished*. If *Sprint Is Not Finished* is false, the Composite Process *Sprint* is finished and *Sprint Review Meeting* and *Sprint Retrospective Meeting* are executed sequentially. Moreover, in the Figure

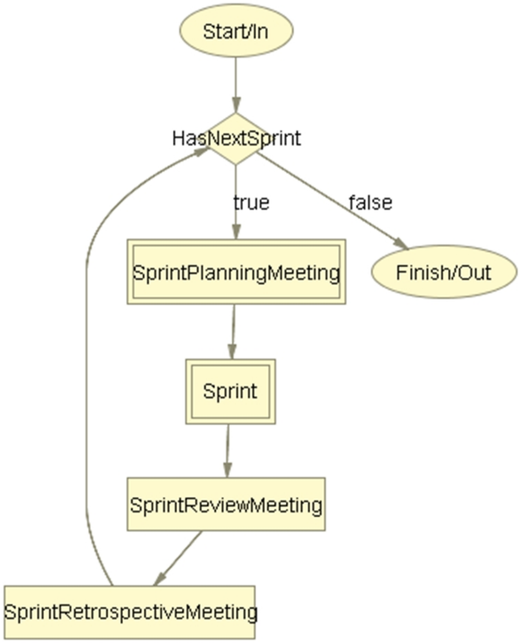


Fig. 10. A Composite Process: *The Game Phase*

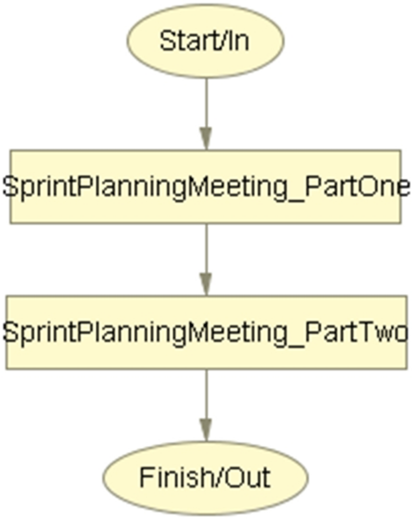


Fig. 11. A Composite Process: *Sprint Planning Meeting*

[13], we use this simple formalized interaction, the Composite Process: *Sprint* to express defining the Formalized Interaction by OWL-WS.

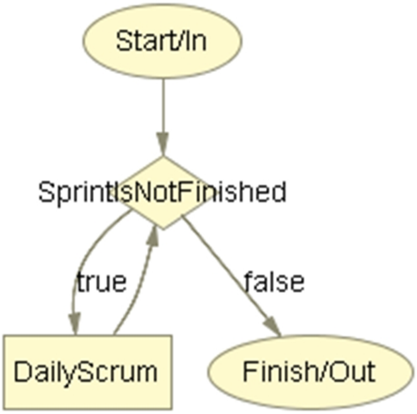


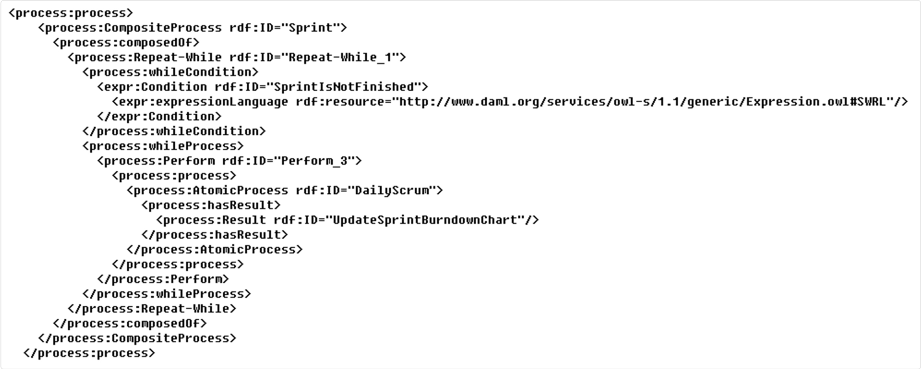
Fig. 12. A Composite Process: *Sprint*

Eventually, *The Postgame Phase* may be seen as a composite process, which may be composed of the processes: *Integration Testing*, *Writing User Document*, *Training Users*, *Marketing Material Preparation* and so on.

These concepts and relationships of Scrum represented by K-CRIO Ontology will be used in a tool for managing knowledge in really enterprise working under Scrum Method. The tool will be developed as Multi-Agent System in the environment Janus [6]. we now take some query examples to address how the ontology could help people managing knowledge during the activities.

- we want to know that one project is executed in which process at present? We suppose ***P*** is the process we want to find. Hence:
  P: FormalizedInteraction (P) $\bigwedge$ *at (P, Doing)* $\bigwedge$ *hasPre-Interaction* (P, P') $\bigwedge$ *at* $(P', Done)$

- we want to know which developers are late for the schedule during all the sprint process?



```
<process:process>
    <process:CompositeProcess rdf:ID="Sprint">
      <process:composedOf>
        <process:Repeat-While rdf:ID="Repeat-While_1">
          <process:whileCondition>
            <expr:Condition rdf:ID="SprintIsNotFinished">
              <expr:expressionLanguage rdf:resource="http://www.daml.org/services/owl-s/1.1/generic/Expression.owl#SWRL"/>
            </expr:Condition>
          </process:whileCondition>
          <process:whileProcess>
            <process:Perform rdf:ID="Perform_3">
              <process:process>
                <process:AtomicProcess rdf:ID="DailyScrum">
                  <process:hasResult>
                    <process:Result rdf:ID="UpdateSprintBurndownChart"/>
                  </process:hasResult>
                </process:AtomicProcess>
              </process:process>
            </process:Perform>
          </process:whileProcess>
        </process:Repeat-While>
      </process:composedOf>
    </process:CompositeProcess>
  </process:process>
```

Fig. 13. A Composite Process: *Sprint* by OWL-WS

We suppose ***D*** is the Developer we want to find. Hence:
D: Developer (D) $\bigwedge$ *hasParticipants (Sprint, D)* $\bigwedge$ *follows (Sprint, BeginningTime)* $\bigwedge$ *follows (Sprint, RealBeginningTime)* $\bigwedge$ *follows (Sprint, EndTime)* $\bigwedge$ *follows (Sprint, RealEndTime)* $\bigwedge$ $\{$*(RealEndTime - RealBeginningTime)* $>$ *(EndTime - BeginningTime)*$\}$

4 Related Works

Formal structures such as OWL and RDFS have been used for Personal Information Management before; the PIMO model [15] aims to represent parts of the Mental Model necessary for tasks involving knowledge. The Mental Model is part of the cognitive system of a person. Subjective to a person, the mental model is individual and cannot be externalized thoroughly. The definition for a Personal Information Model (PIMO) is given as follow: A PIMO is a Personal Information Model of one person. It is a formal representation of parts of the users Mental Model. Each concept in the Mental Model can be represented using a Thing or a subclass of this class in RDF. Native Resources found in the Personal Knowledge Workspace can be categorized, and are occurrences of Thing. The vision is that a Personal Information Model reflects and captures a user's personal knowledge, e.g., about people and their roles, about organizations, processes, and so forth, by providing the vocabulary (concepts and their relationships) for expressing concepts as well as concrete instances. In other words, the domain of a PIMO is meant to be "all things and native resources that are pertinent for the user when doing work involving knowledge". Though "native" information models and structures are widely used, there is still much potential for a more effective and efficient exploitation of the underlying knowledge. Compared to the cognitive representations humans build, there are mainly two shortcomings in native structures:

- Model richness: current state of cognitive psychology assumes that humans build very rich models, representing not only detailed factual aspects, but also episodic and situational informations. Native structures are mostly taxonomy-oriented or keyword-oriented.
- Models coherence: though nowadays (business) life is very fragmented, humans tend to interpret situations as a coherent whole and have representations of concepts that are comprehensive across contexts. Native structures, on the other hand, often reflect the fragmentation of multiple contexts. They tend to be redundant (i.e., the same concepts at multiple places in multiple native structures). Frequently, inconsistencies are the consequence.

In brief, the PIMO shall mitigate the shortcomings of native structures by providing a comprehensive model on a sound formal basis.

Multilayered Semantic Social Network (MSSN) Model [4] proposes a multilayered semantic social network model that offers different views of common interests underlying a community of people, which is working within an ontology-based personalization framework [18], user preferences are represented as vectors $u_i = (u_{i,1}, u_{i,2}, ..., u_{i,N})$ where the weight $u_{i,j} \in [0, 1]$ measures the intensity of the interest of user i for concept c_j in the domain ontology, N being the total number of concepts in the ontology. Similarly, the objects d_k in the retrieval space are assumed to be described (annotated) by vectors $(d_{k,1}, d_{k,2}, ..., d_{k,N})$ of concept weights, in the same vector-space as user preferences. Based on this common logical representation, measures of user interest for content items can be computed by comparing preference and annotation vectors, and these measures can be used to prioritize, filter and rank contents (a collection, a catalog, a search result) in a personal way. The applicability of the proposed model to a collaborative filtering system is empirically studied. Starting from a number of ontology-based user profiles and taking into account their common preferences, the concept space domain is automatically clustered. With the obtained semantic clusters, similarities among individuals are identified at multiple semantic preference layers, and emergent, layered social networks are defined, suitable to be used in collaborative environments and content recommenders.

The AIC Model represents such a system as a tripartite graph with hyperedges [13]. The set of vertices is partitioned into the three (possibly empty) disjoint sets $A = \{a_1, a_2, ..., a_k\}$, $C = \{c_1, c_2, ..., c_i\}$, $I = \{i_1, i_2, ..., i_m\}$ corresponding to the set of actors (users), the set of concepts (tags, keywords) and the set of annotated objects (bookmarks, photos etc). It extends the traditional bipartite model of ontology (concepts and instances) by incorporating actors in the model. In a social tagging system, users tag objects with concepts, creating ternary associations between the user, the concept and the object. Thus the folksonomy is defined by a set of annotations $T \in A \times C \times I$. Such a network is most naturally represented as an hypergraph with ternary edges, where each edge represents the fact that a given actor is associated with a certain instance and a certain concept. In particular, the author defines the representing hypergraph of a folksonomy T as a (simple) tripartite hypergraph $H(T) = (V, E)$ where $V = A \cup C \cup I$, $E = \{\{a, c, i\} \mid (a, c, i) \in T\}$.

The MOISE+ Model [9] structure is built up in three levels: one is the behaviors that an agent playing a role is responsible for (individual), the other is the structure and interconnection of the roles with each other (social), and the last is the aggregation of roles in large structures (collective). In MOISE+, as in MOISE, three main concepts, roles, role relations, and groups, are be used to build, respectively, the individual, social, and collective structural levels of an organization. Furthermore, the MOISE original structural dimension is enriched with concepts such as inheritance, compatibility, cardinality, and sub-groups.

- **Individual level** is formed by the roles of the organization. A role means a set of constraints that an agent ought to follow when it accepts to enter a group playing that role. Following, these constraints are defined in two ways: in relation to other roles (in the collective structural level) and in a deontic relation to global plans (in the functional dimension). In order to simplify the specification, like in Object-Oriented (OO) terms, there is an inheritance relation among roles. If a role p' inherits a role p (denoted by $p \subset p'$), with $p \neq \ p'$, p' receives some properties from p, and p' is a sub-role, or specialization, of p. In the definition of the role properties presented in the sequence, it will be precisely stated what one specialized role inherits from another role. For example, in the soccer domain, the attacker role has many properties of the player role ($p_{player} \subset p_{attacker}$). It is also possible to state that a role specialize more than one role, i.e., a role can receive properties from more than one role. The set of all roles are denoted by R_{ss}. Following this OO inspiration, we can define an abstract role as a role that cannot be played by any agent. It has just a specification purpose. The set of all abstract roles is detonated by $R_{abc}(R_{abc} \subset R_{ss})$. There is also a special abstract role $P_{soc}$where $\forall_{(p\in \ R_{ss})} P_{soc} \subset P$, trough the transitivity of $\subset$, all other roles are specializations of it.
- **Social level** is when the inheritance relation does not have a direct effect on the agents' behavior, there are other kinds of relations among roles that directly constrain the agents. Those relations are called links and are represented by the predicate $link_{(p_s,p_d,t)}$ where p_s is the link source, p_d is the link destination, and $t \in acq, com, aut$ is the link type. In case the link type is acq (acquaintance), the agents playing the source role p_s are allowed to have a representation of the agents playing the destination role p_d (p_d agents, in short).In a communication $link(t = com)$, the p_s agents are allowed to communicate with p_d agents. In an authority $link(t = aut)$, the p_s agents are allowed to have authority on p_d agents, i.e., to control them. An authority link implies the existence of a communication link that implies the existence of an acquaintance link:

$$link_{p_s,p_d,aut} \Longrightarrow link_{p_s,p_d,com} \tag{1}$$

$$link_{p_s,p_d,com} \Longrightarrow link_{p_s,p_d,acq} \tag{2}$$

Regarding the inheritance relation, the links follow the rules:

$$(link_{(p_s,p_d,t)} \wedge p_s \subset p_{s'}) \Longrightarrow link_{(p_{s'},p_d,t)} \tag{3}$$

$$(link_{(p_s,p_d,t)} \wedge p_s \subset p_{d'}) \Longrightarrow link_{(p_s,p_{d'},t)} \tag{4}$$

Take an instance, if the coach role has authority on the player role $link_{(p_{coah},p_{player},aut)}$ and player has a sub-role ($p_{player} \subset p_{attacker}$), by Eq. (4), a coach has also authority on attackers. Moreover, a coach is allowed to communicate with players (by Eq. (1)) and it is allowed to represent the players (by Eq. (2)).

- **Collective Level:** the links constrain the agents after they have accepted to play a role. However we should constrain the roles that an agent is allowed to play depending on the roles this agent is currently playing. This compatibility constraint $p_a \bowtie p_b$ states that the agents playing the role are also allowed to play the role p_b (it is a reflexive and transitive relation). As an example, the team leader role is compatible with the back player role $p_{leader} \bowtie p_{back}$. If it is not specified that two roles are compatible, by default they are not. Regarding the inheritance, this relation follows the rule

$$(p_a \bowtie p_b \wedge p_a \neq p_b \wedge p_a \sqsubseteq p') \Longrightarrow (p' \bowtie p_b) \tag{5}$$

Hence, there should be a series of rules and relationships defined within the collective level.

The software & Systems Process Engineering meta-model (SPEM) allows the Modelling of software processes using OMG (Object Management Group) standard such as the MOF (meta-object facility) and UML: making possible to represent software processes using tools compliant with UML. Daniel Rodriguez et al [14] represents generic processes modeled with SPEM using an underlying ontology based on the OWL representation together with data derived from actual projects. SPEM is generally used to design generic software processes such as Scrum. Therefore, Daniel Rodriguez et al [14] discussed a first approach to create an ontology from the Scrum process as example using SPEM. In Daniel Rodriguez et al [14], Scrum Ontology extends from the *Role* class in the method-content ontology in SPEM. *Method-Content: Role* has *scrum:Pig* and *scrum:Chicken* as subclasses. The *productOwner*, *scrumMaster* and *team* were instances of the *Scrum Chicken role.* Moreover *WorkProduct* as one class in the method content ontology has three subclasses: *Artifact*, *Deliverable* and *Outcome.* The term like SprintBacklog is as an instance or individual of *Artifact* as the part of the scrum Ontology. Furthermore, the process ontology of SPEM has *Activity* class with *Iteration* and *Phase* classes defined as subclasses. Relatively, the *PreGame*, *Game* and *PostGame* are defined in the scrum ontology.

One of the earliest initiatives was the TOVE project that aimed at development of a set of integrated ontologies for modeling all kinds of enterprise, such as commercial and public ones. The TOVE Organization Ontology [8] for Enterprise Modelling is one of these, which puts forward a number of conceptualizations as agents, roles, positions, goals, communication, authority, commitment. Precisely, one organization consists a set of Organization-Agents(OA) having two sub-classes: Individual-Agents and Group-Agents, a set of Organization-Units as recursive subcomponents and an Organization-Goal tree (could divide into sub-goals). An Organization-Role defines a prototypical function of an agent in an organization. Each Organization-Role played by OA has Organization-Goals or Role-Goals, Role-Skills, Role-Process or Organization-Activity, Role-Policy, Role-Communication-Link. Moreover, an Organization-Position defines a formal position that can be filled by OA in the Organization.

5 Conclusion

The goal of this paper was to present K-CRIO, an ontology of organizations for their understanding, analysis and also to enable reasoning. The targeted organizations are those dedicated to the realization of products following a given design process. The definition of this ontology relies on OWL and on the concepts used in the CRIO metamodel. The Scrum software development process is taken as example to illustrate the use of K-CRIO for describing a specific organization.

We are aware that the K-CRIO ontology is not complete. It is a first attempt to build a rich ontology for organizations. Further studies, done with heterogeneous organizations will help us in refining K-CRIO and adding new concepts and relationships. In the meanwhile, the idea is to use K-CRIO as a semantic layer for collaborative softwares in order to enhance Knowledge Management within the targeted organizations.

References

1. http://en.wikipedia.org/wiki/scrum
2. Beco, S., Cantalupo, B., Giammarino, L., Matskanis, N., Surridge, M.: OWL-WS: A workflow ontology for dynamic grid service composition. In: eScience, pp. 148–155. IEEE Computer Society (2005)
3. Bottazzi, E., Ferrario, R.: Preliminaries to a dolce ontology of organizations. International Journal of Business Process Integration and Management 4(4), 225–238 (2009)
4. Cantador, I., Castells, P.: Multilayered Semantic Social Network Modeling by Ontology-Based User Profiles Clustering: Application to Collaborative Filtering. In: Staab, S., Svátek, V. (eds.) EKAW 2006. LNCS (LNAI), vol. 4248, pp. 334–349. Springer, Heidelberg (2006)
5. Cossentino, M., Gaud, N., Galland, S., Hilaire, V., Koukam, A.: A Holonic Metamodel for Agent-Oriented Analysis and Design. In: Mařík, V., Vyatkin, V., Colombo, A.W. (eds.) HoloMAS 2007. LNCS (LNAI), vol. 4659, pp. 237–246. Springer, Heidelberg (2007)
6. Cossentino, M., Gaud, N., Hilaire, V., Galland, S., Koukam, A.: Aspecs: an agent-oriented software process for engineering complex systems. Autonomous Agents and Multi-Agent Systems 20, 260–304 (2010), doi:10.1007/s10458-009-9099-4
7. Deemer, P., Benefield, G., Larman, C., Vodde, B.: The scrum primer. Technical report (2010), http://www.goodagile.com

8. Fox, M.S., Barbuceanu, M., Gruninger, M.: An organisation ontology for enterprise modeling: Preliminary concepts for linking structure and behaviour. Computers in Industry 29(1-2), 123–134 (1996); WET ICE 1995
9. Hübner, J.F., Sichman, J.S., Boissier, O.: A Model for the Structural, Functional, and Deontic Specification of Organizations in Multiagent Systems. In: Bittencourt, G., Ramalho, G.L. (eds.) SBIA 2002. LNCS (LNAI), vol. 2507, pp. 118–128. Springer, Heidelberg (2002)
10. Lin, Y., Hilaire, V., Gaud, N., Koukam, A.: K-CRIO: An Ontology for Organizations Involved in Product Design. In: Cherifi, H., Zain, J.M., El-Qawasmeh, E. (eds.) DICTAP 2011 Part II. CCIS, vol. 167, pp. 362–376. Springer, Heidelberg (2011)
11. Martin, D., Paolucci, M., McIlraith, S., Burstein, M., McDermott, D., McGuinness, D.L., Parsia, B., Payne, T.R., Sabou, M., Solanki, M., Srinivasan, N., Sycara, K.: Bringing Semantics to Web Services: The OWL-S Approach. In: Cardoso, J., Sheth, A.P. (eds.) SWSWPC 2004. LNCS, vol. 3387, pp. 26–42. Springer, Heidelberg (2005)
12. McGuinness, D.L., Van Harmelen, F.: http://www.w3.org/tr/owl-features/
13. Mika, P.: Ontologies are us: A unified model of social networks and semantics. J. Web Sem. 5(1), 5–15 (2007)
14. Rodrguez-Garca, D., Barriocanal, E.G., Alonso, S.S., Nuzzi, C.R.-S.: Defining software process model constraints with rules using owl and swrl. International Journal of Software Engineering and Knowledge Engineering
15. Sauermann, L., Van Elst, L., Dengel, A.: Pimo - a framework for representing personal information models. In: Proceedings of I-Semantics 2007, JUCS, pp. 270–277 (2007)
16. Schreiber, G.: http://www.cs.vu.nl/~guus/public/owl-restrictions/
17. ScrumMethodology.org. Scrum phases (2009), http://www.scrummethodology.org/scrum-phases.html
18. Vallet, D., Mylonas, P., Corella, M.A., Fuentesa, J.M., Castells, P., Avrithis, Y.: A semantically-enhanced personalization framework for knowledge-driven media services. In: IADIS WWW/Internet Conference, pp. 11–18 (2005)

Author Index