

Domenico Bianculli  
Radu Calinescu  
Bernhard Rumpe (Eds.)

LNCS 9509

# Software Engineering and Formal Methods

**SEFM 2015 Collocated Workshops:  
ATSE, HOFM, MoKMaSD, and VERY\*SCART  
York, UK, September 7–8, 2015, Revised Selected Papers**



Springer

*Commenced Publication in 1973*

Founding and Former Series Editors:

Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

## Editorial Board

David Hutchison

*Lancaster University, Lancaster, UK*

Takeo Kanade

*Carnegie Mellon University, Pittsburgh, PA, USA*

Josef Kittler

*University of Surrey, Guildford, UK*

Jon M. Kleinberg

*Cornell University, Ithaca, NY, USA*

Friedemann Mattern

*ETH Zurich, Zürich, Switzerland*

John C. Mitchell

*Stanford University, Stanford, CA, USA*

Moni Naor

*Weizmann Institute of Science, Rehovot, Israel*

C. Pandu Rangan

*Indian Institute of Technology, Madras, India*

Bernhard Steffen

*TU Dortmund University, Dortmund, Germany*

Demetri Terzopoulos

*University of California, Los Angeles, CA, USA*

Doug Tygar

*University of California, Berkeley, CA, USA*

Gerhard Weikum

*Max Planck Institute for Informatics, Saarbrücken, Germany*

More information about this series at <http://www.springer.com/series/7408>

Domenico Bianculli · Radu Calinescu  
Bernhard Rumpe (Eds.)

# Software Engineering and Formal Methods

SEFM 2015 Collocated Workshops:  
ATSE, HOFM, MoKMaSD, and VERY\*SCART  
York, UK, September 7–8, 2015  
Revised Selected Papers

*Editors*

Domenico Bianculli  
Interdisciplinary Centre for ICT Se  
University of Luxembourg (UL)  
Luxembourg

Radu Calinescu  
University of York  
York  
UK

Bernhard Rumpe  
LS Software Engineering  
RWTH Aachen Universität  
Aachen  
Germany

ISSN 0302-9743                      ISSN 1611-3349 (electronic)  
Lecture Notes in Computer Science  
ISBN 978-3-662-49223-9            ISBN 978-3-662-49224-6 (eBook)  
DOI 10.1007/978-3-662-49224-6

Library of Congress Control Number: 2015946575

LNCS Sublibrary: SL2 – Programming and Software Engineering

© Springer-Verlag Berlin Heidelberg 2015

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.

The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

The publisher, the authors and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, express or implied, with respect to the material contained herein or for any errors or omissions that may have been made.

Printed on acid-free paper

This Springer imprint is published by SpringerNature  
The registered company is Springer-Verlag GmbH Berlin Heidelberg

# Preface

This volume contains the technical papers presented in the four workshops collocated with SEFM 2015, the 13th International Conference on Software Engineering and Formal Methods 2015. The workshops were held in York, UK, during September 7–8, 2015.

SEFM 2015 brought together researchers and practitioners from academia, industry, and government to advance the state of the art in formal methods, to facilitate their uptake in the software industry, and to encourage their integration within practical software engineering methods and tools. The satellite workshops provided a highly interactive and collaborative environment for researchers and practitioners from industry and academia to discuss emerging areas of software engineering and formal methods.

The four workshops were:

- ATSE 2015 — The 6th Workshop on Automating Test Case Design, Selection and Evaluation
- HOFM 2015 — The Second Human-Oriented Formal Methods Workshop
- MoKMaSD 2015 — The 4th International Symposium on Modeling and Knowledge Management Applications: Systems and Domains
- VERY\*SCART 2015 — The First International Workshop on the Art of Service Composition and Formal Verification for Self-\* Systems.

A brief description of each workshop, written by its organizers, and the abstracts of the keynote talks follow.

We are grateful to EasyChair for the support with the paper submission and reviewing process for all workshops, and with the preparation of this volume. For each of the workshops at SEFM 2015, we thank the organizers for the interesting topics and resulting talks. We also thank the paper contributors to these workshops and those who attended them. We would like to extend our thanks to all keynote speakers for their support and excellent presentations, and also, to the members of each workshop's Program Committee.

November 2015

Domenico Bianculli  
Radu Calinescu  
Bernhard Rumpe

## **ATSE 2015 Organizers' Message**

The 6th Workshop on Automating Test Case Design, Selection and Evaluation (ATSE 2015), was held this year in conjunction with the 13th International Conference on Software Engineering and Formal Methods (SEFM), in York, UK, on September 7, 2015. Again we had a very interesting workshop that provided a venue for researchers as well as industry participants to exchange and discuss trending views, ideas, state-of-the-art work and work in progress, and scientific results on automated test case design, selection, and evaluation. In all, 40 % of the workshop participants came from industry.

We started off with a keynote by Joachim Wegener from Berner & Mattner (Germany) on “Automatic Generation and Execution of Test Scenarios for Camera-Based Driver Assistance Systems.” This was followed by the keynote of Cristina Seceleanu from MDH (Sweden) on “Testing Function and Time for Embedded Systems.” Subsequently, and based on the submissions received this year for ATSE 2015, the workshop concentrated on three topics:

1. Learning-based testing
2. User experience design and testing
3. Model-based statistical testing

The workshop format was highly interactive. We urged the participants to come to the workshop prepared by having already read the papers. This way we could focus the discussions on the topics and start to set an agenda and lay the foundation for future development.

We enjoyed ATSE 2015 greatly and look forward to the next edition! The program chairs would like to thank all of the reviewers for their excellent work and are grateful to everybody involved in the ATSE 2015 workshop for their support before, during, and after the workshop.

November 2015

Tanja E.J. Vos  
Sigrid Eldh  
Wishnu Prasetya

### **Program Committee**

Pekka Aho	VTT, Finland
Alessandra Bagnato	Softeam, France
Maria Alpuente	Universidad Politécnic de Valencia, Spain
Arie van Deursen	TU Delft, The Netherlands
John Derrick	University of Sheffield, UK
Maria Jose Escalona	Universidad de Sevilla, Spain
Gordon Fraser	University of Sheffield, UK

Herman Geuvers	Radboud University Nijmegen, The Netherlands
Marieke Huisman	Universiteit van Twente, The Netherlands
Teemu Kanstrén	VTT, Finland
Peter M. Kruse	Berner & Mattner, Germany
Beatriz Marin	Universidad Diego Portales, Chile
Karl Meinke	KTH, Sweden
Mehrdad Saadatmand	Mälardalen University, Sweden
Marielle Stoelinga	Universiteit van Twente, The Netherlands
Jan Tretmans	TNO, The Netherlands
Javier Tuya	Universidad de Oviedo, Spain
Marc-Florian Wendland	Fraunhofer, Germany
Carsten Weise	imbus AG, Germany
Burkhardt Wolff	LRI, France

### **Sponsoring Institutions**

Universidad Politécnica de Valencia, Spain  
Berner & Mattner GmbH, Germany  
Ericsson, Sweden



## HOFM 2015 Organizers' Message

The aim of the Human-Oriented Formal Methods (HOFM) workshop series is to bring together researchers and practitioners from academia and industry to exchange ideas and experience in the field of the application of human factors to the analysis and to the optimization of formal methods, as well as to present ongoing research and emerging results in this field. HOFM also aims to develop a future vision and roadmap of usability and automation of formal methods, focusing especially on readability and ease of use.

The Second Human-Oriented Formal Methods (HOFM) Workshop was held on September 7, 2015, in York, UK. This international workshop was affiliated to the 13th International Conference on Software Engineering and Formal Methods (SEFM 2015). The aim of the HOFM workshop series is to establish a community that will investigate the field of application of human factors to the analysis and to the optimization of formal methods. Formal methods (FMs) have been successfully applied in software engineering research for several decades. However, many software engineers largely reject FMs as “too hard to understand and use in practice” while admitting that they are powerful and precise. The reason for this rejection is the lack of usability features: If usability is compromised, methods cannot fit in a real software development process.

There are many applications of FMs to analyze human-machine interaction and to construct user interfaces. However, the field of application of human factors to the analysis and to the optimization of FMs in the sense of usability is almost unexplored. The first and second editions of the workshop showed that there is interest in collaborations and discussions on this topic, and that there are currently more questions than answers in this field. Bad design of interfaces and languages can induce unnecessary human error, cf., e.g., [7]; however, the error information can be used to improve the quality of software and the corresponding development artifacts, also including FMs [1, 2, 5, 8]. “Formal” does not mean “unreadable,” and the readability and usability of FMs might be increased by analyzing human factors related to the specification, modeling, and verification [4].

HOFM 2015 received submissions from 15 authors, affiliated with universities and industry from the UK, Germany, Israel, Norway, Australia, The Netherlands, and Tunisia. Each submission was reviewed by at least three Program Committee members, and five regular papers were accepted for presentation at HOFM 2015.

The HOFM 2015 pre-workshop proceedings, which include all papers presented at the workshop, are available online at the workshop site [3]. All HOFM authors were invited to submit extended versions of their peer-reviewed papers to the post-workshop proceedings, taking into account the feedback from the HOFM reviewers as well as the discussions during the workshop.

An introduction to the second HOFM workshop was given by the keynote talk “Beating Error with Formal Methods,” given by Harold Thimbleby, Swansea University, Wales, UK. In this presentation, Thimbleby emphasized that FMs provide another point of view, namely, mathematical reasoning, on software engineering problems, and this special point of view can help to identify issues that normal human

thinking misses. This introduction led the workshop discussion on the point that FMs are very important for software engineering, especially for the field of safety-critical systems, but their limited understandability might become an obstacle for the broad application of FMs.

The workshop was concluded by an open discussion on the topics of the regular paper talks and the keynote talk as well as on the roadmap for research on human factors in formal methods.

The goal of the open discussion was to stimulate collaboration between researchers and to develop a future vision and roadmap of usability, automation, and other human-oriented aspects of FMs, focusing especially on readability and understandability.

The main focus of the discussion was on the teaching of FMs and the human factors that are related to this learning and teaching activity. Workshop presenters and participants agreed that in order to make progress toward adoption of FMs in industry, we have to work on the popularization of FMs as a part of the university curriculum, also taking into account the different backgrounds and aims of the students.

The discussion further focused on the following questions:

- How can we influence the readability, understandability, and perception of FMs?
- How can we deal with collaborative aspects of specification and verification?
- How can we contribute to the sustainability of the FMs (and through FMs)?

One of the early results of this discussion is paper [6], accepted for presentation at the International Workshop on Automated Testing of Cyber-Physical Systems in the Cloud.

We would like to thank all authors who contributed to HOFM 2015 as well as all workshop participants. We hope that the participants found the program inspiring and relevant to their interests. We also thank the SEFM workshop chairs and local organizers for their help. We would like to express our gratitude to the Program Committee members for their support and thorough reviews.

November 2015

Maria Spichkova  
Heinz Schmidt

## Program Committee

Katherine Blashki	Noroff University College, Norway
Manfred Broy	Technical University München, Germany
Pedro Isaías	Universidade Aberta, Portugal
Lalchandani Jayprakash	IIT Bangalore, India
Peter Herrmann	NTNU Trondheim, Norway
Tim Miller	The University of Melbourne, Australia
Srini Ramaswamy	ABB, USA
Daniel Ratiu	Siemens AG, Germany
Guillermo Rodriguez-Navas	Mälardalen University, Sweden
Bernhard Rumpe	RWTH Aachen, Germany
Bernhard Schätz	fortiss GmbH, Germany

Heinz Schmidt  
Ruimin Shen  
Maria Spichkova  
Judith Stafford

RMIT University, Australia (Chair)  
Shanghai Jiao Tong University, China  
RMIT University, Australia (Chair)  
University of Colorado, USA

## References

1. Dhillon, B.: Engineering Usability: Fundamentals, Applications, Human Factors, and Human Error. American Scientific Publishers (2004)
2. Redmill, F., Rajan, J.: Human Factors in Safety-Critical Systems. Butterworth-Heinemann (1997)
3. Second International Human-Oriented Formal Methods (HOFM) Workshop. <https://hofm2015.wordpress.com/>
4. Spichkova, M.: Design of Formal Languages and Interfaces: “Formal” does not Mean “Unreadable”. IGI Global (2013)
5. Spichkova, M., Liu, H., Laali, M., Schmidt, H.W.: Human factors in software reliability engineering. In: Workshop on Applications of Human Error Research to Improve Software Engineering (WAHESE2015) (2015)
6. Spichkova, M., Zamansky, A., Farchi, E.: Towards a human-centred approach in modelling and testing of cyber-physical systems. In: International Workshop on Automated Testing of Cyber-Physical Systems in the Cloud (cpsATcloud2015) (2015) (to appear)
7. Thimbleby, H., Oladimeji, P., Cairns, P.: Unreliable numbers: error and harm induced by bad design can be reduced by better design. *J. R. Soc. Interface.* **12**(110), 20150685 (2015)
8. Walia, G., Carver, J.: Using error information to improve software quality. In: IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW), pp. 107–107 (2013)

## MoKMaSD 2015 Organizers' Message

The 4th International Symposium on Modelling and Knowledge Management Applications: Systems and Domains (MoKMaSD 2015) was held in York, UK, on September 8, 2015. The aim of the symposium is to bring together practitioners and researchers from academia, industry, government, and non-government organizations to present research results and exchange experiences, ideas, and solutions for modelling and analyzing complex systems and for using knowledge management strategies, technology, and systems in various domain areas such as ecology, biology, medicine, climate, governance, education, and social software engineering. In particular, the focus is on synergistic approaches that integrate modelling and knowledge management/discovery or exploit knowledge management/discovery to develop/synthesize system models.

After a careful review process, the Program Committee accepted nine papers and two oral presentations with extended abstracts for the proceedings. The program of MoKMaSD 2015 was enriched by the keynote speeches by Tias Guns entitled "Constraint Modelling and Solving for Data Mining" and by Guido Sanguinetti entitled "Machine Learning Methods for Model Checking in Continuous Time Markov Chains."

Several people contributed to the success of MoKMaSD 2015. We are grateful to Antonio Cerone, who invited us to chair this edition of the symposium and assisted us in some organizational aspects of the event. We would like to thank the organizers of SEFM 2015, and in particular the general chair, Jim Woodcock, and the workshops chair, Domenico Bianculli. We would also like to thank the Program Committee and the additional reviewers for their work in reviewing the papers. The process of reviewing and selecting papers was significantly simplified through using EasyChair.

We thank all the symposium attendants and hope that this event will enable a good exchange of ideas and generate new collaborations among attendees.

The organization of MoKMaSD 2015 was supported by the research project "Metodologie computazionali per la medicina personalizzata" funded by the University of Pisa (Project reference: PRA\_2015\_0058).

November 2015

Paolo Milazzo  
Anna Monreale

### Program Committee

Luis Barbosa  
Bettina Berendt  
Luca Bortolussi  
Peter Breuer  
Giulio Caravagna  
Antonio Cerone

University of Minho, Portugal  
KU Leuven, Belgium  
University of Trieste, Italy  
Birmingham City University, UK  
University of Milano-Bicocca, Italy  
IMT Institute for Advanced Studies Lucca, Italy

Andrea Esuli	ISTI-CNR, Pisa, Italy
Alexeis Garcia-Perez	Coventry University, UK
Jane Hillston	University of Edinburgh, UK
Joris Hulstijn	Delft University of Technology, The Netherlands
Marijn Janssen	Delft University of Technology, The Netherlands
Donato Malerba	University of Bari, Italy
Stan Matwin	University of Ottawa, Canada
Paolo Milazzo	University of Pisa, Italy (Co-chair)
Anna Monreale	University of Pisa, Italy (Co-chair)
Siegfried Nijssen	KU Leuven, Belgium and University of Leiden, The Netherlands
Adegboyega Ojo	DERI, National University of Ireland, Ireland
Giovanni Pardini	University of Pisa, Italy
Nikos Pelekis	University of Piraeus, Greece
Anna Philippou	University of Cyprus, Cyprus
Giulio Rossetti	ISTI-CNR and University of Pisa, Italy
Marco Scotti	GEOMAR Centre, Kiel, Germany
Filippo Simini	University of Bristol, UK
Manolis Terrovitis	IMIS, Athena Research Center, Greece
Luca Tesei	University of Camerino, Italy

## VERY\*SCART 2015 Organizers' Message

The First Workshop on The Art of Service Composition and Formal Verification for Self-\* Systems (VERY\*SCART) was held in York, UK, on September 8, 2015, and was affiliated with SEFM 2015. The event originated from the fusion of the second edition of VERY\* and the first edition of SCART and was devised for bringing together researchers and practitioners from various areas related to service composition and verification.

In the near future we will be increasingly surrounded by a virtually infinite number of software/hardware services that can be composed to build new applications and systems. To cope with the changing requirements and emergent behaviors, system designers and software engineers need ad-hoc paradigms and technologies to define suitable reconfigurability mechanisms that allow systems to work correctly with respect to the new settlement. The production of application programming interfaces (APIs) is growing exponentially and some companies are accounting for billions of dollars in revenue per year via API links to their services. Moreover, the "Future Internet" (FI) is expected to lead to an ultra-large number of available services, hence increasing their number to billions of services in the near future. Finally, "cyber-physical systems" (CPS) will lead to the pervasive presence of computing and communication devices in everyday life. This situation radically changes the way systems will be produced and used: (a) systems are increasingly produced according to a certain goal and by integrating existing components; (b) the focus of system development is on integration of third-parties components that are only provided with an interface that exposes the available functionalities and, sometimes, the interaction protocol; (c) after deployment they must be able to cope with dynamically changing requirements and emergent behaviors caused by uncertainty in the surrounding environment. This calls for new integration paradigms and patterns, formal composition theories, integration architectures, as well as flexible and dynamic composition and verification mechanisms. Despite the great interest in software composition and self-\* systems, no common formal methods (FM) and software engineering (SE) approaches have been established yet. Developing FI applications and self-adaptive, self-reconfiguring, and self-organizing CPS encompasses a variety of formally grounded and practical aspects, ranging from modelling and analysis issues, to integration code synthesis, implementation and run-time management issues, model checking, and formal verification.

VERY\*SCART 2015 aimed at providing innovative contributions in the research and development of novel FM and SE approaches to the design, development, validation, and execution of FI applications and self\*-systems composed of available components. In particular, the workshop provided the opportunity to discuss how FI and CPS affect the traditional methods and tools, and how facing complexity in terms of scalability, heterogeneity, and dynamicity promotes the integration of FM within SE practices. The goal was to seek answers on how the rigorouslyness of FM assists engineers while designing, developing, validating, and operating systems that are built via correct-by-construction composition. The interplay between SE and FM is by

nature tightly intertwined with the “art” of service composition and formal verification. In order to make this art effective and elevate its maturity to the “readiness level” required for its adoption in practical contexts, novel formally grounded approaches, methods, and tools are required, especially when automation and correctness of the desired composition is of paramount importance.

The workshop was partially supported by the Italian Chapter of the EATCS, the INdAM – GNCS Programme 2015, the Italian PRIN Programme 2010–2011, and the H2020 EU project CHOReVOLUTION. The VERY\*SCART workshop received contributions covering topics related to: run-time adaptive service composition, run-time adaptation, composite cloud and internet applications, distributed coordination and adaptation, automatic choreography synthesis and enforcement, fuzzy description logics for service composition, architectural design for the FI, runtime verification of self-adaptive systems, automatic decision support, automated protocol synthesis, and spatio-temporal reasoning. Each paper was formally peer reviewed by at least three Program Committee members.

Many people contributed to the success of the VERY\*SCART 2015 workshop at SEFM 2015. We would like to acknowledge the SEFM workshop chair for his impeccable support and all Program Committee members for the timely delivery of reviews and constructive discussions given the very tight review schedule. Finally, we would like to thank the authors—without them the event simply would not exist.

November 2015

Marco Autili  
 Davide Bresolin  
 Marcello M. Bersani  
 Luca Ferrucci  
 Alfredo Goldman  
 Manuel Mazzara  
 Massimo Tivoli

## Program Committee

Luciano Baresi	Politecnico di Milano, Italy
Bert van Beek	Technical University of Eindhoven, The Netherlands
Maurice ter Beek	ISTI-CNR, Pisa, Italy
Carlo Bellettini	Università degli studi di Milano, Italy
Domenico Bianculli	University of Luxembourg, Luxembourg
Kelly Rosa Braghetto	University of São Paulo, Brazil
Laura Bocchi	University of Kent, UK
Radu Calinescu	University of York, UK
Mauro Caporuscio	Linnaeus University, Sweden
Miriam Capretz	Faculty of Western Engineering, Canada
Vincenzo Ciancia	ISTI-CNR, Pisa, Italy
Michele Ciavotta	Politecnico di Milano, Italy
Alexander Chichigin	Innopolis University, Russia
Ivica Crnkovic	Mälardalen University, Sweden

Guglielmo De Angelis	CNR-IASI/ISTI, Italy
Stéphane Demri	New York University and CNRS, France
Schahram Dustdar	Vienna University of Technology, Austria
Leo Freitas	Newcastle University, UK
Carlo Alberto Furia	ETH Zürich, Switzerland
Nikolaos Georgantas	Inria, Paris, France
Silvio Ghilardi	Università degli studi di Milano, Italy
Paola Inverardi	University of L'Aquila, Italy
Wojtek Jamroga	Polish Academy of Science, Poland
Alberto Lluch Lafuente	Technical University of Denmark, Denmark
Alexandr Naumchev	Innopolis University, Russia
Luca Pardini	Università di Pisa, Italy
Pascal Poizat	University of Paris Ouest, France
Gwen Salaün	Inria, Grenoble-Rhone-Alpes, France
César Sánchez	IMDEA Software Institute, Madrid, Spain
Nelson Souto Rosa	UFPE, Brazil



# **Keynote Speakers**

# **Automatic Generation and Execution of Test Scenarios for Camera-Based Driver Assistance Systems**

Joachim Wegener  
Berner and Mattner, Germany

## **Keynote Speaker of ATSE 2015**

Advanced driver-assistance systems (ADAS) are one of the key technologies for future innovations in cars and commercial vehicles. Increasingly, these systems take over safety-related functions, e.g., autonomous braking and autonomous steering. Typically, the systems are based on radar sensors or cameras continuously monitoring the vehicle environment. These sensor technologies cannot provide completely reliable detection rates of objects. Correspondingly, the quality assurance for advanced driver-assistance systems is of high importance. In this keynote an approach for automatically testing camera-based driver-assistance systems will be presented. Systematic test design and test scenario generation is based on the classification-tree method, environment simulation and image generation are supported using vehicle dynamics and virtual test drive simulations, such as veDyna or CarMaker, and test execution and test evaluation are performed using the test environment MESSINA. The proposed test solution enables functional testing of ADAS features in different testing phases, e.g., software in the loop for testing functional models in early development phases, hardware in the loop for testing the electronic control units (ECU) with the integrated software, as well as back-to-back tests between functional models and ECUs. Additionally, the high degree of automation enables efficient regression testing.

# **Testing Function and Time for Embedded Systems: From EAST-ADL to Code**

Cristina Seceleanu

Mälardalen University, Sweden

## **Keynote Speaker of ATSE 2015**

Architectural description languages, such as EAST-ADL, provide a comprehensive approach when describing complex automotive embedded systems, as standardized models that encapsulate structural, functional, and extra-functional information. The aim of architectural models is to enable the system's documentation, design, early verification, and even code implementation.

In this talk, we show how the use of such models can be extended further, by proposing a methodology that provides code verification based on the EAST-ADL architectural models. Our methodology relies on the automated model-based test-case generation for both functional and timing requirements of EAST-ADL models extended with timed automata semantics, and validation of system implementation by generating Python test scripts from the abstract test-cases. The Python scripts represent concrete test-cases that are then executed on the system implementation. The entire methodology is implemented as a tool chain, consisting of the ViTAL and Farkle tools, and is validated on an industrial prototype, namely, the Brake-by-Wire system.

# **Beating Error with Formal Methods**

Harold Thimbleby

Swansea University, Wales, UK

## **Keynote Speaker of HOFM 2015**

The reason we use formal methods is because humans make errors, and formal methods provides another point of view, namely, mathematical reasoning, which picks up issues that normal human thinking misses. Nevertheless, there are still residual errors: programmers may think some programming is so obvious that formal methods are not needed, programmers may correctly specify the wrong thing, and programmers may fail to specify all of the behavior of the program. Ironically, user error is rarely handled using formal methods.

We will give some concrete examples from medical systems. Error in hospitals is the third biggest killer (after heart disease and cancer), and inadequate health IT is not helping! It is time to make formal methods more accessible, to use formal methods more, and also to use additional methods to help manage error: formal methods are not sufficient.

# **Constraint Modelling and Solving for Data Mining**

Tias Guns

Declarative Languages and Artificial Intelligence Laboratory,  
KU Leuven, Belgium

## **Keynote Speaker of MoKMaSD 2015**

The use of constraints is prevalent in data mining, most often to express background knowledge or feedback from the user. Constraints are also a well-studied formalism for modelling and solving combinatorial problems, as exemplified by the constraint programming community. In recent years, such constraint technology is increasingly used in the field of (symbolic) data mining. Many challenges exist, however, at the level of modelling the problem (encodings, high-level and declarative languages, new primitives) as well as at the solving level (scalability, redundant constraints, search strategies). We review motivations and recent advances in the use of constraint programming for data-mining problems.

# Machine Learning Methods for Model Checking in Continuous Time Markov Chains

Guido Sanguinetti

School of Informatics, University of Edinburgh, UK

## Keynote Speaker of MoKMaSD 2015

Model checking of temporal properties on stochastic processes is one of the major success stories of formal modelling. The applicability of model checking methods has been greatly extended by the availability of randomized, statistical algorithms such as statistical model checking (SMC). Nevertheless, all of these methods require that a model is specified quantitatively, including a parametrization of the transition rates. Such prior knowledge is often unavailable in many important application fields such as systems biology. In this talk, I will discuss how an SMC procedure can be defined also for models with uncertain rates, by formalizing the task in Bayesian framework and placing a non-parametric prior distribution over how the satisfaction probability of a formula depends on the model parameters. I will introduce the notion of Gaussian process, and show how machine learning ideas can be used also for parameter synthesis and model design. I will also show how similar ideas can be used in more general reachability problems.

# **Automated Integration of Service-Oriented Software Systems**

Paola Inverardi

University of L'Aquila, L'Aquila, Italy

## **Keynote Speaker of VERY\*SCART 2015**

In the near future we will be surrounded by a virtually infinite number of software applications that provide services in the digital space. This situation radically changes the way software will be produced and used: (a) software is increasingly produced according to specific goals and by integrating existing software; (b) the focus of software production will be shifted toward reuse of third-party software, typically black-box, that is often provided without a machine-readable documentation. The evidence underlying this scenario is that the price to pay for this software availability is a lack of knowledge of the software itself, notably of its interaction behavior. A producer will operate with software artifacts that are not completely known in terms of their functional and non-functional characteristics. The general problem is therefore directed to the ability of interacting with the artifacts to the extent the goal is reached. This is not a trivial problem given the virtually infinite interaction protocols that can be defined at application level. Different software artifacts with heterogeneous interaction protocols may need to interoperate in order to reach the goal. This talk focuses on techniques and tools for integration code synthesis, which are able to deal with partial knowledge and automatically produce correct-by-construction service-oriented systems with respect to functional goals. The research approach we propose builds around two phases that elicit and integrate. The first concerns observation theories and techniques to elicit functional behavioral models of the interaction protocol of black-box services. The second deals with compositional theories and techniques to automatically synthesize appropriate integration means to compose the services together in order to realize a service choreography that satisfies the goal.

# **Formal Methods for Cyber-Physical systems**

Davide Bresolin

Alma Mater University of Bologna, Bologna, Italy

## **Keynote Speaker of VERY\*SCART 2015**

Cyber-physical systems are systems integrating computational devices in a physical environment. Examples of such systems are autonomous robotic systems, multi-agent and embedded systems, control systems for aerospace, and automotive, medical, and health-care systems. They all operate under strong safety, performance, reliability, and timing constraints, and are characterized by a dual nature: the computational kernels behave following discrete laws, while the physical components follow the continuous laws of nature. We review some of the challenges that the application of formal verification and design methodologies to cyber-physical systems poses to the academic community. In particular, the definition of suitable temporal logics and other requirement specification languages, and the development of efficient reachability analysis and other algorithmic manipulation techniques.



# Contents

## ATSE 2015

Learning-Based Testing of Distributed Microservice Architectures: Correctness and Fault Injection . . . . .	3
<i>Karl Meinke and Peter Nycander</i>	
The Synergy Between User Experience Design and Software Testing . . . . .	11
<i>A.P. van der Meer, R. Kherrazi, N. Noroozi, and A. Wierda</i>	
Combining Time and Concurrency in Model-Based Statistical Testing of Embedded Real-Time Systems . . . . .	22
<i>Daniel Homm, Jürgen Eckert, and Reinhard German</i>	

## HOFM 2015

Helping the Tester Get It Right: Towards Supporting Agile Combinatorial Test Design . . . . .	35
<i>Anna Zamansky and Eitan Farchi</i>	
Behavioral Types for Component-Based Development of Cyber-Physical Systems . . . . .	43
<i>Jan Olaf Blech and Peter Herrmann</i>	
Refactoring Proofs with Tactician . . . . .	53
<i>Mark Adams</i>	
Exploring the Role of Logic and Formal Methods in Information Systems Education . . . . .	68
<i>Anna Zamansky and Eitan Farchi</i>	
GuideForce: Type-Based Enforcement of Programming Guidelines . . . . .	75
<i>Serdar Erbatur and Martin Hofmann</i>	

## MoKMaSD 2015

Clustering Formulation Using Constraint Optimization. . . . .	93
<i>Valerio Grossi, Anna Monreale, Mirco Nanni, Dino Pedreschi, and Franco Turini</i>	
Towards a Boosted Route Planner Using Individual Mobility Models . . . . .	108
<i>Riccardo Guidotti and Paolo Cintia</i>	

Design of a Business-to-Government Information Sharing Architecture Using Business Rules . . . . .	124
<i>S�elinde van Engelenburg, Marijn Janssen, and Bram Klievink</i>	
Process Mining as a Modelling Tool: Beyond the Domain of Business Process Management . . . . .	139
<i>Antonio Cerone</i>	
On Integrating Social and Sensor Networks for Emergency Management . . . .	145
<i>Farshad Shams, Antonio Cerone, and Rocco De Nicola</i>	
Quantitative Modelling of Residential Smart Grids . . . . .	161
<i>Vashti Galpin</i>	
Attributed Probabilistic P Systems and Their Application to the Modelling of Social Interactions in Primates . . . . .	176
<i>Roberto Barbuti, Alessandro Bompadre, Pasquale Bove, Paolo Milazzo, and Giovanni Pardini</i>	
Probabilistic Modelling and Analysis of a Fish Population . . . . .	192
<i>Chiara Cini, Luca Tesei, Giuseppe Scarcella, Cesar A. Nieto Coria, and Emanuela Merelli</i>	
A Tool for the Modelling and Simulation of Ecological Systems Based on Grid Systems . . . . .	198
<i>Suryana Setiawan, Antonio Cerone, and Paolo Milazzo</i>	
<b>VERY*SCART 2015</b>	
Distributed Coordinated Adaptation of Cloud-Based Applications . . . . .	215
<i>Luciano Baresi, Sam Guinea, and Giovanni Quattrocchi</i>	
Fuzzy Description Logics for Component Selection in Software Design. . . . .	228
<i>Tommaso Di Noia, Marina Mongiello, and Umberto Straccia</i>	
Towards Adapting Choreography-Based Service Compositions Through Enterprise Integration Patterns . . . . .	240
<i>Amleto Di Salle, Francesco Gallo, and Alexander Perucci</i>	
An Experimental Evaluation on Runtime Verification of Self-adaptive Systems in the Presence of Uncertain Transition Probabilities . . . . .	253
<i>Kento Ogawa, Hiroyuki Nakagawa, and Tatsuhiro Tsuchiya</i>	
Towards Automatic Decision Support for Bike-Sharing System Design . . . . .	266
<i>Maurice H. ter Beek, Stefania Gnesi, Diego Latella, and Mieke Massink</i>	
Automated Synthesis of Protocol Converters with BALM-II. . . . .	281
<i>Giovanni Castagnetti, Matteo Piccolo, Tiziano Villa, Nina Yevtushenko, Robert Brayton, and Alan Mishchenko</i>	

An Experimental Spatio-Temporal Model Checker . . . . . 297  
*Vincenzo Ciancia, Gianluca Grilletti, Diego Latella, Michele Loreti,  
and Mieke Massink*

Dependable Composition of Software and Services in the Internet  
of Things: A Biological Approach. . . . . 312  
*Amleto Di Salle, Francesco Gallo, and Alexander Perucci*

**Author Index** . . . . . 325

**ATSE 2015**

# Learning-Based Testing of Distributed Microservice Architectures: Correctness and Fault Injection

Karl Meinke<sup>(✉)</sup> and Peter Nycander

KTH Royal Institute of Technology, 100 44 Stockholm, Sweden  
{karlm,peternyc}@kth.se

**Abstract.** We report on early results in a long term project to apply *learning-based testing* (LBT) to evaluate the *functional correctness of distributed systems* and their *robustness to injected faults*. We present a case study of a commercial product for counter-party credit risk implemented as a distributed microservice architecture. We describe the experimental set-up, as well as test results. From this experiment, we draw some conclusions about prospects for future research in learning-based testing.

**Keywords:** Automated test case generation · Fault injection · Learning-based testing · Microservice · Requirements testing · Robustness testing

## 1 Introduction

### 1.1 Overview

Functional testing of distributed systems presents one of the greatest challenges to any test automation tool. Significant execution problems exist, such as: system latency for individual test cases, global state reset, non-determinism leading to unrepeatable errors, and the existence of faults in communication infrastructure, to name but a few.

Learning-based testing [5] (LBT) is an emerging paradigm for fully automated black-box requirements testing. The basic idea of LBT is to combine machine learning with model checking, integrated in a feedback loop with the system under test (SUT). An LBT architecture iteratively refines and extends an initial model of the SUT by incremental learning, using test cases as learner queries. During this process, it model checks for violation of user requirements. Current LBT technology can construct and judge thousands of test cases per hour (depending on SUT latency), which is potentially useful both for testing complex distributed systems and for fault injection.

### 1.2 Problem Formulation

We consider the problem of applying LBT to *black-box testing the functional correctness of distributed systems* and *testing their robustness to injected faults*.

We describe an experiment to perform requirements testing and fault injection on a *distributed microservice architecture* for counter-party credit risk analysis known as triCalculate. This application is a commercial product developed by TriOptima AB for the Over-The-Counter (OTC) derivatives market. Our experiment had several goals, including an evaluation of:

1. ease of formal modeling of correctness and robustness requirements,
2. ease and efficiency of fault injection and test case tear-down in a distributed system through the use of LBT wrapper constructs
3. success in detecting SUT errors using low fidelity inferred models.

Furthermore, our experiments were made using an existing tool LBTest [6], which lacks any optimisation for distributed system testing. Thus an additional goal was: 4. to evaluate what architectural changes could be made to LBTest, that might improve its performance in this context. In Sect. 5 we make some suggestions for future tool improvement.

## 2 Background

### 2.1 Microservice Architectural Style

The microservice architectural style [4] implements a single application as a suite of many small services that communicate using a language agnostic mechanism such as HTTP. The style is a new approach to Service Oriented Architecture (SOA). The benefits of this style include: technology independence, resilience, scalability and ease of deployment. In triCalculate, network communication is event-based, using the open source message broker RabbitMQ.

### 2.2 Fault Injection

Fault injection (FI) has traditionally been deployed to evaluate the robustness of software in the presence of hardware faults (such as bit-flip errors). A classification in [9] includes: hardware implemented (HIFI), software (SIFI) and model implemented (MIFI). Our approach here is closest to SIFI, but LBT can also support MIFI, through its model inference. A characteristic of fault injection is the combinatorial growth of faults<sup>1</sup>, and large test suites are typically needed. The potential of LBT to rapidly generate test cases may therefore be beneficial in this context. Interesting high-level faults that were considered to inject into triCalculate included: restarting services, starting up several service instances, communication faults, and killing service instances.

---

<sup>1</sup> A *fault* is a triple consisting of a type, a location and a time [9].

### 2.3 Learning-Based Testing

For this experiment, we used LBTest version 1.3.2 [6] which is an LBT tool for testing reactive systems. The architecture of LBTest is illustrated in Fig. 1. It has previously been successfully used to test monolithic financial applications [3]. Reactive systems are modeled and learned as deterministic finite state machines, specified using propositional linear temporal logic (PLTL), and model checked with NuSMV [2]. LBTest has a modular architecture to support a variety of learning algorithms, model checkers and equivalence checkers.

Noteworthy in Fig. 1 is the *communication wrapper* around the SUT, which is responsible for communication and data translation. Wrappers are also responsible for test set-up and tear-down between individual test case executions. These activities can be problematic for distributed systems, requiring complex network management and reset actions. Furthermore, wrappers support the abstraction of infinite state systems into finite state models, through data partitioning. Using the same data abstraction principles, wrappers can also be used to support fault injection. However, to date, there has been no research published on LBT for robustness testing. A more general open question is how to extend LBT to different types of distributed systems, which is a long-term goal of the project [10].

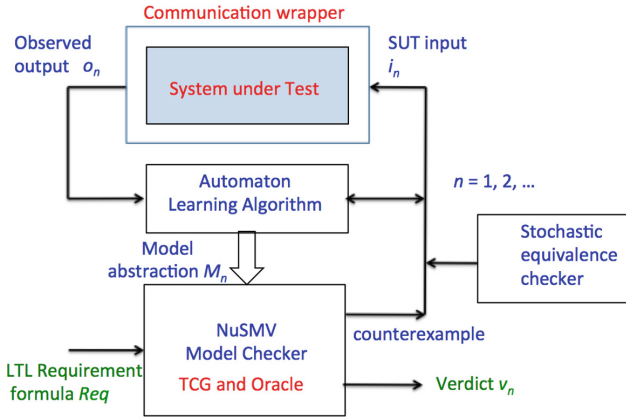
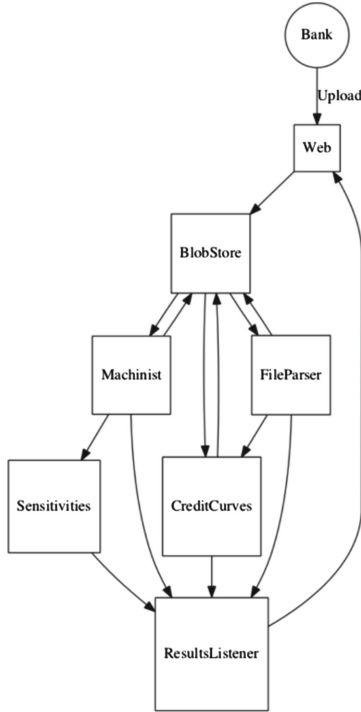


Fig. 1. LBTest architecture

## 3 Experimental Set-up

The SUT triCalculate calculates counter-party credit risks using distributed microservices to host different parts of the calculation. It consists of about 100 KLoC. Figure 2 illustrates the triCalculate architecture in terms of separate microservices and their intercommunication.



**Fig. 2.** triCalculate microservice architecture

The system has just one use case: *a user uploads files and eventually receives output.*

To manage service scheduling and concurrency, we took a grey-box testing approach, by opening up the RabbitMQ message broker to the LBTest wrapper. In particular, the wrapper made non-trivial use of the RabbitMQ API in order to efficiently and reliably tear down inter-service message queues after each test case execution. In [3] we have also shown that efficient tear-down is an important step to fully exploit the high test throughput of LBT methods.

The most relevant correctness property to be tested is that *the end-to-end calculation finishes in a successful way*, i.e. when a file is uploaded then sometime in the future results will be published. Such end-to-end requirements for microservices are quite difficult to test by traditional (i.e. non-formal, manual) methods. We also focused on a single fault scenario: *SUT restart*, to verify that triCalculate handles system crashes and recovery in a robust way. To control fault injection at each use case step, we introduced a single *meta-input variable* with symbolic input values `continue` and `restart`. These values were inputs to the wrapper (which executed the fault injection) rather than the SUT itself. This seems like a flexible and generic approach to SIFI.



The SUT state variables for each observation were obtained from database entries in the central storage unit. These state variables were returned by the wrapper as 6 symbolic output variables, each of which flagged the completion of one use case step (i.e. calculation step). Using these output variables, we could express each of the 6 *sunny day use case steps*, e.g. step 1:

```
G(session = session_none & input = continue ->
X(session = session_ok))
```

This sunny day use case model then had to be extended with alternative *rainy day* use case steps, to model SUT robustness under `restart`, e.g. step 1.b

```
G(session = session_none & input = restart ->
X(session = session_none))
```

To continuously monitor the internal SUT status, an additional output `warning` variable was used, specified by: `G(warning = warning_none)`

## 4 Test Results

In benchmarking LBT methods, our main arbiter of success is the capability to identify known errors (i.e. injected mutations) as well as previously unknown SUT errors within a reasonable time frame. Since LBT is a fully automated testing process, it is feasible to run a tool such as LBTest for several hours.

A secondary measure of testing success is the number of test cases (queries) executed and the size of the inferred state machine model. However, LBT methods always produce a finite abstraction of the SUT, which is usually an infinite state system. It can be difficult to predict the size of this abstraction *a priori*. Therefore besides its absolute size, the degree of convergence of the learned model is another important test performance indicator. Convergence is measured indirectly through stochastic equivalence checking, and can be regarded as a *black-box coverage model*.

Note that test suite size and inferred model size are both heavily influenced by the choice of learning algorithm. Since LBT is still an emerging technique, it is instructive to experiment with different learning algorithms and compare their performance on the same SUT.

To begin our testing experiment, ad-hoc SUT mutations were injected and successfully detected. These helped to confirm the correct construction of both the user requirements and the wrapper.

Testing was then performed on the unmodified SUT using two different automata learning algorithms: IKL [5] and L\*Mealy [7]. L\*Mealy is a straightforward generalisation of Angluin’s well-known L\* algorithm [1] to deal with multi-valued output alphabets. The two algorithms have similar asymptotic complexity properties. IKL is an incremental learning algorithm that produces hypothesis automata with much greater frequency than L\*Mealy. It is well adapted to testing systems that are too big to be completely learned. The heuristic used for stochastic equivalence checking was *first difference*, which is the only heuristic that is practically feasible for SUTs with long test case latency.

Using IKL, a testing session was conducted for 4 h and 3 min. During this time LBTest made a total of 139 queries leading to 7 warnings. A 24 state model was learned after 16 iterations. Using L\*Mealy, a testing session was conducted for 7 h and 32 min. During that time LBTest made a total of 280 queries leading to 2 warnings. A 38 state model was produced after 3 iterations.

In this case study, IKL is slightly more efficient requiring 5.8 queries per learned state versus L\*Mealy’s 7.4 queries. Furthermore, IKL produces about 5 times as many hypothesis automata as L\*Mealy. Thus, IKL allows the model checker a significantly greater role in predicting errors through inductive inference principles.

The final model produced by L\*Mealy was slightly better converged than the final model produced by IKL, as measured by stochastic equivalence checking with the SUT. However, this improvement was marginal given that twice as many queries were used. In fact this small difference suggests that the models produced by both learning algorithms were still highly non-converged.

In this case study, the use-case structure of our PLTL user requirements allowed us to measure *requirements coverage* in a precise way using graph coverage. This approach is similar to the requirement coverage model of [11]. Here the results were quite positive. Every simple path through the use case is covered by at least one path through the final learned model, in both the IKL and L\*Mealy inferred models.

The latency time of the triCalculate SUT was relatively long, on average 1.6 min per test case. Therefore, the inferred models have quite low fidelity (convergence), based on a small set of samples. This latency could be attributed primarily to the computationally intensive SUT, and not to the overhead of machine learning, model checking or message broker communication.

## 5 Conclusions and Future Research

Regarding experimental goal 1, we confirmed the viability of modeling correctness and robustness requirements in a single use-case model, consisting of sunny and rainy day scenarios. The user requirements were captured and formalised using propositional linear temporal logic by a test engineer with no previous experience of using temporal logic.

Regarding goal 2, fault injection was achieved by the use of metavariables to describe environmental behaviour. This approach is simple and supports fault modeling at a high-level of abstraction using symbolic data types. The actual semantics of fault injection (which can be complex) was efficiently implemented in wrapper constructions. This overall approach to SIFI and robustness testing seems generic, powerful and flexible. However, the onus is on the test engineer to make correct and perhaps complex wrapper constructions.

Regarding goal 3, the performance of LBTest compares favourably with current manual techniques to perform end-to-end testing. LBTest was able to discover multiple failed test cases within a reasonable time frame. Although the inferred models had low convergence, at least a high level of user requirements coverage could be achieved.

With respect to goal 4, the non-determinism exhibited by triCalculate showed that new approaches are necessary to improve the inference and modeling of distributed systems. Adapting automata learning algorithms to non-deterministic SUTs is technically straightforward. However, problems may arise with convergence of learning, since some non-deterministic behaviours may have very low probability of occurrence (unrepeatable errors).

In our case study, the problem of poorly converged models was primarily due to high test case latency, which arose from a computationally intensive SUT rather than slow network communication. The general situation for other microservice architectures merits further investigation. However, an important conclusion is that improving the coverage of distributed systems (as measured by learned state space size) is one of the most important and challenging problems for learning-based testing.

One possible approach to improving coverage might be to distribute learning across the network. We could try to infer a set of local automaton models that can be combined by asynchronous parallel composition into a single global model. This approach might be well-suited to microservice architectures, since using microservers should reduce the difficulty of learning the individual local models. Another approach might be to try to re-use inferred models from unit and integration testing within system testing. These two approaches may even be mutually compatible.

An interesting recent evaluation of machine learning in the related context of fault prediction is [8], which takes a less optimistic perspective. It would be interesting to compare the static classifier-based prediction methods considered there, with our own execution-based inductive inference methods.

**Acknowledgement.** K. Meinke wishes to thank VINNOVA and Scania AB for financial support of this research within [10], and R. Svenningsson for valuable discussions on fault injection. P. Nycander wishes to thank TriOptima AB for hosting his Masters thesis research.

## References

1. Angluin, D.: Learning regular sets from queries and counterexamples. *Inf. Comput.* **75**, 87–106 (1987)
2. Cimatti, A., Clarke, E.M., Giunchiglia, E., Giunchiglia, F., Pistore, M., Roveri, M., Sebastiani, R., Tacchella, A.: NuSMV 2: an OpenSource tool for symbolic model checking. In: *Proceedings of International Conference on Computer-Aided Verification (CAV 2002)* (2002)
3. Feng, L., Lundmark, S., Meinke, K., Niu, F., Sindhu, M.A., Wong, P.Y.H.: Case studies in learning-based testing. In: Yenigün, H., Yilmaz, C., Ulrich, A. (eds.) *ICTSS 2013*. LNCS, vol. 8254, pp. 164–179. Springer, Heidelberg (2013)
4. Fowler, M.: *Microservices* (2014). <http://martinfowler.com/articles/microservices.html>
5. Meinke, K., Sindhu, M.A.: Incremental learning-based testing for reactive systems. In: Gogolla, M., Wolff, B. (eds.) *TAP 2011*. LNCS, vol. 6706, pp. 134–151. Springer, Heidelberg (2011)

6. Meinke, K., Sindhu, M.: LBTest: a learning-based testing tool for reactive systems. In: Proceedings of Sixth International IEEE Conference on Software Testing, Verification and Validation (ICST 2013) (2013)
7. Niese, O.: An Integrated Approach to Testing Complex Systems. Ph.D. thesis, University of Dortmund (2003)
8. Shepperd, M., Bowes, D., Hall, T.: Researcher bias: the use of machine learning in software defect prediction. *IEEE Trans. Software Eng.* **40**(6), 603–616 (2014)
9. Svenningsson, R.: Model Implemented Fault-injection for Robustness Assessment. Licentiate Thesis, KTH, Stockholm (2011)
10. VINNOVA FFI project, VIRTUES (Virtualized Embedded Systems for Testing and Development). <http://www.csc.kth.se/~karlm/virtues/>
11. Whalen, M., Rajan, A., Heimdahl, M., Miller, S.: Coverage metrics for requirements-based testing. In: Proceedings of International Symposium on Software Testing and Analysis, ISSA 2006, pp 25–35. ACM Press (2006)

# The Synergy Between User Experience Design and Software Testing

A.P. van der Meer, R. Kherrazi, N. Noroozi<sup>(✉)</sup>, and A. Wierda

Nspyre B.V., Eindhoven, The Netherlands  
neda.noroozi@nspyre.nl

**Abstract.** Formal methods and testing are two important approaches that assist in the development of high quality software. Model-based testing (MBT) is a systematic approach to testing where using formal models enables automatic generation of test cases and test oracle. Although the results of applying MBT in practice are promising, creating formal models is an obstacle for wide-spread use of MBT in industry. In this paper we address how the cooperation between testers and user experience designers can help with the overall challenge of applying MBT. We present a test automation approach based on Task Models and Microsoft Spec Explorer model-based testing tool to improve software testing. Task Model is a formal model to specify the high-level interaction between the user and the graphical user interface (GUI). We developed a tool, called UXSpec, to convert Task Models to the input models of Spec Explorer, allowing us to do functional testing with little modeling effort, due to usage of already existing models. We demonstrate this by applying our approach to a case study.

**Keywords:** Model-based testing · Spec Explorer · ConcurTaskTrees

## 1 Introduction

The speed of software development increases steadily. As a consequence the challenges that each discipline faces increase. For User eXperience (UX) designers a challenge is to ensure that UX designs get implemented correctly, including aspects of performance, exception handling, etc. For software testers the challenge is to build a full understanding of what the correct system is in a limited amount of time. On a high level the solution we have found is to reuse the requirements efforts from UX as input for model-based testing to improve overall development speed and testing quality. Using this approach allows us to benefit from all powerful features of MBT tools without facing issues related to creation of complex test models.

UX work contributes to requirements creation by adding the user perspective; who is the user, what are the tasks that (s)he will use the system for? A tool to create this understanding is making a task analysis. The widely used notation for this is Concurtasktree (CTT) [10]. We found that we can reuse CTT models,

which capture the UI requirements, as basis for model-based testing of the final implementation. For this we have developed the UXSpec tool which converts CTT models to the test models used by Microsoft Spec Explorer tool [1].

Furthermore we have found that the discussions between testers and UX-ers is useful to better coordinate usability testing effort, e.g. what are the most critical scenarios, and which user tasks should be included in a usability test. This qualitative testing complements the testing efforts aimed at coverage and completeness and improves the overall system quality.

In this paper, we first introduce Task Models and Spec Explorer in Sect. 2. We describe some technical details of the implementation of our approach in Sect. 3. In Sect. 4, we report on a case study to demonstrate the feasibility of our tool chain and present some empirical results. We discuss related work in Sect. 5, and finally we write the conclusions in Sect. 6.

## 2 Preliminaries

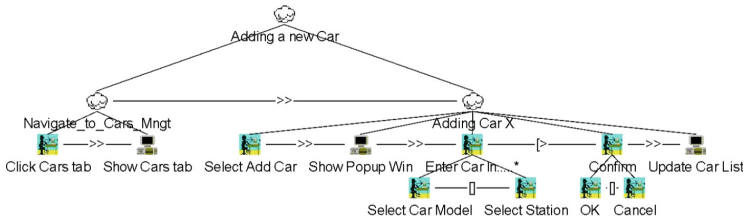
### 2.1 Task Models

Task Models are useful in designing and developing interactive systems. They describe the logical activities (tasks) that have to be carried out in order to reach the users goals [11]. A widely used notation for Task Models is ConcurTaskTree (CTT) [10], which we use in this paper. Four different types of tasks which are supported by CTT models are as follows:

- *interaction task* represent events initiated by a user of the systems
- *application tasks* represent system responses
- *user tasks* represent decision points on the user side
- *abstract tasks* which are further subdivided in other tasks

Connections between tasks are annotated with the temporal operators that describes the dependencies between tasks. Providing a rich set of operators, a set of temporal relationships between tasks such as enabling, disabling, choice, and synchronization are formally defined. A part of the CTT model of a car reservation which will be used as the case study in Sect. 4 is shown in Fig. 1. The sub-tree depicted in Fig. 1 shows the task model for adding a new car to the car reservation system. This CTT model consists of six *interaction* tasks including navigation to the desired interface and providing the required data, and three *application* tasks showing the responses of the system. Semantic of the model is defined by the possible exploration of the model which is governed by the temporal operators. The task of adding a car is decomposed to two interaction tasks: first executing task ‘Navigate to Cars Mng’ and then performing task ‘Add Car X’ (enabling operator:  $\gg$ ). To navigate to the ‘Cars’ tab, the cars tab must first be selected (interaction task ‘Click Cars tab’) and then the system shows the Cars tab (application task ‘Show Cars tab’). A new car is now added by selecting the add car option (interaction task ‘Select Add Car’) which yields a pop-up window is opened (application task ‘Show Pop-up Win’). At this

point the user can enter the information of the new car by entering two possible data (choice operator:  $\square$ ): Select the car model or Select the station. Task ‘Enter Car Information’ is an iterative task (iterative operator:  $*$ ), meaning that it can be repeatedly executed. Once, the user confirm the entered information, (s)he cannot change those information anymore (disabling operator:  $[>]$ ). Eventually, the updated list of the cars is shown to the user (application task: Update Car List). For a complete list of the operators and their meaning see [11].



**Fig. 1.** Fragment of a car reservation system ConcurTaskTree model (in MARIAE tool)

Although it is not visible in Fig. 1, the task model is enriched with some constraints to ensure the availability of certain actions at different states of the system. These constraints are expressed as pre/post conditions of an atomic task. For example, the task named select station in Fig. 1 has a post-condition which guarantees that a station is always selected as one of the requirements of the system. To this end, a string variable named station with the initial value empty is defined in the task. The post-condition, then, is presented by the expression  $station \neq \text{empty}$  where station variable shows the name of selected station. A similar post-condition is added to task ‘select car model’ to ensure that the mandatory field of car model is always set to a non-empty value.

## 2.2 Spec Explorer

Spec Explorer [1] is an extension to Microsoft Visual Studio intended to provide support for MBT. In Spec Explorer, we define a model that describes the expected behavior of the system under test together with a configuration file in which we describe parameters of test cases. The modeling language used in Spec Explorer is an extension to C# with modeling annotation and construction like pre/post condition, final states, variables, etc. The model of a system described by the Spec Explorer modeling language is a possibly infinite state transition system in which the states comprise the possible states of the system variables and transitions between the states are modeled by methods annotated as *Action*. There are two different types of actions in Spec Explorer: controllable and observable actions. Controllable actions defines actions that are under the control of the user (inputs of the system under test), and observable actions describes the (asynchronous) responses of the system. Once the model of a system is complete

in Spec Explorer, different predefined test strategies can be applied to generate test cases, which can be executed directly on the implemented system. Each path in the state space of the defined state transition system represents a possible test, a sequence of controllable and observable actions that the system under test has to be able to follow. More information on Spec Explorer can be found in [9].

### 3 Using Task Models in Model-Based Testing

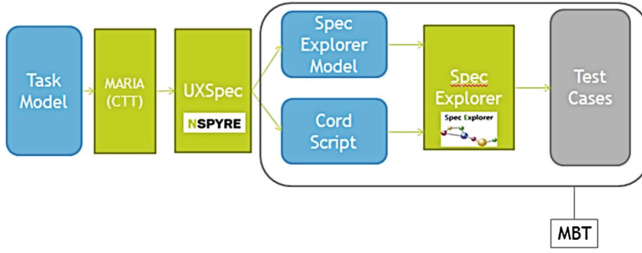
In order to generate tests for a system in Spec Explorer, we need a model that describes the expected behavior and a script file that defines test scenarios, i.e. the kind of tests to be executed. Creating behavioral models is a non-trivial process which typically requires considerable time and efforts in MBT approaches. When we design a system by using CTT models, we already have a model that describes the desired behavior of the system under test. This suggests that if we reuse this model, we will make testing with Spec Explorer easier and cheaper.

Since CTT models cannot be directly used in Spec Explorer, we need to create a Spec Explorer model based on a given Task model in a way that does not affect the semantic of the given CTT model. From a global perspective, we observe that both CTT model and Spec Explorer are fundamentally state-transition-system-based formalisms. This means a system at all times has a well-defined state, which changes in response to triggers received from the environment. In particular, this implies that we can consider a Spec Explorer model compliant to a CTT model if the state machines involved are equivalent in behavior. More precise, we need to make sure that the tests that are constructed by Spec Explorer contain all possible sequences of event triggers that are legal in the CTT model state machine. On the other hand, illegal events should never occur in tests, because the behavior of the component is undefined in such cases, which means that the test can never be failed or passed.

The state transition system of a CTT model can be extracted by identifying its states and the transitions between them. This can be done by simulating the execution of task models: each set of enabled tasks at a same time, called Presentation Task Sets (PTS), in the execution of the task model represents a state of the model, and the relation between tasks defines transitions between states. This step is automatically carried out in MARIAE tool.

In Spec Explorer, the model of the system under test is defined in one or more C# classes with methods which describe events that the system responds to or produces in the response to another event. By analyzing the effects of the methods on the state of the system, Spec Explorer can identify states and the transitions between them. Thus, to reconstruct a CTT model state machine in a Spec Explorer model, we have to create model classes that implement all possible events of the CTT model, in such a way that the resulting state space matches the one in the CTT model. The former is simply achieved by creating a method (defined as a controllable action) for every *interaction* task in CTT





**Fig. 2.** UXSpec workflow and tool chain

models. We achieve the latter by constructing the model, based on an explicit state machine pattern, thus ensuring that all states are explicitly present in the model. In the same way, all events are explicitly covered in methods, ensuring the full coverage of all transitions of CTT models. Furthermore, application of tasks in the task model are translated as probe methods in the Spec Explorer model to check if the effect of user interaction is as expected. The generated Spec Explorer model can be later enriched by further details to each state or/and by defining data constraints in the model which are not available in CTT models. To do the translation from CTT models to Spec Explorer models automatically, we developed a tool, which is named UXSpec. The general work flow around this tool chain is shown in Fig. 2. The UXSpec tool takes the CTT models as input and combines them with some configuration data that are not presented in CTT models, for example how the user interface implementation can be started and stopped. The output of the tool is a Spec Explorer model that can then be used to generate test cases.

UXSpec uses a model transformation in QVTo [5,8], an Eclipse Modeling Framework (EMF) implementation of QVT Operational, to carry out the transformation of task models in CTT notation to models used by Spec Explorer. Because QVTo is based on EMF [3], we translate CTT models into EMF model first. For this, we used an existing tool based on the XML schema provided by the HIIS Laboratory for MARIAE [15], describing the structure of CTT models. This tool was developed by Nspyre as part of an earlier project. In the same project, a transformation was developed that abstracts from format-specific features of CTT models to a generic, more abstract representation. We use this representation as a basis for further processing.

The next step is to generate the C# model file and the Cord Script file, which is implemented by means of two QVTo transformations. The first one generates an EMF C# model based on a C# meta-model created by the MoDisco [4] project. The second generates an EMF Cord Script model based on a Cord Script meta-model of our own design. Because these are both in EMF format, we then have to use templates, in our case based on the Acceleo [2] template engine, to create the textual representations that can be used by Spec Explorer. These templates are generic, in the sense that they can be used for all EMF C# and Cord Script models that use our meta-models.

## 4 Case Study

In this section we demonstrate the feasibility of our approach with an example. The application being used for this purpose is a car reservation system which is based on a large industrial system co-developed by Nspyre. The car reservation system in this section is developed for car rental agencies for booking and managing rental cars. It consists of various user interfaces and each of them has several GUI components. We focus on the functionality of adding a new car to the system. A car is added to the system by identifying its model and the station to which it is assigned as the mandatory fields and its type as an optional field.

To evaluate our approach, we tested the above functionality of the car reservation system with two different techniques: Traditional test automation approach (capture/replay) and (conventional) model-based testing. Capture/replay tools are widely adopted for automatic test execution in which a test is designed and executed for the first time by a human and then a test executor executes scripted tests that record the human interactions with the system under test. For this purpose, we designed test scenarios by using decision table technique. Afterwards, scripted tests of each logical test case are generated by hand. Regarding the system under test was a windows application in this case study, we developed an automatic test executor based on UI Automation framework [12] which translated abstract user interactions of scripted tests to actual UI events, and vice versa.

In the second approach, test cases are automatically generated in Spec Explorer from test models that are manually created. As the common practice in MBT, in order to execute the generated test cases on the system under test, we developed an adapter based on UI Automation framework to connect to the system under test. Finally, we test the system by reusing the CTT model depicted in Fig. 1 and using UXSpec tool to automatically translate the CTT model to a test model in Spec Explorer.

In the remainder of this section, we first explain all steps of the process of automatic generation of test cases in our approach by using UXSpec tool. Afterwards, we compare our approach with the other two approaches.

### 4.1 Testing with UXSpec Tool

Having the task model, the state transition system of the CTT model is automatically generated by MARIAE tool. UXSpec tool starts with the generated state transition system. However, all information of CTT models like pre/post condition of tasks, is not available in the transition system. Therefore, UXSpec use the actual CTT model to extract the necessary information. The output of UXSpec is the preliminary Spec Explorer model in C# format together with a script file, which represents the intended behavior of the system. Figure 3 shows the graphical representation of the Spec Explorer model generated by UXSpec from the task model in Fig. 1. The gray state shows the initial state and each arc shows an enabled interaction task at each state which is translated to a method in the C# model. For example consider the initial state in the CTT model at

which the user can only click on Cars tab (interaction task ‘click cars tab’). This interaction task translated in Spec Explorer model as controllable method ‘P2\_Click\_Cars\_Tab()’. Moreover, to check the effect of tasks (post condition), probe methods are created in the generated Spec Explorer model. For instance, the interaction task ‘select station’ in Fig. 1 has a post-condition which guarantees that a station is always selected. This post-condition is defined in the CTT model by the expression “station != empty” where “station” variable shows the name of selected station. This post-condition is checked by probe method ‘get-Station()’ in the Spec Explorer model. The generated model can be later enriched by further details to each state or/and by defining data constraints in C# model which are not available in task models. For instance, we restricted the set of possible stations and car models to two certain sets of values. These modifications took a small amount of time, i.e. less than half a man-hour. Finally from the modified model, test cases are automatically generated by Spec Explorer. To enable automatic execution of generated test cases over the system under test, we reused the developed adapter from the previous model-based testing project.

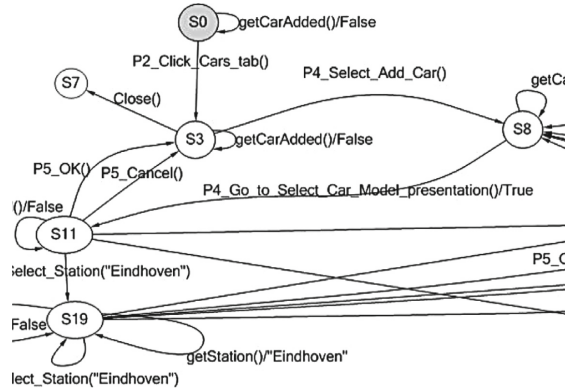


Fig. 3. A part of generated Spec Explorer model of the car reservation system

## 4.2 Results

Based on the case study, we have drawn several conclusions on the approaches discussed in this paper

**Modeling.** Task Models concentrate on activities that a user intends to perform over an interactive application together with temporal relationships between them. Task Models are created in early phases of software development by user experience designer. In cooperation with software architecture, these models are translated into detailed design in later phases. C# models created in Spec Explorer are intended to use for test case generation. They model some parts of

the functionalities of the system under test that are relevant in testing. These models are created by testers to be used only for testing the actual implementation. In our approach, UXSpec reuses existing CTT models to automatically generate C# models in Spec Explorer.

**Technique.** Using the formally defined semantics of temporal relationships between tasks, a designer is able to simulate the flow of the activities of a CTT model before designing the user interface to check if users goal is supported. In contrast, conventional MBT creates tests based on manually created models. UXSpec creates automatically primarily test models from existing CTT models.

**Effort.** Task Models are constructed by user experience designer by focusing on user interactions and abstracting from design and implementation details. In contrast, test models are created by testers to cover only those parts of the functionalities of system which are relevant to testing. UXSpec creates test models by reusing existing task models. Thus, it decreases effort needed for testing.

**Requirement Coverage of Generated Test Cases.** In order to give an indication of the efficiency of our approach; we look at the number of test cases generated by each approach in a same amount of time. In the first approach where the manual scripted tests are used, the number of generated test cases in one hour is 2. This number of test cases partially covered the required test scenarios. In the second approach, the test model of the system was manually developed in Spec Explorer in one hour. Afterwards, 13 test cases were generated in some milliseconds, which cover all possible scenarios. In our approach, the creation of the task model has taken about half an hour. Reusing the task model, UXSpec automatically generates a preliminary model in Spec Explorer. After customizing the generated model, 13 test cases were generated which have a same coverage as those generated in the pure MBT approach. Therefore, by using UXSpec we can reduce time and cost of modeling while preserving the requirement coverage.

## 5 Related Work

In this paper, we focus on improving and accelerating the use of model-based testing for applications interacted via their user interfaces. There are several works on using models in testing user interfaces [13–18]. Challenges of using model-based testing in GUI testing are comprehensively studied in [16, 18]. A pure model-based testing approach is presented in [16]. Similarly, [15] reported on a model-based GUI testing approach in which Uppaal models are used as test models. In contrast to our approach, in both approaches in [15, 16] test models need to be manually developed. Creating test models is not a trivial process and most of the time is time-consuming, particularly when the system under test is large. To overcome this problem in [17], a model-based tool, GUITAR, is developed which automatically creates test models as an event-flow graphs by extracting structures and API calls. This approach can be used only in the final phases of software development, when the system under test is developed.

However, using CTT models enables us to generate test cases in the early phase even before graphical user interfaces of the system with their implementation details are designed and implemented. Analogous to our approach in [13, 14], CTT models are used as input for generating test models of graphical user interfaces. Instead of testing the desired behaviors modeled in task models, the focus of presented approach in [14] is on testing unexpected and undefined behavior by generating task mutations based on a classification of the user errors. The most similar work to the approach presented in this paper is [13], in which customized `concreteTaskTree` models are automatically converted to C# models in Spec Explorer. CTT models used in [13] are enriched by some implementation details that are used later in development of the adapter and for checking the effect of interaction tasks. Including some implementation details in test models restricted the approach in [13] to testing Windows Form applications. Moreover, in [13] only ‘interaction’ tasks are taken in the transformation. But by using standard CTT models, our approach has general applications and it is not limited to a specific type of application and platform. Furthermore, instead of customizing CTT models with some data tag to express the effect of interaction tasks, the approach taken in [13], we use the post-conditions defined for atomic tasks. The post-conditions are then translated as the probe method in Spec Explorer models. Therefore, in contrast with [13], no customization of CTT models is needed in our approach.

## 6 Conclusions

In this paper, we described an approach to automatic testing that is based on combining Task models and model-based testing. We showed that creating the models needed for MBT could be done in a way that builds further on material that is already created earlier in the development process. This reduces rework, improves the understanding for all people involved in the system development and most importantly increases the total software development speed. Task models become shared models that are understood by all disciplines in the project from user experience designers to system architects to developers to testers. Besides improving the communications in software development team, the consistency between requirements defined in the early phases of software development with test cases executed in the later phases is improved as well.

To use Task models in model-based testing, we develop UXSpec tool which via some QVTo transformers translates task models from the MARIAE tool to test models used by Microsoft Spec Explorer tool. This enabled us to automatically create test cases based on a Task models. Although creating this custom connection takes a little bit of time we found that it significantly reduced time and efforts needed for test generation and maintenance. Moreover, the QVTo transformers of UXSpec support transition-based models as the target model. Therefore, the approach we demonstrated in this paper is not limited to a specific test tool and can be used with other MBT tools, such as NModel [6] or PyModel [7] as well.

To demonstrate the feasibility of our approach, we tested a simple part of a car reservation system with different testing techniques: the traditional test automation approach, and the conventional model-based testing in which test models are manually created. Although the scope of our case study is small, our obtained results are promising and showed reduction in time and effort needed for creating models in model-based testing. However, to strengthen our results and to have a better evaluation, we consider applying our approach on a real large industrial system in future.

## References

1. Veanes, M., Campbell, C., Grieskamp, W., Schulte, W., Tillmann, N., Nachmanson, L.: Model-based testing of object-oriented reactive systems with Spec explorer. In: Hierons, R.M., Bowen, J.P., Harman, M. (eds.) FORTEST. LNCS, vol. 4949, pp. 39–76. Springer, Heidelberg (2008). doi:[10.1007/978-3-540-78917-8\\_2](https://doi.org/10.1007/978-3-540-78917-8_2)
2. The Eclipse Foundation: Acceleo. <http://www.eclipse.org/acceleo/>
3. The Eclipse Foundation: Eclipse Modeling Framework Project (EMF). <http://www.eclipse.org/modeling/emf/>
4. The Eclipse Foundation: MoDisco Homepage. <http://www.eclipse.org/MoDisco/>
5. The Eclipse Foundation: QVTo. <http://wiki.eclipse.org/QVTo>
6. Microsoft: NModel. <https://nmodel.codeplex.com/>
7. Jacky, J: PyModel: model-based testing in Python. In: Proceedings of the 10th Python in Science Conference, pp. 43–48 (2011)
8. Steinberg, D., Budinsky, F., Paternostro, M., Merks, E.: EMF: Eclipse Modeling Frame-Work 2.0, 2nd edn. Addison-Wesley Professional, Reading (2009)
9. Model-based Testing with SpecExplorer. <http://research.microsoft.com/en-us/projects/specexplorer>
10. Manca, M., Patern, F., Santoro, C., Spano, L.D.: Considering task pre-conditions in model-based user interface design and generation. In: Symposium on Engineering Interactive Computing Systems, pp. 149–154. ACM (2014)
11. Patern, F., Santoro, C., Spano, L.D., Raggett, D.: MBUI - Task Models, W3C Working Group Note 08 April 2014
12. Windows Automation API: UI Automation. [http://msdn.microsoft.com/enus/library/windows/desktop/ee684009\(v=vs.85\).aspx](http://msdn.microsoft.com/enus/library/windows/desktop/ee684009(v=vs.85).aspx)
13. Silva, J.L., Campos, J.C., Paiva, A.C.R.: Model-based user interface testing with Spec explorer and ConcurTaskTrees. *Electron. Notes Theor. Comput. Sci.* **208**, 77–93 (2008). doi:[10.1016/j.entcs.2008.03.108](https://doi.org/10.1016/j.entcs.2008.03.108)
14. Barbosa, A., Paiva, A.C.R., Campos, J.C.: Test case generation from mutated task models. In : Proceedings of the 3rd ACM SIGCHI Symposium on Engineering Interactive Computing Systems (EICS 2011), pp. 175–184. ACM (2011). doi:[10.1145/1996461.1996516](https://doi.org/10.1145/1996461.1996516)
15. Hjort, U.H., Illum, J., Larsen, K.G., Petersen, M.A., Skou, A.: Model-based GUI testing using UPPAAL at Novo Nordisk. In: Cavalcanti, A., Dams, D.R. (eds.) FM 2009. LNCS, vol. 5850, pp. 814–818. Springer, Heidelberg (2009). doi:[10.1007/978-3-642-05089-3\\_53](https://doi.org/10.1007/978-3-642-05089-3_53)
16. Paiva, A.C.R.: Automated Specification-based Testing of Graphical User Interfaces, Ph.D. thesis, Faculty of Engineering, Porto University, Porto, Portugal (1997)

17. Nguyen, B., Robbins, B., Banerjee, I., Memon, A.: GUITAR: an innovative tool for AU-tomated testing of GUI-driven software. *Autom. Softw. Eng.* **21**, 65–105 (2013). doi:[10.1007/s10515-013-0128-9](https://doi.org/10.1007/s10515-013-0128-9)
18. Alsmadi, I., Samarah, S., Saifan, A., AL Zamil, M.G.: Automatic model based methods to improve test effectiveness. *Univ. J. Comput. Sci. Eng. Technol.* **1**(1), 41–49 (2010)

# Combining Time and Concurrency in Model-Based Statistical Testing of Embedded Real-Time Systems

Daniel Homm<sup>(✉)</sup>, Jürgen Eckert, and Reinhard German

Department of Computer Science 7, University Erlangen-Nuremberg,  
Martensstr. 3, 91058 Erlangen, Germany

{daniel.homm,juergen.eckert,reinhard.german}@fau.de

**Abstract.** Timed usage models (TUMs) represent a model-based statistical approach for system testing of real-time embedded systems. They enable an automatic test case generation and the calculation of parameters that aid the test process. However, a classical TUM only supports sequential uses of the system under test (SUT). It is not capable of dealing with concurrency, which is required for state of the art real-time embedded systems. Therefore, we introduce TUMs with parallel regions. They also allow automatic test case generation, which is carried out similarly to classical TUMs. But, the semi-Markov process (SMP) that is usually used for analysis is not suitable here. We apply Markov renewal theory and define an SMP with parallel regions, which is used to calculate parameters. We validated our analytical approach by simulations.

**Keywords:** System testing · Timed usage models · Concurrency

## 1 Introduction

Timed usage models (TUMs) are applied in the field of system testing for model-based statistical testing of real-time embedded systems [10, 12]. A TUM represents an extension of a Markov chain usage model (MCUM) [13] by time aspects. It specifies possible uses of the system under test (SUT) in a notation similar to state machines. But, it also considers time dependencies related to the use of the SUT. TUMs enable an automatic generation of test cases [12] and the calculation of parameters that aid the test process [11], e.g., they help to decide when to stop testing [9]. Test case generation is achieved by a random walk through the model. Parameters are calculated by mapping the TUM to a semi-Markov process (SMP), which is analyzed subsequently.

Modeling uses in a TUM is restricted to sequences of stimuli (events, interrupts, etc.). Concurrent streams of use are not supported. However, the rising complexity of embedded systems leads to an increase of concurrent aspects. For example, customer requirements in the automotive domain enforce the development of more sophisticated systems [3]. Neglecting concurrent uses during the test of the SUT increases the risk of failures being undetected prior to the release.



In this paper we extend the concept of TUMs by parallel regions. Thus, it is possible to handle both, sequential and concurrent uses of the SUT, with their respective time dependencies and the model stays similar to state machines. Test cases still can be generated automatically from the model by a random walk. But the calculation of parameters becomes a non-trivial task: the SMP used to analyze a TUM assumes a sequential execution of the process in time, which is no longer given due to the introduction of concurrency. Therefore, we apply Markov renewal theory and define an SMP with parallel regions. The calculation of parameters for a TUM with parallel regions is carried out by an analysis of a respective SMP with parallel regions.

## 2 Related Work

Research in the area of model-based testing already tackled the issue of combining time and concurrency aspects of the SUT [2]. There are also commercial tools available, e.g., TPT<sup>1</sup>. However, the results are not transferable to model-based statistical testing.

Model-based statistical testing aims at testing the expected use of the SUT. Test cases are sampled and executed from the set of possible test cases. Statistical methods are used to draw inferences. This has to be considered by concepts that extend the methodology. To the best knowledge of the authors, only one related work deals with the combination of time and concurrency in the field of model-based statistical testing: usage nets [1]. A usage net is a Discrete Deterministic and Stochastic Petri net with colored transitions. Prior to test case generation or parameter calculation, it is transformed into a respective MCUM. Any time dependencies are thereby discretized, i.e., a single state or transition in a usage net is represented by a set of states and transitions in the underlying usage model. Deploying usage nets in practice may cause increased efforts for training, as the notation for most model-based approaches in the field of system testing is similar to state machines [4]. Additionally, the discretization of time dependencies increases the effort that is required to carry out the analysis, as it provokes a state space explosion in the underlying MCUM.

Our approach is based on a notation similar to state machines. This reduces the effort to deploy it in practice. Test case generation is directly carried out on a TUM with parallel regions. For the analysis, discretization of time dependencies is avoided. Instead, we define and use an SMP with parallel regions: a stochastic process suitable as foundation for the calculation of parameters from a TUM with parallel regions.

## 3 Semi-Markov Processes with Parallel Regions

We define an SMP with parallel regions as natural extension of an SMP [6] by composite states. It is represented by a state machine with simple and composite

<sup>1</sup> Time Partition Testing: Systematic automated testing of embedded systems, <http://www.piketec.com>, accessed on May 22, 2015.

states  $s_i \in S$  at the top level. The stochastic matrix  $\mathbf{P} = [p_{i,j}]$  holds the branching probabilities after leaving a state at the top level. The sojourn time of each state  $s_i$  is given by the random variable  $X_i$  with  $F_i(t)$  as distribution, and the steady-state probability is denoted by the respective entry in vector  $\boldsymbol{\pi} = [\pi_i]$ . Different sojourn times can be configured for each state by attaching a respective distribution. However, a composite state is not annotated with a distribution.

A composite state  $s_i$  holds  $r_i$  parallel regions  $s_{i,j}$ . Each region owns a state machine with a final state. A sub-state  $k$  of region  $j$  of composite state  $s_i$  is referred to as  $s_{i,j,k}$ . Besides final states, each sub-state has a distribution annotated. If a composite state is entered, then each region will enter its initial state. A composite state will be left, if all regions have reached their final state. There are no further synchronizations between the regions. An exemplary SMP with parallel regions is depicted in Fig. 1. Note that, we call a final state an absorbing state and refer to it as  $s_{i,j,m}$  with  $m = |s_{i,j}|$  ( $s_{i,j}$  can also be interpreted as set of its sub-states).

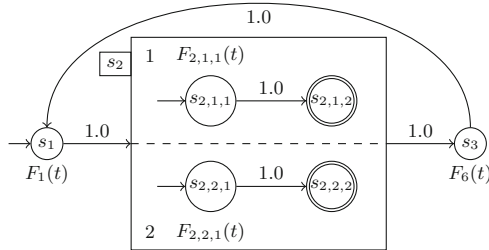


Fig. 1. Exemplary SMP with parallel regions.

### 3.1 Analysis

The analysis focuses on the calculation of the steady-state probabilities. They represent the fraction of time spent in each state in the long run and are used to calculate further parameters (see Sect. 3.2).

Prior to analysis it is necessary to define regeneration points, i.e., time instants when the process is memoryless [6]. We set regeneration points upon leaving a state at top level. This yields an embedded Markov chain (EMC) that consist of top level states only. A solution to the steady-state probabilities in the EMC can be obtained by solving the linear system of equations  $\mathbf{u} = \mathbf{u}\mathbf{P}$  with the normalization condition  $\mathbf{u}\mathbf{e} = 1$  [6]. For the final solution at top level it is necessary to consider the mean sojourn times for each EMC state. Therefore, we define matrix  $\mathbf{C} = [c_{i,j}]$ .

$$c_{i,j} = \begin{cases} E[X_i], & \text{if } i = j \\ 0 & , \text{ otherwise} \end{cases} \quad (1)$$

For simple states, the mean sojourn time is derived by  $E[X_i] = \int_0^{\infty} \overline{F}_i(t) dt$ . For composite states, it is derived by a transient analysis of the regions, which is stated below. Finally, the solution to the steady-state probabilities with regard to the individual mean sojourn times is obtained by solving

$$\boldsymbol{\pi} = \frac{\mathbf{u}\mathbf{C}}{\mathbf{u}\mathbf{C}\mathbf{e}}. \quad (2)$$

The mean sojourn time in a composite state depends on the time spent in each region in isolation until absorption. Its calculation requires the definition of some parameters:

- the mean sojourn time in a region  $s_{i,j}$  is given by the random variable  $X_{i,j}$  with distribution  $\sigma_{i,j}(t)$
- the time spent in a region  $s_{i,j}$  depends on the sojourn times of its sub-states  $s_{i,j,k}$ , which is given by the random variable  $X_{i,j,k}$  with distribution  $F_{i,j,k}(t)$
- the vector  $\mathbf{F}_{i,j}(t)$  holds the distributions of the sojourn time in each state in region  $s_{i,j}$
- the vector  $\mathbf{f}_{i,j}(t)$  holds the respective density function for each state in  $s_{i,j}$
- the branching probabilities after leaving state  $k$  and entering state  $l$  in region  $s_{i,j}$  are given by the stochastic matrix  $\boldsymbol{\Delta}_{i,j} = [\delta_{i,j,k,l}]$
- the steady-state probabilities for sub-states are given by vector  $\boldsymbol{\pi}_{i,j} = [\pi_{i,j,k}]$
- the transient probabilities of states in region  $s_{i,j}$  in isolation are given by the vector  $\mathbf{v}_{i,j}(t) = [v_{i,j,k}(t)]$ , starting at  $t = 0$
- the matrix  $\mathbf{V}_{i,j}(t) = [v_{i,j,k,l}(t)]$  holds the conditional transient probabilities in isolation, i.e., the probability that region  $s_{i,j}$  is in state  $l$  at time  $t$  given it was in state  $k$  at time 0
- the mean sojourn times in states of region  $s_{i,j}$  are given by vector  $\boldsymbol{\sigma}_{i,j} = [\sigma_{i,j,k}]$
- the matrix  $\boldsymbol{\Sigma}_{i,j} = [\sigma_{i,j,k,l}]$  holds the conditional mean sojourn times in  $s_{i,j}$

Note that, we denote vectors and matrices that are restricted to non-absorbing states with a bar  $\bar{\phantom{x}}$ , e.g., the vector  $\bar{\boldsymbol{\sigma}}_{i,j}$  denotes the mean sojourn times for all states in region  $s_{i,j}$  except for its final state. This does not affect distributions, i.e., the complement of a distribution is still given as  $\bar{F}(t) = 1 - F(t)$ .

For the calculation of the mean sojourn time spent in a composite state we consider three stochastic processes: Continuous Time Markov Chains (CTMCs) with exponentially distributed sojourn times, Discrete Time Markov Chains (DTMCs) with geometrically distributed sojourn times with time step  $\tau$ , and SMPs with sojourn times described by general distributions.

First, the mean sojourn time is calculated for all non-absorbing states. In case composite state  $s_i$  contains only one region, i.e.,  $r_i = 1$ , the mean sojourn times in non-absorbing states is determined by solving only a linear system of equations. For a CTMC, it is given by solving  $-\bar{\mathbf{v}}_{i,j}(0) = \bar{\boldsymbol{\sigma}}_{i,j} \bar{\mathbf{Q}}_{i,j}$ , with  $\bar{\mathbf{Q}}_{i,j}$  as generator matrix. For a DTMC, the equation

$$-\bar{\mathbf{v}}_{i,j}(0) = \bar{\boldsymbol{\sigma}}_{i,j} (\bar{\mathbf{P}}_{i,j} - \mathbf{I}) \quad (3)$$

has to be solved, with  $\bar{\mathbf{P}}_{i,j}$  as transition matrix. In case of an SMP, it is necessary to derive the conditional mean sojourn times  $\bar{\Sigma}_{i,j}$  first. The mean sojourn times  $\bar{\sigma}_{i,j}$  are calculated subsequently.

$$\bar{\Sigma}_{i,j} = \text{diag} \left( \int_0^{\infty} \bar{\mathbf{F}}_{i,j}(t) dt \right) + \bar{\Delta}_{i,j} \bar{\Sigma}_{i,j} \quad (4)$$

$$\bar{\sigma}_{i,j} = \bar{\mathbf{v}}_{i,j}(0) \bar{\Sigma}_{i,j} \quad (5)$$

In case a composite state  $s_i$  contains multiple parallel regions, i.e.,  $r_i > 1$ , a transient analysis has to be performed for each region separately. If a region is described by a CTMC, the transient analysis is concerned with solving the transient probabilities  $\bar{\mathbf{v}}_{i,j}(t) = \bar{\mathbf{v}}_{i,j}(0)e^{\mathbf{Q}_{i,j}t}$  of that region. If a region is described by a DTMC, the transient analysis has to consider the branching probabilities in the calculation of the transient probabilities  $\bar{\mathbf{v}}_{i,j}(k\tau) = \bar{\mathbf{v}}_{i,j}(0)\bar{\Delta}_{i,j}^k$ . In case a region is described by an SMP, its transient probabilities  $\bar{\mathbf{v}}_{i,j}(t)$  are retrieved based on Markov renewal theory [6] by calculating the conditional transient probabilities  $\bar{\mathbf{V}}_{i,j}(t)$  first, with global kernel matrix  $\bar{\mathbf{E}}_{i,j}(t) = \text{diag}(\bar{\mathbf{F}}_{i,j}(t))$ , local kernel  $\mathbf{K}'_{i,j}(t) = \text{diag}(\bar{\mathbf{f}}_{i,j}(t))\bar{\Delta}_{i,j}$  and  $*$  as convolution operation.

$$\bar{\mathbf{V}}_{i,j}(t) = \bar{\mathbf{E}}_{i,j}(t) + \mathbf{K}'_{i,j}(t) * \bar{\mathbf{V}}_{i,j}(t) \quad (6)$$

$$\bar{\mathbf{v}}_{i,j}(t) = \bar{\mathbf{v}}_{i,j}(0)\bar{\mathbf{V}}_{i,j}(t) \quad (7)$$

The effort required to calculate the transient probabilities in case of an SMP is given by  $O(|s_{i,j}|^2)$ . It can be reduced by using supplementary variables [6]. Note that, in call cases (CTMC, DTMC, SMP) it is possible to derive a closed form solution if the topology is acyclic.

The mean sojourn times  $\bar{\sigma}_{i,j}$  spent in non-absorbing states in region  $s_{i,j}$  can be derived directly from the transient probabilities.

$$\bar{\sigma}_{i,j} = \int_0^{\infty} \bar{\mathbf{v}}_{i,j}(t) dt \quad (8)$$

The complement of the sum over all transient distributions for non-absorbing states yields the distribution  $G_{i,j}(t)$ , which specifies the time to absorption in isolation within region  $s_{i,j}$ .

$$G_{i,j}(t) = 1 - \bar{\mathbf{v}}_{i,j}(t)\mathbf{e} \quad (9)$$

Due to the synchronization upon leaving a composite state  $s_i$ , its mean sojourn time  $E[X_i]$  is given as maximum over the distributions that specify the time to absorption in isolation for each region.

$$E[X_i] = \int_0^{\infty} 1 - \prod_{j=1}^{r_i} G_{i,j}(t) dt \quad (10)$$

The value derived for  $E[X_i]$  is used as entry  $c_{i,i}$  in matrix  $\mathbf{C}$  from (2).

With the solution to (2), the steady-state probability can be calculated for any sub-state at composite state level. It represents a fraction of the steady-state probability for the composite state and depends on the mean sojourn time spent in the sub-state with regard to the mean sojourn time in the composite state:

$$\pi_{i,j} = \pi_i \frac{\sigma_{i,j}}{c_{i,i}}. \quad (11)$$

Note that, the mean sojourn time for the final state of a region is given as  $\sigma_{i,j,m} = E[X_i] - \bar{\sigma}_{i,j}\mathbf{e}$ , the difference between the mean sojourn time spent in the composite state  $s_i$  and the sum of mean sojourn times spent in non-absorbing states in region  $s_{i,j}$ .

### 3.2 Calculation of Parameters

The SMP can be used to calculate parameters related to durations. For example, the mean time  $E[d]$  until the final state of the SMP is reached. It is calculate as the average waiting time  $W$  by means of Little's Law  $L = \lambda W$  [7]. As long-term average number of customers  $L$  in our process we use the fraction of time spent in non-absorbing states of the SMP in the long run, which is given by  $1 - \pi_n$  with  $\pi_n$  as steady-state probability of the final state at top level. The long-term arrival rate is given by  $\lambda = \frac{\pi_n}{E[X_n]}$ .

$$E[d] = W = \frac{L}{\lambda} = \frac{1 - \pi_n}{\pi_n} E[X_n] \quad (12)$$

The introduction of parallel regions to an SMP enables the calculation of two new parameters. The mean active time within a region  $s_{i,j}$  of a composite state  $s_i$  can be derived as  $E[a_{i,j}] = \bar{\sigma}_{i,j}\mathbf{e}$ , the sum over the mean sojourn times of its non-absorbing states. The mean sojourn time  $\sigma_{i,j,m}$  that is spent in the final state of a region  $s_{i,j}$  can be interpreted as mean waiting time  $E[w_{i,j}]$  within that region.

## 4 Timed Usage Models with Parallel Regions

A TUM with parallel regions consists of states  $s_i \in S$  and transitions  $a_{i,j} \in A$  that originate in state  $s_i$  and have state  $s_j$  as target. A state represents the externally visible state of the SUT while it is used and is either a simple or a composite state. Each simple state has a distribution  $F_i(t)$  annotated that describes the sojourn time. Each composite state contains parallel regions, i.e., it specifies concurrent streams of use. No distribution is added to a composite state, as its sojourn time depends on the evolution within its regions. Each region specifies again a TUM that may utilize composite states with parallel regions. This allows to nest concurrent streams of use. Final states are used to terminate a region. A transition specifies a stimulus  $y$ , i.e., an input that is

applied to the SUT. Additionally, each transition consists of a probability  $p_{i,j}$  and a distribution  $F_{i,j}(t)$  describing the sojourn time. An example TUM with parallel regions is depicted in Fig. 2a. It is explained in more detail in Sect. 5.

Test cases can be derived automatically from a TUM with parallel regions. The process is straightforward: A random walk is applied, beginning at the start state and ending at the final state at top level. If a simple state is reached, a sojourn time will be sampled from its distribution. Subsequently, an outgoing transition is selected with regard to the branching probabilities together with a respective firing time from the annotated distribution. If a composite state is encountered, a path is sampled from each of its regions analog to the top level.

Parameters can be calculated prior to any test case generation. Therefore, it is required to map the model to an SMP with parallel regions.

#### 4.1 Mapping to an SMP with Parallel Regions

The mapping represents a natural extension of the mapping known for classical TUMs. Each simple state  $s_i$  is mapped to a simple state  $s'_i$  in the SMP. The distribution  $F_i(t)$  is added to state  $s'_i$ . Each composite state  $s_i$  is mapped to a composite state  $s'_i$  in the SMP. No distribution is attached to composite state  $s'_i$ , as there is none available at the original state  $s_i$ .

Each transition  $a_{i,j}$  is mapped as follows: First, a new state  $s'_{ij}$  is introduced in the SMP with distribution  $F_{i,j}(t)$ . A transition  $a'_{i,ij}$  is created with transition probability equal to  $p_{i,j}$ . Afterwards, a new transition  $a'_{ij,j}$  with probability 1 is created, it leads from state  $s'_{ij}$  to state  $s'_j$ . Note that, it is not required to consider stimuli in the mapping. They are not required for the analysis.

Finally, the parallel regions of each composite state  $s_i$  are mapped to corresponding parallel regions of composite state  $s'_i$  in the SMP. The same mapping rules apply to the contents of each region as for the top level of the TUM.

#### 4.2 Calculation of Parameters

The analysis of the SMP underlying a TUM with parallel regions yields the steady-state probabilities and the mean sojourn times (see Sect. 3). They can aid the test process. The mean active time  $E[a_{i,j}]$  and mean waiting time  $E[w_{i,j}]$  of a region  $s_{i,j}$  in a TUM, under the condition that the respective composite state is entered, and the mean test case duration  $E[d]$  are direct results of the analysis of the underlying SMP. Note that, the mean sojourn time of the final state is not included in the test case duration.

The expected number of occurrences of a state in a test case, given one starts in the start state of the usage model, is no direct result of the SMP analysis. In order to calculate it, we make use of its definition: For a state  $s_i$  in a DTMC with final state  $s_n$  the expected number of occurrences  $E[n_{1,i}]$  within a test case, given one starts in the start state  $s_1$  of the usage model, is obtained via (13) [8]. It holds for DTMCs, as the same fixed time step  $\tau$  is spent in each state upon entry and the final state occurs exactly once within a test case.

$$E[n_{1,i}] = \frac{\pi_i}{\pi_n} \quad (13)$$

We apply the underlying idea to an SMP with parallel regions. All attached times are ignored and the whole process is considered as DTMC. This also applies to regions of composite states. The expected number of occurrences  $E[n_{1,i}]$  for any state  $s_i$  within the EMC is obtained via (13) with the steady-state probabilities  $\mathbf{u}$  calculated for the EMC (see Sect. 3.1). In order to calculate the expected number of occurrences for a sub-state  $s_{i,j,k}$ , we consider a reduced process where only the top level and the respective region  $s_{i,j}$  of the composite state  $s_i$  are considered, i.e., we blend out all other parallel regions and regions of different composite states. Leveraging (1), (2) and (11), we obtain the expected number of occurrences for a sub-state  $s_{i,j,k}$  as

$$E[n_{1,ijk}] = \frac{\pi_{i,j,k}}{\pi_n} = \frac{\pi_i \frac{\sigma_{i,j,k}}{E[X_i]}}{\frac{u_n \cdot c_{n,n}}{\mathbf{uCe}}} = \frac{u_i \cdot c_{i,i} \cdot \sigma_{i,j,k}}{\mathbf{uCe} \cdot c_{i,i}} \frac{\mathbf{uCe}}{u_n \cdot c_{n,n}} = \frac{u_i \cdot \sigma_{i,j,k}}{u_n \cdot c_{n,n}}. \quad (14)$$

The value  $E[n_{1,ijk}]$  only depends on: the steady-state probabilities from the EMC which are the same for the original and the reduced process and therefore only have to be calculated once, the mean sojourn time of the final state in the usage model which is equal to the chosen step size  $\tau$  as we consider the whole process as DTMC, and the mean sojourn time of the respective sub-state which is obtained via (3). Note that, the mean sojourn time for the final state of region  $s_{i,j}$  can also be obtained by (3), since we blend out other parallel regions. The expected number of occurrences of a transition in a test case is obtained in the same way, as transitions in a TUM are extended to states/sub-states in the underlying SMP (see Sect. 4.1).

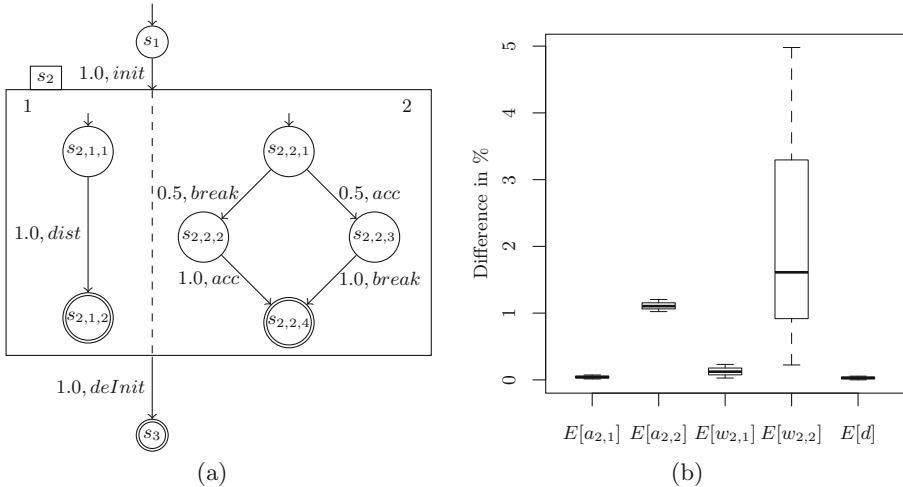
The mean residence time of a state  $s_i$  in a test case is given by  $E_r[s_i] = E[n_{1,i}] \cdot E[X_i]$ , i.e., its number of occurrences within a test case multiplied with its mean sojourn time. Of course, the mean sojourn time used here is derived with regard to the distribution attached to the state. The mean residence time of a transition in a test case is calculated in the same way. The expected execution time for a stimulus is derived as sum over all expected residence times for transitions, that have the stimulus attached.

## 5 Validation

We applied our approach to an example system from the automotive domain: the adaptive cruise control (ACC). It provides comfort to the driver by automatically maintaining a defined velocity. If a vehicle in front prohibits to drive at desired speed, it will slow down in order to maintain a safe distance. The ACC allows concurrent streams of use, e.g., the driver may change the speed and/or distance, while the vehicle in front may break or accelerate. We specified a respective TUM with parallel regions. It is depicted in Fig. 2a. After initialization, the ACC maintains a safe distance to the vehicle in front at a velocity that is lower than configured by the driver. In the first region of composite state  $s_2$ , the driver changes the distance to be maintained to the vehicle in front. The second region specifies the actions of the vehicle in front. Test cases generated from the model

examine whether the acceleration induced by the ACC stays below a defined threshold. The aim of this threshold is to avoid abrupt movements. Distributions for sojourn times in states and firing times of transitions are omitted in the figure to keep it clear. We used exponential distributions for non-absorbing states and geometrical distributions for transitions. Of course, different times may be used.

We carried out the analysis on the example usage model and calculated all metrics from Sect. 4.2. In order to validate the analysis, we built a simulation. Therefore, we used Papyrus<sup>2</sup> to specify the model and Syntony [5] to transform it to the simulation framework OMNeT++<sup>3</sup>. The simulation was run multiple times. The results, some shown in Fig. 2b, confirmed the values that had been obtained by our approach. Parameters related to the expected number of occurrences of a state/transition were identical. We measured a marginal deviation between the simulation and analysis results only for parameters related to durations. The difference is depicted in Fig. 2b. The value for  $E[w_{2,2}]$  stands out. This is due to the non-deterministic synchronization effects in the simulation upon exit of the composite state. However, the absolute value is negligibly small.



**Fig. 2.** (a) TUM with parallel regions for ACC example and (b) relative difference between simulation and analysis results.

## 6 Conclusions and Future Work

With the introduction of parallel regions to TUMs we were able to model both, sequential and concurrent uses of the SUT, with regard to time dependencies.

<sup>2</sup> Papyrus: Graphical editing tool for UML 2, <http://www.eclipse.org/modeling/mdt/papyrus>, accessed on May 22, 2015.

<sup>3</sup> OMNeT++: An object-oriented modular discrete event network simulation framework, <http://www.omnetpp.org>, accessed on May 22, 2015.



This was previously not possible due to restrictions in classical TUM. Thereby, we preserved the major properties of the model: the automatic test case generation and calculation of parameters that aid the test process. The test case generation stayed a straightforward process. For the analysis, we avoided discretization of time dependencies and introduced an SMP with parallel regions instead.

In our future work we plan to provide an additional test stop criteria that considers the variability of possible uses resulting from concurrent aspects.

## References

1. Böhr, F.: Model based statistical testing of embedded systems. In: Proceedings of the 4th International Conference on Software Testing, Verification and Validation Workshops (ICSTW 2011), Berlin, Germany, pp. 18–25, March 2011
2. Bringmann, E., Kramer, A.: Model-based testing of automotive systems. In: Proceedings of the 1st International Conference on Software Testing, Verification, and Validation (ICST 2008), Lillehammer, Norway, pp. 485–493, April 2008
3. Broy, M.: Challenges in automotive software engineering. In: Proceedings of the 28th International Conference on Software Engineering (ICSE 2006), Shanghai, China, pp. 33–42, May 2006
4. Dias Neto, A.C., Subramanyan, R., Vieira, M., Travassos, G.H.: A survey on model-based testing approaches: a systematic review. In: Proceedings of the 1st ACM International Workshop on Empirical Assessment of Software Engineering Languages and Technologies (WEASEL/Tech 2007), Atlanta, Georgia, USA, pp. 31–36 (2007). <http://doi.acm.org/10.1145/1353673.1353681>
5. Dietrich, I.: Syntony: A Framework for UML-Based Simulation, Analysis and Test with Applications in Wireless Networks. Dr. Hut, Germany (2010)
6. German, R.: Performance Analysis of Communication Systems. Wiley, Chichester (2000)
7. Little, J.D.C.: A proof for the queuing formula:  $L = \lambda W$ . *Oper. Res.* **9**(3), 383–387 (1961)
8. Prowell, S.J.: Computations for Markov chain usage models. University of Tennessee, Tech. report (2003)
9. Prowell, S.J.: A stopping criterion for statistical testing. In: Proceedings of the 37th Hawaii International Conference on Systems Sciences (HICSS 2004), Big Island, Hawaii, USA, January 2004
10. Siegl, S., Dulz, W., German, R., Kiffe, G.: Model-driven testing based on markov chain usage models in the automotive domain. In: Proceedings of the 12th European Workshop on Dependable Computing (EWDC 2009), Toulouse, France, May 2009
11. Siegl, S., German, R.: Model-driven testing with timed usage models in the automotive domain. In: Proceedings of the 20th International Symposium on Software Reliability Engineering (ISSRE 2009), Mysuru, India, November 2009
12. Siegl, S., Hielscher, K.S., German, R.: Introduction of time dependencies in usage model based testing of complex systems. In: 4th Annual IEEE Systems Conference (SysCon 2010), San Diego, California, USA, pp. 622–627, April 2010
13. Whittaker, J.A., Poore, J.H.: Markov analysis of software specifications. *ACM Trans. Softw. Eng. Methodol.* **2**(1), 93–106 (1993)

**HOFM 2015**

# Helping the Tester Get It Right: Towards Supporting Agile Combinatorial Test Design

Anna Zamansky<sup>(✉)</sup> and Eitan Farchi

University of Haifa, Haifa, Israel  
annazam@gmail.com

**Abstract.** Combinatorial test design (CTD) is an effective test planning technique that reveals faulty feature interaction in a given system. CTD takes a systematic approach to formally model the system to be tested, and propose test cases ensuring coverage of given conditions or interactions between parameters. In this position paper we propose a framework for supporting *agile* CTD, a human-centered methodology, which takes into account the human tester's possible mistakes and supports revision and refinement. In this approach a combinatorial model of the system and test plans are constructed in an incremental and iterative way, providing the tester with the ability to refine and validate the constructions. We propose a formal framework which can be used as a theoretical foundation for the development of agile CTD support tools, and describe a use case of an envisioned tool.

## 1 Introduction

As software systems become increasingly complex, verifying their correctness becomes more challenging. Formal verification approaches are highly sensitive to the size of complexity of software, and might require extremely expensive resources. Functional testing, on the other hand, is prone to omissions, as it always involves a selection of what to test from a potentially enormous space of scenarios, configurations or conditions that is typically exponential in nature. The process of test planning refers to the design and selection of tests out of a test space aiming at reducing the risk of bugs while minimizing redundancy of tests. *Combinatorial Test Design* (CTD) [6, 10] is an effective test planning technique, in which the space to be tested, called a *combinatorial model*, is represented by a set of parameters, their respective values and restrictions on the value combinations [8]. CTD approaches can be applied at different phases and scopes of testing, including end-to-end and system-level testing and feature-, service- and application program interface-level testing.

The main challenge of CTD consists of finding a small (and ideally minimal) set of test cases (a subset of the space to be tested), which ensures coverage of given conditions, or interactions between variables (such as pairs, three-way, etc.) Experiments show that a test set that covers all possible pairs of parameter values can typically detect between 50 to 75% of the bugs in a program [9].

Other experimental work has shown that typically all bugs can be revealed by covering the interaction of between 4 to 6 parameters [3].

One important task in CTD is the construction of a *combinatorial model* of a system, which includes a set of parameters, their respective values and logical restrictions on value combinations. Typically, the parameters of the model do not map directly to user inputs of the system, but are rather a high-level abstraction of them. Identifying correctly the set of parameters for the model, their values - and most importantly - the restrictions on them is a laborious task which requires abstract thinking and a considerable level of familiarity with formal logic. Another task is the planning of tests over the constructed model, which satisfy a chosen coverage requirement (e.g., pairwise coverage). The complex and error-prone nature of both of these tasks leads to the need for automatic tools supporting the human tester.

While there are numerous studies on human errors (see, e.g. [1,5]), and in particular on human factors in software development [4,7], they have been only marginally addressed in the context of combinatorial test design. In this position paper we propose an *explicitly human-centered* methodology for CTD, which takes into account the possibility of the tester's error and provides the tester with systematic tools to refine and validate the constructed models and test plans. We propose the term '*agile test design*' to refer to such approach, reflecting the incremental and iterative way in which models and test plans are constructed. We propose a formal framework which can be used as a theoretical foundation for the development of agile CTD tools, and describe a use case of an envisioned tool.

## 2 A Vision for Agile Test Design

Typically, the CTD methodology is applied in the following stages. First the tester constructs a combinatorial model of the system by providing a set scenarios which are executable in the system. After choosing the coverage requirements, the second stage is constructing a test plan, i.e., proposing a set of tests over the model, so that full coverage with respect to a chosen coverage strategy is achieved. In practice, taking into account the human factor [1,5], errors are possible at both of these stages. Moreover, error discovery in the test plan may cause the tester to return to the model and refine it, and vice versa. To reflect the iterative and incremental nature of the two stages of CTD, we propose the term '*agile test planning*'. This notion is based on the assumption that the tester's activity is not errorproof: errors can happen, both in the model and the test plan, and should be taken into account. Therefore, *correctness* of the combinatorial model is not assumed at the stage of test planning, and the tester may go back to refining the model at any point.

The basic unit manipulated by the tester is a *test*. A test is represented by a vector of values assigned to systems' parameters. The combinatorial model of the system is a set of all tests which describe possible (or executable) behaviors of the system. A test plan is a set of tests out of the model space which satisfy some chosen coverage policy (such as pairwise testing).

We divide the space of tests into three basic types, according to the information available from the tester:

1. *validated* - these are the tests that the tester confirmed as executable (according to some chosen confirmation strategy).
2. *rejected* - these are the tests that the tester rejected as impossible, either by explicitly removing them from the model, or by providing a logical condition that rules them out.
3. *uncertain* - these are the tests for which not enough information has been provided to classify them as validated/rejected.

Agile test design, therefore, can be thought of as an iterative process, the goal of which is to validate or reject each possible test, arriving at a coherent combinatorial model of the system, together with a test plan satisfying the chosen coverage. Referring to the above three types of tests as colors: green, red and yellow respectively, the goal is eliminating all yellow tests by turning them either to green or to red. This can be done by raising a series of questions to the tester, which help determine if tests are missing in the test plan (and so the combinatorial model should be expanded, turning yellow to green), or the combinatorial model contains non-executable tests (and so it should be reduced, turning some yellow tests to red).

The methodology described above allows for a great amount of flexibility, both for user querying strategies, and for validation/rejection strategies. For instance, we may want to minimize the time it takes the tester to arrive to a coherent solution, or the cognitive load of the tester (by presenting him only small portions of tests in each iteration).

### 3 The Formal Framework

#### 3.1 Combinatorial Models and Test Plans

**Definition 1 (System space).** *A system space is a finite set of system parameters  $\mathbf{P} = \{\mathbf{A}_1, \dots, \mathbf{A}_n\}$  together with their corresponding associated values  $\{\mathbf{V}(\mathbf{A}_1), \dots, \mathbf{V}(\mathbf{A}_n)\}$ . For any value  $a \in \mathbf{V}(\mathbf{A}_i)$ , we say that  $a$  has type  $\mathbf{A}_i$ , denoted by  $\text{type}(a) = \mathbf{A}_i$ .*

**Definition 2 (Interactions, scenarios).** *An interaction is an element  $\mathcal{I} \subseteq \bigcup_1^n \mathbf{V}(\mathbf{A}_i)$  such that for distinct  $a, b \in \mathcal{I}$ ,  $\text{type}(a) \neq \text{type}(b)$ . An interaction of size  $n$  (where  $n$  is the number of system parameters) is called a scenario.*

*Example 1.* As our running example, let us consider a system in which there are two servers  $S_1$  and  $S_2$ , which can be either active (up) or inactive (down). Moreover, there are two operations Send and Ping, which can be performed by either one of the servers. We can set the system parameters to  $\mathbf{P} = \{S_1, S_2, Op\}$ , where:

$$V(S_1) = \{up^1, down^1\}$$

$$V(S_2) = \{up^2, down^2\}$$

$$V(Op) = \{Send_1, Send_2, Ping_1, Ping_2\}$$

So, e.g., an assignment  $S_1 = up^1$ ,  $S_2 = down_2$  and  $Op = Send_1$  represents a situation where the first server is up, performing the send operation and the second is down.

Moreover,  $\{up^1, Send_0\}$  and  $\{up^1, down^2, Ping_0\}$  are interactions; the latter is also a scenario.

**Definition 3 (Coverage).** We say that a set of scenarios  $T$  covers a set of interactions  $C$  if for every  $c \in C$  there is some  $t \in T$ , such that  $c \subseteq t$ .

*Example 2.* Let the global system space be  $\mathbf{P} = \{\text{FileOps}, \text{PathName}, \text{OS}\}$ , where

$$\mathbf{V}(\text{FileOps}) = \{open, close, read, write\}$$

$$\mathbf{V}(\text{PathName}) = \{relative, absolute\}$$

$$\mathbf{V}(\text{OS}) = \{unix, windows\}$$

Then we can define interactions  $\mathcal{I}_1 = \{open, relative\}$  and  $\mathcal{I}_2 = \{close, absolute\}$ . We can define further define scenarios  $t_1 = \{open, relative, unix\}$  and  $t_2 = \{close, absolute, windows\}$ . Note that  $\{t_1, t_2\}$  covers  $\{\mathcal{I}_1, \mathcal{I}_2\}$ .

**Definition 4 (Combinatorial model).** A combinatorial model  $\mathcal{E}$  is a set of scenarios (which defines all scenarios executable in the system).

**Definition 5 (Test plan).** A test plan is a triple  $P = (\mathcal{E}, C, T)$ , where  $\mathcal{E}$  is a combinatorial model,  $C$  is a set of interactions called coverage requirements, and  $T$  is a set of scenarios called tests, where  $T$  covers  $C$ .

*Example 3.* The most standard coverage requirement in the domain of combinatorial test design is *pairwise testing* [9, 10]: considering every (executable) pair of possible values of system parameters. In terms of the above definition, a pairwise test plan can be formulated as any pair of the form  $P = (\mathcal{E}, C_{pair}(\mathcal{E}), T)$ , where  $C_{pair}$  is the set of all interactions of size 2 which can be extended to scenarios from  $\mathcal{E}$ .

### 3.2 Representing Models and Plans as Logical Theories

Recall that  $\mathbf{P} = \{\mathbf{A}_1, \dots, \mathbf{A}_n\}$  is our set of system parameters and  $\{\mathbf{V}(\mathbf{A}_1), \dots, \mathbf{V}(\mathbf{A}_n)\}$  their corresponding ranges of values.

**Definition 6.**  $F_{\mathbf{P}}$ , the set of well formed formulas of  $\mathbf{P}$  is defined inductively as follows:

- $\mathbf{A}_i$  :  $a \in F$  for every  $1 \leq i \leq n$  and  $a \in \mathbf{V}_{\mathbf{P}}(\mathbf{A}_i)$ . We call such formulas atomic.
- If  $\psi, \varphi \in F_{\mathbf{P}}$ , then  $\neg\psi, (\varphi \wedge \psi), (\varphi \vee \psi), (\varphi \rightarrow \psi) \in F_{\mathbf{P}}$ .

When the parameter is clear from context, we shall write just  $a$  instead of  $\mathbf{A}_i : a$ . So examples of formulas for  $\mathbf{P}$  defined in Example 1 are: *open*, *relative*,  $((open \wedge relative) \rightarrow \neg unix)$ .

Every formula of  $F_{\mathbf{P}}$  naturally induces a set of scenarios (i.e., assignments to all system parameters, or in other words, tests) in the following sense:

**Definition 7.** Let  $\psi \in F_{\mathbf{P}}$  and let  $s$  be some scenario. We say that  $s$  satisfies  $\psi$ , denoted by  $s \models \psi$  iff:

- $\psi = \mathbf{A}_i : a$  and  $a \in s$ .
- $\psi = \varphi_1 \wedge \varphi_2$ ,  $s \models \varphi_1$  and  $s \models \varphi_2$ .
- The rest of the cases are defined similarly.

For a formula  $\psi$ , we define  $mod(\psi) = \{s \mid s \models \psi\}$ . For a theory  $\Gamma = \{\psi_1, \dots, \psi_n\}$ , we denote  $mod(\Gamma) = mod(\psi_1) \cap \dots \cap mod(\psi_n)$ .

The above definition provides a useful link between logical theories and sets of tests: a set of formulas  $\Gamma$  naturally defines a subset of tests by  $mod(\Gamma)$ . Thus the tester may use formulas of the above form to specify sets of tests of interest.

### 3.3 Agile CTD: Iterative Uncertainty Elimination

Let us now describe the framework of agile CTD more formally, using the notions introduced above. The goal of the tester is to provide a valid test plan  $P = (\mathcal{E}, C, T)$ . By a validity we mean here that (i)  $T$  satisfies all coverage requirements in  $C$ , and (ii)  $T$  is a subset of  $\mathcal{E}$ . A coherent solution is reached when both (i) and (ii) are satisfied, if one of them is violated our envisioned tool supports an interactive resolution of such violations.

It is important to note that in our framework we do not assume the availability of  $\mathcal{E}$ , the combinatorial model, rather it is extracted when the tester specifies a set of specific test cases  $T$ , as well as some logical restrictions (in the form of formulas as defined above) on the combinatorial model, which provide only partial information about  $\mathcal{E}$ . Once the tester is satisfied with the constructed plan, a support tool may be invoked. The tool may automatically extract the combinatorial model from the provided tests and logical restrictions. Each test in the model has a status: those appearing in the test plan are validated (green), those ruled out by the logical conditions are rejected (red), the remaining ones are uncertain (yellow). The tool then may raise a sequence of questions that help determine the status of each of the yellow tests in the model. Different sequences may be proposed according to different considerations.

### 3.4 A Possible Use Case

Let us describe a possible agile test planning use case. Suppose the coverage requirement is pairwise coverage. Returning to the system from Example 1, suppose that not all test cases are executable and further restrictions should be imposed. For instance, assume that server 1 can perform both send and ping

operations, while server 2 can only perform ping operations. Moreover, only one of the servers can be up at the same time. If these are the only restrictions, the set of the executable scenarios can be described (in the sense of Definition 7) by the logical theory  $\Gamma = \{up^1 \leftrightarrow down^2, up^2 \rightarrow Ping_0 \vee Ping_1\}$ .

Suppose, however, that the tester erroneously thinks that all combinations are possible and provides no logical restrictions for the model at this point. This induces the whole set  $\mathcal{E}_0 = V(S_1) \times V(S_2) \times V(Op)$  as the underlying combinatorial model. The tester further proposes the following test cases:

$S_1$	$S_2$	Op
$down^1$	$up^2$	$Ping_0$
$down^1$	$up^2$	$Ping_1$
$up^1$	$down^2$	$Send_0$
$up^1$	$down^2$	$Send_1$
$up^1$	$down^2$	$Ping_0$
$up^1$	$down^2$	$Ping_1$

Once the tester submits the test plan, the six tests above are colored green (validated) in the model, the rest remaining yellow as no additional restriction ruling them out have been provided. At this point the tester's mistake may be discovered, as pairwise coverage is not achieved: the following interactions remain uncovered:

$$\{up^1, up^2\}, \{down^1, down^2\}, \{up^2, Send_0\}, \{up^2, Send_1\}$$

This could be either due to the fact that the tester forgot to add some tests, or he intentionally left them out as they are not executable in the system. In the former case he will add further concrete test cases, which will turn green in the model. In the latter case, however, he needs to refine the combinatorial model of the system, either explicitly or by adding logical restrictions, which will make some tests red in the model. In any case, the level of uncertainty (the number of yellow-colored tests) will be reduced at each iteration.

To maintain scalability to large parameter sets, a useful strategy can be to consider smaller projections at each iteration. So, e.g, the tool may focus on the projection  $\{S_1, S_2\}$ , asking the tester whether he wants to add tests to cover the interactions  $\{up^1, up^2\}, \{down^1, down^2\}$ . A negative answer implies that some scenarios should be excluded from the model. In this case a logical condition  $up^1 \leftrightarrow down^2$  "explaining" this discrepancy can be suggested to refine the model. It is an interesting direction for further investigation to investigate strategies for looking for the logical conditions which would "make the most sense" to the tester. Suppose that the user may now accept this model refinement, leading to further yellow tests turning red.

The remaining missing interactions  $\{up^2, Send_0\}, \{up^2, Send_1\}$  are in the projection  $\{S_2, Op\}$ , on which the tool may focus next. We can again ask the tester whether he wants to add tests to cover them. He replies negatively, and the logical condition  $up^2 \rightarrow Ping_0 \vee Ping_1$  "explaining" this discrepancy can be suggested to refine the model.



## 4 Summary and Future Work

The tester's main tasks in combinatorial test design, which include modelling the system and test planning, are laborious and error-prone. In this position paper we presented a vision for automatic support tools for CTD testers, which support an 'agile' iterative test design. This approach takes into account the error-prone nature of the human tester's tasks, and supports the tester in refining and correcting his outputs by proposing a series of queries. We proposed here a framework for formalization of agile CTD, which can be used as a theoretical basis for developing future automatic support tools.

An immediate direction for further research is an implementation of a prototype of the proposed tool. We plan to implement it using the environment of IBM Functional Coverage Unified Solution (FoCuS [2]), which is a tool for test-oriented system modeling, for model based test planning, and for functional coverage analysis. Another challenge is proposing methods to quantify the added value such tool may provide to the human testers. A starting point here could be proposing measurable criteria for the quality of combinatorial models. These criteria could then be used to compare manually constructed combinatorial models to models constructed with the help of our tool.

When developing tools to support agile CTD, there is a level of freedom when considering different minimization strategies, which in their turn induce the order and type of questions posed by the tool to the user. One possible strategy is the minimization of time to complete a test plan. In this case questions answers to which make more tests certain should be preferred. Another strategy could be minimization of the cognitive load of the tester. In this case we may propose first logical restrictions involving a small number of parameters, or capturing a small number of tests which can be visualized on screen.

Human factors have so far received little attention in combinatorial test design. It is our hope that this paper will start a discourse on the practical needs of testers in the process of test design and combinatorial system modelling.

## References

1. Dhillon, B.S.: Engineering product usability: a review and analysis techniques. *WSEAS Trans. Circuits Syst.* **2**, 86–94 (2005)
2. [http://researcher.watson.ibm.com/researcher/view\\_group.php?id=1871](http://researcher.watson.ibm.com/researcher/view_group.php?id=1871)
3. Kuhn, D.R., Wallace, D.R., Gallo, Jr., A.M.: Software fault interactions and implications for software testing. *IEEE Trans. Softw. Eng.* **30**(6), 418–421 (2004)
4. Malz, C., Sommer, K., Göhner, P., Vogel-Heuser, B.: Consideration of human factors for prioritizing test cases for the software system test. In: Harris, D. (ed.) *HCI 2011*. LNCS, vol. 6781, pp. 303–312. Springer, Heidelberg (2011)
5. Mioch, T., Osterloh, J.-P., Javaux, D.: Selecting human error types for cognitive modelling and simulation. In: Cacciabue, P.C., Hjälm Dahl, M., Luedtke, A., Riccioli, C. (eds.) *Human Modelling in Assisted Transportation*, pp. 129–138. Springer, Heidelberg (2011)
6. Nie, C., Leung, H.: A survey of combinatorial testing. *ACM Comput. Surv. (CSUR)* **43**(2), 11 (2011)

7. Pirzadeh, L.: Human factors in software development: a systematic literature review. M.Sc thesis, Chalmers University of Technology (2010)
8. Segall, I., Tzoref-Brill, R., Zlotnick, A.: Common patterns in combinatorial models. In: Proceedings of the IEEE Fifth International Conference on Software Testing, Verification and Validation (ICST), pp. 624–629. IEEE (2012)
9. Tai, K.C., Lie, Y.: A test generation strategy for pairwise testing. *IEEE Trans. Software Eng.* **28**(1), 109–111 (2002)
10. Zhang, J., Zhang, Z., Ma, F.: Introduction to combinatorial testing. In: Zhang, J., Zhang, Z., Ma, F. (eds.) *Automatic Generation of Combinatorial Test Data*, pp. 1–16. Springer, Heidelberg (2014)

# Behavioral Types for Component-Based Development of Cyber-Physical Systems

Jan Olaf Blech<sup>1</sup>(✉) and Peter Herrmann<sup>2</sup>

<sup>1</sup> RMIT University, Melbourne, Australia  
janolaf.blech@rmit.edu.au

<sup>2</sup> NTNU, Trondheim, Norway  
herrmann@item.ntnu.no

**Abstract.** Spatial behavioral types encode information on the tempo-spatial behavior of components acting in the physical space. That makes it possible to utilize the well established concept of type systems with its well studied benefits for programming languages, e.g., fast automatic detection of incompatibilities and coercion, also in the cyber-physical world of domains such as embedded systems. So, spatial behavioral types support development and better maintenance of systems leading to a reduction of errors, improvement of safety and, in consequence, lower expenditure. In this position paper, we summarize existing work and develop our ideas for a spatial behavioral type concept. In particular, we turn our attention to making the spatial behavioral types easily usable by non-experts. Besides of a semantics that resembles traditional types systems, our method offers a syntax based on easily comprehensible regular expressions while systems can be verified using fully-automatic tools.

## 1 Introduction

Most programming languages use type systems that facilitate the automatic analysis of program code for errors. Behavioral types [3,4] are an enhancement of this well-known concept. In contrast to simple type systems, they do not only model interfaces on a purely syntactical level but also take the behavior of software into consideration (see [8]). Behavioral types support Human Factors in two ways:

- They provide an easily comprehensible modeling language for abstract component specifications. This can rely on formal specification mechanisms such as regular expressions that are frequently used by non-experts.
- In the context of developing component-based software, behavioral types provide means to specify and check component contracts that consider the current states of a component and its environment. This makes an easy and mostly fully automatic analysis of the conformance of a component with its environment possible (see, e.g., [14,32]). The checks may run in the background of development environments or deployed systems and interact with user-interfaces in an intuitive way.

In the approach presented here, we bring the spatial behavior of cyber-physical components into the type system world. In domains such as the automotive industry and industrial automation, standards like ISO 26262 and IEC 61499 gain increasing popularity. Since several of these standards support a component view, we believe that the behavioral types concept can play an important role in supporting the development, maintenance and service activities of the components in a supportive and user-friendly way.

We continue with a summary of related work in Sect. 2. To ease the understanding of our approach, we introduce a motivating example in Sect. 3 followed by a description of the core concepts of the behavioral types in Sect. 4. Thereafter, we discuss the particular properties of the spatial behavioral types in Sect. 5. The text is completed by some concluding remarks.

## 2 Existing Approaches

Different behavioral type-like approaches to specify interfaces of component systems and reason about these specifications have been proposed in the past. The current state-of-the-art, however, focusses almost exclusively on software aspects.

Interface automata [3] are one form of behavioral types. Component descriptions are based on timed automata. The focus of interface automata is on communication protocols between components. Interface automata do not target all type relevant aspects discussed in this paper, e.g., physical and spatial aspects or the checking the behavior at runtime of a component by using some form of monitoring. The main focus is an compatibility checks of software components interacting at compile time. Behavioral types are part of the Ptolemy framework [35]. Here, one focus is on the software part of real-time systems such as execution time of code.

The idea of having well defined specifications defining interfaces of software component systems has been made popular by design-by-contract [36] like approaches during the late 80s and early 90s. The focus of the classical approach is on contracts for object oriented systems. Other work that is related to our behavioral types comprises specification and contract languages for component based systems that have been studied in the context of web services. For example, the approach presented in [2] comprises request and response operations as a means for specifying behavior. Process algebra-like approaches including deductive techniques are presented for web services [18, 20]. In [18], emphasis of the formalism is put on compliance, a correctness guaranty for properties like deadlock and livelock freedom. Another algebraic approach to service composition is described in [23]. Means restricting the interface behavior of OSGi for facilitating analysis are featured in [16].

A variant is used in the model-based system engineering technique SPACE [34] and its tool-set Reactive Blocks. Using UML activities, systems are modeled by composing descriptions of subfunctions that are arranged in so-called building blocks. A building block is provided by an External State Machine (ESM) [32] that is a UML state machine describing the interface behavior of

the building block. Due to formal semantics of the UML activities and state machines [33], one can verify by model checking at design time that a building block complies with both its own ESM and those of the blocks it incorporates in its behavioral description. The approach has already been used in the context of cyber-physical systems [27,30].

JML [22] can be applied to specify pre- and postconditions for Java programs. Although not a type system, it can be utilized to specify aspects of behavior for Java based systems. In addition, assertion like behavioral specifications have been studied for access permissions [21]. Behavioral types comprising behavioral checks at runtime for component based systems were proposed in [4]. The focus is on the definition of a suitable formal representation expressing types and investigating their methodical application in the context of a model-based development process. A language for behavioral specification of software components, in particular of object oriented systems, is introduced in [31]. Compared to the more requirement-based descriptions proposed in our paper, the specifications used in [31] are still relatively close to an implementation. Additional work on refinement of automata based specifications is studied in [38]. A survey with a focus on pre-/postcondition and invariant-based annotations for programming languages can be found in [28].

The runtime verification community has developed frameworks which can be used to generate monitors checking behavioral type conformance at runtime in order to detect and report type violations. The MOP framework [37] provides the integration of specifications into Java source code files and generates AspectJ aspects realizing runtime monitoring. A framework that regards the trade-off between checking specifications at runtime and at development time is provided in [17]. The framework described in [6] facilitates also the generation of Java monitors but leaves the instrumentation aspect, i.e., the connection of the monitor to the deployed system, to the implementation. Other topics explored in this context comprise the efficiency and expressiveness of monitoring [5,7]. Monitoring of performance and availability attributes of Java/OSGi-based systems has been studied in [41]. A focus is on the dynamic reconfiguration ability of OSGi. Work building on the .Net framework for runtime monitor integration is described in [26]. Runtime monitors for interface specifications of web-service in the context of a concrete e-commerce service are described in [25]. Behavioral conformance of web-services and corresponding runtime verification were investigated in [19]. Furthermore, in [24] runtime monitoring of web-services is studied, in which runtime monitors are derived from UML diagrams. Runtime enforcement of safety properties is especially important in the context of cyber-physical systems, since a deviation is not only reported, but a countermeasure is provided. In [39], security automata are used to enforce security properties. These automata are able to halt the underlying program when a deviation from the expected behaviors is detected. A similar approach to protect software components against malicious behavior, is provided in [29]. Behavioral types-based runtime monitoring for industrial automation is also studied in [44], where the

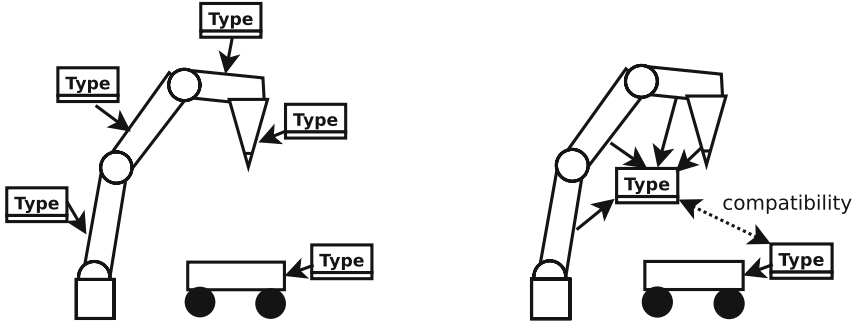


Fig. 1. Behavioral types for a robot

abstract behavioral types specification yields a monitor that runs directly on PLC.

Tangible user interfaces [40] provide a good way of exemplifying underlying principles of cyber-physical behavioral types. Here, each component in the interface may bear cyber-physical characteristics and can be described with a behavioral specification, e.g., using a domain specific language.

### 3 Motivating Example

Figure 1 shows the interaction of a robot with a vehicle. Both components may change their positions, directions and orientations over time. We can represent the spatial attributes of a component as subtypes of its spatial behavioral type. For instance, the position of the vehicle at a certain point of time can be expressed by the space it physically occupies. The space may be represented by coordinates in a geometric coordinate system stored in the behavioral type. Using these subtypes, one can verify spatial properties, e.g., two components do not collide if their physically occupied spaces never overlap at any point of time.

Cyber-physical components can be made up of other components. For example, the robot consists of three segments and a tool that are attached via joint devices. Each subcomponent has its own spatial behavior that can be represented by a separate spatial behavioral type. The type encodes the movement of its sub-component which depends on the spatial behavior of the attached devices. Similar subcomponents like the lower two segments of the robot may be represented by the same types. The two type instances relate to different attached segments which leads to different positions and orientations.

The picture on the right side of Fig. 1 depicts the composition of the behavioral types of the three segments and the tool to a single type that superimposes the spatial movements of the subcomponents and represents the spatial behavior of the overall robot. The composition makes it easier to check if the spatial behaviors of the robot and vehicle are compatible, e.g., that the vehicle is in a certain distance when the robot loads something onto it.

The concept facilitates also the reconstruction of cyber-physical components. For instance, when a segment of the robot is replaced by another one, only its local behavioral type has to be exchanged. If the type of the replacing component is a refinement of the one of the replaced component, one can assure certain spatial properties without demanding a new proof. E.g., if we could prove that a certain vehicle cannot collide with the robot since it never passes an area reachable by the robot arm, this property will also hold if a robot segment is replaced by a shorter one.

## 4 Core Concepts of Behavioral Types

We present some core concepts on behavioral types to support a development process of component based systems. To ease their application for users familiar with traditional *type systems*, (spatial) behavioral types should provide a number of features [14]:

- **Abstraction:** Behavioral types represent aspects of programs, components, or systems. They provide an *abstraction from details concerning interaction with their environment as well as their internal structure* [14]. For example, the driver of a physical component may require that the interaction with its environment follows a particular protocol that may form a part of the abstract view of the component provided by its behavioral type. According to the classification in [8], abstraction is a typical example of *behavioral contracts* that one can find, e.g., in UML and programming languages like Eiffel. It is also alike to the ESMs used in SPACE and Reactive Blocks [32] (see Sect. 2).
- **Type conformance:** Type conformance is used to relate a component to its own behavioral type. For instance, one can check whether a component interacting with its environment obeys the protocol listed in its behavioral type under all circumstances. These proofs can be taken at development time of a system, during changes of its component structure, or at runtime. Type conformance resembles classical static type checks performed by compilers of imperative programming languages.
- **Type refinement:** Behavioral types should support stepwise refinement, i.e., developing systems by making their models gradually more detailed using correctness preserving steps [1]. Thus, they have to be based on a formal semantics that allows to *ensure the correct implementation of abstract specifications by concrete components* [14]. This is quite similar to inheritance in object-oriented programming languages in which, as long as a program is developed “top-down”, first more abstract functionality is created using super classes, while later on the functionality is refined by applying inherited classes.
- **Type compatibility:** To facilitate the composition of components to systems, one has to check whether a component fulfills not only its one behavioral type but also those of its environment in order to guarantee that the components interact with each other in the desired way. This can be investigated at development time and at runtime when dynamically reconfiguring a system. The development time-based analysis is alike static type checks while

the dynamic tests resemble introspection in component-structured software (see [43]) as well as runtime monitoring.

- **Type inference:** As shown in the example description, subsystems can consist of various components. Often, type compatibility proofs can be easier if a subsystem is represented by a single behavioral type. Therefore, the formalism of behavioral types should *allow to infer the type of a composed component from the types of its constituents* [14]. Type inference corresponds to combining various software components to more comprehensive blocks resp. the composition of various building blocks to more extensive ones in Reactive Blocks [34].

To fulfill all these properties, the selected formalism has to take certain dependencies into consideration. For instance, type conformance has to guarantee with respect to type refinement that for a pair of components  $A$  and  $\hat{A}$  conforming to a pair of behavioral types  $T_A$  and  $T_{\hat{A}}$  with  $A$  subsuming the behavior of  $\hat{A}$ ,  $T_{\hat{A}}$  should be a refinement of  $T_A$ . Moreover, type refinement, type compatibility and type inference should agree that if a type  $T_A$  compatible to a given type  $T_B$  is refined by another type  $T_{\hat{A}}$ , also  $T_{\hat{A}}$  should be compatible to  $T_B$  by definition. Similarly, if in a composed type  $T_S$  one type  $T_A$  is replaced by a refined type  $T_{\hat{A}}$  leading to a new composed type  $T_{\hat{S}}$ , then  $T_{\hat{S}}$  should be a refinement of  $T_S$ . Also, for application in a development process, a behavioral type should not only be explicitly provided for a component and checked for conformance, but may be specifically constructed for this component. This is desirable in a seamless model-based development process. Finally, as type checking of expressive behavioral types is in general undecidable, an adequate level of expressiveness is needed making type checking feasible without over-restricting the expressiveness of the behavioral types.

We have studied behavioral types in the context of the OSGi framework [10–13]. Here, we use a simple finite automaton model  $(\Sigma, L, l_0, E)$  that consists of an alphabet  $\Sigma$  of labels, a set  $L$  of locations, an initial location  $l_0$  and a set  $E$  of transitions. A transition is a tuple  $(l, \sigma, l')$  with  $l, l' \in L$  and  $\sigma \in \Sigma$ . This formalism allows, e.g., runtime checking of type conformance [9]. Further, it is suitable to theorem proving as discussed in [14]. Nevertheless, the concept of behavioral types is suited to a diversity of formalisms. For instance, we currently experiment with temporal logic for cyber-physical systems (see [42]).

Behavioral types can be used for runtime-verification of systems, supplying a monitor being executed in parallel with a system implementation. The monitor corresponds to a behavioral type and checks all behavioral constraints specified via the type. It observes the system behavior and reports violations. The generation of the monitors from behavioral types can be performed automatically.

Furthermore, as already mentioned, the use of behavioral types facilitates the dynamic reconfiguration of systems based on type information and the discovery (both at runtime and development time) of components in a SOA like setting.



## 5 Spatial Behavioral Types

Spatial behavioral types extend the notion of behavioral types to cyber-physical components. Such components can comprise physical structures like the arms and tools attached to a robot as shown in the introductory example. Furthermore, controllers, network infrastructure elements, sensors and actuators are good candidates for spatial behavioral types.

Like purely software-related behavioral types, the spatial types comprise general aspects, e.g., protocols defining the interaction of a component with its environment. In addition, they define specific spatial aspects like the physical occupation of a physical components as well as its position, direction and speed. Depending on the application domain, further aspects as the acceleration of an object can be added. The representation of the spatial behavior is usually quite simple since we can restrict ourselves to describe positions by coordinates in the  $x$ ,  $y$  and  $z$  axes in the Euclidian space.

An advantage of this proceeding is that, similar to software components, we do not need to model a physical unit with its full complexity from scratch. Instead, we can start with relatively abstract spatial behavioral types that are stepwise refined to more complex ones until we finally get one that considers all relevant spatial properties of the real component. This facilitates the handling of complexity in the development process vastly. Moreover, we can verify crucial spatial type compatibility properties, e.g., freedom of collisions or keeping a certain distance between the robot tool and the vehicle when a good is loaded or unloaded, based on the abstract models.

Of course, we have to guarantee that these proofs stay valid also for the refined models. For that, we use over- and underapproximation of physical properties. For example, in abstract models of a component, we can overapproximate the spatial area occupied by a component. That allows us to perform type compatibility proofs already with the coarse-grained models that will also hold for more detailed ones as long as the refined models do not exceed the overapproximated elongation of the original ones. An example for underapproximation is the assumption of low sensor ranges in abstract models. Thus, we can refine the sensor models reusing proofs for the original ones as long as their ranges is not shorter than those of the refinements.

Behavioral types are applicable for the specification of software controlled cyber-physical entities. Furthermore, they can be used to describe other entities such as overapproximations of human behavior, or elements in an environment where a cyber-physical system is deployed. Such elements can comprise static descriptions of obstacles such as walls and pillars in closed spaces or open-environment features such as lakes and mountains or infrastructures such as roads or rail-lines.

## 6 Conclusion

We motivated spatial behavioral types as an advanced concept for type systems. In the context of component-based system development, spatial behavioral types

lift type systems to the component level also taking behavior of the underlying component into account. Thereby, they provide user-friendly means to specify and check contracts since easily comprehensible syntactical constructs are used, the core features of the method resemble techniques well-known from programming languages resp. component-based development, and the verification tools work automatically. In the past, realization of behavioral type systems for software was studied. In this publication, we propose the use of these type concepts also for components that comprise physical aspects which can be expressed using the concepts of Euclidian geometry.

Currently, we integrate spatial behavioral types into the model-based engineering technique SPACE [34] in order to facilitate the design of controllers for cyber-physical systems. Moreover, we work on the combination of spatial behavioral types with the verification tool-suite BeSpaceD [15] such that a highly automatic analysis of spatiotemporal properties will be possible. This will ease the application of the behavioral types for cyber-physical components further.

## References

1. Abadi, M., Lamport, L.: The existence of refinement mappings. *Theor. Comput. Sci.* **82**(2), 253–284 (1991)
2. Acciai, L., Boreale, M., Zavattaro, G.: Behavioural contracts with request-response operations. *Sci. Comput. Program.* **78**(2), 248–267 (2013)
3. de Alfaro, L., Henzinger, T.A.: Interface automata. In: *Symposium on Foundations of Software Engineering*. ACM (2001)
4. Arbab, F.: Abstract behavior types: a foundation model for components and their composition. In: de Boer, F.S., Bonsangue, M.M., Graf, S., de Roeper, W.-P. (eds.) *FMCO 2002*. LNCS, vol. 2852, pp. 33–70. Springer, Heidelberg (2003)
5. Barringer, H., Falcone, Y., Havelund, K., Reger, G., Rydeheard, D.: Quantified event automata: towards expressive and efficient runtime monitors. In: Gianakopoulou, D., Méry, D. (eds.) *FM 2012*. LNCS, vol. 7436, pp. 68–84. Springer, Heidelberg (2012)
6. Barringer, H., Goldberg, A., Havelund, K., Sen, K.: Rule-based runtime verification. In: Steffen, B., Levi, G. (eds.) *VMCAI 2004*. LNCS, vol. 2937, pp. 44–57. Springer, Heidelberg (2004)
7. Bauer, A., Leucker, M.: The theory and practice of SALT. In: Bobaru, M., Havelund, K., Holzmann, G.J., Joshi, R. (eds.) *NFM 2011*. LNCS, vol. 6617, pp. 13–40. Springer, Heidelberg (2011)
8. Beugnard, A., Jézéquel, J.-M., Plouzeau, N., Watkins, D.: Making components contract aware. *Computer* **32**(7), 38–45 (1999)
9. Blech, J.O.: Ensuring OSGi component based properties at runtime with behavioral types. In: *10th Workshop on Model Design, Verification and Validation Integrating Verification and Validation in MDE* (2013)
10. Blech, J.O.: Towards a Formalization of the OSGi Component Framework (2012). [arxiv.org/abs/1208.2563v1](https://arxiv.org/abs/1208.2563v1)
11. Blech, J.O.: Towards a framework for behavioral specifications of OSGi components. In: *10th International Workshop on Formal Engineering Approaches to Software Components and Architectures*. *Electronic Proceedings in Theoretical Computer Science* (2013)

12. Blech, J.O., Falcone, Y., Rueß, H., Schätz, B.: Behavioral specification based runtime monitors for OSGi services. In: Margaria, T., Steffen, B. (eds.) *ISoLA 2012, Part I*. LNCS, vol. 7609, pp. 405–419. Springer, Heidelberg (2012)
13. Blech, J.O., Rueß, H., Schätz, B.: On Behavioral Types for OSGi: From Theory to Implementation (2013). [arxiv.org/abs/1306.6115](http://arxiv.org/abs/1306.6115)
14. Blech, J.O., Schätz, B.: Towards a formal foundation of behavioral types for UML state-machines. In: *5th International Workshop UML and Formal Methods, Paris*. ACM SIGSOFT Software Engineering Notes (2012)
15. Blech, J.O., Schmidt, H.: Towards modeling and checking the spatial and interaction behavior of widely distributed systems. In: *Improving Systems and Software Engineering Conference, Melbourne* (2013)
16. Bliudze, S., Mavridou, A., Szymanek, R., Zolotukhina, A.: Coordination of software components with BIP: application to OSGi. In: *6th International Workshop on Modeling in Software Engineering*. ACM (2014)
17. Bodden, E., Hendren, L.: The clara framework for hybrid typestate analysis. *Int. J. Softw. Tools Technol. Transf. (STTT)* **14**, 307–326 (2012)
18. Bravetti, M., Zavattaro, G.: A theory of contracts for strong service compliance. *Math. Struct. Comput. Sci.* **19**(3), 601–638 (2009)
19. Cao, T.D., Phan-Quang, T.T., Félix, P., Castanet, R.: Automated runtime verification for web services. In: *International Conference on Web Services*. IEEE Computer Society (2010)
20. Castagna, G., Gesbert, N., Padovani, L.: A theory of contracts for web services. *ACM Tran. Program. Lang. Syst.* **31**(5), 1–61 (2009)
21. Cataño, N., Ahmed, I.: Lightweight verification of a multi-task threaded server: a case study with the plural tool. In: Salaün, G., Schätz, B. (eds.) *FMICS 2011*. LNCS, vol. 6959, pp. 6–20. Springer, Heidelberg (2011)
22. Chalin, P., Kiniry, J.R., Leavens, G.T., Poll, E.: Beyond assertions: advanced specification and verification with JML and ESC/Java2. In: de Boer, F.S., Bonsangue, M.M., Graf, S., de Roever, W.-P. (eds.) *FMCO 2005*. LNCS, vol. 4111, pp. 342–363. Springer, Heidelberg (2006)
23. Fiadeiro, J.L., Lopes, A.: Consistency of service composition. In: de Lara, J., Zisman, A. (eds.) *Fundamental Approaches to Software Engineering*. LNCS, vol. 7212, pp. 63–77. Springer, Heidelberg (2012)
24. Gan, Y., Chechik, M., Nejati, S., Bennett, J., O’Farrell, B., Waterhouse, J.: Runtime monitoring of web service conversations. In: *2007 Conference of the Center for Advanced Studies on Collaborative Research*. ACM (2007)
25. Hallé, S., Bultan, T., Hughes, G., Alkhalaf, M., Villemare, R.: Runtime verification of web service interface contracts. *Computer* **43**, 59–66 (2010)
26. Hamlen, K.W., Morrisett, G., Schneider, F.B.: Certified in-lined reference monitoring on.NET. In: *2006 Workshop on Programming languages and Analysis for Security*. ACM (2006)
27. Han, F., Blech, J.O., Herrmann, P., Schmidt, H.: Model-based engineering and analysis of space-aware systems communicating via IEEE 802.11. In: *To appear in 39th Annual International Computers, Software & Applications Conference (COMPSAC)*. IEEE Computer (2015)
28. Hatcliff, J., Leavens, G.T., Leino, K.R.M., Müller, P., Parkinson, M.: Behavioral interface specification languages. *ACM Comput. Surv.* **44**(3), 16:1–16:58 (2012). Article 16
29. Herrmann, P.: Trust-based protection of software component users and designers. In: Nixon, P., Terzis, S. (eds.) *iTrust 2003*. LNCS, vol. 2692, pp. 75–90. Springer, Heidelberg (2003)

30. Herrmann, P., Blech, J.O., Han, F., Schmidt, H.: A model-based toolchain to verify spatial behavior of cyber-physical systems. In: 2014 Asia-Pacific Services Computing Conference (APSCC). IEEE Computer (2014)
31. Johnsen, E.B., Hähle, R., Schäfer, J., Schlatte, R., Steffen, M.: ABS: a core language for abstract behavioral specification. In: Aichernig, B.K., Boer, F.S., Bonsangue, M.M. (eds.) Formal Methods for Components and Objects. LNCS, vol. 6957, pp. 142–164. Springer, Heidelberg (2011)
32. Kraemer, F.A., Herrmann, P.: Automated encapsulation of UML activities for incremental development and verification. In: Schürr, A., Selic, B. (eds.) MODELS 2009. LNCS, vol. 5795, pp. 571–585. Springer, Heidelberg (2009)
33. Kraemer, F.A., Herrmann, P.: Reactive semantics for distributed UML activities. In: Hatcliff, J., Zucca, E. (eds.) FMOODS 2010, Part II. LNCS, vol. 6117, pp. 17–31. Springer, Heidelberg (2010)
34. Kraemer, F.A., Slätten, V., Herrmann, P.: Tool support for the rapid composition, analysis and implementation of reactive services. *J. Syst. Softw.* **82**(12), 2068–2080 (2009)
35. Lee, E.A., Xiong, Y.: A behavioral type system and its application in ptolemy II. *Formal Aspects Comput.* **16**(3), 210–237 (2004)
36. Meyer, B.: Applying "design by contract". *Computer* **25**(10), 40–51 (1992)
37. Meredith, P.O., Jin, D., Griffith, D., Chen, F., Roşu, G.: An overview of the MOP runtime verification framework. *Int. J. Softw. Tech. Technol. Transfer* **14**, 249–289 (2011)
38. Prehofer, C.: Behavioral refinement and compatibility of statechart extensions. In: Formal Engineering Approaches to Software Components and Architectures. Electronic Notes in Theoretical Computer Science (2012)
39. Schneider, F.B.: Enforceable security policies. *ACM Trans. Inf. Syst. Secur.* **3**, 30–50 (2000)
40. Shaer, O., Hornecker, E.: Tangible user interfaces: past, present, and future directions. *Found. Trends Hum. Comput. Inter.* **3**(12), 1–137 (2010)
41. Souza, F., Lopes, D., Gama, K., Rosa, N., Lima, R.: Dynamic event-based monitoring in a SOA environment. In: Meersman, R., et al. (eds.) OTM 2011, Part II. LNCS, vol. 7045, pp. 498–506. Springer, Heidelberg (2011)
42. Spichkova, M., Blech, J.O., Herrmann, P., Schmidt, H.: Modeling spatial aspects of safety-critical systems with FOCUS<sup>ST</sup>. In: Model-Driven Engineering, Verification, and Validation in MDE, Satellite Event of MoDELS2014, CUR-WS Proceedings, vol. 1235, pp. 49–58, Valencia (2014)
43. Szyperski, C.: Component Software - Beyond Object Oriented Programming. Addison-Wesley Longman, New York (1997)
44. Wenger, M., Blech, J.O., Zoitl, A.: Behavioral type-based monitoring for IEC 61499. To appear in Emerging Technologies and Factory Automation (ETFA). IEEE (2015)

# Refactoring Proofs with Tactician

Mark Adams<sup>1,2</sup>(✉)

<sup>1</sup> Proof Technologies Ltd., Worcester, UK

Mark@proof-technologies.com

<sup>2</sup> Radboud University, Nijmegen, The Netherlands

**Abstract.** Tactician is a tool for refactoring tactic proof scripts for the HOL Light theorem prover. Its core operations are packaging up a series of tactic steps into a compact proof with tactical connectives, and the reverse operation of unravelling compact proofs into interactive steps. This can be useful for novices learning from legacy proof scripts, as well as for experienced users maintaining their proofs. In this paper, we give an overview of Tactician’s core capabilities and provide insight into how it is implemented.

## 1 Introduction

Although now over 30 years old, Paulson’s subgoal package [9] is still the predominant mode of interactive proof in various contemporary theorem provers, including Coq [4], HOL4 [10], HOL Light [6] and ProofPower [2], and is still used by some in the Isabelle [11] community. In recent years it was used extensively in the verification of the seL4 operating system microkernel [7] in Isabelle/HOL [8], and in the Flyspeck mathematics formalisation project [5] in HOL Light, for example.

Despite its widespread use, user facilities remain basic and lack useful extended features. One desirable facility is automated proof refactoring, for transforming a proof script into a more suitable form, where “more suitable” depends on the user’s needs, and might mean more compact, more efficient, more readable, more maintainable or easier to step through interactively. The understandability of various large proof scripts from Flyspeck, for example, would improve greatly if their tactic proofs could be cleaned up to be more coherent and easier to step through.

In this paper, we describe Tactician, a tool for refactoring HOL Light tactic proof scripts, explaining how key parts of its implementation work. In Sect. 2, we provide the motivation for automated proof refactoring. In Sect. 3, we show examples of Tactician’s two main refactoring operations. In Sect. 4, we explain how Tactician refactors proofs. In Sect. 5, we explain how it captures tactic proofs in state. In Sect. 6, we report on experiences and limitations. In Sect. 7, we present our conclusions. This paper extends an earlier workshop paper [1], by explaining the proof refactoring mechanism, filling out more detail about how tactic proofs are captured, and covering experiences and limitations.

Tactician is open source and can be downloaded from [12]. We explain how it works with extracts from its implementation in the OCaml dialect of ML, although these extracts are simplified for illustrative purposes.

## 2 Background and Motivation

The subgoal package is simple in concept and yet remarkably effective in practice. The user starts with a single main goal to prove. They then apply a series of tactic steps to break down the goal into hopefully simpler-to-prove subgoals. Each tactic application results in one or more new subgoals, or otherwise completes its subgoal, in which case focus shifts to the next subgoal. The proof is complete when each leaf in the tree of subgoals has been proved.

Behind the scenes, the subgoal package is keeping everything organised by maintaining a proof state that includes a list of remaining subgoals and a justification function for constructing the formal proof of the main goal from the formal proofs of the remaining subgoals. Tactics are implemented as functions that take a goal and return a subgoal list plus a justification function. The proof state is updated every time a tactic is applied, incorporating the tactic’s resulting subgoals and justification function. Once the proof is complete, a theorem stating the main goal can be extracted using the final justification function.

```

g ‘!f:A->B. (!y. ?x. f x = y) <=> (!P. (?x. P(f x)) <=> (?y. P y))’;;
e (GEN_TAC);;
e (EQ_TAC);;
e (MESON_TAC []);;
e (DISCH_THEN (MP_TAC o SPEC ‘\y:B. !x:A. ~(f x = y)’));;
e (MESON_TAC []);;
top_thm ();;

```

**Fig. 1.** An interactive “g and e” style proof of HOL Light’s SURJECTIVE\_EXISTS\_THM.

The user writes their proof script as a series of commands in the ML programming language. The first draft is typically written in the interactive proof style, which starts with the command to set the proof goal (called **g** in HOL Light), proceeds with a potentially long series of tactic steps each using the tactic application command (called **e** in HOL Light), and often ends with a command to extract the theorem result (called **top\_thm** in HOL Light), as illustrated in Fig. 1. This version is usually not beautiful, but does the job of proving the theorem. For clarity, the user may have inserted indentations and/or ML comment annotations to make explicit where the proof branches, but if not the proof script does not convey any information about the structure of the proof. Worse still, the proof steps may be interspersed with other, unrelated ML commands in the script, further obscuring the proof.

What often happens next is that the proof script is cleaned up, or *refactored*, to become more succinct. This refactoring will typically involve packaging-up

the tactic steps into a single, compound tactic, and then using the batch proof command (called `prove` in HOL Light) for performing the completed proof and extracting the theorem result, as illustrated in Fig. 2. If done well, the resulting compound tactic will be neater and more concise because it can factor out repeated use of tactics, for example when the same tactic is applied to each subgoal of a given goal.

```

prove
  ('!f:A->B. (!y. ?x. f x = y) <=> (!P. (?x. P(f x)) <=> (?y. P y))',
   GEN_TAC THEN
   EQ_TAC THENL
     [ALL_TAC; DISCH_THEN (MP_TAC o SPEC '\y:B. !x:A. ~(f x = y)')] THEN
   MESON_TAC []);;

```

**Fig. 2.** A batch “prove” style proof of `SURJECTIVE_EXISTS_THM`.

Tactic connectives are used to package up steps into compound tactics. The binary `THEN` connective applies its right-hand side (RHS) tactic to all of the subgoals resulting from its left-hand side (LHS) tactic. The binary `THENL` differs in that its RHS argument is a list of tactics rather than a single tactic, where the  $n$ th tactic in this list gets applied to the  $n$ th subgoal resulting from the LHS tactic. The identity tactic `ALL_TAC` makes no change to the goal, which can be useful in the RHS list of a `THENL` to make a branch skip the list.

These packaged-up proofs feature heavily in the source code building up the standard theory library of the HOL4 and HOL Light systems. They were also used in Flyspeck for a few years, because they bring the proof together as a single ML statement, making it easier to manage.

In light of this, one can see that a capability for automatically refactoring tactic proof scripts between the interactive “g and e” style and the packaged-up “prove” style would be valuable to the user, for three reasons:

**Tidying.** Having created a new interactive proof, the process of tidying it and packaging it up into a batch proof can be long and tedious, and for proofs that run into dozens of lines it can be easy to miss opportunities to make the proof more concise or readable. Doing this automatically could both save effort and result in better proofs;

**Learning.** Most of the best examples of tactic proofs are stored as packaged-up batch proofs. Novice users currently have to laboriously unravel these into interactive proofs if they want to step through these masterpieces and learn how the experts prove their theorems. This can be even more tedious than packaging a proof up, because the user does not know which tactics apply to more than one subgoal and thus need to occur more than once in the unpackaged proof. Automation would improve access to the wealth of experience that is held in legacy proof scripts;

```

prove
  ('!x. &0 < x ==> &0 < inv(x)',
  GEN_TAC THEN
  REPEAT_TCL DISJ_CASES_THEN
    ASSUME_TAC (SPEC 'inv(x)' REAL_LT_NEGTOTAL) THEN
  ASM_REWRITE_TAC[] THENL
    [RULE_ASSUM_TAC(REWRITE_RULE[REAL_INV_EQ_0]) THEN ASM_REWRITE_TAC[];
    DISCH_TAC THEN
    SUBGOAL_THEN '&0 < --(inv x) * x' MP_TAC THENL
      [MATCH_MP_TAC REAL_LT_MUL THEN ASM_REWRITE_TAC[];
      REWRITE_TAC[REAL_MUL_LNEG]]] THEN
  SUBGOAL_THEN 'inv(x) * x = &1' SUBST1_TAC THENL
    [MATCH_MP_TAC REAL_MUL_LINV THEN
    UNDISCH_TAC '&0 < x' THEN REAL_ARITH_TAC;
    REWRITE_TAC [REAL_LT_RNEG; REAL_ADD_LID; REAL_OF_NUM_LT; ARITH]]);;

```

**Fig. 3.** The original batch proof of `REAL_LT_INV` from HOL Light’s `real.ml`.

**Maintenance.** Proofs need to be maintained over time, due to changes in the theory context in which the theorems are proved. If the proofs are packaged up, then they will need to be unpackaged, debugged and then repackaged, which again would be considerably easier with automated support.

### 3 Usage

Tactician can be loaded into a HOL Light session at any stage, and from that point onwards silently captures proofs as they are executed, outputting refactored versions upon request. We illustrate its usage here with `REAL_LT_INV`, proved in the HOL Light standard theory library (see Fig. 3).

#### 3.1 Proof Unravelling

Trying to manually unpick the original packaged proof of `REAL_LT_INV` is an arduous task. From looking at the text of the script, use of `THENL` reveals that the proof branches at various points, but it is not clear exactly where these branches start, which tactics get applied to more than one branch, and which branches are finished off within the RHS of the `THENL` connectives rather than carry on further into the proof script.

Using Tactician, the proof can be automatically unravelled into “g and e” style, with branch points optionally annotated to reveal its true structure (see Fig. 4). We can see that the application of `REPEAT_TCL` resulted in three subgoals, with the first finishing in the first branch of the first `THENL`, the second finished off by `ASM_REWRITE_TAC` prior to the first `THENL`, and the third continuing beyond the second branch of the first `THENL` to split and continue to the end of the packaged proof. Once expressed in “g and e” style, the proof is ready to be replayed interactively by stepping through individual tactic applications.



```

g '!x. &0 < x ==> &0 < inv(x)';;
e (GEN_TAC);;
e (REPEAT_TCL DISJ_CASES_THEN
  ASSUME_TAC (SPEC 'inv x' REAL_LT_NEGTOTAL));;
(* *** Branch 1 *** *)
e (ASM_REWRITE_TAC []);;
e (RULE_ASSUM_TAC (REWRITE_RULE [REAL_INV_EQ_0]));;
e (ASM_REWRITE_TAC []);;
(* *** Branch 2 *** *)
e (ASM_REWRITE_TAC []);;
(* *** Branch 3 *** *)
e (ASM_REWRITE_TAC []);;
e (DISCH_TAC);;
e (SUBGOAL_THEN '&0 < --inv x * x' MP_TAC);;
(* *** Branch 3.1 *** *)
e (MATCH_MP_TAC REAL_LT_MUL);;
e (ASM_REWRITE_TAC []);;
(* *** Branch 3.2 *** *)
e (REWRITE_TAC [REAL_MUL_LNEG]);;
e (SUBGOAL_THEN 'inv x * x = &1' SUBST1_TAC);;
(* *** Branch 3.2.1 *** *)
e (MATCH_MP_TAC REAL_MUL_LINV);;
e (UNDISCH_TAC '&0 < x');;
e (CONV_TAC REAL_ARITH);;
(* *** Branch 3.2.2 *** *)
e (REWRITE_TAC [REAL_LT_RNEG;REAL_ADD_LID;REAL_OF_NUM_LT;ARITH]);;

```

Fig. 4. An interactive version of REAL\_LT\_INV.

### 3.2 Proof Packaging

Tactician can automatically package up a “g and e” style proof into a batch proof, identifying opportunities to factor out repeated use of tactics to make the batch proof more concise. We concentrate here on how it can do a better job at making the batch proof readable than what has been done manually in the HOL Light standard theory library.

A good technique to be used in packaging up a tactic proof is to make the main branch of the proof clear by separating it out from minor branches that get discharged in relatively few steps. This is done by completing any minor branches within the RHS of a THENL connective as soon as they arise in the proof, but keeping the proof of the main branch outside of this by performing a null step in its corresponding place in the THENL RHS by use of the identity tactic ALL\_TAC. Thus all subsequent steps after the THENL RHS are concerned only with the main branch, and the THENL RHS is only concerned with the minor branches.

This technique is used throughout the HOL Light standard theory library, although various opportunities to use it have been missed, as is the case with

```

prove
  ('!x. &0 < x ==> &0 < inv(x)',
  GEN_TAC THEN
  REPEAT_TCL DISJ_CASES_THEN
    ASSUME_TAC (SPEC 'inv(x)' REAL_LT_NEGTOTAL) THEN
  ASM_REWRITE_TAC [] THENL
    [RULE_ASSUM_TAC (REWRITE_RULE [REAL_INV_EQ_0]) THEN
    ASM_REWRITE_TAC [];
    ALL_TAC] THEN
  DISCH_TAC THEN SUBGOAL_THEN '&0 < --(inv x) * x' MP_TAC THENL
    [MATCH_MP_TAC REAL_LT_MUL THEN ASM_REWRITE_TAC []; ALL_TAC] THEN
  REWRITE_TAC [REAL_MUL_LNEG] THEN
  SUBGOAL_THEN 'inv(x) * x = &1' SUBST1_TAC THENL
    [MATCH_MP_TAC REAL_MUL_LINV THEN UNDISCH_TAC '&0 < x' THEN
    REAL_ARITH_TAC;
    REWRITE_TAC [REAL_LT_RNEG; REAL_ADD_LID; REAL_OF_NUM_LT; ARITH]]);;

```

**Fig. 5.** A repackaged version of `REAL_LT_INV`, making use of `ALL_TAC`.

`REAL_LT_INV`. We can see from its unravelled proof in Fig. 4 that Branch 3 and subsequently Branch 3.2 represent the main branch of the proof, although from Fig. 3 we can see that these branches are partly worked on inside the RHSs of `THENL` connectives and partly worked on outside.

As can be seen in Fig. 5, when using `Tactician` to package up the unravelled proof, the two opportunities to separate out the main branch are not missed, as is evident by the two occurrences of `ALL_TAC`. Thus the packaged proof is easier to follow, as well as keeping a better shape.

### 3.3 Output

The most common usage of `Tactician` is to output to screen an individual refactored proof. However, there are various extended features to support the management of large proof projects. It is capable of bulk processing a file of proof scripts and exporting refactored versions of each to disk. There are also commands for exporting the graph of a proof to help visualise its goal tree, for presenting statistics about each proof including size and usage metrics, and for exporting a graph showing the dependencies between theorems.

## 4 Implementing Proof Refactoring

We now explain the mechanism by which tactic proofs are refactored. This is based around combining basic transformation operations performed on an abstract representation of a tactic proof called a *hiproof*.

## 4.1 Hiproofs

In order for a proof script to be refactored, it should first be represented in a suitable form that holds all necessary information about its structure, so that it can be correctly transformed, and sufficient information about its components, so that the result can be outputted as a standalone proof script that can be executed as a substitute for the original.

Hiproofs (see [3]) are an algebraic representation of tactic proofs that we find are at just the right level of abstraction for our purposes. They represent a proof in terms of the goal structure resulting from the execution of the proof, with each goal represented by the tactic applied to it. There is an atomic class for a tactic application on an individual goal, an identity hiproof for a null tactic application, an empty hiproof for a completed subgoal, a binary sequential composition operator written as semicolon, a binary tensor operator  $\otimes$  for grouping hiproofs in parallel (for goals with multiple subgoals) and a labelling operator. The grammar is given by:

$$h ::= \textit{tactic} \mid \textit{id} \mid \textit{empty} \mid h;h \mid h \otimes h \mid [\textit{label}] h$$

Our representation differs from the algebra in [3] to better fit implementation of proof refactoring. There is an additional atomic class for an active subgoal, to cater for incomplete proofs. Another difference is that our tensor operator groups a list of hiproofs rather than a pair, enabling the empty hiproof to be replaced with a tensor of length 0. Note that we write our hiproof notation differently, so that tensor is written with angled brackets and comma separators.

$$h ::= \textit{tactic} \mid \textit{active} \mid \textit{id} \mid h;h \mid \langle h, h, \dots \rangle \mid [\textit{label}] h$$

In Tactician, the labelling construct is used differently from its original intended use, where the label is supposed to represent an abstract view of a subproof that can be “zoomed in” to show the detail of the hiproof inside if desired. Instead we use labelling primarily to capture usage of the **THEN** and **THENL** tactic connectives (which both represent sequential composition).

In Fig. 6, we show a toy example of a packaged proof script and its hiproof representation. Note that, unlike the proof script, the hiproof conveys the full structure of the proof, explicitly showing that **REFL\_TAC** gets applied to the two subgoals resulting from **CONJ\_TAC**. For illustrative purposes, we have kept the parentheses around compound sequential composition. Note that these brackets accumulate on the left in packaged proofs, because the **THEN** and **THENL** connectives are left-associative, because they are normally used to apply their RHS argument to all of the remaining subgoals in the proof.

In Fig. 7, we show an interactive version of the toy example. The hiproof has no labels since tactic connectives are not used. Note that the sequential composition brackets accumulate on the right in “g and e” style proofs, because the LHS argument reflects a single goal in interactive proof, and the RHS argument gets applied just to its subgoals.

Each hiproof has an input and output arity, corresponding to the number of input and output goals of the hiproof respectively. To be well-formed, the

```

prove ('!x. x = x /\ x = x',
      GEN_TAC THEN CONJ_TAC THEN REFL_TAC');;

[Label THEN]
  ([Label THEN] (Tactic GEN_TAC; Tactic CONJ_TAC);
   <Tactic REFL_TAC, Tactic REFL_TAC>)

```

**Fig. 6.** A toy example of a packaged proof, together with its hiproof.

```

g '!x. x = x /\ x = x';;
e (GEN_TAC);;
e (CONJ_TAC);;
e (REFL_TAC);;
e (REFL_TAC);;

Tactic GEN_TAC; (Tactic CONJ_TAC; <Tactic REFL_TAC, Tactic REFL_TAC>)

```

**Fig. 7.** An interactive version of the toy proof, together with its hiproof.

output arity of the LHS of a sequential composition must equal the input arity of the RHS. The arity of a hiproof can be calculated from its atoms. Tactic application has input arity 1 and output arity depending on the application. An active subgoal has input arity 1 and undefined output arity. The identity hiproof has input and output arity 1. Sequential composition has input arity of the LHS hiproof and output arity of the RHS hiproof. Tensor has input and output arity of the sum of its hiproofs' input and output arities respectively. And a labelled hiproof has the input and output arity of its hiproof.

## 4.2 Hiproof Transformations

The insight into the different way in which the brackets accumulate between packaged and interactive proofs provides the basis for the refactoring operations. To convert between one style and the other, the brackets need to be shifted from one side to the other. In reality, the refactoring operations have to do much more than simply shift brackets, and are compositions are various primitive refactoring transformations, but the shifting of brackets is the most fundamental aspect.

**Unravelling a Hiproof.** To refactor into an unravelled proof, it is first necessary to remove any labels from the hiproof, that capture the use of **THEN** and **THENL** connectives, that would be present if the original proof were a packaged proof. Then it is possible to recursively apply the associativity rule for sequential composition in the left-to-right direction (1), to group the sequential composition brackets on the right.

$$(h1; h2); h3 \longrightarrow h1; (h2; h3) \tag{1}$$

However, the ultimate aim is to output an interactive proof, which cannot involve tensors as the LHS of a sequential composition. Thus part of the right-grouping operation is to remove any such tensors by distributing them through their sequential composition RHS (2). This distribution must preserve the well-formedness of the hiproof, and the components of the resulting tensor must be lined-up so that arities match (thus  $h3a$  below is not necessarily simply  $h1a; h2a$ ).

$$\langle h1a, h1b, .. \rangle; \langle h2a, h2b, .. \rangle \longrightarrow \langle h3a, h3b, .. \rangle \quad (2)$$

The hiproof is then ready to be printed out as ML, with the option of inserting a comment for every branching point to show the structure of the proof.

**Packaging-Up a Hiproof.** To refactor into a packaged proof, we first remove any labels, since the original proof might also be a packaged proof. The next task is to compact the hiproof by spotting opportunities for performing the same tactic in parallel to each component of a tensor, enabling THEN to be used in place of THENL and thus making the proof more concise. These common tactics can be at the start or the end of tensors, corresponding to the initial few tactics after a proof branches or the final few tactics at the end of each of a branch’s subbranches respectively. Thus we first group sequential composition brackets on the right and compare initial segments of a tensor’s hiproofs, separating out any common initial segments. We then do the same for common final segments of a tensor’s hiproofs by left-grouping sequential composition brackets, using the associativity rule in the right-to-left direction (3).

$$(h1; h2); h3 \longleftarrow h1; (h2; h3) \quad (3)$$

Next, if there is a branch in a tensor that is much longer than its sibling branches, then it is extracted out of the tensor using the ALL\_TAC technique explained in Sect. 3.2. The heuristic we use for identifying a “much longer” branch is that the branch has size of at least five, and this size is at least three times the maximum size of its siblings, where size is calculated as the number of ML atoms in its ML text. Experience has shown that this heuristic delivers good results. Having applied the heuristic, the hiproof has been optimised with separated tensors for parallel application of the same tactic. Each sequential composition is then labelled with THENL unless its RHS has input arity 1 or if the RHS is a tensor with the same tactic as each component, in which case it is labelled as THEN. The packaged hiproof is then ready to be printed out as ML.

## 5 Implementing Tactic Recording

In the last section, we explained how a tactic proof represented as a hiproof is refactored in Tactician. However, this assumes that the proof has been appropriately captured in program state. In this section, we explain how Tactician captures proofs, which accounts for the bulk of its implementation. First we briefly discuss the requirements for capturing tactic proofs in a suitable form.

## 5.1 Requirements for Capturing Tactic Proofs

It is not particularly important that the original text of the proof can be recreated in every last detail, including those parts that are redundant or that don't get executed. For our purposes, we are more interested in what does get executed, and what happens when it gets executed, which tells us more about which proof refactorings would be valid. Thus capturing the proof by a static syntactic transformation of the original proof script would not suit our purposes. Rather, the proof needs to somehow be dynamically recorded as it is executed, to capture what is actually used. We call this *tactic proof recording*.

Note that the subgoal package already dynamically captures interactive tactic proofs, simply as a list of subgoals (or actually a stack of such lists, so that interactive steps can be undone if required). However, this form is not suitable for our purposes because it does not explicitly capture the structure of the tree of goals, and neither does it carry the names of tactics used or their arguments, which we require in order to output a proof script.

We identify seven main requirements for our tactic recording mechanism:

1. To fully capture all the information needed to recreate a proof script;
2. To capture the parts of the proof script that actually get used;
3. To capture the information in a form that suitably reflects the full structure of the original proof, including the structure of the goal tree and hierarchy corresponding to the explicit use of tactic connectives in the proof script;
4. To capture information at a level that is meaningful to the user, i.e. with atoms corresponding to the ML binding names for the tactics and other objects mentioned in the proof script;
5. To be capable of capturing both complete and incomplete proofs;
6. To work both for both interactive and packaged-up proofs;
7. To work for legacy proofs, without requiring modification to the original proof script.

## 5.2 The Basic Recording Mechanism

Our recording mechanism is designed to meet the above requirements. It maintains a proof tree in program state, in parallel with the subgoal package's normal state. The proof tree has nodes corresponding to the initial and intermediate goals in the proof, and branches from each node corresponding to the goals' subgoals, reflecting the structure of the executed proof. Each node carries information about its goal, including a statement of the goal, an abstract representation of the ML text of the tactic that got applied to the goal, and a unique goal identity number. Active subgoals are labelled as such in place of the tactic's abstract ML text, thus enabling incomplete proofs to be represented. The tree gets added to as interactive tactic steps are executed, and deleted from if steps are undone. A hiproof that captures the proof's structure and the ML text of the tactic applied in each goal can thus be dumped from the proof tree at any stage, from which the proof can be refactored and printed (see Sect. 4).

The crucial means by which the stored proof tree is updated in line with the subgoal package state is based on the goal ids. HOL Light's existing datatype for goals is extended to carry such an identity number. These identity-carrying goals are called *xgoals*.

```
type goalid = int;;
type xgoal = goal * goalid;;
```

Tactics are adjusted, or *promoted*, to work with xgoal inputs and outputs. These promoted tactics, called *xtactics*, have a datatype that is a trivial variant of HOL Light's original, with xgoals instead of goals. Proofs are performed using xtactics in place of tactics. The pretty printer for xgoals is written to ignore the goal id and print xgoals like goals, so that users see normal feedback when performing subgoal package proofs.

```
type xgoalstate = xgoal list * justification;;
type xtactic = xgoal → xgoalstate;;
```

The implementation of a promoted tactic first involves breaking up its xgoal argument into the original unpromoted goal and its goal id. The original, unpromoted tactic then gets applied to the unpromoted goal to result in a list of new subgoals and a justification function. These new subgoals are promoted into xgoals with unique ids, which then get inserted as branches of the node in the proof tree at the location determined by the input goal's id, along with abstract ML text for the tactic, based on the tactic's name. The promoted new subgoals and the justification function are returned as the result of the xtactic.

Rather than individually promote each tactic, we write a generic wrapper function for automatically promoting a supplied tactic of a given ML type, that takes the name of the tactic and the tactic itself as arguments. Below is the wrapper function for basic tactics that take no other arguments. The local value `obj` captures the abstract ML text of the tactic (see Sect. 5.3).

```
let tactic_wrap name (tac:tactic) : xtactic =
  fun (xg:xgoal) →
  let (g,id) = dest_xgoal xg in
  let (gs,just) = tac g in
  let obj = Mname name in
  let xgs = extend_gtree id (Gatom obj) gs in
  (xgs,just);;
```

This wrapper gets used to overwrite unpromoted tactics with their promoted versions, so that existing proof scripts can be replayed without adjustment.

```
let REFL_TAC = tactic_wrap"REFL_TAC" REFL_TAC;;
let STRIP_TAC = tactic_wrap"STRIP_TAC" STRIP_TAC;;
```

It is necessary to write wrapper functions for each ML type of tactic that can occur in a proof script. As the datatypes become more complex, so does the implementation of their corresponding wrapper functions. Slightly more complex than `tactic_wrap` is `term_tactic_wrap`, a function for promoting tactics that take

a term argument. Its implementation is similar to `tactic.wrap`, except that here `obj` has the expression syntax of an ML name binding applied to a HOL term, enabling a refactored proof to refer to the term argument supplied in the proof.

```
let term_tactic_wrap name (tac:term→tactic) : term→xtactic =
  fun (tm:term) (xg:xgoal) →
  let (g,id) = dest_xgoal xg in
  let (gs,just) = tac g in
  let obj = Mapp (Mname name, [Mterm tm]) in
  let xgs = extend_gtree id (Gatom obj) gs in
  (xgs,just);;
```

Similar wrapper functions can be written for tactics that take other basic arguments, such as integers, strings or HOL types, or even structures of such arguments, such as a list of terms. Each time, the form of the `obj` local value varies to reflect the ML text of the tactic written with its arguments.

### 5.3 The Abstract ML Datatype

We use a datatype for capturing the ML text of how a tactic is written in a proof script, so that we can print out our refactored proof. We want this datatype to be capable of representing all the ML expression forms that commonly occur in tactic proofs, and to represent these as abstract syntax, so that we can easily perform syntax-based operations on these expressions when needed. Thus we define our recursive `mobject` datatype. It is capable of referring to ML binding names, literals of basic datatypes (such as integers, strings, booleans, HOL terms and HOL types), structures (such as lists and tuples), function application, anonymous functions and local variables. It also supports the function composition operator separately, since this is often treated specially.

```
type mobject =
  Mname of string
| Mint of int
| Mstring of string
| Mbool of bool
| Mterm of term
| Mtype of hol_type
| Mtuple of mobject list
| Mlist of mobject list
| Mapp of mobject * mobject list
| Mfcomp of mobject list
| Mlambda of lvarid * mltype * mobject
| Mlvar of lvarid * mltype;;
```

### 5.4 Capturing Theorems

HOL theorems have no literal representation in the `mobject` datatype. This is because, unlike for HOL types and HOL terms, printing the value of a theorem



in an outputted proof script is of no use to the user, because it cannot be parsed in to recreate the theorem (in LCF-style theorem provers, at least). Instead it is necessary to print a theorem using its ML binding name, if it has one, or otherwise the ML text of its proof. Thus the theorem datatype `thm` is extended to carry abstract ML text. We call these extended theorems `xthms`. Like for `xgoals`, the pretty printer for `xthms` ignores the extension and prints an `xthm` in the same way as a `thm`, so that users see normal theorems when they view `xthms`.

```
type xthm = thm * mobject;;
```

Each existing named theorem is promoted to have the `xthm` datatype, with its ML binding name for its abstract ML text, using the following wrapper:

```
let thm_wrap name (th:thm) : xthm =
  let obj = Mname name in
  mk_xthm (th,obj);;
```

All functions that take or return theorems also need to be promoted to work with `xthms`. We write wrapper functions for promoting any such functions of a given type, in a similar way to the wrapper functions for promoting tactics, although `xthm` wrappers are simpler to write because they do not need to incorporate their results into the proof tree.

```
let term_rule_wrap name (r:term→thm→thm) : (tm→xthm→xthm) =
  fun (tm:term) (xth:xthm) →
  let (th0,obj0) = dest_xthm xth in
  let th = r tm th0 in
  let obj = Mapp (Mname name, [Mterm tm; obj0]) in
  mk_xthm (th,obj);;
```

## 5.5 Automated Promotion

To avoid the user having to adapt proof scripts to explicitly promote ML values, Tactician automatically promotes all values. For values existing in the ML session when Tactician is loaded, this is done by executing a promotion function for each value in the OCaml environment, which is available to the user via the OCaml `Toploop` module and use of `Obj.magic`.<sup>1</sup> For values entered after Tactician has loaded, this is done by adjusting the OCaml `toploop` to execute a hook each time a value is added in the session.

## 6 Experiences and Limitations

Tactician has been tested on tens of thousands of lines of proof script files from the HOL Light theory libraries, including the Multivariate library, and the Flyspeck project. Tests involve running a proof script through a HOL Light session

<sup>1</sup> This is similar to a trick used in HOL Light to capture the theorem values in the ML session, implemented in HOL Light's `update_database.ml`.

with Tactician loaded, exporting the refactored proof script from the session, and replaying the exported script in a separate HOL Light session.

In typical usage, Tactician comfortably handles the large (30 lines) and very large (100+ lines) proof scripts found in HOL Light and Flyspeck. For example NADD\_COMPLETE (100 lines) from the standard theory library’s `rearith.ml`, and PROPER\_MAP (115 lines) from Multivariate’s `topology.ml`. However, processing thousand-line proof script files often fails, not due to the size of the script, but due to the likelihood of hitting one of the four current limitations of Tactician. We list the limitations in decreasing order of their frequency of occurrence:

**Concrete Manipulation of the Goal.** Since Tactician works on a different concrete goal datatype, it cannot process ML that uses the usual concrete constructors and destructors that manipulate goals as ML pairs (e.g. `fst` and `snd`). These need to be replaced with the abstract goal constructors and destructors provided by Tactician (e.g. `goal_hyp` and `goal_concl`).

**ML Type Annotations.** Tactician cannot process ML with type annotations for the “promotable” datatypes (i.e. `thm`, `conv`, `goal`, `tactic`, etc.). Any such type annotations must either have such datatypes changed to their promoted equivalents (i.e. `xthm`, `xconv`, `xgoal`, `xtactic`, etc.), or be removed.

**Promotion of Obscure ML Datatypes.** Tactician cannot automatically promote bindings with some obscure ML datatypes. However, these are very rarely encountered, and it is easy to write new promotion functions for them.

**Promotion of “Unpromotable” ML Datatypes.** Tactician cannot properly promote bindings that take a function as an argument if the function does not return a promotable datatype. The only common example of this in HOL Light is `PART_MATCH`, whose first argument is a term-to-term function. It outputs “<???” in place of this argument, and warns the user.

We have found that it is always easy to overcome these limitations, either by manually adjusting the files, or, in the case of obscure ML datatypes, to extend Tactician. This usually takes about a minute per thousand-line file that fails, or a few minutes for multi-thousand line files. Tactician has been enhanced over the past few years to solve other limitations, and there is prospect of further enhancement to reduce further the frequency of hitting limitations. For example, the ML type annotations limitation can be solved by giving promoted datatypes the same name as their unpromoted equivalents, since there is no need for the unpromoted datatypes at the user level once Tactician has been loaded.

## 7 Conclusions

Tactic proofs are still used extensively in theorem proving, including in recent ground-breaking projects. However, there is little to help users refactor tactic proof scripts for maintenance or understanding. Tactician is a tool designed specifically to cater for this need.

In this paper we have explained the basics of how Tactician works, including how it captures proofs in memory, and how it transforms these proofs into suitable refactored forms. These facilities have been shown to work on the largest

proof scripts from the HOL Light theory libraries and the Flyspeck project, although four current limitations mean that it is often necessary to perform a minute or two of manual preparation on large proof script files.

The principles explained have been implemented for the HOL Light system, but also apply to any implementation of the subgoal package. Most parts of Tactician should be straightforward to port to other systems, although the automatic promotion mechanism is specific to OCaml implementations, and adaptations to the subgoal package main functions are specific to HOL Light.

## References

1. Adams, M., Aspinall, D.: Recording and refactoring HOL light tactic proofs. In: Workshop on Automated Theory eXploration, in Association with the 6th International Conference on Automated Reasoning (2012)
2. Arthan, R., Jones, R.: Z in HOL in ProofPower. In Issue 2005-1 of the British Computer Society Specialist Group Newsletter on Formal Aspects of Computing Science (2005)
3. Aspinall, D., Denney, E., Lüth, C.: Tactics for hierarchical proof. *Math. Comput. Sci.* **3**(3), 309–330 (2010). Springer
4. Bertot, Y., Casteran, P.: Interactive Theorem Proving and Program Development: Coq’Art: The Calculus of Inductive Constructions. Texts in Theoretical Computer Science an EATCS Series. Springer, Heidelberg (2004)
5. Hales, T., Adams, M., Bauer, G., Dat, T.D., Harrison, J., Truong, H.L., Kaliszzyk, C., Magron, V., McLaughlin, S., Thang, N.T., Truong, N.Q., Nipkow, T., Obua, S., Pleso, J., Rute, J., Solovyev, A., An, T.H., Trung, T.N., Diep, T.T., Urban, J., Ky, V.K., Zumkeller, R.: A Formal Proof of the Kepler Conjecture. [arXiv:1501.02155v1](https://arxiv.org/abs/1501.02155v1) [math.MG]. [arxiv.org](https://arxiv.org/abs/1501.02155v1) (2015)
6. Harrison, J.: HOL light: an overview. In: Berghofer, S., Nipkow, T., Urban, C., Wenzel, M. (eds.) TPHOLs 2009. LNCS, vol. 5674, pp. 60–66. Springer, Heidelberg (2009)
7. Klein, G., Elphinstone, K., Heiser, G., Andronick, J., Cock, D., Derrin, P., Elkaduwe, D., Engelhardt, K., Kolanski, R., Norrish, M., Sewell, T., Tuch, H., Winwood, S.: seL4: formal verification of an OS Kernel. In: Proceedings of the ACM SIGOPS 22nd Symposium on Operating Systems Principles, pp. 207–220. ACM (2009)
8. Nipkow, T., Paulson, L., Wenzel, M.: Isabelle/HOL: A Proof Assistant for Higher-Order Logic. LNCS, vol. 2283. Springer, Heidelberg (2002)
9. Paulson, L.: Logic and Computation: Interactive proof with Cambridge LCF. Cambridge University Press, Cambridge (1987)
10. Slind, K., Norrish, M.: A brief overview of HOL4. In: Mohamed, O.A., Muñoz, C., Tahar, S. (eds.) TPHOLs 2008. LNCS, vol. 5170, pp. 28–32. Springer, Heidelberg (2008)
11. Wenzel, M., Paulson, L.C., Nipkow, T.: The Isabelle framework. In: Mohamed, O.A., Muñoz, C., Tahar, S. (eds.) TPHOLs 2008. LNCS, vol. 5170, pp. 33–38. Springer, Heidelberg (2008)
12. Tactician homepage: <http://www.proof-technologies.com/tactician/>

# Exploring the Role of Logic and Formal Methods in Information Systems Education

Anna Zamansky<sup>1,2</sup>(✉) and Eitan Farchi<sup>1,2</sup>

<sup>1</sup> University of Haifa, Haifa, Israel  
annaz@is.haifa.ac.il

<sup>2</sup> IBM Research Labs, Haifa, Israel  
farchi@il.ibm.com

**Abstract.** This position paper contributes to the ongoing debate on the role played by logic and formal methods courses in the computing curricula. We report on an exploratory empirical study investigating the perceptions of Information Systems students on the benefits of a completed course on logic and formal specification. Participants indicated that the course had fostered their analytical thinking abilities and provided them with tools to handle abstraction and decomposition. This provides a starting point for a discourse on the benefits of formal methods courses for IS practitioners.

## 1 Introduction

Several ways in which formal methods should be incorporated into the computing curricula have been proposed (see, e.g., [1, 10, 12–16, 18, 19]), but there is still no consensus on what and how to teach. Moreover, there is an ongoing debate on the role of the very foundations of formal methods - logic and discrete mathematics in CS education. While the early CS curricula were strongly mathematically oriented, many voices are recently calling for a less mathematically rigorous curriculum, claiming this type of knowledge is not really used by practitioners in industry [2, 8]. In the Information Systems discipline, which draws a large portion of its body of knowledge from CS, the importance of logic and formal methods is even less recognized. Indeed, the ACM IS curriculum guidelines [17] mention statistics and probability as required core IS topics and discrete mathematics only as an optional one, leaving logic and formal methods outside mainstream IS topics.

We believe that this state of affairs is most unfortunate. Although we tend to agree that a typical IS major may need a less extensive mathematical background than a CS major, formal methods are an essential tool for software quality control, i.e., activities for checking (by proof, analysis or testing) that a software system meets specifications and that it fulfills its intended purpose. But - even more importantly - learning formal methods fosters analytical thinking, provides the tools to deal with abstractions, and makes the students comfortable with complex mathematical notations. As stated by J. Wing in [19]: “Thinking

in terms of formal methods concepts forces the designer to take a more abstract perspective of a system than that taken with an algorithmic or operational approach. This more abstract thinking invariably provides the designer with new insights and a deeper understanding of the systems desired behavior”.

As convincing as this may sound, as noted by Wing, “the biggest obstacle is getting “buy-in” from our colleagues: convincing co-instructors, curricula committees and administrators that integrating formal methods is a good thing to do”. This requires collecting empirical evidence on the above mentioned benefits of studying formal methods. In this position paper we take a step towards filling this gap by reporting on our findings collected when teaching a graduate course “Logic and Formal Specification”, taught at the Information Systems Department of the University of Haifa. The details of the course design and implementation are provided in [21]. Here we focus on the human factor, namely the students’ perceptions of the benefits of the course. We present the results of our exploratory study involving 22 participants who completed our course. The students were asked about their perception of its benefits, both for an IS practitioner in general, and for them personally. The most striking observation emerging from our data analysis was that most of the students reported an effect on their *cognitive* processes: fostering analytical thinking, mental decomposition of complex problems into simpler ones, and using abstraction. This provides a starting point for collecting further empirical data on the benefits of teaching formal methods to IS practitioners.

## 2 Related Work

Several works address the relevance of discrete mathematics, logic and formal methods for practitioners, mainly in the context of Computer Science and Software Engineering (to the best of our knowledge, no work addressed the IS domain in this context). In [9] the suitability of the standard logic syllabus to the needs of CS practitioners is questioned: “The current syllabus is often justified more by the traditional narrative than by the practitioners needs... The proof of the Completeness Theorem is a waste of time at the expense of teaching more the important skills of understanding the manipulation and meaning of formulas”. According to [9], the needs of CS practitioners are to: (i) understand the meaning and implications of modeling the environment as precise mathematical objects and relations; (ii) understand and be able to distinguish intended properties of this modeling and side-effects; (iii) be able to discern different level of abstraction, and (iv) understand what it means to prove properties of modeled objects.

In [18,19], J. Wing stresses the importance of integrating formal methods into the existing CS curriculum by teaching their common conceptual elements, including state machines, invariants, abstraction, composition, induction, specification and verification. She states discrete mathematics and mathematical logic as crucial prerequisites. Further concrete proposals on the integration of formal methods into CS curriculum are made in [1,10,12–16]. Some of these studies include some form of empirical evaluation. Their methodology is mainly based

on objective assessment, comparing the performance of some control group to other groups of students with respect to programming skills [10] and general problem-solving skills, including using abstraction [13, 14]. However, empirical evidence for the benefits of formal methods courses is still very sparse.

In this paper we take a different approach to providing such evidence. Instead of directly assessing the students' skills and abilities, we turn to them for help with our investigation. After all, investigating the students' perceptions may provide new insights into the way in which taking the course affected their cognitive processes. Moreover, their attitudes towards the practical value of such courses may be helpful with the prediction of their future acceptance of formal methods in industry. It should be noted that we apply a qualitative research approach using open-ended questionnaires, as opposed to the quantitative approach taken in [8], which also surveyed "what subject matter practitioners themselves actually find most important in their work", but used a closed-ended questionnaire, with questions such as "How useful have the details of this specific material been to you in your career as a software developer?" using a scale from 0 to 5, leaving no place for exploring cognitive aspects affected by the taught courses.

### 3 The Exploratory Study of Students Perceptions

The course "Logic and Formal Specification" has been taught at the IS department at the University of Haifa for several years by both of the authors<sup>1</sup>. The course is a mandatory course for graduate students, and its length is one semester, 4 h per week. Many of the students return to their studies after several years in the industry. This poses a twofold challenge when designing a course in logic and formal methods. First, they have a solid understanding of topics that have direct relevance to practice and are reluctant to study topics whose relevance to their daily practice is indirect. Secondly, they have forgotten the basic concepts of discrete mathematics which they studied years ago.

Our course design is based on previous proposals on the adaptation of the traditional logic and formal methods syllabi to the needs of modern practitioners [4, 9, 10, 16, 19]. As such, the course aims to equip the students with the following abilities: (1) read, write and understand formal specifications, (2) be able to formalize informal specifications, (3) analyze specifications and detect sources of incompleteness, inconsistency and complexity, (4) reason about specifications, and (5) check a system against a specification. Based on the above, the taught material includes (a) Basic principles for reasoning about sets; (b) Induction

---

<sup>1</sup> Perhaps it is important to mention here the authors' relevant background. The first author is an associate professor at the Information Systems Department at the University of Haifa with active research interests in applied logic. The second author is the manager of the Software Performance and Quality research group at the IBM Haifa Research Laboratory, and a member of the IBM corporate Board of Software Quality. Both of the authors have several years of experience in teaching logic and formal methods to various audiences of students.

and invariants; (c) Propositional and first-order logic and (d) Formal specification using the Z language. Further details about the course and the ways we propose to overcome the above mentioned challenges with respect to the target audience are provided in [21].

In what follows we describe the results of an exploratory interview we carried out in order to gain a deeper understanding into the students' perception of the benefits of studying formal methods for IS practitioners. For this purpose, we chose an open-ended questionnaire [11] over indepth interviews to ensure the anonymity of our participants, which was an important concern in our context. We used an open-ended questionnaire as we wanted to minimize any pre-suppositions on the participants' responses (as opposed to e.g., the closed-ended questionnaire of [8]).

The answers were collected by the first author from twenty two graduate students who completed the course in the years 2013–2014. This sample included 8 female and 15 male students; 12 students out of 22 had no prior experience in industry. The questionnaire included the following open-ended questions.

- Q1.** Is it important for practitioners whose work is related to software development to study logic and formal methods? Why?
- Q2.** In what way (if at all) is the course's content useful for Information Systems practitioners?
- Q3.** What (if at all) were the course's contributions for you personally?
- Q4.** How relevant was your background from Discrete Mathematics course? In what way (if at all) was it helpful?
- Q5.** In what ways would you recommend to improve the course?

In what follows we refer mainly to the answers received to questions Q2 and Q3. Only three students responded that logic and formal methods are not useful (Q2):

1. I worked at two different places in industry, and never have I seen the courses' content put to any use...
2. It is not necessary for software development.
3. It depends on the work environment. I think it's not useful.

Two of them thought the course was not useful for them personally (Q3).

Out of those who responded positively to both questions, one of the most striking observations was the extensive use of formulations related to mental processes, such as "thinking", in particular "analytical/logical thinking" in answers to both questions.

E.g., answers to question Q2 included:

1. It improves **thinking** about problem modeling.
2. I think that it opens directions for **thinking** about how things really work under the surface.
3. The world of software is based on understanding the needs and modeling them in precise terms. Many such models require **logical thinking**.

4. The course's contents develop and deepen **ways of thinking**.
5. The course helps shaping **thinking** that can help in programming.
6. The course improves **analytical thinking**.
7. The course is very helpful in improving **thinking that is not necessarily algorithmic**. A different one, out of the box.
8. Of course! **Correct and systematic thinking** of IS practitioners helps in requirements specification.

Notably, no participants provided concrete examples of direct use of the courses' content in answering Q2. Yet several of them took a confident stand when speaking of their own personal experience in Q3:

1. I have already applied the new skills at work, using truth tables and proofs.
2. It improved my modeling skills. I'm certain!
3. I am now using the tools when reading scientific papers.
4. I was surprised to see how helpful the tools we studied are in practice.

Moreover, when answering question Q3, several participants referred again (implicitly or explicitly) to an improvement in their *mental* processes:

1. The course introduced order into complex topics. It gave me tools to simplify complex problems and find easy and efficient solutions.
2. It made me think in a modular way, providing me with the ability to grasp more complex models.
3. It improved my ability to refer to problems schematically.
4. It provided me with an abstract view on the problems of software design.
5. It made me realize there are systematic solutions to problems that seem unsolvable at first.
6. I learned to reduce complex problems to simpler ones.

Table 1 summarizes the main skill categories that emerged during text analysis of questions Q2 and Q3, providing the number of students that used formulations related to these categories.

**Table 1.** Categories emerging from answers to Q2 and Q3 and number of students using each category

	Q2 (general IS practitioner)	Q3 (personal experience)
Thinking	8	8
Understanding	7	8
Formulation	5	3
Modelling	1	0
Research	0	3
General knowledge	0	5



## 4 Summary and Future Research

To make logic and formal methods more central in the IS curriculum, further empirical evidence on the benefits of such courses for practitioners is required. While IS practitioners rarely apply formal methods directly, but rather use tools where they are “hidden” [5], exploring the effect of such courses on the cognitive processes of students seems the most fruitful direction. Our results highlights the potential of the methodology of exploring students’ perceptions and attitudes in this context. This could be beneficiary for the education community also from another aspect. While the key role of abstract thinking for acquiring computation-related skills has been stressed by Kramer [6], ways of teaching it are still not well understood (some relevant general suggestions can be found in [7]). The positive effect formal methods courses may have on the cognitive processes of the students provides a good starting point for exploring concrete ways in which we abstract thinking can be taught.

We plan to further extend the data collection and analysis to larger student populations, including both undergraduate and graduate students taking relevant courses, as well as to experienced IS practitioners. In future research we plan to further investigate the impact of factors such as industrial experience and maturity of the subjects of the study. We also plan to reformulate the questions to allow a further sharper quantitative analysis.

The observations of this paper also bring forward the somewhat controversial notion of *computational thinking* (CT). This is a term introduced by J. Wing in 2006 [20], describing the mental activity in formulating a problem to admit a computational solution, which is crucial for many disciplines at our times. Naturally, it has also been pointed out as a key capability for future IS practitioners [3]. Note the striking relation of some of the above mentioned responses to key attributes of what J. Wing classifies as computational thinking [20]: (i) reformulating a seemingly difficult problem into one we know how to solve; (ii) using abstraction and decomposition when attacking a large complex task or designing a large complex system; and (iii) choosing an appropriate representation for a problem or modeling the relevant aspects of a problem to make it tractable. The link between teaching formal methods and the development of computational thinking skills deserves further exploration (also due to the increasing public interest in CT).

## References

1. Barland, I., Felleisen, M., Fisler, K., Kolaitis, P., Vardi, M.Y.: Integrating logic into the computer science curriculum. In: Annual Joint Conference on Integrating Technology into Computer Science Education (2000)
2. Glass, R.L.: A new answer to how important is mathematics to the software practitioner? *IEEE Softw.* **17**(6), 136 (2000)
3. Hardy, G.M., Everett, D.L.: *Shaping the Future of Business Education: Relevance, Rigor, and Life Preparation*. Palgrave Macmillan, London (2013)

4. Harvey, V.J., Wu, P.Y., Turчек, J.C., Longenecker, H.E.: Coordinated topic presentations for information systems core curriculum and discrete mathematics courses. In: Proceedings of ISECON 2005 (2005)
5. Hussmann, H.: Indirect use of formal methods in software engineering. In: ICSE-17 Workshop on Formal Methods Application in Software Engineering Practice, Seattle (WA), USA, pp. 126–133. Citeseer (1995)
6. Kramer, J.: Is abstraction the key to computing? *Commun. ACM* **50**(4), 36–42 (2007)
7. Kramer, J., Hazzan, O.: The role of abstraction in software engineering. In: Proceedings of the 28th International Conference on Software Engineering, pp. 1017–1018. ACM (2006)
8. Lethbridge, T.C.: What knowledge is important to a software professional? *Computer* **33**(5), 44–50 (2000)
9. Makowsky, J.A.: From Hilberts program to a logic tool box. *Ann. Math. Artif. Intell.* **53**(1–4), 225–250 (2008)
10. Page, R.L.: Software is discrete mathematics. *ACM SIGPLAN Not.* **38**, 79–86 (2003)
11. Patten, M.L.: *Questionnaire Research: A Practical Guide*. Pycszak Publisher, Glendale (2001)
12. Skevoulis, S., Makarov, V.: Integrating formal methods tools into undergraduate computer science curriculum. In: 36th Annual on Frontiers in Education Conference, pp. 1–6. IEEE (2006)
13. Kelley Sobel, A.E.: Empirical results of a software engineering curriculum incorporating formal methods. *ACM SIGCSE Bull.* **32**(1), 157–161 (2000)
14. Kelley Sobel, A.E., Clarkson, M.R.: Formal methods application: an empirical tale of software development. *IEEE Trans. Software Eng.* **28**(3), 308–320 (2002)
15. Sotiriadou, A., Kefalas, P.: Teaching formal methods in computer science undergraduates. In: International Conference on Applied and Theoretical Mathematics (2000)
16. Tavolato, P., Vogt, F.: Integrating formal methods into computer science curricula at a university of applied sciences. In: TLA+ Workshop at the 18th International Symposium on Formal Methods, Paris, Frankreich (2012)
17. Topi, H., Valacich, J.S., Wright, R.T., Kaiser, K., Nunamaker, Jr., J.F., Sipior, J.C., de Vreede, G.J.: Is 2010: Curriculum guidelines for undergraduate degree programs in information systems. *Commun. Assoc. Inf. Syst.* **26**(1), 18 (2010)
18. Wing, J.M.: Teaching mathematics to software engineers. In: Alagar, V.S., Nivat, M. (eds.) *AMAST 1995*. LNCS, vol. 936, pp. 18–40. Springer, Heidelberg (1995)
19. Wing, J.M.: Weaving formal methods into the undergraduate computer science curriculum. In: Rus, T. (ed.) *AMAST 2000*. LNCS, vol. 1816, pp. 2–7. Springer, Heidelberg (2000)
20. Wing, J.M.: Computational thinking. *Commun. ACM* **49**(3), 33–35 (2006)
21. Zamansky, A., Farchi, E.: Teaching logic to information systems students: challenges and opportunities. In: *Tools for Teaching Logic* (2015)

# GuideForce: Type-Based Enforcement of Programming Guidelines

Serdar Erbatur<sup>(✉)</sup> and Martin Hofmann

Ludwig-Maximilians-Universität, Munich, Bavaria, Germany  
{serdar.erbatur,hofmann}@ifi.lmu.edu

**Abstract.** In this paper, we introduce the GuideForce project, whose aim is to develop automatic methods based on type systems and abstract interpretation that are capable of checking that programming guidelines related to secure web programming are correctly and reasonably applied. We outline the project plan and motivation and then describe a pilot study carried out with Soot, a Java-based program analysis framework. While still maintaining high accuracy and efficiency, the focus on guidelines adds a new human-oriented component to static analysis.

**Keywords:** Program analysis · Type systems · Language-based security · String analysis

## 1 Introduction

Modern software typically must be able to interact with the whole world. While until recently such worldwide interaction was quite rare now almost any business software has a web interface allowing anyone to interact with the software be it only by entering something into the username/password fields that are openly accessible to anyone.

Since almost anyone writes software exposed to the resulting security threats, one can no longer rely on high skill and experience of specialists who were formerly only having to deal with such risks. To address this issue programming guidelines and best practices have been developed, see e.g. [www.owasp.org](http://www.owasp.org), that summarise and condense the expert knowledge and make it available to a larger community (secure coding) [30]. Whether or not such programming guidelines are applied and whether they have been correctly applied, is however left to the good will of the programmers.

For this reason, we have proposed the project GuideForce (funded by DFG since 11/2014) in which we want to develop automatic methods based on type systems that are capable of checking that programming guidelines have been correctly and reasonable applied without compromising the flexibility of writing code. Besides further developing type system methodology this also requires us

---

This research is funded by the German Research Foundation (DFG) under research grant 250888164 (GuideForce).

to devise a formalism to rigorously define such policies which typically are given in plain English and by examples. In order that users will actually trust the system and perceive it as a useful tool it will be necessary to achieve a rather high degree of accuracy. For example, if an already sanitized user input is stored in a string buffer and later on read out it is not necessary to re-sanitize it. If the system does not recognize such a situation users will neglect its warnings in the future. Similarly, to ensure appropriate authorization prior to accessing sensitive data then, if such access happens within a method of a class admitting the invariant that authorization has taken place prior to creation of any of its objects then the system must be able to discover this. All this means that cutting edge techniques such as careful analysis of strings, objects, and control flow, must be harnessed and further developed in this project. In order to guarantee appropriate feedback to the user and to achieve seamless integration we will use type-theoretic formulations of these methods resulting then in a single customizable type system capable of enforcing a large span of guidelines and best practices for secure web programming.

In [15], we developed a type-based string analysis for the enforcement of guidelines to detect code injection attacks, and a tool has been implemented using OCaml [7].

For GuideForce, however, we plan not to rely on a standalone implementation, but rather to build our tools on top of the Soot framework or one of its competitors. Soot is a Java-based program analysis framework developed by the Sable research group in McGill University [6, 23, 39]. To gain some experience we developed a small pilot study in Soot that we describe in Sect. 4 below. This will allow us to take profit of Soot’s front end and to base our analyses on intermediate code. We also hope that the use of a well-established framework will ease maintenance and changes in the development team. Furthermore, we will be able to take profit of Soot’s built-in fixpoint engines for dataflow analysis. While our analyses will mostly be type-based it is well-known that type inference can be understood as a dataflow problem, see e.g. [29].

The key scientific innovations of the project are the focus on guidelines rather than risks and the development of a configurable type system. In Sect. 3 we describe the GuideForce project in more detail and provide examples of guidelines to illustrate our approach. In Sect. 4 we describe our pilot study which is a taintedness analysis for the prevention of Xpath injection and developed on top of Soot.

We stress that the purpose of this project-start paper is not so much to describe finished results, but to motivate and advertise the goals of our recently begun project so as to create opportunities for interaction.

*Human orientation.* The focus on guidelines and the use of type systems are thus what distinguishes our approach from most existing ones. While still maintaining high accuracy and efficiency which are the traditional objectives in static analysis this adds a new human-oriented component to the field. Types present even the intermediate results of a static analysis in a concise yet human-readable manner and also allow, in the form of manually provided type annotation for interaction

with the analysis process. Guidelines, on the other hand, have been designed by human experts with human users in mind. Enforcing guidelines not only helps to prevent vulnerabilities and attacks, but also has a forensic and thus human-oriented component: if a programmer or subcontractor can show that all guidelines have been followed they might not be responsible for a possible unforeseen attack.

## 2 Related Work

The use of formal approaches to guarantee adherence to a secure coding guideline rather than directly preventing the absence of vulnerabilities is a recent idea; the only directly related work we are aware of is [8] where guidelines are formalised as temporal logic formulas. The novel aspects of our approach in comparison with [8] are the precise tracking of control flow, even interprocedural, the analysis of string values as is required to model guidelines for preventing code injection attacks [9,15], the tracking of class invariants, and the presentation as a type system akin to the Java type system which will allow for meaningful feedback to the user.

There is a huge body of literature on the use of formal methods, in particular type systems for directly preventing attacks rather than enforcing a guideline and the methods used there are also relevant for us and, indeed, without the progress in those areas in the last 10 years a project like ours could not be carried out. In the following summary we focus on our running example, namely prevention of code injection attacks.

Test-based techniques [20] identify points in an application that are subject to code injection attacks, and build attacks that target such points using knowledge about typical attack patterns. Black-box testing in particular does not make any assumption about the implementation of an application. By design, this method in general cannot guarantee the absence of attack possibilities; they rather aim to decrease their occurrences by applying reasonable testing heuristics.

Another approach to prevent code injection is to monitor or instrument a program. The goal is to raise an exception if an action is detected that does not conform to a certain policy. Since it is in general hard to decide which parts of a generated code are intended by the program and which are injected by the user, the program is either instrumented to mark user-dependent parts [37] or combined with a static analysis [16].

The most interesting approach for us is to statically analyse a program in order to verify that code injection attacks cannot occur during the execution of a program [9,21,26,41,42]. These approaches usually rely on string analysis to compute the possible values that a string variable can take during runtime, or on taint analysis to determine the origin of a string object, or most commonly on a combination of both.

The work we referred so far comes mainly from academia. On the industry side, many static analysis tools that help with secure web programming have been developed such as CheckMarx [1], AppScan [4], Coverity [2] and Fortify [3]. These tools protect against vulnerabilities for various languages (e.g. C,

Java) and also claim compliance with guidelines offered by institutions such as OWASP, MISRA, SANS, and Mitre CWE up to different levels. Despite this fact expressed in their data sheets, the question of how the commercial tools formalize the guidelines differs from one tool to another and the common practice is to hardwire a given guideline into the tool. One of our goals is to develop a configuration mechanism to formalize the guidelines so that they will be written separately from GuideForce source code and are open to independent review by experts.

Furthermore such a configuration mechanism will make design parametric with respect to guidelines. A developer will only need to write down the configuration file when analyzing their code and be able to check the code against as many guidelines as they want at any point of software development cycle. We also aim at bringing the guidelines rather than vulnerabilities into the focus of academic study.

*Type systems.* One may note that none of these methods provides a formal guarantee. While guaranteeing the complete absence of vulnerabilities is not feasible one can very well guarantee that a particular guideline has been enforced. Type systems [32] are particularly apt for establishing rigorous guarantees for they draw a clear distinction between the declarative statement of a program property by means of a typability relation, and the automatic verification of the property using a type inference algorithm. Type systems have been successfully used not only to prove data type safety, but also to enforce security requirements such as non-interference properties used in information flow security [24, 25, 27, 36]. Of special interest are also the recent works on providing strong typing guarantees for scripting languages such as [17, 18]. In another recent work [19], the authors define a type system along with an inference system and implement a type-based taint analysis on top of Checker Framework to analyze Java web applications.

One may note that strong type systems are used in mainstream languages such as Java, C# and play a central role in new programming languages such as Scala and F# which come originally from research and are now beginning to be used in the productive sector. One may thus argue that programmers are familiar with the form of feedback offered by types.

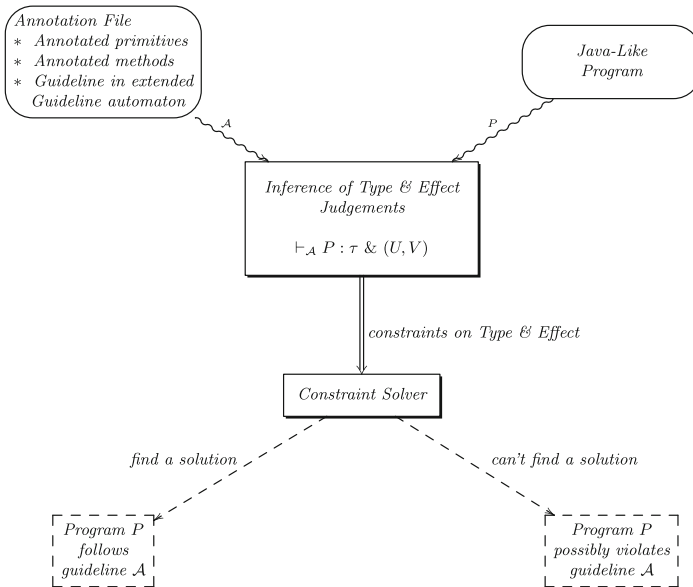
An early work proposing the use of type systems in the context of secure programming is [40] where a type system is used to check runs of functional programs against a security automaton [34].

*Human-oriented approaches.* Several researchers have already pointed out that the benefits of static analysis go beyond finding bugs in source code. In [28], various applications of static analysis to other than finding bugs such as code-review and computing human-readable properties of the software were outlined. The Agile Manifesto [5] emphasized that the focus should be human and interactions among others. In [14] authors give a detailed explanation of static analysis as part of code review, explain secure programming guidelines (for web applications etc.) and provide exercises with Fortify code analyzer [3]. The industrial

tools including Fortify, mentioned earlier in this section, allow user interaction; developer receives highlighted source code in a human-readable format.

### 3 Our Method

In this section we sketch how the final outcome of this project might look like. The workflow of our system is depicted in Fig. 1.



**Fig. 1.** GuideForce (Type-Based Guideline Enforcement)

The input to the GuideForce system (Type-Based Guideline Enforcement) consists of two parts: an annotation file formalising a guideline and containing instrumentation information and an automaton (“guideline automaton”) expressing the required behaviours and a source program in Java (or an appropriate subset thereof). The program is then automatically annotated according to the annotation instructions in the annotation file and subjected to inference of effect types, the effects being derived from the guideline automaton. We note that automata have been used in the past to describe program properties to prevent bugs, i.e., in SLAM project from Microsoft [11] and in Metal project from Stanford [13] both of which are used to analyze C code. However, the difference of our approach is that we later construct type and effect systems from automata in the next phases. This process results in a list of constraints typically over finite sets whose solutions correspond to valid typing derivations. If a solution can be found then this implies that the source program definitely abides

by the formalised guideline. If no solution can be found then the program might violate the guideline and the reason for this belief can be clearly pinpointed in the form of a typing error that will be reported to the user.

To illustrate the format and the workflow of our approach we now describe two simple concrete instances. The first one about sanitization of user input is modelled after [15] and related to prevention of SQL injection attack.

**Example 1. Sanitization** In order to avoid code injection, programmers are required to sanitize strings to be output through a browser window by calling appropriate framework methods that remove potentially dangerous components such as JavaScript code or dangerous URLs. Strings that stem from the user are regarded as potentially dangerous and in need of sanitization whereas string literals contained in the program are regarded as safe. The guideline in plain English is as follows:

*String output must be sanitized.*

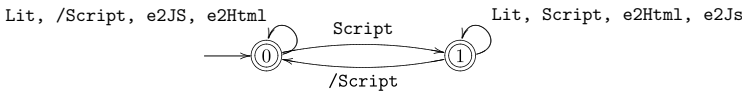
The following code gives an example which follows this guideline.

```
class Sanitization {
  void main () {
    String input = getUserInput();
    String s ="<script>"+ escapeToJs(input)
      + "</script>";
    output(s);
    s ="<body>"+ escapeToHtml(input) + "</body>";
    output(s);}}
```

Output strings within tags `<script>` and `</script>` must be sanitized by calling the framework method `escapeToJs` and those within tags `<html>` and `</html>` must be sanitized by calling the framework method `escapeToHtml`.

We decorate strings with labels describing their origin, whether they contain tags like `<script>`, `</html>`, etc., and if any sanitization they have undergone.

Given this (imaginary) instrumentation we can then specify the guideline by the following automaton:



Here `e2JS` tags the output of `escapeToJs` and `e2Html` tags the output of `escapeToHtml`. String literals `<script>` and `</script>` are mapped to `Script` and `/Script` respectively. All other strings are tagged as `Lit`. It is not hard to see that in the above code snippet, the tags of strings to be output will be `Script · e2JS · /Script` and `Lit · e2Html · Lit` respectively. They are accepted by the above automaton. This information is supplied by users and stored in the annotation file whose details we omit here for lack of space.

**Example 2. Authorization** Let us consider the following guideline:

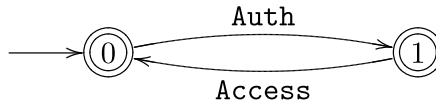
*Any access to sensitive data must be done after authorization.*



This guideline requires the programmer to call function `auth` before `access`. The following code fragment follows the guideline:

```
class Authorization {
    void main () {
        String s;
        while(true) {
            try {
                auth();s = access();
            }
            catch (AuthorizationFailed e) {
                s ="Invalid access";
            }
            output(s);}}}
```

Assuming that any successful call to the framework method `auth` issues an `Auth` event and that every call of framework method `access` issues an `Access` event we can specify the policy with the following finite state machine.



The guidelines that we formalised here are only two examples and by no means restricted to sanitization and authorization. Other examples include appropriate initialisation and finalisation of data structures and resources, appropriate bounds checking, integrity checking of downloaded code, execution with appropriate privileges, etc.

The Soot-based implementation lets us design a human-readable output file from a given source file. Using built-in visualization tools in Soot, we will be able to reproduce the source code with highlighted lines where the (formalized) guidelines might be violated. Furthermore, we plan to provide the user a detailed explanation of the guideline and also how the highlighted lines violate it. Thus, this mechanism increases the readability and understandability of output of the analysis by human developers.

## 4 Taintedness Analysis for XPath Injection

To see whether Soot fits our purpose, we decided to develop an analysis with it for a very simple but still meaningful guideline. For this purpose, we selected a guideline from OWASP [31] to prevent XPath injection. XPath injection is a type of code injection that allows structured access to a node in a XML document tree. In this simple example the use of types was not necessary; instead we employed a rather straightforward taintedness analysis.

In the rest of this section, we first explain a sample XPath injection attack, and then explain our implementation according to flow analysis framework provided by Soot. Our analysis is twofold. First, we develop an intraprocedural

analysis to analyze method bodies based on forward flow analysis provided by Soot. Second we develop extend the intraprocedural analysis to an interprocedural analysis to analyze Java classes and methods of an application that uses XML documents to store sensitive data.

In fact our interprocedural analysis can be seen as an instance of the summary-based (also called functional) approach introduced by Pnueli and Sharir [35]. An efficient framework that is based on the work in [35] is IFDS given by Reps, Horwitz and Sagiv [33] which is based on graph reachability. Soot has an extension, called Heros<sup>1</sup>, that implements the IFDS framework; however, we preferred to implement our own interprocedural analysis directly within Soot since this is considerably simpler and also, as we believe, will facilitate the extension to later more complex type-based disciplines.

We also remark that recently several tools for taint analysis have appeared, notably Andromeda [38] for Java programs and Flowdroid [10] for Android applications.

#### 4.1 XPath Injection Example

Similar to other types of code injection, an attacker enters (malicious) strings such that the resulting XPath query grants unintended access to XML data. We reproduce here the Java-based example given in the OWASP XPath injection page [31]. For more general information and other possible attacks and countermeasures, see [22]. According to the scenario in the OWASP example, an application includes the following XML document to keep personal data:

```
<?xml version="1.0"encoding="utf-8"?>
<employees>
  <employee id="AS789"fname="John"lname="Doo"salary="70000"/>
  <employee id="AS719"fname="Isabela"lname="Dobora"salary="90000"/>
  <employee id="AS219"fname="Eric"lname="Lambert"salary="65000"/>
</employees>
```

An XPath query to select the nodes in the XML document<sup>2</sup> is:

```
/employees/employee[@id='employeeID']
```

The example uses the following lines of code to embed this query into source code to select nodes from the document:

```
String eID = request.getParameter("employeeID");
String xpathExpr = "/employees/employee[@id=' +eID +']";
compile(xpathExpr); /* shortened */
```

This implementation is vulnerable to attacks that exploit XPath syntax and allow attackers to enter malicious input. For instance, a user could enter ' or '1'='1 into the field `employeeID`. Then, the query formed by the application amounts to

<sup>1</sup> <https://github.com/Sable/heros>.

<sup>2</sup> Above, `fname` is shorthand for first name, `lname` for last name and `salary` for annual salary.

```
/employees/employee[@id=' ' or '1']='1']
```

which in turn returns all the records in the XML document.

A programming guideline to avoid XPath injection is to sanitize the user input as for other kinds of injection attacks. As explained earlier, checking if this guideline is followed can be reformulated as a taint analysis. This example is actually simpler than the one described in the introduction as there is only one way to sanitize input rather than several ones depending on the context.

## 4.2 Soot Implementation

We explain the first part of implementation, namely an intraprocedural forward flow analysis for which Soot provides a framework. To analyze a method body, we use Soot's Jimple intermediate representation.

In a Jimple body, we call a variable *tainted* (i) if it is a string variable whose value is obtained through user input (i.e. the `getParameter()` method in the previous subsection) or (ii) if it is a concatenation of two strings where one or both of them are tainted. If we restrict ourselves to intraprocedural analysis we must also deem all return values of method calls tainted as well as contents of fields (object variables). Below, we describe an interprocedural extension that deals with the first restriction; dealing with the second one will be left for the future.

We use the standard Java class `Boolean` to represent the values *tainted* and *untainted*, i.e., `true` for *tainted* and `false` for *untainted*. Each (local) variable in a Jimple body is mapped to a boolean taintedness value. To model the mapping between variables and boolean values, we define a map as `TreeMap<Value, Boolean>` and implement flowsets with a new class `AbstractState` that includes the map as a field. The reason that the key set is `Value` rather than `Local` stems from the fact that `Value` is the least common superclass of `Local` (local variables) and `IdentityRef` (formal parameters). Our abstract state also contains a boolean field `broken` indicating whether or not the guideline has been violated up to this point. This would be the case, if we apply the framework method `compile` to a tainted argument.

```
class AbstractState{
    TreeMap<Value, Boolean> locals;
    Boolean broken;
}
```

Then, we instantiate `ForwardFlowAnalysis<N,A>` in Soot as `ForwardFlowAnalysis<Unit, AbstractState>`. According to the Soot framework, flow analysis requires to implement some methods after extending the flow analysis class; a constructor and the methods `copy()`, `merge()` `newInitialFlow()`, `entryInitialFlow()` and `flowThrough()`.

Merging is implemented using union and disjunction. The analysis starts with the assumption that all variables are untainted and that we are not broken yet. The method `flowThrough()` describes how the abstract state is changed upon the execution of basic statements, i.e. assignments and method calls.

```

java.lang.String r1, r2, r6;
r1 := @parameter0: java.lang.String;
r2 = "abdce";
r3 = ... getParameter(java.lang.String)>("employeeID");
r6 = ... m(...);

```

At the end of the execution of this block the “values” `r1`, `r2` will be untainted whereas `r3` and `r6` will be. The field `broken` will be false, however.

**Interprocedural Analysis.** The local variable `r6` in the example above was deemed tainted just because we did not track taintedness across method boundaries. In order to achieve that in a meaningful way we need to track the dependency of taintedness of locals (and also broken-ness) on the taintedness of parameters.

We therefore use a more refined abstract state where `Booleans` are replaced by elements of the lattice  $L = \mathcal{P}(P) \cup \{\top\}$  where  $P$  is the set of formal parameters including `this`, `IdentityRef` in Soot. A lattice element  $\{x, y, z\}$  represents taintedness if one of  $x, y, z$  are tainted or rather definite untaintedness if none of  $x, y, z$  are tainted.

```

class Lattice {
    TreeSet<IdentityRef> params;
    Boolean isTop; /* if this is set can ignore params */
}
class AbstractState{
    TreeMap<Value, Lattice> locals;
    Lattice broken;
}

```

In this case, the initial abstract state will set each formal parameter  $x$  to  $\{x\}$  whereas the other locals are set to  $\emptyset$ .

```

public String m(String x, String y){
    int t;
    String z;
    z = x + y;
    compile(x)
    return z; }

```

In this example, the abstract state is initially  $(\{x \mapsto \{x\}, y \mapsto \{y\}, z \mapsto \emptyset, t \mapsto \emptyset\}, \emptyset)$ . The resulting abstract state is expected to be  $(\{x \mapsto \{x\}, y \mapsto \{y\}, z \mapsto \{x, y\}, t \mapsto \emptyset\}, \{x\})$  signifying in particular that  $z$  is tainted if either of the arguments is and running the body will violate the guideline if  $x$  is tainted.

We then obtain the desired interprocedural analysis by making iterated calls to the intraprocedural analysis class. First, we set up a *table*; a treemap that maps each method to a *summary*, where a summary comprises two lattice elements, one explaining the taintedness of the return value of the method and another its broken-ness, i.e., under what conditions on the taintedness of the formal parameters running the method might violate the guideline.

```
class Summary { Lattice ret; Lattice broken; }
```

The summaries for the built-in methods `append`, `compile` and `getParameter` are as follows:

	ret	broken
<code>append</code>	$\{\text{this}, p_0\}$	$\emptyset$
<code>compile</code>	$\emptyset$	$\{p_0\}$
<code>getParameter</code>	$\top$	$\emptyset$

Thus, `getParameter` in itself never violates the guideline (`broken` =  $\emptyset$ ), but its return value is always tainted (`ret` =  $\top$ ). Similarly, running `append` does not per se violate the guideline, but its result is tainted once `this` or  $p_0$  are. Running `compile` on a tainted input violates the guideline, (`broken` =  $\{p_0\}$ ); here  $p_0$  denotes the first (and only) formal parameter of the method), the return value of `compile` is never tainted.

Next, for each method body and its control flow graph we compute the abstract states at exit points by using `getTail()` and `getFlowAfter()`.

```
for (SootClass c : cls) {
  for (SootMethod m : c.getMethods()) {
    Body b = m.retrieveActiveBody();
    UnitGraph graph = new ExceptionalUnitGraph(b);
    XPathIntra intra = new XPathIntra(graph, mtable);
    Lattice ret = Lattice.latticeZero();
    Lattice broken = Lattice.latticeZero();
    for (Unit s : graph.getTails()) {
      AbstractState abs = intra.getFlowAfter(s);
      broken.union(abs.getBroken());
      if (s instanceof ReturnStmt){
        Value loc = ((ReturnStmt) s).getOpBox().getValue();
        ret.union(abs.getLocals().get(loc));}
      newmtable.put(m, new Summary(ret, broken));}
  }
}
```

Herein, `cls` refers to the entire set of classes comprising an application to be analysed. After the above code has been executed we then check whether the updated method table `newmtable` is equal to `mtable` (in the extensional sense) in which case we have reached a fixpoint and `mtable` contains the results of our interprocedural analysis. Otherwise, we replace `mtable` with `newmtable` and start over. It is easy to see by induction on the number of iterations that `newmtable` is pointwise a superset of `mtable` so that a fixpoint will be reached and the iteration is guaranteed to terminate.

We also remark that interprocedural analysis required us to implement flowsets (for intraprocedural analysis) with a tree-like structure, whereas it is common to implement flowsets with hash sets. This was due to strange interactions with hashing that occurred during interprocedural analysis.

**Example 3.** The current implementation successfully handles the motivating example above and also the following more artificial one which demonstrates that we support method polymorphism.

```

class D{
    public static void main(String[] args){
        m(getParameter());
        compile(m("a"));
    }
    public String m(String s){
        return s;
    }
}

```

However, as already mentioned, we do not presently handle class fields. To do that properly, a region typing [12] or a points-to analysis would be necessary. The following example is therefore rejected since we over-cautiously consider all content of fields potentially tainted.

#### Example 4

```

class E{
    public static void main(String[] args){
        F f1 = new F(getParameter());
        F f2 = new F("test");
        compile(f2.s);
    }
}
class F{
    public String s;
    public F(String s){this.s=s;}
}
}

```

## 5 Conclusion

We have described our plans for a configurable, type-based analysis for the enforcement of programming guidelines. In contrast to most existing uses of formal methods in the context of secure programming, we aim at an easily configurable analysis that guarantees adherence to well-specified guidelines rather than the absence of vulnerabilities. While guidelines often implicitly form the basis of code analysis tools they are here the primary goal of the analysis. While of course the ultimate goal is to avoid attacks we find that it may, not least for reasons of ultimate responsibility, be better to ensure automatically that expert advice has been followed and that it is crystal-clear which piece of advice (guideline) has been followed. We therefore consider the precise formalisation of guidelines an important concept which is now, of course to be demonstrated as feasible. Our approach focuses not only finding bugs but also human factors of analysis. One of our goals is to provide human-readable output which presents detailed information about how the guideline is violated in the source code. Our decision to implement on top of Soot framework complies with this goal since Soot has built-in visualization tools.

As a first starting point in this direction we have developed, within Soot, a concrete analysis enforcing adherence to a guideline against XPath injection

proposed by the OWASP consortium. The next steps will consist of implementing the region-based type system from [12] within Soot which will improve context sensitivity of our analysis. On top of that, we can then develop further guidelines including those described semi-formally in Sect. 3.

## References

1. Checkmarx CxSAST. <https://www.checkmarx.com/>
2. Coverity. <http://www.coverity.com/>
3. Fortify Static Code Analyzer. <http://www8.hp.com/us/en/software-solutions/static-code-analysis-sast/index.html>
4. IBM Secure AppScan Source. <http://www-03.ibm.com/software/products/en/appscan-source>
5. Manifesto for Agile Software Development. <http://agilemanifesto.org/>
6. Soot - A framework for analyzing and transforming Java and Android Applications. <http://sable.github.io/soot/>
7. Type-Based Java String Analysis (2012). <http://jsa.tcs.ifi.lmu.de/>
8. Aderhold, M., Cuellar, J., Mantel, H., Sudbrock, H.: Exemplary formalization of secure coding guidelines. Technical report TUD-CS-2010-0060, TU Darmstadt, Germany (2010)
9. Annamaa, A., Breslav, A., Kabanov, J., Vene, V.: An interactive tool for analyzing embedded SQL queries. In: Ueda, K. (ed.) APLAS 2010. LNCS, vol. 6461, pp. 131–138. Springer, Heidelberg (2010)
10. Arzt, S., Rasthofer, S., Fritz, C., Bodden, E., Bartel, A., Klein, J., Le Traon, Y., Octeau, D., McDaniel, P.: Flowdroid: precise context, flow, field, object-sensitive and lifecycle-aware taint analysis for android apps. In: O’Boyle, M.F.P., Pingali, K. (eds.) ACM Conference on Programming Language Design and Implementation, PLDI 2014, Edinburgh, United Kingdom, 09–11 June 2014, p. 29. ACM (2014)
11. Ball, T., Rajamani, S.K.: The SLAM project: debugging system software via static analysis. In: Launchbury, J., Mitchell, J.C. (eds.) The 29th ACM Symposium on Principles of Programming Languages, Portland, OR, USA, 16–18 January 2002, pp. 1–3. ACM (2002)
12. Beringer, L., Grabowski, R., Hofmann, M.: Verifying pointer and string analyses with region type systems. In: Clarke, E.M., Voronkov, A. (eds.) LPAR-16 2010. LNCS, vol. 6355, pp. 82–102. Springer, Heidelberg (2010)
13. Chelf, B., Engler, D.R., Hallem, S.: How to write system-specific, static checkers in metal. In: Dwyer, M.B., Palsberg, J. (eds.) Proceedings of the Workshop on Program Analysis for Software Tools and Engineering, PASTE 2002, Charleston, South Carolina, USA, 18–19 November 2002, pp. 51–60. ACM (2002)
14. Chess, B., West, J.: Secure Programming with Static Analysis, 1st edn. Addison-Wesley Professional, Reading (2007)
15. Grabowski, R., Hofmann, M., Li, K.: Type-based enforcement of secure programming guidelines — code injection prevention at SAP. In: Barthe, G., Datta, A., Etalle, S. (eds.) FAST 2011. LNCS, vol. 7140, pp. 182–197. Springer, Heidelberg (2012)
16. Halfond, W.G.J., Orso, A.: Preventing SQL injection attacks using AMNESIA. In: 28th IEEE and ACM SIGSOFT International Conference on Software Engineering (ICSE 2006) - Formal Demos track (May 2006)

17. Heidegger P., Bieniusa, A., Thiemann, P.: Access permission contracts for scripting languages. In: Field, J., Hicks, M. (eds.) Proceedings of the 39th ACM Symposium on Principles of Programming Languages, POPL 2012, Philadelphia, Pennsylvania, USA, 22–28 January 2012, pp. 111–122. ACM (2012)
18. Heidegger, P., Thiemann, P.: Recency types for analyzing scripting languages. In: D’Hondt, T. (ed.) ECOOP 2010. LNCS, vol. 6183, pp. 200–224. Springer, Heidelberg (2010)
19. Huang, W., Dong, Y., Milanova, A.: Type-based taint analysis for Java web applications. In: Gnesi, S., Rensink, A. (eds.) FASE 2014 (ETAPS). LNCS, vol. 8411, pp. 140–154. Springer, Heidelberg (2014)
20. Huang, Y.-W., Huang, S.-K., Lin, T.-P., Tsai, C.-H.: Web application security assessment by fault injection and behavior monitoring. In: WWW 2003: Proceedings of the 12th International Conference on World Wide Web, pp. 148–159. ACM, New York, NY, USA (2003)
21. Jovanovic, N., Kruegel, C., Kirda, E.: Pixy: a static analysis tool for detecting web application vulnerabilities (short paper). In: SP 2006: Proceedings of the 2006 IEEE Symposium on Security and Privacy, pp. 258–263. IEEE Computer Society, Washington, DC, USA (2006)
22. Klein, A.: Blind XPath Injection (2004). [http://www.packetstormsecurity.org/papers/bypass/Blind\\_XPath\\_Injection\\_20040518.pdf](http://www.packetstormsecurity.org/papers/bypass/Blind_XPath_Injection_20040518.pdf)
23. Lam, P., Bodden, E., Lhoták, O., Hendren, L.: The soot framework for java program analysis: a retrospective. In: Cetus Users and Compiler Infrastructure Workshop (CETUS 2011) (2011)
24. Laud, P.: Secrecy types for a simulatable cryptographic library. In: Atluri, V., Meadows, C., Juels, A. (eds.) ACM Conference on Computer and Communications Security, pp. 26–35. ACM (2005)
25. Laud, P., Uustalu, T., Vene, V.: Type systems equivalent to data-flow analyses for imperative languages. *Theor. Comput. Sci.* **364**(3), 292–310 (2006)
26. Livshits, V.B., Lam, M.S.: Finding security vulnerabilities in java applications with static analysis. In: SSYM 2005: Proceedings of the 14th Conference on USENIX Security Symposium, pp. 18–18. USENIX Association, Berkeley, CA, USA (2005)
27. Mantel, H., Sudbrock, H.: Types vs. PDGs in information flow analysis. In: Albert, E. (ed.) LOPSTR 2012. LNCS, vol. 7844, pp. 106–121. Springer, Heidelberg (2013)
28. Moy, Y.: Static analysis is not just for finding bugs. *CrossTalk J. Defense Softw. Eng.* **23**(5), 5–8 (2010)
29. Nielson, F., Nielson, H.R., Hankin, C.: Principles of Program Analysis. Springer, Berlin (1999)
30. The owasp application security verification standard project. <http://www.owasp.org/index.php/ASVS>. Accessed 23 June 2013
31. OWASP. XPATH Injection Java (2012). [https://www.owasp.org/index.php/XPATH\\_Injection\\_Java](https://www.owasp.org/index.php/XPATH_Injection_Java)
32. Pierce, B.C.: Types and Programming Languages. MIT Press, Cambridge (2002)
33. Reps, T.W., Horwitz, S., Sagiv, S.: Precise interprocedural dataflow analysis via graph reachability. In: Cytron, R.K., Lee, P. (eds.) POPL 1995: 22nd ACM Symposium on Principles of Programming Languages, San Francisco, California, USA, 23–25 January 1995, pp. 49–61. ACM Press (1995)
34. Schneider, F.B.: Enforceable security policies. *ACM Trans. Inf. Syst. Secur.* **3**(1), 30–50 (2000)
35. Sharir, M., Pnueli, A.: Two approaches to interprocedural data flow analysis. In: Muchnick, S.S., Jones, N.D. (eds.) Program Flow Analysis - Theory and Applications, pp. 189–233. Prentice-Hall, Englewood Cliffs (1981)



36. Smith, G.: A new type system for secure information flow. In: CSFW, pp. 115–125. IEEE Computer Society (2001)
37. Su, Z., Wassermann, G.: The essence of command injection attacks in web applications. In: Proceedings of the 33rd Annual Symposium on Principles of Programming Languages, pp. 372–382, Charleston, SC, January 2006. ACM Press, New York, NY, USA
38. Tripp, O., Pistoia, M., Cousot, P., Cousot, R., Guarnieri, S.: ANDROMEDA: accurate and scalable security analysis of web applications. In: Cortellessa, V., Varró, D. (eds.) FASE 2013 (ETAPS 2013). LNCS, vol. 7793, pp. 210–225. Springer, Heidelberg (2013)
39. Vallée-Rai, R., Co, P., Gagnon, E., Hendren, L.J., Lam, P., Sundaresan, V.: Soot - a java bytecode optimization framework. In: MacKay, S.A., Johnson, J.H. (eds.) Proceedings of the Conference of the Centre for Advanced Studies on Collaborative Research, 8–11 November 1999, Mississauga, Ontario, Canada, pp. 13. IBM (1999)
40. Walker, D.: A type system for expressive security policies. In: Wegman, M.N., Reps, T.W. (eds.) POPL 2000, Proceedings of the 27th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, Boston, Massachusetts, USA, 19–21 January 2000, pp. 254–267. ACM (2000)
41. Wassermann, G., Su, Z.: Sound and precise analysis of web applications for injection vulnerabilities. In: Proceedings of the ACM SIGPLAN 2007 Conference on Programming Language Design and Implementation, San Diego, CA, June 2007. ACM Press, New York, NY, USA
42. Wassermann, G., Su, Z.: Static detection of cross-site scripting vulnerabilities. In: Proceedings of the 30th International Conference on Software Engineering, Leipzig, Germany, May 2008. ACM Press, New York, NY, USA

**MoKMaSD 2015**

# Clustering Formulation Using Constraint Optimization

Valerio Grossi<sup>1</sup> (✉), Anna Monreale<sup>1,2</sup>, Mirco Nanni<sup>2</sup>,  
Dino Pedreschi<sup>1</sup>, and Franco Turini<sup>1</sup>

<sup>1</sup> KDDLab, University of Pisa, Largo B. Pontecorvo, 3, Pisa, Italy  
{valerio.grossi,anna.monreale,dino.pedreschi,franco.turini}@di.unipi.it

<sup>2</sup> KDDLab, ISTI-CNR, Via G. Moruzzi, 1, Pisa, Italy  
{anna.monreale,mirco.nanni}@isti.cnr.it

**Abstract.** The problem of clustering a set of data is a textbook machine learning problem, but at the same time, at heart, a typical optimization problem. Given an objective function, such as minimizing the intra-cluster distances or maximizing the inter-cluster distances, the task is to find an assignment of data points to clusters that achieves this objective. In this paper, we present a constraint programming model for a centroid based clustering and one for a density based clustering. In particular, as a key contribution, we show how the expressivity introduced by the formulation of the problem by constraint programming makes the standard problem easy to be extended with other constraints that permit to generate interesting variants of the problem. We show this important aspect in two different ways: first, we show how the formulation of the density-based clustering by constraint programming makes it very similar to the label propagation problem and then, we propose a variant of the standard label propagation approach.

## 1 Introduction

One of the most important and more diffuse task in data mining and machine learning is clustering. This data mining method partitions a set of data objects into subsets without any supervisory information such as data labels. Each subset is called cluster and contains objects very similar to each other, and dissimilar to objects in other clusters. The set of clusters resulting from a cluster analysis can be referred to as a clustering. In the literature different clustering algorithms have been proposed suitable to operate on various kind of data objects such as text, multimedia, databases, spatio-temporal databases, networks and so on [8, 15, 18, 23]. Most of the clustering algorithms can be seen as typical optimization problems since they try to optimize a criterion specifying the clustering quality.

In this paper, we propose a formalization of some clustering problems using the constraint programming (CP) paradigm. In particular, we introduce a CP model for *K-medoids*, a prototype-based clustering, *DBSCAN*, a density-based clustering and *Label Propagation* a community discovery algorithm (i.e., a cluster algorithm working on network data). The advantage that derives from the use

of this kind models is twofold. On one side, relying on CP paradigm we are able to find clustering representing the optimal solution. This is an important aspect because often imperative algorithms find local optima due to the high complexity of the clustering problems. On the other side, by using these models we can exploit the expressive power of the constraint programming formulation that makes it easy to extend standard clustering problems with new user-specified constraints, that can express some background knowledge about the application domain or the data set, improving the clustering results.

The key contribution of this paper is focused on the fact that by our CP models we show how by slightly changing some constraints in the formulation of the standard problems we can easily obtain new and interesting variants that capture new properties and characteristics of the clusters. In particular, we introduce a new variant of the Label Propagation algorithm and we show that by formulating the DBSCAN as a community discovery problem it become very similar to Label Propagation.

The remaining of the paper is organized as follows. Section 2 discusses the clustering problem and describes the imperative algorithms of *K-medoids*, *DBSCAN* and *Label Propagation*. In Sect. 3, we introduce the constraint programming model for the three clustering algorithms. Section 4 shows the expressive power of CP formulation of the clustering models by introducing some variants of the standard clustering methods. Section 5 discusses the state-of-the-art and, lastly Sect. 6 concludes the paper.

## 2 Clustering Problem

Clustering is one of the most important and well-known data mining methods. The goal of a cluster algorithm is to group together data objects according to some notion of similarity. Therefore, it assigns data objects to different groups (clusters) so that objects that were assigned to the same groups are more similar to each other than to objects assigned to another clusters.

Clustering algorithms can work on different types of data such as text, multimedia, databases, spatio-temporal databases, networks and so on. The main requirement is to have objects described by a set of features or relationships that the algorithm uses to measure the similarity between objects and determine the object's cluster. In the literature, different algorithms have been proposed and each one is designed to find clusters with specific properties [6, 14, 15, 18]. Other clustering algorithms have been designed to work in particular data having a specific structure such as network data. In this field, the clustering algorithms are commonly called *community discovery algorithms*. The concept of a “community” in a (web, social, or informational) network is intuitively understood as a set of individuals that are very similar, or close, to each other, more than to anybody else outside the community. An exhaustive survey about this kind of algorithms are discussed in [8].

Most of the clustering algorithms can be seen as typical optimization problems because in their search of finding a grouping of data objects, they try to

optimize a criterion, such as minimizing the intra-cluster distances or maximizing the inter-cluster distances.

We formally define a clustering  $\mathcal{C}$  as a set of clusters  $C_1, \dots, C_k$  that represent a partition of a set of data points  $P = \{p_1, \dots, p_n\}$  obtained by optimizing a specific criterion; so we have  $C \subseteq P, \forall C \in \mathcal{C} : C \subseteq D, \bigcup \mathcal{C} = D, \bigcap \mathcal{C} = \emptyset$ . Note that we consider non-overlapping clusters. Each point  $p_i$  is represented by an  $m$ -dimensional vector. In order to determine the similarity between data objects we consider a distance function  $d(p_i, p_j)$  between two data objects. The data objects are described by attributes, which may be qualitative or quantitative. In case of quantitative attributes, we consider the euclidean distance between two  $m$ -dimensional vectors.

The optimization criterion determines the quality of the clustering. There are many different ways to measure the goodness of a clustering:

*Sum of Squared Inter-cluster Distances.* Given some distance function  $d(\cdot, \cdot)$  over points, e.g., the Euclidean distance, we can measure the sum of squared distances over each cluster as follows:  $\sum_{C \in \mathcal{C}} \sum_{i,j \in |C|, i < j} d^2(p_i, p_j)$ .

*Sum of Squared Error to Centroid.* A more common measure is the “error” of each cluster, that is, the distance of each point in the cluster to the mean (centroid) of that cluster.

Considering the *centroid* of a cluster as the mean of the data points that belong to it, i.e.,  $m_C = \frac{\sum_{p \in C} p}{|C|}$  then, the sum of squared error is measured as:  $\sum_{C \in \mathcal{C}} \sum_{p \in C} d^2(p, m_C)$ .

*Sum of Squared Error to Medoids.* Instead of using the mean (centroid) of the cluster, we can also use the medoid of the cluster, that is, the point that is most representative of the cluster. Let the medoid of a cluster be the point with smallest average distance to the other points:  $m_C = \operatorname{argmin}_{y \in C} \frac{\sum_{p \in C} d^2(p, y)}{|C|}$ . The sum of squared error to the medoids is then measured as follows:  $\sum_{C \in \mathcal{C}} \sum_{p \in C} d^2(p, m_C)$ .

*Cluster Diameter.* Another measure of coherence is to measure the diameter of the largest cluster, where the diameter is the largest distance between any two points of a cluster. This leads to the following measure of maximum cluster diameter:  $\max_{C \in \mathcal{C}} \max_{i,j \in |C|, i < j} d(p_i, p_j)$ .

*Inter-cluster Margin.* The margin between two clusters is the minimal distance between any two points that belong to the different clusters. The margin gives an indication of how different the clusters are from each other (e.g. how far apart they are). This can be optimized using the following measure of minimum inter-cluster margin:  $\min_{i,j \in |C|, i < j} \min_{p_1 \in C_i, p_2 \in C_j} d(p_1, p_2)$ .

## 2.1 Prototype-Based Clustering

Among the most studied and applied clustering methods there are the prototype-based ones that require the number of clusters to be known in advance. These

techniques create a partitioning of the data where each cluster (partition) is represented by a prototype called the *centroid* of the cluster. Two popular algorithms employing this approach are *K-means* and *K-medoids*. K-means represents each centroid as the average of all points in the cluster, while K-medoids considers the most representative actual point in the cluster. Both methods are heuristic algorithms, that operate as follows: given a user-specified value  $k$ , the algorithm selects  $k$  initial centroids. Successively, each point is assigned to the closest centroid based on a distance measure. Finally, the centroids are updated iteratively based on the points assigned to the clusters. This process stops when centroids do not change. The quality of the clustering is computed as the sum of squared distances of each cluster compared to its centroid. The goal is to minimize this value. Therefore, given a set of  $n$  points, and a  $k$  value, the aim is to find an assignment of points that minimizes the Sum of Squared Error to centroid (SSE):

$$SSE = \sum_{C \in \mathcal{C}} \sum_{p \in C} d^2(p, m_C)$$

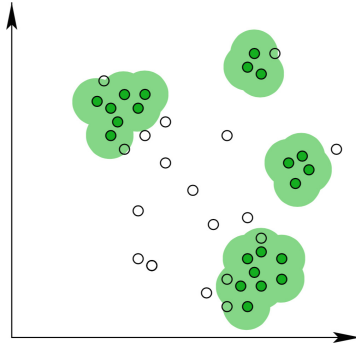
where  $m_C$  is the centroid of the cluster  $C$  (i.e., the mean of the data points that belong to  $C$  in K-means while the medoid in K-medoids).

## 2.2 Density-Based Clustering

The approaches presented in Sect. 2.1 require to know in advance the number of clusters to be found. Moreover, they tend to provide clusters with globular shapes. Unfortunately, in many real applications, we are in the presence of non-globular regions or regions that are quite dense surrounded by areas with low density, typically formed by noise. In this perspective, clusters can also be defined implicitly by the regions of higher data density, separated from each other by regions of lower density. The price for this flexibility is a difficult interpretation of the obtained clusters. One of the most famous algorithm based on the notion of density of regions is DBSCAN [15]. This approach does not rely on an optimization algorithm, it is based on measuring the data density at a certain region of the data space and then, defining clusters as regions that exceed a certain density threshold. The final clusters are obtained by connecting neighboring dense regions. Figure 1 shows an example for the two-dimensional space. Four groups are recognized as clusters and they are separated by an area where the data density is too low. DBSCAN identifies three different classes of points:

**Core points.** These points are in the interior of a density-based cluster. A point is a core point if the number of points within a given neighborhood around the point as determined by the distance function and a user-specified distance parameter,  $\epsilon$ , exceeds a certain threshold,  $MinPts$ , which is also a user-specified parameter.

**Border points.** These points are not core points, but fall within the neighborhood of a core point. A border point can fall within the neighborhoods of several core points.



**Fig. 1.** Example of density-based clusters [5].

**Noise points.** A noise point is any point that is neither a core point nor a border point.

The DBSCAN algorithm is the following:

1. Label all points as core, border, or noise points.
2. Eliminate noise points.
3. Put an edge between all core points that are within  $\epsilon$  of each other.
4. Make each group of connected core points into a separate cluster.
5. Assign each border point to one of the clusters of its associated core points.

### 2.3 Label Propagation

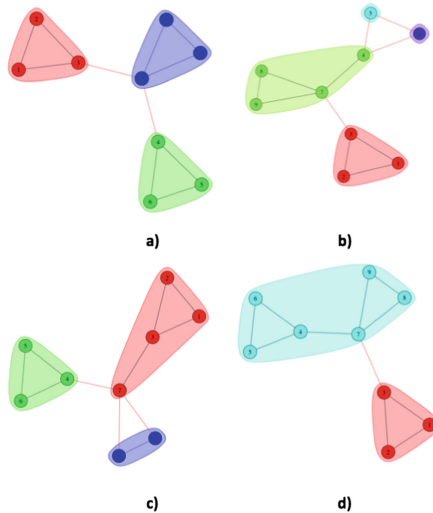
In the network field a task very similar to clustering is *community discovery*, which can be seen as a network variant of standard data clustering. The concept of a “community” in a (web, social, or informational) network is intuitively understood as a set of individuals that are very similar, or close, to each other, more than to anybody else outside the community [8]. This has often been translated in network terms into finding sets of nodes densely connected to each other and sparsely connected with the rest of the network. An interesting community discovery algorithm is the Label Propagation algorithm [23] that detects communities by spreading labels through the edges of the graph and then labeling nodes according to the majority of the labels attached to their neighbors, iterating until a general consensus is reached.

**Iterative Label Propagation (LP).** Suppose that a node  $v$  has neighbors  $v_1, v_2, \dots, v_k$  and that each neighbor carries a label denoting the community that it belongs to. Then,  $v$  determines its community based on the labels of its neighbors. [23] assumes that each node in the network chooses to join the community to which the maximum number of its neighbors belong to. As the labels propagate, densely connected groups of nodes quickly reach a consensus on a unique label. At the end of the propagation process, nodes with the same

labels are grouped together as one community. Clearly, a node with an equal maximum number of neighbors in two or more communities will take one of the two labels by a random choice. For clarity, we report here the procedure of the LP algorithm. Note that, in the following  $C_v(t)$  denotes the label assigned to the node  $v$  at time (or iteration)  $t$ .

1. Initialize the labels at all nodes in the network. For any node  $v$ ,  $C_v(0) = v$ .
2. Set  $t = 1$ .
3. Arrange the nodes in the network in a random order and set it to  $V$ .
4. For each  $v_i \in V$ , in the specific order, let  $C_{v_i}(t) = f(C_{v_{i1}}(t-1), \dots, C_{v_{ik}}(t-1))$ .  $f$  here returns the label occurring with the highest frequency among neighbors and ties are broken uniformly randomly.
5. If every node has a label that the maximum number of their neighbors have, or  $t$  hits a maximum number of iterations  $t_{max}$  then stop the algorithm. Else, set  $t = t + 1$  and go to (3).

The drawback of this algorithm is the fact that *ties are broken uniformly randomly*. This random behavior can lead to different results for different executions and some of these results cannot be optimal. In Fig. 2, we show how given the same network as input of LP we obtain four different results.



**Fig. 2.** The result of four executions of LP Algorithm.

## 2.4 Constraint-Based Clustering

Clustering algorithms are generally used in an unsupervised way, i.e., they do not require any external knowledge to be run. However, sometimes in real application



domains, it is often the case that the data analyst possesses some background knowledge (about the application domain or the data set) that could be useful in clustering the data. Therefore, incorporating user knowledge into algorithms could make them better both in terms of efficiency and quality of results. These constraints help to reduce the task complexity and to find clusters that satisfy user-specified constraints.

The introduction of user-specified constraints in the standard algorithms leads to the study of a new branch of clustering algorithms [3, 22, 24, 27]. The possible constraints that can be specified are *cluster-level constraints*, that define some specific requirements on clusters such as the minimum or the maximum number of elements, and *instance-level constraints*, that define a property of two data object such as the fact that they must be or cannot be in the same cluster [25]. In the following we recall the definition of the most important user constraints:

- *Must-link constraint* requires that two points belong to the same cluster, so given two points  $p_i$  and  $p_j$  it is expressed by:  $\exists C \in \mathcal{C} : p_i \in C \wedge p_j \in C$ .
- *Cannot-link constraint* requires that two points belong to a different clusters, so given two points  $p_i$  and  $p_j$  it is expressed by:  $\forall C \in \mathcal{C} : \neg(p_i \in C \wedge p_j \in C)$ .
- *Cluster size constraints* require that the found clusters have a minimum or a maximum numbers of elements. These constraints are respectively expressed by:  $\forall C \in \mathcal{C} : |C| \geq \alpha$  and  $\forall C \in \mathcal{C} : |C| \leq \alpha$ .
- *Cluster diameter constraint* requires that each cluster must have a diameter at most  $\beta$ . In this case, we express this constraint by:  $\forall C \in \mathcal{C}, \forall p_i, p_j \in C : d(p_i, p_j) \leq \beta$ .
- *Margin constraint* requires that the margin between any two different clusters to be above a certain threshold  $\delta$ . This constraint is also called  $\delta$ -constraint and is expressed as follows:  $\forall C, C' \in \mathcal{C}, \forall p_i \in C, p_j \in C' : d(p_i, p_j) \geq \delta$ .

### 3 Modeling Clustering by Constraint Programming

This paper proposes a constraint programming formulation of some of the most famous clustering methods: *K-medoids*, *DBSCAN* and *Label Propagation*. Constraint Programming (CP) is a declarative programming paradigm where the relations between variables are defined in terms of constraints. Constraints do not specify a step or sequence of steps to be executed (as in imperative programming), but rather the properties of a solution to be found. CP is a powerful paradigm in solving combinatorial search problems. A Constraint Satisfaction Problem (CSP) is a triple  $(V, D, Y)$  composed of a set of variables  $V$ , a set of domains  $D$  and a set of constraints  $Y$  over the variables  $V$ . Sometimes a CSP can be associated to an objective function to be optimized to find a solution. In this case we have a Constraint Optimization Problem that is a quadruple  $(V, D, Y, f)$ , where we have also the objective function  $f$  that must be minimized or maximized.

The clustering problems, described as an optimization problem, must optimize one of the criterion specified in Sect. 2 representing the quality of the clustering.

In this paper, we introduce the formalization of some clustering methods as optimization problems by using constraint programming formulation. We denote by  $A$  the matrix representing the possible clusters where  $a_{i,j} = 1$  if data point  $p_i$  belongs to the cluster  $j$  and  $a_{i,j} = 0$  otherwise.

Clearly, we re-define all the user-specified constraints described above by using the constraint programming formulation. Therefore, we have:

- *Must-link constraint* between two points  $p_i$  and  $p_j$  can be expressed by:  $\exists t \in \mathcal{C} : a_{i,t} + a_{j,t} = 2$ .
- *Cannot-link constraint* between two points  $p_i$  and  $p_j$  can be expressed by:  $\forall t \in \mathcal{C} : a_{i,t} + a_{j,t} \leq 1$ .
- *Cluster size constraints* can be expressed as follows:  $\forall t \in \mathcal{C} : \sum_{\forall i} a_{i,t} \geq \alpha$  and  $\forall t \in \mathcal{C} : \sum_{\forall i} a_{i,t} \leq \alpha$ .
- *Cluster diameter constraint*, given a threshold  $\beta$ , can be formulate as:  $\forall p_i, p_j. i < j$  and  $d(p_i, p_j) \geq \beta \rightarrow \forall t \in \mathcal{C} a_{i,t} + a_{j,t} = 2$ .
- *Margin constraint*, given a threshold  $\delta$ , can be expressed as:  $\forall p_i, p_j. i < j$  and  $d(p_i, p_j) \geq \delta \rightarrow \forall t \in \mathcal{C} a_{i,t} + a_{j,t} \leq 1$ .

### 3.1 CP Model for K-medoid Clustering

In this section, we provide a constraint formalization of the K-medoid clustering starting from the definition of SSE to medoids introduced in Sect. 2. In this context, given a set of points and a parameter  $K$  specifying the number of clusters to find, the aim is to find an assignment of points as well as medoids such that the sum of squared error to medoids is minimized. We recall that the problem of finding clusters providing an optimal solution considering Euclidean metric measure for a general value of  $K$ , e.g. minimizing SSE, is a NP Hard problem [17].

The proposed model assumes that both the *assignment of points to clusters* and *the actual medoids* are discovered during solution search. More precisely we do not explicitly constrain the medoid according to its cluster and inversely, we do not impose any constraints among the points. As shown in Table 1, we introduce all the variables representing point membership (4) and medoids (5). The matrix  $A$  stores the clustering partition. In particular, the final model is represented by an assignment of the points to the clusters. Moreover, the model provides the selection of most representative points as medoids in  $m_{i,j}$ . The K-medoid algorithm is based on euclidean distance computed by the function  $d(i, j)$ .

Our formalization defines the optimization function, as proposed in Sect. 2, i.e. it sums for all points  $i \in P$  and cluster  $C_j \in \mathcal{C}$ , the squared distance between  $i$  and the medoid of  $j$ . Furthermore, two additional constraints are required to enforce that a point can only belong to one cluster (7) implying that  $C_i \cap C_j = \emptyset$ ,  $\forall C_i, C_j \in \mathcal{C}$ , and each cluster can have only one medoid (8).

**Table 1.** A K-medoid based clustering formulation.

$$\begin{aligned}
 & \underset{\mathcal{C}}{\text{minimize}} \quad SSE(\mathcal{C}), & (1) \\
 & \text{s.t.} \\
 & \quad P \text{ is the set of points} & (2) \\
 & \quad K \text{ number of required clusters} & (3) \\
 & \quad a_{i,C_j} \in \{0, 1\} : a_{i,C_j} = 1 \text{ iff point } i \in P \text{ is assigned to cluster } C_j \in \mathcal{C} & (4) \\
 & \quad m_{i,C_j} \in \{0, 1\} : m_{i,C_j} = 1 \text{ iff point } i \in P \text{ is the medoid of cluster } C_j \in \mathcal{C} & (5) \\
 & \quad SSE(C_j \in \mathcal{C}) = \sum_{i \in P} a_{i,C_j} \sum_{h \in P \& h \neq i} m_{h,C_j} d^2(i, h) & (6) \\
 & \quad \sum_{C_j \in \mathcal{C}} a_{i,C_j} = 1 \quad \forall i \in P & (7) \\
 & \quad \sum_{i \in P} m_{i,C_j} = 1 \quad \forall C_j \in \mathcal{C} & (8) \\
 & \quad \left| \bigcup_{C \in \mathcal{C}} C \right| = |P| & (9) \\
 & \quad |\mathcal{C}| = K & (10)
 \end{aligned}$$

**Table 2.** DBSCAN formulation.

$$\begin{aligned}
 & \text{maximize} \left( \sum_{l_j \in L} \min(1, \sum_{i \in P} k_{i,l_j}) \right), & (11) \\
 & \text{s.t.} \\
 & \quad P \text{ is the set of points} & (12) \\
 & \quad L = \{l_1, \dots, l_n, l_{n+1}\} \text{ is an ordered set of colors (or labels)} & (13) \\
 & \quad a_{i,j} \in \{0, 1\} : a_{i,j} = 1 \text{ iff distance between points } i, j \in P : d(i, j) \leq \epsilon & (14) \\
 & \quad k_{i,l_j} \in \{0, 1\} : k_{i,l_j} = 1 \text{ iff point } i \in P \text{ has color } l_j \in L & (15) \\
 & \quad r_i \in \{0, 1\} : r_i = 1 \text{ iff point } i \in P \text{ is a core point, i.e., } \sum_{j \in P} a_{i,j} \geq \text{minp} & (16) \\
 & \quad \sum_{l_j \in L} k_{i,l_j} = 1 \quad \forall i \in P & (17) \\
 & \quad r_h = 1 \wedge r_p = 1 \wedge a_{h,p} = 1 \wedge k_{h,l_j} = 1 \Rightarrow k_{p,l_j} = 1 & (18) \\
 & \quad r_i = 0 \Rightarrow k_{i,l_j} = 1 & (19) \\
 & \quad \text{where } l_j = \min\{l_j \in L \setminus \{l_{n+1}\} | a_{h,i} = 1 \wedge r_h = 1 \wedge k_{h,l_j} = 1\} \cup \{l_{n+1}\}
 \end{aligned}$$

### 3.2 CP Model for DBSCAN Clustering

Let us now introduce a constraint programming model for the density-based clustering DBSCAN. In particular, we reformulate the problem in the context of networks by considering the set of points  $P$  as nodes and setting an edge between two nodes  $i$  and  $j$ , if the distance between  $i$  and  $j$  is less than a given  $\epsilon$ . Clearly, in this way we have that the neighbors of a node (point)  $i$  are the set of points within a distance  $\epsilon$ . Our intended objective is to capture the basic idea of that “each node has the *same* label of all its neighbors”.

DBSCAN algorithm does not rely on an optimization algorithm however, its constraint programming formulation shows how, re-defining this task as a community discovery problem in a network, this approach becomes very similar to the Label Propagation approach that finds clusters of nodes in networks [23]. Moreover, we show how the expressivity introduced by the formulation of the problem by constraint programming makes the standard problem easy to be extended with other constraints that permit to generate interesting variants of the same problem.

More in detail, the model in Table 2 is described as follows. Variables  $k_{i,l_j}$  denotes the color (label  $l_j$ ) of a point (node  $i$ ). Variable  $a_{i,j}$  indicates the presence or absence of an edge between two nodes  $i$  and  $j$ . Variables  $r_i$  denote whether node  $i$  is a core point or not.

We also consider two domains: (a) the set of points  $P$ , and (b) the ordered set of colors  $L$ . Note that in the set of colors we have the color  $l_{n+1}$  that is an additional color used for coloring the noise points. The model imposes that a point has one and only one color, and that all the connected core points must have the same color (18). Another requirement is that each point that is not a core point takes the same color of the core points that are connected to it. If it does not have any core point around, then this point takes the additional color  $l_{n+1}$  because it is a noise (19). Such a constraint also captures the special case in which the point  $i$  can be connected to more than one core with different colors. In this case, the model assigns to  $i$  the color of the core point that in the ordered set  $C \setminus \{l_{n+1}\}$  has a lower rank. Finally, the model is intended to maximize the number of different colors. Notice that a solution where all points have distinct colors does not satisfy Constraint (18) because connected points do not have the same color.

### 3.3 CP Model for Label Propagation

Let us now propose a constraint programming model for the community discovering problem based on label propagation. Our aim is to capture the basic idea that “each node takes the label of the majority of its neighborhood”. The model is reported in Table 3 and described as follows. Variables  $a_{i,j}$  indicate the presence of an edge between nodes  $i$  and  $j$ . Variables  $k_{i,l_j}$  denote the color (label)  $l_j$  of a node  $i$  in the network. There are three domains: (a) the set of nodes  $N$ , (b) the set of edges  $E$ , and (c) an ordered set of colors  $L$ . A node can be assigned one and only one color. Variables  $n_{i,l_h}$  denote the number of neighbors of node

**Table 3.** Label Propagation formulation.

$$\text{maximize}(\sum_{l_j \in L} \min(1, \sum_{i \in N} k_{i,l_j})), \tag{20}$$

s.t.

$$E \text{ is the set of edges} \tag{21}$$

$$N \text{ is the set of nodes} \tag{22}$$

$$L = \{l_1, \dots, l_n\} \text{ is an ordered set of colors (or labels)} \tag{23}$$

$$a_{i,j} \in \{0, 1\} : a_{i,j} = 1 \text{ iff the edge between nodes } i \text{ and } j, (i, j) \in E \tag{24}$$

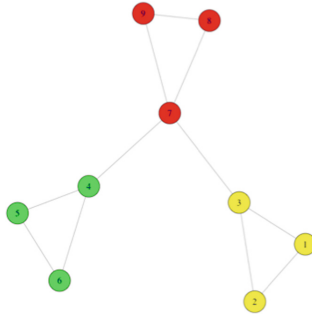
$$k_{i,l_j} \in \{0, 1\} : k_{i,l_j} = 1 \text{ iff node } i \in N \text{ has color } l_j \in L \tag{25}$$

$$\sum_{l_j \in L} k_{i,l_j} = 1 \quad \forall i \in N \tag{26}$$

$$\sum_{\forall j, a_{i,j}=1} k_{j,l_h} = n_{i,l_h} \quad l_h \in L \tag{27}$$

$$\forall i \in N : k_{i,l_j} = 1 \text{ where } l_j = \min\{l_j \in L | \max\{n_{i,l_1}, \dots, n_{i,l_n}\} = n_{i,l_j}\} \tag{28}$$

$i$  with assigned color  $l_h$ . The model assigns to the node  $i$  the color  $l_h$  if it is the most popular among its neighbors, as shown in (28). Such a constraint also captures the case of ties. In such a case, node  $i$  is assigned the color that has the lowest rank in the ordered set  $L$ . Finally, the model maximizes the number of different colors in the network, as shown in (20).



**Fig. 3.** The result of the execution of CP-LP model.

This model highlights the similarity between Label Propagation and the Density-based clustering problem, and thanks to the constraint programming formulation we can notice that the model for density-based clustering is a variant of the standard label propagation. Indeed, the only difference is due to the fact that the Density-based model requires that “each node has the *same* label

**Table 4.** Formulation of a variant of standard Label Propagation.

$$\text{maximize} \left( \sum_{l_j \in L} \min \left( 1, \sum_{i \in N} k_{i,c_j} \right) \right), \quad (29)$$

*s.t.*

$$E \text{ is the set of edges} \quad (30)$$

$$N \text{ is the set of nodes} \quad (31)$$

$$L = \{l_1, \dots, l_n\} \text{ is an ordered set of colors (or labels)} \quad (32)$$

$$a_{i,j} \in \{0, 1\} : a_{i,j} = 1 \text{ iff the edge between nodes } i \text{ and } j, (i, j) \in E \quad (33)$$

$$k_{i,l_j} \in \{0, 1\} : k_{i,l_j} = 1 \text{ iff node } i \in N \text{ has color } l_j \in L \quad (34)$$

$$g_i \in \{0, 1\} : g_i = 1 \text{ iff the node } i \text{ has degree } \geq \gamma \quad (35)$$

$$\sum_{l_j \in L} k_{i,l_j} = 1 \quad \forall i \in N \quad (36)$$

$$\sum_{\forall j. a_{i,j}=1} k_{j,l_h} g_j = n_{i,l_h} \quad l_h \in L \quad (37)$$

$$\forall i \in N : k_{i,l_j} = 1 \text{ where } l_j = \min\{l_j \in L \mid \max\{n_{i,l_1}, \dots, n_{i,l_n}\} = n_{i,l_j}\} \quad (38)$$

of all its neighbors”, and not the *most frequent* label. Constraints (18) and (19) in Table 2 and Constraint (28) in Table 3 express this difference. By executing our model we obtain the optimal solution depicted in Fig. 3, where we consider as input the same network in Fig. 2.

## 4 Variants of the Standard Clustering Algorithms

The expressivity introduced by the formulation of the problems by constraint programming makes the standard methods easy to be extended with other constraints that permit to generate interesting variants.

For each formulation of the clustering problems introduced above, we could easily specify some constraints on the cluster size, or on the maximum diameter of the clusters and so on, by adding one of the user-specified constraints defined in Sect. 3. However, we can also extend the problems by changing one of the constraints of the standard formulation. We extended the standard Label propagation problem as follows.

Given the neighborhood of a node, we give to each node an importance on the basis of its degree; more specifically, in order to compute the most frequent label of its neighborhood the nodes with a degree less than a specific threshold  $\gamma$  are not considered. This means that the propagation of labels is guaranteed by *important* nodes (i.e., nodes with a minimum degree equal to  $\gamma$ ), while the nodes with a low degree takes a label in a passive way. We recall that the degree of a node  $i$  is the number of nodes that are directly connected to  $i$ .

As reported in Table 4, the model presented in Sect. 3.3 can be easily transformed in order to consider this new constraint. We can simply add a variable  $g_i$  that is equal to 0 if the node  $i$  has a degree less than  $\gamma$  (Constraint (35)); then, the computation of the neighborhood takes into consideration also this variable to filter out the nodes with low degree (37).

## 5 Related Work

Initial approaches in constrained clustering focused on the introduction of instance-level constraints, especially *must*- and *cannot*-link [26, 27]. Several properties are related to instance-level constraints [11]. Furthermore, [10, 12, 13] define  $\delta$  and  $\epsilon$ -constraint forcing clustering with specific spatial properties among items. COP-KMeans [27] represents a popular approach for the integration of instance level constraints in the classical K-means approach. The algorithm takes as input the data, the number of cluster required, a set of must-link, and a set of cannot-link. COP-KMeans ensures that when the clustering is updated none of the specified constraints is violated, performing a hard constraint satisfaction. [22] proposes a modification of COP-KMeans introducing an ensemble approach for managing constraint priorities in order to provide a model even though all the constraints cannot be satisfied. Similarly, PC-KMeans [3] permits that some constraints can be violated enabling a soft constraint satisfaction. In this case every must- and cannot-link has a weight  $w$  associated to it. By setting the value  $w$ , the algorithm looks for a trade-off between minimizing the total distance between points and cluster centroids, and the cost of violating the constraints. Other approaches do not require that any constraint must be satisfied but they perform metric learning from constraints or adopt an hybrid solution including both constraints satisfaction and metric learning [4]. The set of constraints is mapped into distance measures [2]. The basic idea of these approaches is to weight the features differently, based on their importance in the computation of the distance measure based on the available constraints [7, 28]. E.g. [28] parametrizes the Euclidean distance using a symmetric positive-definite matrix, simultaneously minimizing the distance between must-linked instances and maximizing the distance between cannot-linked instances. A probabilistic model based on Hidden Markov Random Fields (HMRF) can be used for incorporating constraints into prototype-based clustering. In [4] the objective function is derived from the posterior energy of the HMRF framework, and the authors propose an EM-based clustering algorithm (HMRF-KMeans) for finding (local) minimum of this objective function. The distance measure is estimated during clustering to validate the user-specified constraints as well as to incorporate data variance. MPCK-Means [7] learns an individual metric for each cluster allowing violations by imposing penalties. Recently, [16, 21] proposed the use of constraint programming for modeling data mining tasks. In particular, [16] addressed the problem of searching frequent k-patterns, that cover the whole dataset, while [21] proposed a CP formulation of the constraint-based sequence mining task, that has the goal to find sequences of symbols in a large number of input sequences and that

satisfies some constraints specified by the user. In [20], instead authors proposed an approach based on Integer Linear Programming where the model, knowing a set of candidate clusters, searches for the best clustering among the subset of clusters. Integer Linear programming is also used in [1], where a constraint-based approach for K-means is investigated by using column generation; this work is based on [19]. In [9] authors proposed a declarative and generic framework, based on CP, which enables to design clustering tasks by specifying an optimization criterion and some constraints either on the clusters or on pairs of objects.

## 6 Conclusion

In this paper, we have introduced a CP model for a medoid based clustering. Moreover, we have proposed a model for a density based clustering and one for a community discovery problem corresponding to a clustering algorithm in network data. We showed the powerful of the expressivity introduced by the constraint programming formulation that enables an easy integration of the standard clustering problems with other constraints. We presented this aspect in two different ways: first, we showed how the formulation of the density-based clustering by constraint programming makes it very similar to the label propagation problem and then, we proposes a variant of the standard label propagation approach.

**Acknowledgements.** This work was supported by the European Commission under the project Inductive Constraint Programming (ICON) contract number FP7-284715.


## References

1. Babaki, B., Guns, T., Nijssen, S.: Constrained clustering using column generation. In: Simonis, H. (ed.) CPAIOR 2014. LNCS, vol. 8451, pp. 438–454. Springer, Heidelberg (2014)
2. Bar-Hillel, A., Hertz, T., Shental, N., Weinshall, D.: Learning distance functions using equivalence relations. In: ICML, pp. 11–18 (2003)
3. Basu, S., Banerjee, A., Mooney, R.J.: Active semi-supervision for pairwise constrained clustering. In: SDM (2004)
4. Basu, S., Bilenko, M., Mooney, R.J.: A probabilistic framework for semi-supervised clustering. In: KDD, pp. 59–68 (2004)
5. Berthold, M.R., Borgelt, C., Hppner, F., Klawonn, F.: Guide to Intelligent Data Analysis: How to Intelligently Make Sense of Real Data, 1st edn. Springer, London (2010)
6. Bezdek, J.C.: Pattern Recognition with Fuzzy Objective Function Algorithms. Kluwer Academic Publishers, Norwell (1981)
7. Bilenko, M., Basu, S., Mooney, R.J.: Integrating constraints and metric learning in semi-supervised clustering. In: ICML, ACM (2004)
8. Coscia, M., Giannotti, F., Pedreschi, D.: A classification for community discovery methods in complex networks. *Stat. Anal. Data Min.* **4**(5), 512–546 (2011)
9. Dao, T.-B.-H., Duong, K.-C., Vrain, C.: A declarative framework for constrained clustering. In: Blockeel, H., Kersting, K., Nijssen, S., Železný, F. (eds.) ECML PKDD 2013, Part III. LNCS, vol. 8190, pp. 419–434. Springer, Heidelberg (2013)



10. Davidson, I., Ravi, S.S.: Clustering with constraints: feasibility issues and the k-means algorithm. In: SDM (2005)
11. Davidson, I., Ravi, S.S.: Identifying and generating easy sets of constraints for clustering. In: Proceedings, The Twenty-First National Conference on Artificial Intelligence and the Eighteenth Innovative Applications of Artificial Intelligence Conference (AAAI), pp. 336–341 (2006)
12. Davidson, I., Ravi, S.S.: The complexity of non-hierarchical clustering with instance and cluster level constraints. DMKD **14**(1), 25–61 (2007)
13. Davidson, I., Ravi, S.S.: Using instance-level constraints in agglomerative hierarchical clustering: theoretical and empirical results. Data Min. Knowl. Discov. **18**(2), 257–282 (2009)
14. Dunn, J.C.: A fuzzy relative of the isodata process and its use in detecting compact well-separated clusters. J. Cybern. **3**(3), 32–57 (1974)
15. Ester, M., Kriegel, H.-P., Sander, J., Xu, X.: A density-based algorithm for discovering clusters in large spatial databases with noise. In: Simoudis, E., Han, J., Fayyad, U.M. (eds.) KDD, pp. 226–231. AAAI Press (1996)
16. Guns, T., Nijssen, S., Raedt, L.D.: k-pattern set mining under constraints. IEEE Trans. Knowl. Data Eng. **25**(2), 402–418 (2013)
17. Hansen, P., Aloise, D.: A survey on exact methods for minimum sum-of-squares clustering. <http://www.math.iit.edu/Buck65files/msscStLouis.pdf>, pp. 1–2, January 2009
18. Hartigan, J.A., Wong, M.A.: Algorithm AS 136: a k-means clustering algorithm. J. Roy. Stat. Soc. Ser. C (Appl. Stat.) **28**(1), 100–108 (1979)
19. Merle, O.D., Hansen, P., Jaumard, B., Mladenović, N.: An interior point algorithm for minimum sum of squares clustering. SIAM J. Sci. Comput. **21**, 1485–1505 (1997)
20. Mueller, M., Kramer, S.: Integer linear programming models for constrained clustering. In: Pfahringer, B., Holmes, G., Hoffmann, A. (eds.) DS 2010. LNCS, vol. 6332, pp. 159–173. Springer, Heidelberg (2010)
21. Negrevergne, B., Guns, T.: Constraint-based sequence mining using constraint programming. In: Michel, L. (ed.) CPAIOR 2015. LNCS, vol. 9075, pp. 288–305. Springer, Heidelberg (2015)
22. Okabe, M., Yamada, S.: Clustering by learning constraints priorities. In: ICDM, pp. 1050–1055 (2012)
23. Raghavan, U.N., Albert, R., Kumara, S.: Near linear time algorithm to detect community structures in large-scale networks. Phys. Rev. E **76**(2), 036106+ (2007)
24. Ruiz, C., Spiliopoulou, M., Menasalvas, E.: C-DBSCAN: density-based clustering with constraints. In: An, A., Stefanowski, J., Ramanna, S., Butz, C.J., Pedrycz, W., Wang, G. (eds.) RSFDGrC 2007. LNCS (LNAI), vol. 4482, pp. 216–223. Springer, Heidelberg (2007)
25. Wagstaff, K., Cardie, C.: Clustering with instance-level constraints. In: ICML, pp. 1103–1110 (2000)
26. Wagstaff, K., Cardie, C.: Clustering with instance-level constraints. In: AAAI/IAAI, p. 1097 (2000)
27. Wagstaff, K., Cardie, C., Rogers, S., Schrödl, S.: Constrained k-means clustering with background knowledge. In: ICML, pp. 577–584 (2001)
28. Xing, E.P., Ng, A.Y., Jordan, M.I., Russell, S.: Distance metric learning, with application to clustering with side-information. In: Advances in Neural Information Processing Systems, vol. 15, pp. 505–512. MIT Press (2002)

# Towards a Boosted Route Planner Using Individual Mobility Models

Riccardo Guidotti<sup>1,2</sup> and Paolo Cintia<sup>1,2</sup>

<sup>1</sup> KDDLab, University of Pisa, Largo B. Pontecorvo, 3, Pisa, Italy  
{riccardo.guidotti,paolo.cintia}@di.unipi.it

<sup>2</sup> KDDLab, ISTI-CNR, Via G. Moruzzi, 1, Pisa, Italy  
{riccardo.guidotti,paolo.cintia}@isti.cnr.it

**Abstract.** Route planners generally return routes that minimize either the distance covered or the time traveled. However, these routes are rarely considered by people who move in a certain area systematically. Indeed, due to their expertise, they very often prefer different solutions. In this paper we provide an analytic model to study the deviations of the systematic movements from the paths proposed by a route planner. As proxy of human mobility we use real GPS traces and we analyze a set of users which act in Pisa and Florence province. By using appropriate mobility data mining techniques, we extract the GPS systematic movements and we transform them into sequences of road segments. Finally, we calculate the shortest and fastest path from the origin to the destination of each systematic movement and we compare them with the routes mapped on the road network. Our results show that about 30–35% of the systematic movements follow the shortest paths, while the others follow routes which are on average 7 km longer. In addition, we divided the area object of study in cells and we analyzed the deviations in the flows of systematic movements. We found that, these deviations are not only driven by individual mobility behaviors but are a signal of an existing common sense that could be exploited by a route planner.

## 1 Introduction

Route planners are systems which help users selecting a route between two locations. When providing directions, web and mobile mapping services generally suggest the shortest route. Popular route planning system such as Google Maps, Open Street Maps etc. generate diverging directions using powerful libraries of roads and road attributes [16]. However, they often ignore both the time at which a route is to be traveled and, more important, the preferences of the users they serve. Since cities are becoming crowded and jammed, smart route planning are gathering an increasing interest. In such a context, a route planner which takes into account users' preferences [8], and which exploits the crowd expertise w.r.t urban mobility in order to identify the best route, can be more desirable and helpful than an ordinary route planner [6].

A route planner which exploits individual mobility models to improve the planning will have a real advantage from these models only if the users do not

follow the shortest path in their systematic movements but deviate from them. Consequently, the target of this work is twofold. The first one is to understand and estimate how much the systematic movements of a user are different from the shortest paths between the origin and destination locations. The intuition is that a user which lives and acts in a certain territory do not automatically select the shortest path. This can happen for many reasons: e.g. traffic conditions, road quality, for passing close to the cheapest petrol station, for avoiding roads with control of speed etc. However, independently from the reasons, if there is a divergence between the systematic route with origin point  $o$  and destination point  $d$ , and the shortest route from  $o$  to  $d$  suggested by a route planner, then also other users could benefit from this kind of knowledge which comes from individual expertise on a certain area. This lead to the second and main target: a boosted route planner that, when is possible, proposes as alternative to the shortest path a route which is frequently followed by someone. This planner would be a route planner coming from the wisdom of the crowd in mobility.

By exploiting *individual mobility profile* models [15] and *trajectory map-matching* [5] for a set of users in Pisa and Florence province, we retrieved the systematical movements of the users, named *routines*, and we mapped these routines along a road network. By calculating the shortest path from the origin  $o$  to the destination  $d$  of each routine with an ordinary route planner we obtained the movements a user would have followed when there is not expertise of the area. Then we compared the routines with the corresponding shortest path. Thanks to this analysis we are able to (i) quantify how much human mobility differs from the shortest path and, on the other hand how good can be an approximation of human mobility made with the shortest paths, (ii) at which level appears the divergence between the routine and the shortest path w.r.t. origin/destination, and (iii) which are the road intersections, areas and flows of movements in which users mobility detaches more in comparison with the shortest paths.

Our experiments show that about 30–35% of the routines follow the shortest paths, while the others follow routes which are on average 7km longer. In addition, 20% of the routines deviate at the very beginning from the suggested paths. Despite these differences, 60% of the route returned by the planner would belong to the individual mobility profiles. Consequently, even if the analyzed drivers follow routines quite similar to the routes suggested by a route planner, they deviate from them not to minimize the travel distance but for some other unknown reasons. Finally, we discovered a sort of collective “common sense”: when moving from a certain origin to a certain destination nearly all the drivers deviate in the same area. This indicates that different users which systematically drive along the same roads develop similar individual mobility behaviors.

Preliminary techniques are illustrated in Sect. 2. In Sect. 3 we propose our analytic model. We show in Sect. 4 the results of our analysis. In Sect. 5 we summarize some related works on route planning. Finally, Sect. 6 reports a summary of the contributions of the paper and possible future works.

## 2 Preliminaries

Movements are usually performed by people in specific areas and time instants. These people are called *users* or *drivers* and each movement is composed by a sequence of spatio-temporal points  $(x, y, t)$  where  $x$  and  $y$  are the coordinates, while  $t$  is the time stamp. We call *trajectory* the movements of a user described by a sequence of spatio-temporal points:

**Definition 1 (Trajectory).** A trajectory  $m$  is a sequence of spatio-temporal points  $m = [(x_1, y_1, t_1), \dots, (x_n, y_n, t_n)]$  where the spatial points  $(x_i, y_i)$  are sorted by increasing time  $t_i$ , i.e.,  $\forall 1 \leq i \leq k$  we have  $t_i < t_{i+1}$

The set of all the trajectories traveled by a user  $u$  makes her *individual history*:

**Definition 2 (Individual History).** Given a user  $u$ , we define the individual history of  $u$  as the set of traveled trajectories denoted by  $H_u = \{m_1, \dots, m_k\}$ .

### 2.1 Individual Mobility Profiles

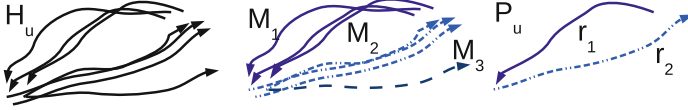
It is possible to extract the systematic movements of a user  $u$  by following the profiling procedure proposed in [15]. This approach groups the trajectories using a clustering algorithm equipped with a *distance function* defining the concept of trajectory similarity:

**Definition 3 (Trajectory Similarity).** Given two trajectories  $m'$  and  $m''$ , a trajectory distance function  $\text{dist}$  and a distance threshold  $\varepsilon$ , we say that  $m'$  is similar to  $m''$  ( $m' \sim m''$ ) iff  $\text{dist}(m', m'') \leq \varepsilon$ .

The result is a partitioning of the original dataset from which the *clusters* with few trajectories and those containing noise are filtered out. Finally, the *representative trajectory* are extracted from the remaining clusters. These representative trajectories are called *routines* and the set of routines is called *mobility profile*:

**Definition 4 (Routine and Mobility Profile).** Let  $H_u$  the individual history of a user  $u$ ,  $ms$  a minimum size threshold,  $\text{dist}$  a distance function and  $\varepsilon$  a distance threshold. Given a grouping function  $\mathcal{M} = \text{group}(H_u, ms, \varepsilon, \text{dist})$ , such that  $\mathcal{M} = \{M_1 \dots M_k\}$  where  $M_i \subset H_u$ , we define a routine  $r_i$  as the medoid trajectory of a group  $M_i$ . The set of routines extracted from  $\mathcal{M}$  is called mobility profile and is denoted by  $P_u = \{r_1 \dots r_k\}$ .

A *mobility profile* describes an abstraction in space and time of the systematic movements: the user's real movements are represented by a set of trajectories delineating the generic paths followed. Moreover, the exceptional movements are ignored due to the fact they will not be part of the profile. Figure 1 depicts an example of mobility profile extraction. We name  $\text{getmedoids}(\mathcal{M})$  the function that takes in input the output of  $\text{group}()$  and returns the routines, i.e. the medoid trajectories  $\{r_1 \dots r_k\}$  of the groups in  $\mathcal{M}$  describing the mobility profile  $P_u$ .



**Fig. 1.** The user *individual history* ( $H_u$ ), the clusters identified by the grouping function ( $M_1, M_2, M_3$ ) and the extracted *individual routines* ( $P_u = \{r_1, r_2\}$ ) forming the *individual mobility profile*.

## 2.2 Trajectory Map Matching

A trajectory  $m$  coming from GPS or GSM dataset generally does not contain the relative reversed road network segments. Such enrichment might not be straightforward, especially when raw trajectory data have a high sampling rate. This lack of information can be restored by means of some *map matching* techniques. We adopted the *gravity model* [5] as method to match each single trajectory point to the road segment it belongs to:

**Definition 5 (Gravity Force Attraction).** Given a point  $p_i$  and a set of road segments describing the road network  $S = \{s_1, \dots, s_r\}$  where  $s_j = \{p_{start}, p_{end}\}$ , we define the gravity force attraction of a segment  $s_j$  for a point  $p_i$  as:

$$GFA(p_i, s_j) = w_{(p_i, s_j)}^d = w_{(p_i, s_j)}^\theta$$

where  $w_{(p_i, s_j)}^d = 1 - \frac{\text{dist}(p_i, s_j)}{\sum_{s_k \in S} \text{dist}(p_i, s_k)}$ ,  $w_{(p_i, s_j)}^\theta = 1 - \frac{\text{ang}(p_i, r_j)}{\sum_{s_k \in S} \text{ang}(p_i, s_k)}$ ,  $\text{dist}$  is the euclidean distance between a point and a segment, and  $\text{ang}$  is the absolute difference between the direction of the point and the direction of the segment.

This model can be applied over the whole road network segments. However, in real applications the set of segments  $S$  to be considered can be very large. For this reason, it is possible to use a *nearest neighbor* approach and consider only a subset  $S_k \subset S$  containing the  $k$  segments closest to a given point.

Given a GPS trajectory  $m = \{p_1, \dots, p_n\}$  and a set of road segments  $S$ , it is possible to assign each point  $p_i$  to the segment with the most powerful force  $\bar{s}_j = \sigma(p_i, S, k) = \text{argmax}_{s_j \in S_k} (GFA(p_i, s_j))$ . The Gravity Model adopted has also been used to estimate the traveltime of each matched road segment; once every trajectory point have been matched, the typical travel time of a segment  $s$ , given  $P$  the set of points matched to  $s$ , is defined as  $\frac{\sum_{p_i \in P} \text{speed}(p_i) * GFA(p_i, s)}{\sum_{p_i \in P} GFA(p_i, s)}$ . Road network travel times have been estimated from a dataset composed by 9.8 millions car travel.

**Definition 6 (Trajectory Map Matching).** Given a trajectory  $m$  and a set of road segments  $S$ , we refer to  $m^*$  as the trajectory  $m$  on the road segment network  $S$ , i.e. the points of  $m^*$  belong to the segments in  $S$ :

$$m^* = \text{mapmatch}(m, S, k)$$

where  $m^* = [p_1^*, \dots, p_n^*] = [\bar{s}_1, \dots, \bar{s}_{n-1}]$  and  $[p_i^*, p_{i+1}^*] = \bar{s}_j = \sigma(p_i, S, k)$

Thus,  $m$  can be transformed in the map matched version  $m^* = [p_1^*, \dots, p_n^*]$  containing points which belong to the road segments  $S$ , where  $p_1^*, \dots, p_n^*$  maximize the attractions with  $p_1, \dots, p_n$ , i.e.  $m^*$  is the best representation of  $m$  on  $S$ . Note that it is sufficient to set  $k$  greater than one to guarantee the denominators be different to zero. A refinement is needed to obtain the map matched trajectory  $m^*$ , i.e. a path must be added for each couple of points which are not directly connected. This is a common case when low sampled GPS data are involved: in this scenario, a GPS point every  $\sim 90$  s is recorded. To find such path followed by the driver, we used a Time-Aware heuristic as described in [4]. This map-matching method takes the GPS travel time between the two consecutive GPS point as input and returns the path connecting the two points that better fit the input travel time. It is worth to consider that the road network is a directed graph, thus including and correctly recognizing one way segments.

### 3 Proposed Analytic Model

In the following we describe the analytic model adopted to discover how much the shortest/fastest path can approximate the systematic movements of a user, and how much a route planner could improve its performances by using the wisdom of systematic drivers.

Given a set of users  $U$  and set of road segments  $S$ , for each user  $u \in U$ , we calculate the individual mobility profile

$$P_u = \text{getmedoids}(\text{group}(H_u, ms, \varepsilon, \text{dist}))$$

Then for each routine  $r_i \in P_u$ , we map match the routine on the road network

$$r_i^* = \text{mapmatch}(r_i, S, k)$$

We name *map matched individual mobility profile*  $P_u^* = \{r_1^*, \dots, r_k^*\}$  the profile of a user  $u$  containing the routines mapped on the road network.

We define a route planner

$$\bar{m} = \text{routeplanner}_{\text{type}}(o, d, S)$$

as a function which returns the best path  $\bar{m} = [o, \bar{p}_2, \dots, \bar{p}_{n-1}, d]$  w.r.t. the type of search  $\text{type} \in \{s, f\}$  (where  $s$  stands for *shortest* and  $f$  stands for *fastest*) on the road segments  $S$  where  $o$  is the origin point and  $d$  is the destination point.

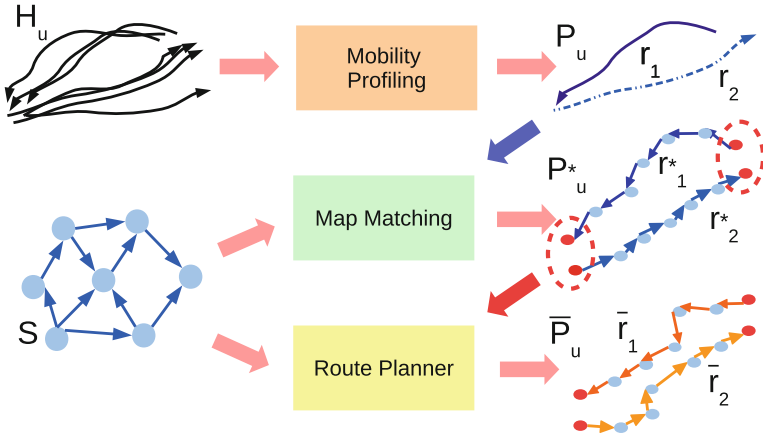
Finally, for each routine  $r_i^* = [o_i, \dots, d_i] \in P_u^*$  we calculate the path returned by the route planner  $\bar{r}_i = \text{routeplanner}_{\text{type}}(o_i, d_i, S)$  on the origin and destination. We indicate with  $\bar{P}_u^{\text{type}} = \{\bar{r}_1, \dots, \bar{r}_k\}$  the *shortest/fastest individual mobility profile* of a user  $u$  containing the paths returned by the route planner.

Summing up, given a set of users  $U$  and their individual history  $H_u \forall u \in U$ , and the road network segments set  $S$  we obtain:

1.  $P_u \forall u \in U$  with the *Mobility Profiles* step as result of the application of  $\text{group}()$  and  $\text{getmedoids}()$  using  $H_u$  for each  $u \in U$ ;

2.  $P_u^* \forall u \in U$  through the *Map Matching* step as result of the application of *mapmatch()* for each  $r_i \in P_u, \forall u \in U$ ;
3.  $\bar{P}_u^{type} \forall u \in U$  by means of the *Route Planner* step as result of the application of *routeplanner()* on the origin and destination points  $o_i, d_i$  of for each  $r_i^* \in P_u^*, \forall u \in U$ .

Figure 2 shows the steps of the analytic mobility model. In the next section we will observe the differences between  $P_u^*$  and  $\bar{P}_u^s, \bar{P}_u^f$ . We remark that the *shortest path* is the path which minimizes the distance, while the *fastest path* is the path which minimizes the travel time.



**Fig. 2.** Steps of the analytic mobility model. Input: individual history  $H_u$ , road network segments set  $S$ . Output: individual map matched mobility profile  $P_u^*$ , individual shortest/fastest mobility profile  $\bar{P}_u^{type}$ .  $P_u$  is calculated by using the *Mobility Profiling* functions. Then, the *Map Matching* module produces  $P_u^*$  by using the routines in  $P_u$ . Finally,  $\bar{P}_u^{type}$  is obtained by using the *Route Planner* on the origin and destination points (highlighted in the red dotted circles) of the routines in  $P_u^*$ .

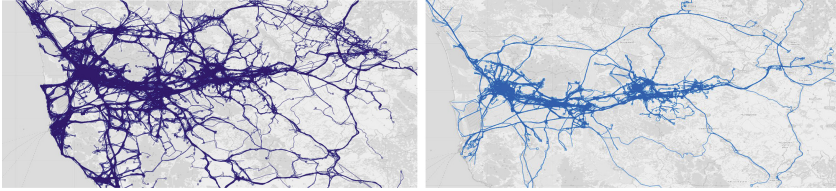
## 4 Experiments

In the following we evaluate how much systematic users described by their map matched individual mobility profile  $P_u^*$  deviate from the shortest and fastest routes contained in the shortest mobility profile  $P_u^s$  and fastest mobility profile  $P_u^f$  for the provinces of Pisa and Florence. Moreover we analyze which are the nodes on the road network  $S$ , the areas and the flows more affected by deviations.

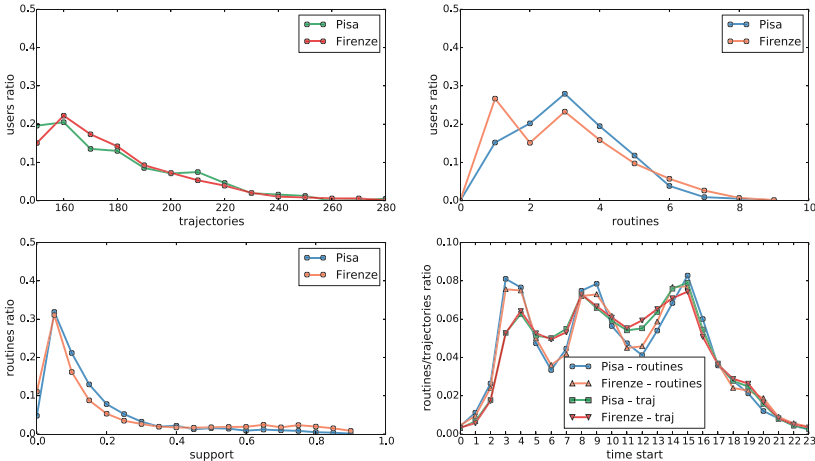
### 4.1 Dataset

As a proxy of human mobility, we use real GPS traces collected for insurance purposes by *Octo Telematics S.p.A*<sup>1</sup>. This dataset contains 9.8 million car travels

<sup>1</sup> <http://www.octotelematics.com/it>.



**Fig. 3.** (Left) A sample of the considered trajectories in Pisa province. (Right) Mobility profiles extracted in Pisa province.



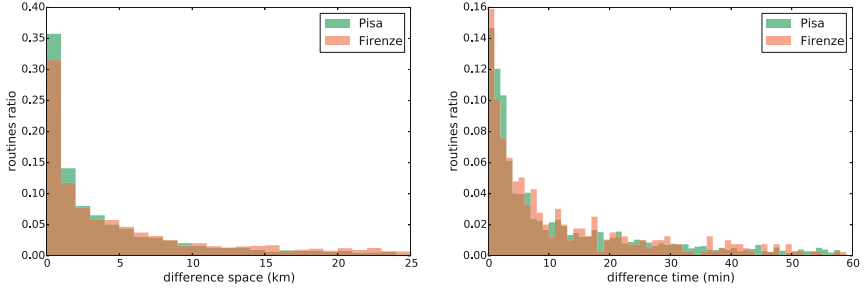
**Fig. 4.** Distributions of number of trajectories (top - left), number of routines (top - right), routine relative support (bottom - left), trajectories and routines starting time (bottom - right).

performed by about 160,000 vehicles active in a geographical area focused on Tuscany (Italy) in a period from 1st May to 31st May 2011. Figure 3-left depicts a sample of the considered trajectories. In our analysis we split geographically the dataset in provinces to consider the fact that each area has its type of mobility with characteristics depending on the surface, on the topology and on the number of inhabitants. In this paper we present the results obtained for the provinces of Pisa and Florence. A user is analyzed in one province if at least one of his/her trajectories passes through that province. In particular we analyzed a subset of 3,000 representative users which have traveled along a total of about 500.000 trajectories. The individual history  $H_u$  represents our input data.

## 4.2 Mobility Profiles Analysis

To perform the *Mobility Profiling* step, we used as profiling function  $profile()$  the clustering algorithm Optics [1], and as distance function  $dist()$  a function which compares the points distances along the trajectories (or an interpolation of





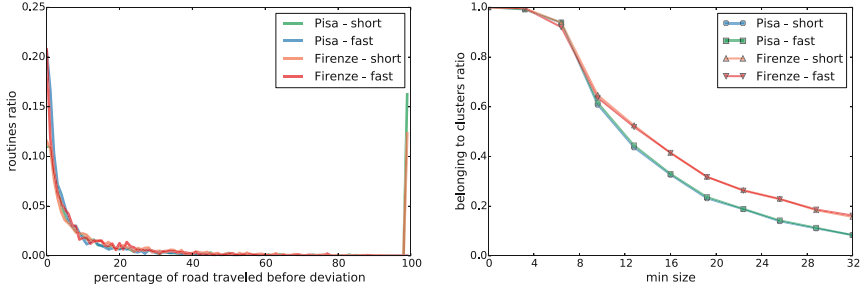
**Fig. 5.** (Left) Space difference distribution in km between the routines in  $P_u^*$  and the corresponding routines in  $\bar{P}_u^s$ . (Right) Time difference distribution in minutes between the routines in  $P_u^*$  and the corresponding routines in  $\bar{P}_u^f$ .

them) and returns the average of these comparisons. In order to obtain sound and reliable routines we performed some preliminary tests to set the best parameters to extract the mobility profiles  $P_u$ . We choose  $\varepsilon = 500$  m and  $ms = 8$  since a routine is a movement that must be repeated a significant number of time during a month. Figure 3-right depicts an example of profile extracted in Pisa province modeling the users' systematic movements.

In Fig. 4(top) we can observe the distributions of the number of trajectories and number of routines per user (left and right respectively). All the users selected have more than 150 trajectories and most of them has 160 with an average of about 200 trajectories. Most of the individual mobility profiles  $P_u$  contain 1 – 4 routines. The average length of a routine is about 8.87 km ( $\pm 8.96$  km of standard deviation), while the average duration is about 20 min ( $\pm 12$  min standard deviation). In Fig. 4(bottom - left) we can observe that most of the routines have a relative support of 0.2 of the trajectories. This means for example that given a user with 160 trajectories and a routine with support equals to 0.2, then that routine is supported by about 30 trajectories, i.e. a trajectory per day on average in the observation period. Finally, the starting time distributions of trajectories and routines is depicted in Fig. 4(bottom - right). Note how the starting time distribution of the routines, more than the starting time distribution of the trajectories, follows a clear M-shape pattern. This highlight how the routines capture the systematic movements from home to work in the morning and from work to home in the afternoon.

### 4.3 Deviation Analysis

As first experiment we analyzed the deviation in term of space difference from the routines in  $P_u^*$  to those in shortest path  $\bar{P}_u^s$ , and the deviation in term of time difference from the routines in  $P_u^*$  to those in fastest path  $\bar{P}_u^f$ . In particular, for each user  $u \in U$  analyzed, for each routine in  $r_i^* = \{o_i, \dots, d_i\} \in P_u^*$ , we calculated the difference in length with the corresponding route in  $\bar{P}_u^{\{s,f\}}$ , i.e. the route  $\bar{r}_i$  which starts in  $o_i$  and ends in  $d_i$ . Note that the following results



**Fig. 6.** (Left) Distribution of the percentage of road traveled before the routine deviates from the shortest/fastest path. (Right) Ratio of shortest and fastest routes belonging to the clusters of the corresponding routines by varying the *minsize* parameter.

**Table 1.** median, average and standard deviation of the space difference (km), time difference (min) and relative percentage of road traveled before the deviation (*pbd*).

	Short - space diff			Fast - time diff			Short - pbd			Fast - pbd		
	med	avg	std	med	avg	std	med	avg	std	med	avg	std
Pisa	02.31	07.16	13.56	07.42	26.92	58.13	07.07	25.14	35.52	07.96	23.19	32.33
Florence	03.64	10.22	18.45	07.31	19.06	29.90	02.97	07.58	13.54	01.05	01.58	21.58

are biased by the route planner used: by applying different route planners the shortest and fastest path obtained could be different.

In Fig. 5 we can observe the space and time differences distributions. With respect to the shortest path (left in the figure), in both dataset there is a consistent set of routines with space difference equals to zero. This indicates that 30–35 % of the routines (for Pisa and Florence respectively) follow the shortest path suggested by the route planner. The remaining routines differentiate on average of 7 km (see Table 1). On the other hand, in Fig. 5(right) none of the routines follows exactly the fastest path. Just few routines, i.e. the 10 %, follow the fastest routes with less than a minute of difference. All the others differentiate consistently (20 min on average Table 1). In addition, we observed that 15 % of the drivers in Pisa and 10 % of the drivers in Florence have the individual mobility profile exactly equal to the shortest mobility profile ( $P_u^* = \bar{P}_u^s$ ). On the contrary, none of the user has all the routines equal to the fastest path, i.e.  $P_u^* = \bar{P}_u^f$

In Fig. 6 left is reported the percentage of road traveled before the deviation (*pbd*), both for Pisa and Florence. It is obtained by observing after how much  $r_i^*$  deviates from  $\bar{r}_i$  after the start point  $o_i$  (for  $\bar{r}_i \in \bar{P}_u^s$  and  $\bar{r}_i \in \bar{P}_u^f$ ). We can notice how 20 % of the systematic movements deviate from the shortest/fastest paths at the very beginning. The distribution is a long tailed power law with average percentage before deviation of 7 % and 3 % for Pisa and Florence respectively (see Table 1). Furthermore, how already observed, there is a consistent subset of routines (12–15 %) which do not deviate from the shortest path. This does not occur for the fastest path.

Finally, we studied the percentage of shortest/fastest movements which would have belonged to the clusters by varying the *minsize* (*ms*) parameter (Fig. 6 right). We calculated for each user  $u \in U$  the trajectory distance (using the same distance function *dist* applied for the clustering) between the short/fast paths  $\bar{r}_1 \dots \bar{r}_k$  and the trajectories belonging to the corresponding cluster  $M_1 \dots M_k$ . For *minsize* = 8 (the value used for the clustering), 60% of the movements returned by the route planner would have belonged to the clusters in both shortest and fastest path. This indicates that the movements returned by the route planner are similar enough to the trajectories belonging to the cluster to be considered part of them. This fact is quite interesting if we consider that the space and time difference between routines and suggested routes are in some cases not negligible, and that the routines generally deviate not far from the origin point.

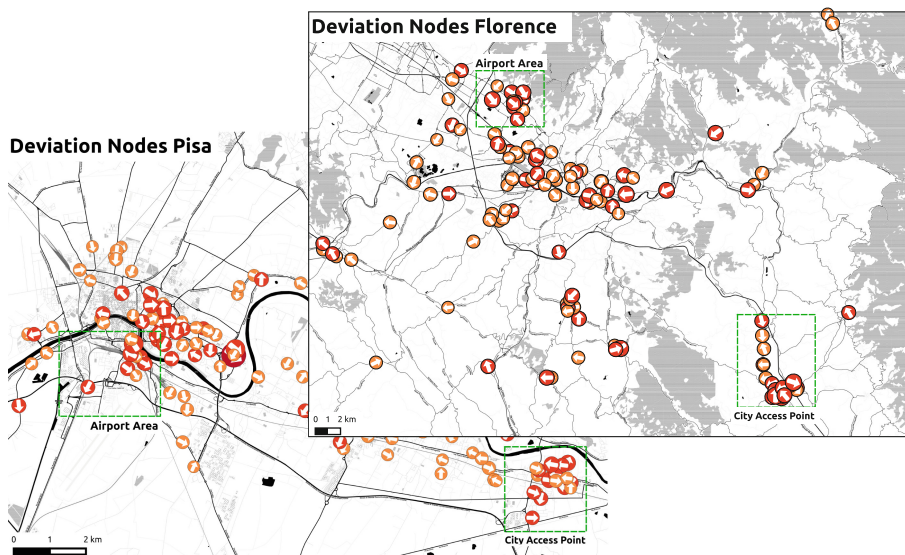


Fig. 7. Deviation nodes supported by with at least 100 deviations.

The conclusion is that systematic drivers generally deviate from the routes suggested by a route planner at the very beginning of their movements, and that in general they do not optimize their travel time but try to minimize the travel distance. However, even the drivers deviate from the short/fast routes, these routes are in many cases very similar to the routines systematically followed.

#### 4.4 Towards a Boosted Route Planner

Before presenting the analysis of this section we remark that routines are movements repeated many times (on average 15 times) during the observation period. Thus, if drivers systematically deviate from what is supposed to be the shortest

(or the fastest) path there should be a valid reason. Given a user moving for the first time in a certain area, it could be better for him/her to follow the routines described by “expert driver” instead of the routes suggested by a route planner.

A route planner could be boosted by exploiting the knowledge given by the individual mobility models. Such a route planner should consider various information: *(i)* the road intersections where the systematic drivers deviate more, *(ii)* the areas where those intersections are concentrated, and *(iii)* the main flows of movement containing deviations. In the following we analyze these three factors to understand their impact and which are their possible uses. Due to lack of space in the following we focus the analysis only on the deviation of the routines against the shortest path.

We refer to the road intersections as *deviation nodes*. They correspond to the first nodes in the set of road segments  $S$  from which the routines in  $P_u^*$  deviate from the route in  $\bar{P}_u^s$ . To count the number of deviations, instead of considering only the number of routines, we weighted each routine  $r_i^* \in P_u^*$  with the number of trajectories that support it. In Fig. 7 we can observe the deviation nodes in which there are at least 100 trajectories which deviate. The darker and the bigger is a marker, the higher is the number of deviations performed by the routines on that node. As expected, for both cities, the highest numbers of deviation nodes appear into the city center. This confirms the fact that in the city is very difficult to follow the shortest paths. Moreover, in both cities we can observe some particular areas not in the city center (those highlighted in the green dotted squares) with an high number of deviations. They correspond in both cases *(i)* to the main access points to/from the city center, and *(ii)* to the roads close to the airports. This is a signal that these areas are probably affected by consistent traffic and the systematic users which have to pass through them prefer longer but less stressful routes.

To analyze the deviations’ areas we divided the territory using a grid with cells of 2.5 km of radius. The heatmap of the deviations is shown in Fig. 8. The darker is a cell, the higher is the number of trajectories which support the routines deviating there. For these images no filters are applied. The first insight is that the users acting in province of Florence have an active role even in the mobility of Pisa but the viceversa is not true. Indeed, most of the cells with more deviation in Pisa occur also in the Florence heatmap. From the intersection of the two images emerges that most of the systematic deviations take place along the main road between Pisa and Florence (named SGC Fi-Pi-Li) with a concentration in the area around Empoli. This probably happens because most of the people living in Empoli, which is in province of Florence, go systematically to Pisa for working. For example, instead of following SGC Fi-Pi-Li that is an highway but has a lot of traffic, many drivers could prefer as alternative the road SS67 which runs along SGC Fi-Pi-Li but has much more turns and is not an highway. In Fig. 9(left) we report the distribution of the number of cells per routines’ deviations. It is a power low distribution indicating that there are few cells where most of the systematic users decide to take alternative routes. Those are the cells that more than the others the boosted root planner should consider when suggesting the routes which exploit the wisdom of the crowd.

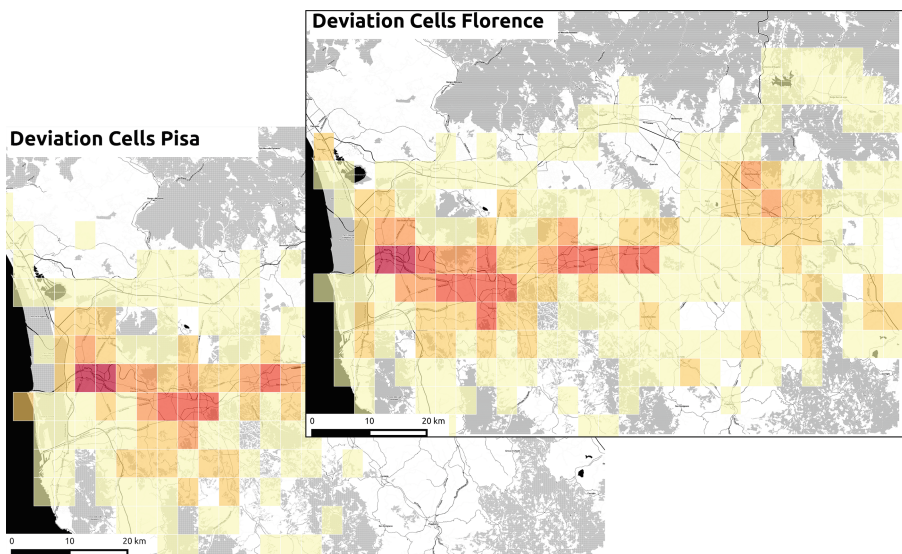


Fig. 8. Heatmap of the deviation cells.

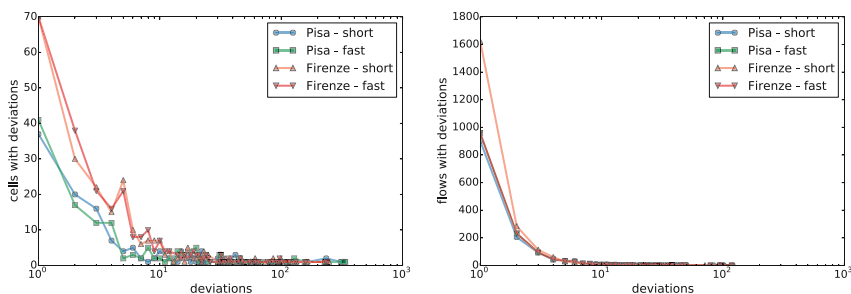
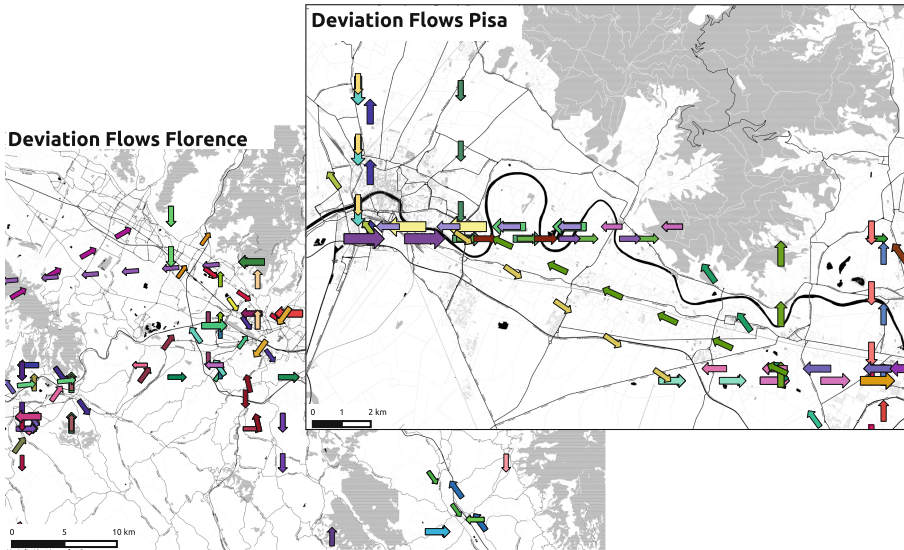


Fig. 9. Distributions of the number of cells with deviations (left), and of the number of flows with deviation (right).

We defined a flow as a triple of cells (*origin*, *deviation*, *destination*) where *origin* is the cell origin of the routine, *deviation* is the cell where  $r_i^*$  deviates from  $\bar{r}_i$ , and *destination* is the ending cell of the routine. In Fig. 10 we can observe the flows containing the routines supported by at least 100 trajectories. Through this approach we can observe the main flows along with most of the drivers deviate from the shortest paths. We can observe how in Pisa province there are various flows of entrance to and exit from the city center. The flow with more deviations (the purple biggest arrows) are just under the city center starting from the airport area up to the suburbs. They are surrounded by a large number of in-coming and out-coming flows. We remark that in many cases the deviation from the shortest path appears at the very beginning of the movement. Thus the flows reported mainly highlight the part of the movement after the deviation.

Some deviation flows do not have a mutual reverse flow of the same importance. For these cases the deviation is more evident only in one direction. On the other hand, in province of Florence, the flows in the city center are on average shorter than those outside. In addition, the biggest flows are present in the airport area (big green arrow in the center) and close to the exit of the highways (big blue arrow bottom right and big aqua green arrow in the center). Figure 9(right) shows the distribution of the number of flows per routines' deviations. Similarly to the cells, the distribution is long tailed indicating a small set of flows where many routines deviates from the shortest/fastest path. A route planner having this kind of knowledge should recommend paths which run along these flows and are similar to the individual routines. Indeed, by applying appropriate weights on the road network segments in  $S$  the route planner could provide solutions boosted by the routes systematically followed by expert drivers.



**Fig. 10.** Deviation flows supported by with at least 100 deviations.

Finally, we analyzed the difference between the flows described above and the flows built using only origins and destinations. In other words given a origin-destination flow ( $origin, destination$ ) how many flows ( $origin, deviation, destination$ ) pass through the same  $deviation$ ? We name this indicator *flow similarity in deviation*. This value give us a hint of how much a certain deviation is stable along a flow. A flow similarity in deviation of X% indicates the percentage of ( $origin, deviation, destination$ ) flow on the number of origin-destination flows ( $origin, destination$ ) which pass through the same  $deviation$  cell. E.g. given the following origin-destination flows  $\{A \rightarrow B, X \rightarrow Y\}$  and the flows  $\{A \rightarrow C \rightarrow B, A \rightarrow C \rightarrow B, A \rightarrow D \rightarrow B, X \rightarrow Z \rightarrow Y, X \rightarrow Z \rightarrow Y\}$ , then

the percentage of flow difference is 80%. In our dataset of Pisa and Florence we obtained the following results: *Pisa*: 83% (short), 78% (fast), *Florence*: 87% (short), 85% (fast). These high percentages are a clear signal that the deviation along the various flows are not a matter of individuals, but that are known and subscribed from the majority of the drivers. It is a sort of “common sense” which surprisingly emerges at collective level even though all the mobility models used in the proposed analysis are individual.

## 5 Related Work

Route planners are designed to provide information about the possible journeys in a certain area. Generally route planners refer to means of transportation which are either private or public. However, the application prompts a user to input an origin and a destination and it recommends some routes which are considered to be the best for that query.

Route planners generally use some smart variations of well known shortest path algorithms to search a graph of nodes (modeling access points to the network) and edges (modeling links between nodes) [8]. Different cost weights such as distance, cost etc. can be associated with edges and nodes. However, it is generally quite difficult to plan high quality routes [11]: (i) the notion of “route quality” is different from person to person, and (ii) available route networks rarely contain all the information needed for proposing the best route (e.g. traffic information, road quality etc.). Thus, even though the search can be optimized w.r.t. different criteria, e.g. the shortest, the fastest, the cheapest [13] and even the happiest ones [14], there is not guarantee that the route provided will be considered “the best” by the majority of the users.

Various effort in different directions have been made to improve route planning applications. In particular, personalized route services able to deal with individual users preferences have been investigated recently. For example in [12] complex users preferences were modeled into a route planner by means of the fuzzy set theory. In [9] the authors provided improved individual route plans for Dublin inhabitants by exploiting both historical data and estimated traffic flows. Still according to an estimation of future travels obtained by mining public transport data, in [7] were recommended personalized tickets for London public transport network. Another framework for personalized trip recommendations considering user preferences and temporal properties was proposed in [10]. In [16] were introduced real-time information coming from GPS-equipped taxi together with historical data for an improved route planner which uses traffic conditions and driver behavior for selecting the best path. Finally, a multi-modal journey planner can consider at the same time various means of transport and minimize the uncertainty of catching a certain means [3], or it can provide for the same journey personalized public and private transportation solutions [2].

## 6 Conclusion

In this work we analyzed the deviation of the systematic movements from the shortest and fastest paths suggested by a route planner on a set of drivers in Pisa and Florence provinces. We found that systematic drivers deviate from the routes suggested by a route planner at the very beginning of their movements, and that they generally try to minimize the travel distance more than the travel time. Moreover, we observed that the shortest paths are in many cases very similar to the systematic movements from which they deviate. Through our analytic model we were able to select the areas and the flows with the highest number of systematic deviation. We discovered that given a flow from an origin  $o$  to a destination  $d$  nearly all the users which systematically move from  $o$  to  $d$  deviate in the same area. Our analysis shows that, for some unknown reasons, the traveled systematic movements give to the drivers a feeling that their route is better than the shortest or fastest paths suggested by a route planner. This kind of knowledge can be exploited by a route planner which can weight the cost on the edges with the number of supported trajectories instead of with the length or with travel time. Following this approach, a user which travels for the first time in a certain area could be helped in selecting the route by the wisdom of the drivers which systematically pass there. Also a city manager could gain worth information from our analysis. Indeed, he/she could favor the cars circulation along the routes followed by systematic drivers and improve the others which are in fact not exploited enough.

**Acknowledgements.** This work has been partially supported by the European Commission under the SMARTCITIES Project n. FP7-ICT-609042, PETRA. We thank Roberto Trasarti and Mirco Nanni for the help that lead to the creation of this paper.

## References

1. Ankerst, M., Breunig, M.M., Kriegel, H.-P., Sander, J.: Optics: ordering points to identify the clustering structure. In: ACM SIGMOD, vol. 28. ACM (1999)
2. Botea, A., Braghin, S., Lopes, N., Guidotti, R., Pratesi, F.: Managing travels with petra: the rome use case. In: ICDE. IEEE (2015)
3. Botea, A., Nikolova, E., Berlingerio, M.: Multi-modal journey planning in the presence of uncertainty. In: ICAPS (2013)
4. Cintia, P., Nanni, M.: An effective time-aware map matching process for low sampling GPS data. Technical report cnr.isti/2015-TR-011
5. Cintia, P., Trasarti, R., Cruz, L., Costa, C., de Macedo, J.A.F.: A gravity model for speed estimation over road network. In: 2013 IEEE 14th International Conference on Mobile Data Management (MDM), vol. 2, pp. 136–141. IEEE (2013)
6. Giannotti, F., Nanni, M., Pedreschi, D., Pinelli, F., Renso, C., Rinzivillo, S., Trasarti, R.: Unveiling the complexity of human mobility by querying and mining massive trajectory data. VLDB J. **20**(5), 695–719 (2011)
7. Lathia, N., Capra, L.: Mining mobility data to minimise travellers' spending on public transport. In: Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1181–1189. ACM (2011)



8. Letchner, J., Krumm, J., Horvitz, E.: Trip router with individualized preferences (trip): incorporating personalization into route planning. In: Proceedings of the National Conference on Artificial Intelligence, vol. 21, p. 1795. AAAI Press/MIT Press, Menlo Park/Cambridge/London (1999, 2006)
9. Liebig, T., Piatkowski, N., Bockermann, C., Morik, K.: Predictive trip planning-smart routing in smart cities. In: EDBT/ICDT Workshops, pp. 331–338 (2014)
10. Lu, E.H.-C., Chen, C.-Y., Tseng, V.S.: Personalized trip recommendation with multiple constraints by mining user check-in behaviors. In: ICAGIS, pp. 209–218. ACM (2012)
11. McGinty, L., Smyth, B.: TURAS: a personalised route planning system. In: Mizoguchi, R., Slaney, J.K. (eds.) PRICAI 2000. LNCS, vol. 1886, p. 791. Springer, Heidelberg (2000)
12. Mokhtari, A., Pivert, O., Hadjali, A.: Integrating complex user preferences into a route planner: a fuzzy-set-based approach (2009)
13. Pelletier, M.-P., Trépanier, M., Morency, C.: Smart card data use in public transit: a literature review. *Transp. Res. Part C Emerg. Technol.* **19**(4), 557–568 (2011)
14. Quercia, D., Schifanella, R., Aiello, L.M.: The shortest path to happiness: recommending beautiful, quiet, and happy routes in the city. In: Conference on Hypertext and Social Media, pp. 116–125. ACM (2014)
15. Trasarti, R., Pinelli, F., Nanni, M., Giannotti, F.: Mining mobility user profiles for car pooling. In: KDD. ACM (2011)
16. Yuan, J., Zheng, Y., Xie, X., Sun, G.: Driving with knowledge from the physical world. In: KDD, pp. 316–324. ACM (2011)

# Design of a Business-to-Government Information Sharing Architecture Using Business Rules

Sélinde van Engelenburg<sup>(✉)</sup>, Marijn Janssen, and Bram Klievink

Faculty of Technology, Policy and Management,  
Delft University of Technology, Delft, The Netherlands  
{S.H.vanEngelenburg, M.F.W.H.A.Janssen,  
A.J.Klievink}@tudelft.nl

**Abstract.** Information sharing between businesses and government agencies is of vital importance, yet business are often reluctant to share information, e.g. as it might be misused. Taking this into account is however often overlooked in the design of software architectures. In this research we apply a design science approach to develop an software architecture that is acceptable by businesses. From a case study we derive the requirements an architecture should meet in order to contribute to increasing willingness to share information. In this paper the architecture is developed and evaluated according to the requirements. We recommend the use of different types of business rules that provide businesses with control over their data, in combination with encryption and decryption of data to provide access to parts of the data within an organization.

**Keywords:** Software-architecture · Information sharing · Business rules · Encryption · Decryption · Supply chain · Customs

## 1 Introduction

Easy and seamless information sharing can have advantages for both businesses and government agencies [1–3]. There are however some factors, such as the competitive advantage of having information that other parties do not have, that might make businesses unwilling to share their information [2]. Some research has been conducted on building trust between stakeholders and on governance and identifying gains of using architectures facilitating information sharing, in order to support the adoption of such architectures [3–5]. In this paper, the main premise is that in order for businesses to be willing to share information, businesses and government agencies first will want to make sure that the sharing and use of information does not harm their interests and that it complies with legislation [2, 6]. An architecture could very well help to ensure businesses of this, but currently, there is no knowledge about what such an architecture should look like.

In this research, we take a design science approach to develop a software architecture facilitating information sharing between businesses and government agencies. This architecture should have properties such that businesses are more willing to share information when they use the architecture, than when they share information directly.

Usually the focus of research is on finding explanations or predictions for e.g. human or organizational behaviour. However, in design science ways of expanding the boundaries of human and organizational capabilities are looked for by creating new and innovative artefacts [7]. Central is the development and evaluation of IT-artefacts with the intention to solve organizational problems [7]. The evaluation of IT-artefacts provides feedback information that is used to improve the quality of the product [7]. The IT-artefact developed in this research is a software architecture. The organizational problem we intend to solve is that of the unwillingness of businesses to share information in cases when it could be advantageous for the them or for the government agencies. The organizational problem was studied based on literature and this resulted in requirements for the architecture. The architecture was then developed and evaluated based on these requirements. For the kernel theories of our research, we want to refer to the related research in Sect. 3.

The research in this paper is related to research from many different domains, e.g., knowledge management and research on confidentiality and trust. Some of the innovative nature of this research results from not limiting its scope to a single domain or point of view. Concerning the organizational problem, the scope is limited to the case of information sharing between companies in an international supply chain and customs. This case is appropriate for our research due to its complexity and the involvement of various types of actors, amongst others. For the architecture, the scope is limited to the flow of data and its structure. The contents and form of data and the components are outside of the scope of this paper. These would be highly complex and therefore it is important to first determine whether the structure of the architecture will result in meeting the requirements. The evaluation of the architecture is restricted to the requirements we determined. Of course there may be other requirements that are of importance as well, but they are subject to further research.

In Sect. 2 of this paper, we sketch the case of information sharing between businesses in a supply chain and customs. In addition, the requirements will be elicited from the point of view of both businesses and government agencies. In the following section, we will start by discussing related research. Subsequently, we present the aforementioned software-architecture. In Sect. 5 we make the connection between the requirements and the architecture and we evaluate the architecture.

## 2 Requirements to Share Information

### 2.1 Companies in a Supply Chain and Customs

A supply chain can be described as complex network, which consists of many different stakeholders, including shippers, deep-sea carriers, port operators, and customs organizations [8]. Mentzer et al. [9] argue that supply chains can be defined in various ways. According to Tsay et al. [10] modern usage of the term supply chain is consistent with the following: “*a supply chain is two or more parties linked by a flow of goods, information and funds*” [10]. In concurrence with this description, with companies in a supply chain, we refer to the companies that are linked by a flow of goods, information and funds. However, our main focus will be on the companies that have a role in

transporting containerized goods, such as shippers, carriers and freight forwarders. These companies are especially interesting for our case, since they deal more directly with customs and the use of containers increases the need for information sharing. We will elaborate on this below. The government agency in this case is customs. The role of customs is that of a gate-keeper, excises-, duties- and tax-collector and they are responsible for monitoring the flow of goods and interfere with it if there are safety, security and other public policy reasons to do so [6].

We selected this case because of its complexity, amongst others, which is due to the inclusion of various types of actors, both from government and business. These parties have different roles and interests. Adding to the complexity is the international character, due to which visibility on the whole chain is limited and various legislations may play a role.

The limited visibility presents a particular problem, as in containerized transport it is not feasible to open each container to look what is inside [11]. This poses a problem for customs who try to monitor the flow of goods and try to determine whether companies comply with regulation [12]. To companies it is often also very important to know what is in containers. Just like customs, companies want to reduce security and safety risks. For instance carriers want to know the weight of goods in containers so that they can make a stowage plan that ensures the stability of the ship. Better data may also help actors in optimizing supply chains [13].

Good quality information on what is inside containers is therefore very important to the companies in the supply chain as well as to customs. The information that customs and companies need often is available in other places in the supply chain. For instance the manufacturer of the goods that are transported has a lot of details on them, such as their weight. The shipper who packed the box has information on the contents of containers [12]. However, much of the information that companies have and that could benefit customs is not provided to customs [13]. Companies often only have access to information that is altered, inaccurate and vague as well [4, 12, 14].

Governments seek to reduce the administrative burden to attract economic activity to their country. Consequently, the sharing of additional information is often voluntary, which requires that businesses are willing to do so. This is of course very important for our research, since it focusses on the willingness to share information. As this complexity and these tensions have been studied before, there is material available for use in this study.

## 2.2 Requirements

A way to provide both companies in a supply chain and customs with high quality information, is by making it possible and easier for them to share information that already is available. Research by Fawcett et al. [2] suggests that willingness is key to information sharing in supply chains, but is often overlooked and misunderstood. An important factor influencing the willingness to share information is the need to keep information confidential [15]. For competitive (e.g. fear of being bypassed in the chain) or security (e.g. confidentiality on high value goods) reasons, companies may be hesitant to share information with others [2, 3].

There may also be a challenge in the willingness to receive additional information. For example, according to international rules, the description of goods carriers receive from the shipper influences their liability in case of damage or loss. If a shipper does not provide the carrier with information on the value of goods and a full description, then the liability of the carriers is limited to a certain amount per package. This has as a result that carriers may not even want to have this information. [12] Therefore they cannot provide this information to customs or other companies in the supply chain that they are in direct contact with, who might benefit from this information.

The last factor influencing the willingness to share information is the confidence that the sharing of information or its use is in compliance with legislation. The legal status of information gathering and sharing is often unclear, since different legal considerations may play a role [6]. This is a barrier for companies to share information with customs. Clarity is not improved by the fact that with whom data can be shared legally, depends on the country in which the goods are moving in [16] and that different sources of law, such as national and European law, might be applicable at the same time. Moreover, legislation may change frequently [17]. Uncertainty about the legal status of information, as well as the legality of the methods for obtaining it, may lead for instance carriers to shielding their data from other parties [6].

From analysing the literature on information sharing between companies in a supply chain and customs discussed above, we can abstract three requirements that influence the willingness to share information, namely:

- Keeping information confidential when needed.
- Ensuring there is no obstruction for information sharing from the possible increase of liability when businesses receive information.
- Ensuring the sharing of information and its use is in compliance with legislation.

### 3 Related Background

There exists a vast amount of research related to the research in this paper. This makes it impossible to discuss the related research from all different domains in its entirety. We do want to discuss some different kinds of architectures that are related to the architecture in this paper. For a more comprehensive overview of related research we refer to the work of Sahin and Robinson [18] and Yang and Maxwell [19].

Bharosa et al. [1] present two different software architectures for information sharing. The first is called Standard Business Reporting in which a standardized data representation format and semantics are used by businesses to file official reports. It incorporates a government gateway that is used to move messages from businesses to the appropriate government agency and return a receipt. The other architecture they discuss is a Continuous Control Monitoring architecture. This architecture incorporates an intermediary platform that the business uses to push key performance data to, which are then monitored by the government agencies. While in both cases in this study monitoring compliance by government agencies and limiting administrative burden do play a role, the architectures themselves are very different, which has likely to do with

the fact that they are applied in different domains, namely mainly in the financial domain and in a meat supply chain.

In the domain of our case of information sharing a notable concept that has been proposed is that of a data pipeline. The purpose of the data pipeline is to capture data at the source and to improve the coordination of border management and reduce administrative burden for businesses [12, 13, 20]. This architecture differs in structure and flow of data from the architectures described above as well.

The description of the different possible architectures above shows that architectures for facilitating information sharing can be quite different from each other. This implies that the architecture we develop should be very flexible in order to be of use in different circumstances. Our solution to this is to make the architecture such that it can be incorporated in architectures in which the flow of data itself is central (such as those described above) in order to increase the willingness to share information.

Our research is related to some research on the use of business rules to capture legal knowledge, which is relevant as in our architecture business rules are used to capture legal knowledge as well. Gong and Janssen [17] propose a framework that can be used to automatically derive business processes from such business rules.

## 4 Towards a Software Architecture

The Software Engineering Standards Committee [21] defines an architecture as “*The fundamental organization of a system embodied in its components, their relationships to each other, and to the environment, and the principles guiding its design and evolution*”. The architecture we developed, consists of a decision component and a component that allows access to data according to the decision. Metadata, business rules, global rules and context information on the requester of access to data is used as input for the decision component to reach a decision. Access to data is prevented or granted by respectively encrypting parts of data and decrypting parts of data using decryption keys.

In this section we will describe the architecture we developed. Section 5 contains an overview of the requirements and the solutions in the architecture.

### 4.1 The Decision Component and Providing of Decryption Keys

The decision component can be used by businesses or government agencies that have received encrypted data to request a key to decrypt the parts of the data that they are allowed to see according to the decision. A decision in this case should not be viewed as a simple yes or no, but contains a specification on which parts of the data the requester is allowed to access and which not. Based on the decision of the decision component, a key is provided to the requester which they can use to decrypt the parts of the data they are allowed access to according to the decision.

The decision of the decision component is based on business rules that are provided by the owners and by the senders of the data. When a decision is requested, these business rules are requested by the decision component from the owners and senders of

the data. In addition to the business rules, the decisions are based on global rules as well. The global rules play a role in all decisions and are element of the decision component itself. Furthermore, the requesters of access provide context information on themselves and their intention to use the data as input to the decision component. They also provide metadata on the data they want to receive a decryption key for.

After the decision component has received all required rules, metadata and context information and has reached a decision using these, there are two possibilities. The first is that according to the decision the requester is not allowed access to any part of the data, in which case the requester is informed of this. The second possibility is that according to the decision the requester is allowed access to parts of the data. In that case the decision and other needed information is send to a component that generates a decryption key that can be used only by the requester of the information to decrypt exactly the parts of the information they are allowed access to according to the decision. This key is then send to the business or government agency that has requested access and they can use this key to get access to the appropriate parts of the data.

## 4.2 Business Rules

In order to make a decision on whether a business or government agency can have access to parts of data, the decision component needs the business rules that are specified by the owners and previous senders of the data. The decision component obtains these business rules by requesting them from these owners and senders. In order for the decision component to do this, the metadata that the requesters send the decision component with their request, should contain information for each part of the data on who the owners and previous senders are. If the business rules are requested each time a decision has to be made, it is possible for owners and senders of data to change their business rules and thereby influencing all decisions subsequent to this adaptation.

A business rule can be defined in various ways. Graham [22] defines business rules with an emphasis on the form and expressive power as follows: “*A business rule is a compact, atomic, well-formed, declarative statement about an aspect of a business that can be expressed in terms that can be directly related to the business and its collaborators, using simple unambiguous language that is accessible to all interested parties: business owner, business analyst, technical architect, customer, and so on. This simple language may include domain-specific jargon.*”. However, since the focus of this paper is more on the function of the business rules in the architecture, the definition of Ross [23] fits our purposes better. He defines business rules as a directive intended to influence or guide business process behaviour.

Business rules in the described architecture can be used by owners and senders of information to specify who does and does not have access to which parts of the data they own or send and in what cases. Since they are used by the decision component to reach a decision, they can control access to their data by specifying these rules.

Business rules can be general, e.g. specifying that all information can be used by anyone if they have a certain goal. They could also be very specific and e.g. specify that only a certain company gets access to a specific part of the data. To illustrate some

ways in which business rules could be used, we provided some examples. It is important to mention that the use of first order logic (FOL) in the example is not meant as a recommendation to use FOL to express the rules in the architecture since their form is outside of the scope of this paper. FOL was chosen to make the example intelligible to most readers. In the example, symbols have their usual meaning and arguments that are capitalized denote variables.

$$\begin{aligned} &HasRole(X, customer) \wedge Has\_Role(Z, manufacturer) \wedge \\ &InformationOn(Y, Z) \rightarrow \neg Access(X, Y) \end{aligned} \quad (1)$$

$$Goal(X, Y, security\_check) \rightarrow Acces(X, Y) \quad (2)$$

Example (1) shows a business rule that might be specified by a seller who does not want customers to access information on the manufacturer of goods. In example (2), there is a business rule that could be used to express that if the goal of the requester of access to part of the data is to perform a security check, access to that part of the data is allowed.

Global businesses rules are basic rules that the decision component includes for each decision and that are not organization specific. They could be used to incorporate some general common sense in the decision process and to make sure that access to data is allowed only in accordance with legislation. We provided some examples below. It is important to note that they are not meant as a proposal for a specific design as well. The global rules are expressed in FOL, symbols have their usual meaning and arguments that are capitalized denote variables.

$$GoalGathered(X, Y) \wedge GoalUse(Z, X, Q) \wedge Y \neq Q \rightarrow \neg Access(Z, X) \quad (3)$$

$$SecurityStatus(emergency) \wedge HasRole(X, customs) \rightarrow Access(X, all) \quad (4)$$

The global rule in (3) expresses that access to data is not allowed for a requester if their goal for using the data is different from the goal for which it was gathered. In (4) a global rule is expressed that is used to grant customs access to all information in case of an emergency situation.

Since global rules are the same for all data, they all could be saved in the same location and be retrieved when needed in the decision process. If they are changed, e.g. because of changes in regulation, this change influences all subsequent decisions.

### 4.3 Metadata and Context Information

In order to make decisions based on the rules, metadata and context information are needed to determine which rules are applicable. Ma [24] provides a comparison of twenty-seven definitions of metadata. Zuiderwijk et al. [25] defines metadata as “structured, encoded data that describes characteristics of information bearing entities to aid in the identification, discovery, assessment, and management of the described entities”. Often, metadata is simply defined as “data about data” [26, 27]. Examples of



metadata that could be sent with the encrypted data, is information on the owners and previous senders of the encrypted data or the goals for which parts of the data initially was gathered. It also could be important to incorporate for instance information on the way in which different parts of data are linked.

The metadata should be sent with the encrypted data itself. There are several reasons for this. The receivers of encrypted data probably would like to receive at least some basic information on the data that they have received and it is easiest to send this together with the data itself. Furthermore, information on the owners and previous senders of the data is metadata as well and is needed for the decision component in order to determine from who business rules should be requested. Of course there might be cases in which it is not desirable for receivers of information to have access to all metadata. In that case, parts of the metadata could be encrypted. The decision component should receive the metadata, together with the request for access to data itself from the business or government agency that is the requester.

The last input that is needed for a decision by the decision component is context information on the requester of access to the data and their intent to use the data. Such context information of course is available with the requesters of access to data themselves. Businesses and government agencies making a request should send this context information together with their request to the decision component. Some sort of authentication could be sent as part of the context information as well.

#### **4.4 Regulating Access via Encryption and Decryption of Parts of Data**

In many cases, access to data is regulated by sending or not sending data to others or by allowing or not allowing others access to a database. This differs for the architecture we developed. Namely, in this architecture, encryption and decryption of parts of data is used to regulate access to data. This means that it is possible for businesses to send encrypted data to each other and to government agencies directly without thereby automatically granting them access to all the data they are sending. As a result, the access to data and the location where the data is saved are not linked. In other words, physical access to information does not imply logical access in this case.

Because of the use of encryption and decryption to regulate access to data, the flow of encrypted data itself can be very flexible and can be adapted to specific needs and circumstances. There is no obvious obstruction to using any kind of flow of information between businesses and government agencies. It is for instance possible that there is a direct data flow between the users of the architecture. In that case, businesses and government agencies could announce that they have data and send it upon request or they can take the initiative to send data when they think it is necessary. Another possible example is using a physical shared data space.

The fact that access and location are no longer linked, means that it is possible for businesses and government agencies to share the information that they have received with others as well, without automatically granting them access. This allows for the possibility for organizations to enrich data or combine the data that they have received in useful ways and to share it with others. In order for this to work, the enriched data or combined data should be encrypted as well before sending and the rules of the owners

of the original data, previous senders and the new sender should be applicable on the new data that is based on them. An example of the enrichment of data could be if a company added the weight of containers to the data, based on received information on the weight of goods and their own information on in which containers goods are. If they would send the enriched data, the business rules of the owner of the information on the weight of the goods, as well as the previous senders of this information would be applicable, in addition to of course the business rules of the company itself.

## 5 Evaluation of the Architecture

### 5.1 An Illustration

To illustrate the way in which the architecture we developed works and how the components and users relate to each other, we have provided an example related to our case study. Figure 1 shows the flow of information in the architecture, while Fig. 2 is an UML sequence diagram. In the figures, there are two businesses and a government agency. Other ways of sharing information than shown in the example are possible, but sending information directly is the most simple and thus the clearest way to illustrate the structure of our architecture. The flow of the encrypted data follows the physical flow of goods in the supply chain. Business 1 sends encrypted information to business 2. Business 2 enriches the information with their own and sends the enriched information to the government agency. Of course, usually, business 2 would in this case request access to the information as well. This request is left out of the diagrams to guard their intelligibility, as is the encryption of the data and generation of rules and such. After the government agency has received the encrypted data, it uses the decision component to obtain a decision. Then a key is generated and shared with customs based on the decision. Note that the decision process pulls the business rules from the businesses and that this happens after the data is requested.

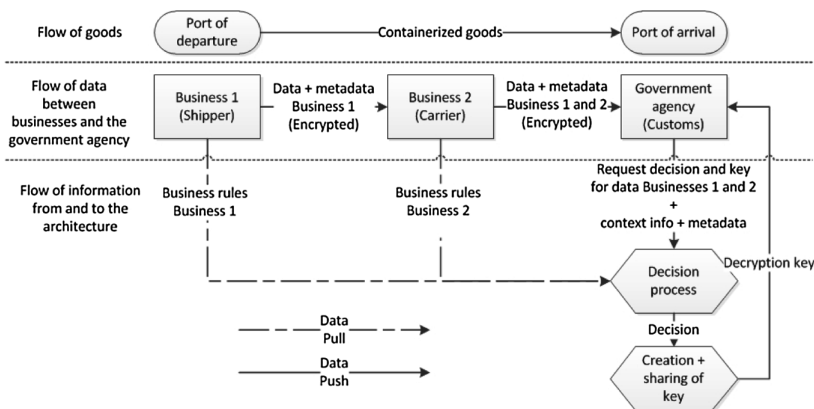


Fig. 1. The flow of information in an example for the described architecture

In Fig. 2 the procedures that are executed by the architecture and its users can be observed as well as what happens in case access to data is not allowed according to the decision process. To guard the intelligibility of the diagram, the components of the architecture itself are not shown separately as is done in Fig. 1.

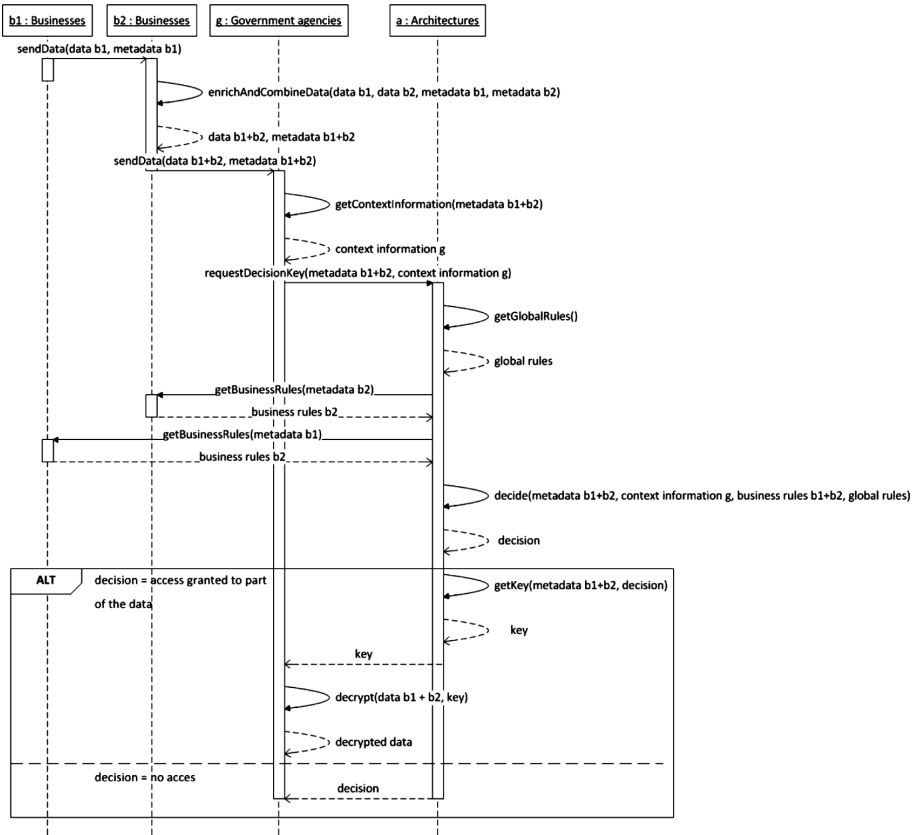


Fig. 2. UML sequence diagram for the example for the described architecture

Next we will evaluate the architecture presented in the previous subsection by analysing the way the requirements are met by the architecture. In addition, we discuss some other advantages and disadvantages of the architecture described.

### 5.2 Keeping Information Confidential When Needed

In the architecture businesses and government agencies can control who does and does not have access to their information by specifying business rules. The responsibility to specify these rules so that their interests are not hurt, lie with the parties themselves. If they think that certain information for instance will be of competitive advantage, they

can specify business rules which deny their competitors access to this information. Since each time a decision is made by the decision component, the appropriate business rules are pulled from the systems of the owners and senders of data, businesses can modify, add or remove them if this is needed and this will result in changes on who gets access to information. Businesses are thus provided with a means to control their data and to make sure it is kept confidential when needed.

At first sight, the level of control businesses can exert over access to their data in the architecture is not much unlike other situations in which businesses or government agencies for instance choose to send data that is not encrypted to some parties and not to others. There are however some important differences. The first is that in the described architecture, the rules for who gets access to what parts of data and who does not are explicit and applied in a consistent manner. Making a refined specification, might make businesses and government agencies realize more clearly what parts of their data actually needs to be kept confidential and for whom. A consequence of this is a more refined distinction between data that other parties can and cannot have access to, which in turn results in increased willingness to share data. A problem with the fact that businesses and government agencies have to specify business rules is that it could be a lot of work and they might not want to make such an investment if it is not clear how they could benefit from it.

The second difference is that parties sending received data to others, do not need to be afraid of breaking confidentiality or damaging trust between parties, if the data that they send is encrypted. When access to data and the location where data is saved are no longer linked, sending data to others does not automatically mean granting them access. Even if data is received from other sources, access to data is still controlled by the owners of data and the previous senders because their business rules stay applicable. This makes it easier for parties to send data that they have received and enriched or combined with their own data, since it no longer has the consequence of breaking confidentiality. As access to data is no longer needed to make sure that resending it does not break confidentiality, businesses that cannot access data themselves can resend it easier as well. For these reasons, it is expected that willingness to share information will increase.

To the original owners of information, it might be reassuring to know that wherever their data is, their business rules are applicable and access is only granted following these rules. Furthermore, they might be reassured by the fact that these rules are as well applicable on enriched or combined data that is based on their data, eliminating the risk that their data can be abstracted from this by parties from who the data should be kept confidential. This might increase their willingness to share their information. Of course there is still a possibility that other parties might send decrypted information to others and breaking confidentiality in that way. The risks of this are not higher than parties sharing data illicitly when businesses or government agencies granted them access to data directly themselves. In the case of the described architecture, there clearly is the option of sending information encrypted and thereby respecting the rules that are specified by the owners and senders of the data. Therefore, when information is send decrypted, the attempt to circumnavigate the control of the owners and senders of information on their data is much more clear.

### **5.3 Ensuring There is No Obstruction for Information Sharing from the Possible Increase of Liability When Businesses Receive Information**

For the second requirement, it is not completely clear how liability is influenced by the use of the architecture. When businesses or government agencies receive certain information, it could increase their liability. In the case of the architecture, they could receive and store such information, but not have access to it, or even know that it exists and that they have it stored. While it is not completely clear from a legal point of view whether they are still liable, we could look at what is reasonable. One could say in this case that in a sense they did not receive or do not possess the information that is in the data. It is reasonable that in that case the receiving and storage of the encrypted data should not make them liable. In consequence, the willingness to store this information and subsequently share it with others would probably improve. A decision by the decision component could serve as proof that data indeed cannot be accessed if needed. There of course is a difference between cases in which encrypted data cannot be accessed and cases in which the business or government agency chooses not to access the data. In the second case, the position that receiving and storing this data should not increase liability is harder to defend.

### **5.4 Ensuring the Sharing of Information and Its Use is in Compliance with Legislation**

The last requirement has to do with the confidence that the sharing of information and its use is in compliance with legislation. The global rules that the decision component uses, could be used to make sure that access is only granted to parts of data if this complies with legislation. These rules could be adapted in case legislation changes, without users having to keep track of such changes in legislation and their influence on whether they can or cannot share information themselves. At the moment, it is unclear whether the same legislation that is applicable to data that is directly accessible is applicable to data that is encrypted and cannot directly be accessed. However, for the same reasons as in the situation with the increase of liability, it might be reasonable to say that this should not be the case. If encrypted data could be shared freely and access is only granted to parts of data when this is in compliance with legislation, it is reasonable for users of the architecture to be confident that they comply with legislation if they only send encrypted information and only access information according to the decisions of the decision component. This in turn might lead to an increase in their willingness to share information.

While at first sight this would be an ideal situation for the users of the architecture, there might be some problems as well. The responsibility to be compliant in a sense shifts from the users of the architecture to the organization that specifies the global rules, resulting in some ethical and legal difficulties. Furthermore, the organization specifying the global rules, has a lot of power, since they have an influence on all requests for access and this may not be desirable. A solution could be to let the decision

component be governed by the users themselves, solving some of the ethical difficulties with responsibility and distribute the power between the users.

### 5.5 Other Properties of the Architecture

The described architecture has some other interesting properties worth discussing. The first is the security of the data in the architecture. If encrypted data falls in the wrong hands, someone can attempt to decrypt it themselves illegally. Since in our architecture they can have the encrypted data stored in their own systems, they could go about their business uninterrupted. The data should thus be encrypted well enough to make it not worth attempting to decrypt it illegally or so that this takes such a long time that by the time they succeed, the data has lost its worth. Hence, the quality of encryption is vital. There are some advantages of the described architecture considering security as well. If someone decrypts data illegally, they can only access that data and e.g., not a full database. Furthermore, there does not need to be a single component through which all data passes, which would have possessed its own risks.

Another property of the described architecture is that it is very flexible. There is no clear obstacle for the architecture to be part of any information sharing architecture, since the way information is sent to others is and does not need to be specified. Furthermore, it allows for the constant adaptation of business rules and global rules to new (legal) circumstances, and changes of interests and needs.

## 6 Conclusion and Suggestions for Further Research

The architecture we developed empowers business by providing them control of their information sharing. The fact that business rules can be specified by the owners and senders of information and that these are applicable even when information is not received directly from its original source or when it is combined or enriched, gives owners the control to keep their data confidential when needed. In addition, in the architecture the sharing of data that is received by others, that is enriched or combined, is especially made easier by using the combination of business rules and encryption. Furthermore, it seems that using global rules to make sure that data access complies with legislation, is an option for increasing willingness to share information.

Overall, an architecture incorporating business rules, global rules, a decision component and encrypted data has enough potential to merit further investigation. Especially since the proposed architecture is very flexible and could be combined with other architectures, combining their advantages as well. We do recommend that this is coupled with investigating the legal framework such an architecture would exist in.

The subject of control management and especially role based access and attribute based access such as described in [28, 29] seems very relevant for future research, in particular when working out what the different kinds of business rules and global rules should look like. Metadata and context information plays an important role in the architecture as well, making it an important topic for further research. Research on knowledge representation as well as existing formats and standards for metadata are

relevant for this. In order to reason with the rules and information, theories from the domain of automated reasoning and decision support are significant as well and should be taken into account. Generating and distributing keys and encryption of data is vital for the architecture, but far from trivial. Research on encryption and computer security should therefore have an important part in future research. Other ways of increasing security, e.g., by using authentication or signing of data, for this architecture should be investigated as well.

## References

1. Bharosa, N., Janssen, M., van Wijk, R., de Winne, N., van der Voort, H., Hulstijn, J., Tan, Y.-H.: Tapping into existing information flows: The transformation to compliance by design in business-to-government information exchange. *Gov. Inf. Q.* **30**, S9–S18 (2013)
2. Fawcett, S.E., Osterhaus, P., Magnan, G.M., Brau, J.C., McCarter, M.W.: Information sharing and supply chain performance: the role of connectivity and willingness. *Supply Chain Manag. Int. J.* **12**, 358–368 (2007)
3. Klievink, B., Janssen, M., Tan, Y.-H.: A stakeholder analysis of business-to-government information sharing: the governance of a public-private platform. *Int. J. Electron. Gov. Res.* **8**, 54 (2012)
4. Klievink, B., Lucassen, I.: Facilitating adoption of international information infrastructures: a living labs approach. In: Wimmer, M.A., Janssen, M., Scholl, H.J. (eds.) *EGOV 2013*. LNCS, vol. 8074, pp. 250–261. Springer, Heidelberg (2013)
5. Overbeek, S., Klievink, B., Hesketh, D., Heijmann, F., Tan, Y.-H.: A Web-based data pipeline for compliance in international trade. In: *Proceedings of the CEUR Workshop*, vol. 769, pp. 32–48 (2011)
6. Janssen, M., Smeele, F.: *JUridical and context-aware Sharing of informaTION for ensuring compliance (JUST)* (2013)
7. Hevner, A.R., March, S.T., Park, J., Ram, S.: Design science in information systems research. *MIS Q.* **28**, 75–105 (2004)
8. Van Baalen, P., Zuidwijk, R., van Nunen, J.: Port inter-organizational information systems: capabilities to service global supply chains. *Found. Trends® Technol. Inf. Oper. Manag.* **2**, 81–241 (2009)
9. Mentzer, J.T., DeWitt, W., Keebler, J.S., Min, S., Nix, N.W., Smith, C.D., Zacharia, Z.G.: Defining supply chain management. *J. Bus. Logist.* **22**, 1–25 (2001)
10. Tsay, A.A., Nahmias, S., Agrawal, N.: Modeling supply chain contracts: a review. In: Tayur, S., Ganeshan, R., Magazine, M. (eds.) *Quantitative Models for Supply Chain Management*, pp. 299–336. Springer, New York (1999)
11. Levinson, M.: The world the box made. In: *The Box: How the Shipping Container Made the World Smaller and the World Economy Bigger*. Princeton University Press, Princeton (2010)
12. Hesketh, D.: Weaknesses in the supply chain: who packed the box. *World Cust. J.* **4**, 3–20 (2010)
13. Klievink, B., van Stijn, E., Hesketh, D., Aldewereld, H., Overbeek, S., Heijmann, F., Tan, Y.-H.: Enhancing visibility in international supply chains: the data pipeline concept. *Int. J. Electron. Gov. Res.* **8**, 14–33 (2012)
14. Lee, H.L.H., Whang, S.: Information sharing in a supply chain. *Int. J. Manuf. Technol.* **1**, 79–93 (2000)

15. Urciuoli, L., Hintsä, J., Ahokas, J.: Drivers and barriers affecting usage of e-Customs — a global survey with customs administrations using multivariate analysis techniques. *Gov. Inf. Q.* **30**, 473–485 (2013)
16. van Stijn, E., Hesketh, D., Tan, Y.-H., Klievink, B., Overbeek, S., Heijmann, F., Pikart, M., Butterly, T.: Annex 3: The Data Pipeline. Connecting International Trade?: Single Windows and Supply Chains in the Next Decade, pp. 158–183. United Nations Economic Commission for Europe (2011)
17. Gong, Y., Janssen, M.: A framework for translating legal knowledge into administrative processes: dynamic adaption of business processes. In: Cerone, A., Persico, D., Fernandes, S., Garcia-Perez, A., Katsaros, P., Ahmed Shaikh, S., Stamelos, I. (eds.) SEFM 2012 Satellite Events. LNCS, vol. 7991, pp. 204–211. Springer, Heidelberg (2014)
18. Sahin, F., Robinson, E.P.: Flow coordination and information sharing in supply chains: review. *Implications Dir. Future Res.* **33**, 1–32 (2002)
19. Yang, T.-M., Maxwell, T.A.: Information-sharing in public organizations: A literature review of interpersonal, intra-organizational and inter-organizational success factors. *Gov. Inf. Q.* **28**, 164–175 (2011)
20. Stijn, E. Van, Klievink, B., Janssen, M., Tan, Y.-H.: Enhancing business and government interactions in global trade. In: Third International Engineering Systems Symposium CESUN 2012. pp. 18–20 (2012)
21. Software Engineering Standards Committee: IEEE Recommended Practice for Architectural Description of Software-Intensive Systems (2000)
22. Graham, I.: *Business Rules Management & Service Oriented Architecture*. Wiley, Chichester (2006)
23. Ross, R.G.: *Principles of the Business Rule Approach*. Addison-Wesley Longman Publishing Co., Inc., Boston (2003)
24. Ma, J.: Managing metadata for digital projects. *Libr. Collect. Acquis. Tech. Serv.* **30**, 3–17 (2006)
25. Zuiderwijk, A., Jeffery, K.G., Janssen, M.: The potential of metadata for linked open data and its value for users and publishers. *JeDEM-e-J. e-Democracy Open Gov.* **4**, 2012 (2012)
26. Jeffery, K.G.: Metadata: the future of information systems. In: Brinkkemper, J., Lindencrona, E., Sølberg, A. (eds.) *Information Systems Engineering: State of the art and research themes*. Springer, London (2000)
27. Schuurman, N., Deshpande, A., Allen, D.M.: Data integration across borders: a case study of the Abbotsford-Sumas aquifer (British Columbia/Washington State) 1. *JAWRA J. Am. Water Resour. Assoc.* **44**, 921–934 (2008)
28. Sandhu, R.R.S., Coynek, E., Coyne, E.J., Feinstein, H.L., Youman, C.E.: Role-based access control models. *Comput. (Long. Beach. Calif)* **29**, 38–47 (1996)
29. Hu, V.C., Ferraiolo, D., Kuhn, R., Schnitzer, A., Sandlin, K., Miller, R., Scarfone, K.: Guide to attribute based access control (ABAC) definition and considerations. *NIST Spec. Publ.* **800**, 162 (2014)



# Process Mining as a Modelling Tool: Beyond the Domain of Business Process Management

Antonio Cerone<sup>(✉)</sup>

IMT Institute for Advanced Studies, Lucca, Italy  
antonio.cerone@imtlucca.it

**Abstract.** Process mining emerged in the field of business process management (BPM) as an innovative technique to exploit the large amount of data recorded by information systems in the form of event logs. It allows to discover not only relations and structure in data but also control flow, and produces a process model, which can then be visualised as a process map. In addition to discovery, process mining supports conformance analysis, a technique to compare an *a priori* model with the event logs to detect deviations and inconsistencies.

In this paper we go beyond the domain of BPM and illustrate how process mining and conformance analysis can be used in a number of contexts, in and across the areas of human-computer interaction and learning.

## 1 Introduction

*Process mining* is an emerging discipline based on model-driven approaches and data mining. It is a process management technique used to extract information from event logs consisting of activities (business activities, communication activities, collaboration activities, etc.) and then produce a graphical representation of the process control flow, detect relations between components/individuals involved in the process and infer data dependencies between process activities [11].

In order to be successfully processed, event logs have to meet a number of structural properties, that is, to contain adequately organised and clustered data. This level of structural organisation can be attained by applying text mining techniques, in particular semantic indexing, that is, by assigning a meaningful subject to the data. Recently, incremental fully automatic semantic mining algorithms have been developed within the semantic platform associated with the 2012 SQL Server: they produce weighted physical indexes, which can then be queried through the SQL interface [5]. Using the semantic platform associated with the SQL Server, queries can be defined based on a catalog in which keywords, key-phrases and conditional activities are categorised in terms of states of a process. The resultant output is a pre-processed log to be fed to a process mining tool, such as DISCO (Discover Your Process) [2], to produce a process model, presented as a Petri net-like visualisation (often called process map) and its associated PNML representation, possibly together with relevant statistical

data. The capability of discovering statistical data about the analysed process makes process mining a useful tool in performance evaluation.

In the area of business process management (BPM), process mining has been used not only to discover a process model and represent it as a process map, but also to extend a pre-existing *a priori* model by enriching it with new aspects and perspectives illustrated by the discovered *a posteriori* process model, and to compare, by using a technique called *conformance analysis*, the *a priori* model with the event logs (and thus implicitly with the *a posteriori* process model).

*Conformance analysis* originated from Rozinat and van der Aalst's work in the area of BPM [7]. Conformance analysis, also called *conformance checking*, is the detection of deviations and inconsistencies between an *a priori* model, which is based on theoretical perspectives and/or data collected and analysed using social science research methods, and the traces generated by the event logs. In fact, conformance checking seems appropriate well beyond BPM. In particular, it can be applied to the analysis of social networks and peer-production systems, and the first attempts in this direction have been done in the areas of collaborative learning and Free/Libre Open Source Software (FLOSS) development [3, 4].

In this paper we go beyond the domain of BPM and illustrate how process mining and conformance analysis can be used in a number of contexts, in and across the areas of human-computer interaction (HCI) and learning. Section 2 illustrates how to apply process mining to social networks in order to extract behavioral patterns that provide evidence of learning processes (Sect. 2.1) and skill acquisition (Sect. 2.2). Section 3 concerns real-time applications of process mining. Section 4 concludes the paper.

## 2 Modelling from Observed Behavioural Patterns

One important application of cognitive psychology to HCI is the observation of human behaviour, during interaction with interfaces, devices or within online communities, to extract behavioral patterns of users or control operators. In this section we present a case study on the extraction of learning processes from behavioral patterns of FLOSS contributors and propose how to extend our approach to modelling skill acquisition not only in FLOSS communities but, generally, in interacting with a specific device/interface/application.

### 2.1 Modelling Learning Processes: The FLOSS Case Study

Social Networks can be seen as collaborative environments in which interactions among peers support the building of knowledge both at individual and community level. Learning processes occur naturally within such environments and produce evidences of their existence in the contents of communications between community members and in the digital artifacts shared or produced by the community, such as web pages, documents, audio and video clips, software, etc.

FLOSS communities also present this learning potential. They are open participatory ecosystems in which actors not only create source code but also produce and organise a large variety of resources that include implicit and explicit knowledge, communication logs, documentation and tools. Collaboration in FLOSS projects is highly mediated by the usage of tools, such as versioning systems, mailing lists, reporting systems, etc. These tools serve as repositories which can be data mined to understand the identities of the individuals involved in a communication, the topics of their communication, the amount of information exchanged in each direction, as well as the amount of their contribution in terms of code commits, bug fixing, produced reports and documentation, sent emails and posted comments/messages. This large amount of data can be selectively collected and then analysed not only by using inferential statistics to identify activity patterns but also by using ontology engineering formalisms that support the extraction of semantic information [8,9].

In recent work [3], we identify three phases of the learning process occurring in a FLOSS environment, *initiation*, *progression* and *maturation*, and two categories of FLOSS contributors, *novice* and *expert*. For each phase and category of contributor, we make use of semantic search in SQL to retrieve data from posts and emails, in order to identify those activities, carried out by FLOSS members, that may contribute to the members' learning processes. The choice of the key-words and key-phrases that drive the semantic search is based on a number of studies that analyse FLOSS communities using social science means to identify questions and answers that normally occur during collaboration and communication in FLOSS environments [8]. States of the process are associated with lists of generic key-words/key-phrases while specific activities are associated with lists of more discriminative key-words/key-phrases. Examples of states are: *observation* and *contact establishment*, for the initiation phase; *revert*, *post* and *apply*, for the progression phase; *analyse*, *commit*, *develop*, *revert* and *review*, for the maturation phase. Example of activities are: *formulate question*, *identify expert* and *post message* as novice's activities of the *observation* state; *run source code* as expert's activity of the *apply* state; *submit code* and *submit bug report* as novice's activities of the *commit* state; *write source code* as novice's activity of the *develop* state. The resultant three catalogs, one for each phase of the learning process, are used to build organised event logs out of the unstructured data. Using DISCO process mining tool [2], a visual representation of a process model is extracted from the event log [3].

A number of pilot studies have analysed communications in FLOSS communities in terms of participants, quantity and sometimes topics by using questionnaires and surveys or written student reports describing the encountered risks as research instruments. These previous works were the basis for our definition of *a priori* models of the collaboration and learning processes occurring in FLOSS communities [1]. Using conformance analysis, these *a priori* models are compared with the event logs, thus detecting a number of deviations. Finally, such deviations are interpreted on the discovered *a posteriori* model in order to reconcile it with the corresponding original *a priori* model.

## 2.2 Modelling Skill Acquisition

The modelling and conformance analysis approach described in Sect. 2.1 refers to the learning process at a specific phase of the contributor’s growth as a member of the FLOSS community, according to the two points of view of the novice looking for guidance and the expert providing support. However, transition between learning phases is not instantaneous but proceeds as a gradual evolution determined by the acquisition of new skills and their exploration in the social and productive contexts of the FLOSS community.

Understanding the aspects of skill acquisition, its individual variations and the social, technological and organisational factors that naturally encourage, constrain or hinder it is essential to design an appropriate learning model based on the exploitation of FLOSS projects. Given the diversity with which skill acquisition occurs for different individuals, and the consequent difficulties in collecting comprehensive data through social science research methods, it is hard to develop an *a priori* model of this important learning process.

In this context, process mining could be used as a primary modelling tool. As we have seen in Sect. 2.1, in order to produce catalogs for semantic search, appropriate key-words and key-phrases can be identified and associated with states and activities. However, states would now describe acquired skills, such as *coding*, *reviewing*, *testing* and *documenting*, while activities would still be the same as we identified in our previous work [3]. Furthermore, process mining could be used to associate quantitative information, such as frequency, number of repetition and approval rate, with activities. Quantitative information would be then integrated by functions that evaluate the level of skill acquisition. For example, the transition to the state *coding* can occur only if the ratio between the frequencies of *commit source code* and *write source code* is sufficiently high. Transition between states is triggered when the value of the function associated with the skill represented by the target state is above a given threshold.

Social networks are an important source of information about learning and other cognitive processes not only in the case of interaction within an online community, as in the case of FLOSS communities, but also in the context of the usage of a specific device/interface/application. Similarly to the diversity with which skill acquisition occurs for different individuals, users show large varieties in the modality of interacting with or using the device/interface/application/online resource. Moreover, the large number of features offered by these kinds of hardware and software artifacts gives users plenty of choices in developing strategies for using a specific artifact to achieve their goals. Furthermore, on the one hand, user’s creativity results in modalities of use and exploitation that were not considered by the artifact designers and developers. For example, short message service (sms) was initially introduced in the 1980s as an additional feature of mobile phones, but has nowadays become, for many users, the main or only purpose of using a mobile phone. On the other hand, artifact “pseudo-intelligence” tries to anticipate user’s (unpredictable) behaviour, thus leading to unexpected errors. For example, a text processing program might continuously rearrange the order of the items in a toolbar depending on the frequency of their use, thus

confusing users and inducing errors. This complex situation cannot be captured by collecting data through social science research methods. As a consequence, also in modelling usage or tasks or task failures it is basically impossible to develop comprehensive *a priori* models. Thus, the application of process mining to online reviews and user community communications could extract important information about behavioral patterns underlying usage strategies, task performance and task failure.

Finally, online reviews of products contain a large amount of information about their standard and non-standard usage as well as their pitfalls and failures. Moreover, new hardware and software products give rise to new online communities of users who exchange opinions on the product, report their usage experiences, post requests for help, reply by providing advices and, most important, learn from each other, thus improving their skills and evolving from being novices to being experts. We claim that both online reviews and user community communications can be mined to extract information about user's skill acquisition in using the product. Therefore, a process mining based approach could be used also in this context to define a skill acquisition model to be used for evaluating the quality of the product and improving new releases in terms of learnability and usability.

### 3 Towards Real-Time Process Mining

In various domains a large amount of data is collected using geographical information system (GIS) in association with a wireless sensor network. This is the case, for example, in: *ethology*, by equipping individuals of animal species with tracking devices in order to monitor animal behaviour and migration routes; *transportation*, by exploiting GPS devices on cellphones or cars; *ecology*, where wireless sensor networks are used to collect real-time information about environmental conditions. In some cases, such as in ethology, data processing occurs normally only after data collection has been completed, whereas, in other cases, data must be processed in real time in order to decide corrective or emergency action to be carried out promptly. For example, in transportation, traffic may be redirected in real time to avoid congestion, while, in ecology, real-time data flow allows researchers to react rapidly to events, thus extending the laboratory to the field [6].

We envisage the use of real-time process mining to produce visual presentations of bottlenecks and alternative routes, from traffic event logs, for traffic management, as well as informed, visual presentations of the real-time situations, from sensor network and social network logs, for emergency management.

### 4 Conclusion

In this paper we have extensively discussed the possible use of process mining as an effective modelling and validation tool to support the design and validation of a number of frameworks and systems spanning across various application

domains. We have envisaged that process mining could be effectively applied to a variety of domains other than BPM: learning, HCI, cognitive modelling, traffic management and emergency management. As a support to our claims, we have shown the successful use of process mining and conformance analysis in the area of learning, by referring to our previous work on process mining FLOSS repositories, which aimed to validate an *a priori* model of the learning processes naturally occurring within FLOSS communities [3, 4]. In our approach, process mining is applied to FLOSS communities to discover dynamic processes (learning processes, that is, processes that produce learning). This is different from van der Aalst and Song's approach to discover and analyse social networks from event logs [10]. In fact, their approach consists in extracting information about the activity performers described by the event logs, whereas ours consists in extracting information about control flow and building statistics about the occurrence of activities.

## References

1. Cerone, A.: Learning and activity patterns in OSS communities and their impact on software quality. In: Proceedings of OpenCert 2011, ECEASST, vol. 48 (2012)
2. Günther, C., Rozinat, A.: DISCO: discover your process. In: Proceedings of the Demonstration Track of BPM 2012, CEUR Workshop Proceedings, vol. 940, pp. 40–44. CEUR-WS.org (2012)
3. Mukala, P., Cerone, A., Turini, F.: Mining learning processes from FLOSS mailing archives. In: Janssen, M., Mäntymäki, M., Hidders, J., Klievink, B., Lamersdorf, W., van Loenen, B., Zuiderwijk, A. (eds.) I3E 2015. LNCS, vol. 9373, pp. 287–298. Springer, Heidelberg (2015)
4. Mukala, P., Cerone, A., Turini, F.: Process mining event logs from FLOSS data: state of the art and perspectives. In: Canal, C., Idani, A. (eds.) SEFM 2014 Workshops. LNCS, vol. 8938, pp. 182–198. Springer, Heidelberg (2015)
5. Mukerjee, K., Porter, T., Gherman, S.: Linear scale semantic mining algorithms in Microsoft SQL server's semantics platform. In: Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 213–221. ACM (2011)
6. Porter, J., et al.: Wireless sensor networks for ecology. *BioScience* **55**(7), 561–572 (2005)
7. Rozinat, A., van der Aalst, W.M.P.: Conformance checking of processes based on monitoring real behavior. *Inf. Syst.* **33**(1), 64–95 (2008)
8. Scacchi, W., Feller, J., Fitzgerald, B., Hissam, S.A., Lakhani, K.: Understanding free/open source software development processes. *Softw. Process Improv. Pract.* **11**(2), 95–105 (2006)
9. Sowe, S.K., Cerone, A.: Integrating data from multiple repositories to analyze patterns of contribution in FOSS projects. In: Proceedings of OpenCert 2010, ECEASST, vol. 33 (2010)
10. van der Aalst, W.M.P., Song, M.S.: Mining social networks: uncovering interaction patterns in business processes. In: Desel, J., Pernici, B., Weske, M. (eds.) BPM 2004. LNCS, vol. 3080, pp. 244–260. Springer, Heidelberg (2004)
11. van der Aalst, W.M.P., Stahl, C.: Modeling Business Processes: A Petri Net-Oriented Approach. The MIT Press, Cambridge (2011)

# On Integrating Social and Sensor Networks for Emergency Management

Farshad Shams, Antonio Cerone<sup>(✉)</sup>, and Rocco De Nicola

IMT Institute for Advanced Studies, Lucca, Italy  
{farshad.shams,antonio.cerone,rocco.denicola}@imtlucca.it

**Abstract.** The 2010 earthquake in Haiti is often referred to as the turning point that changed the way social media can be used during disasters. The development of strategies, technologies and tools to enhance user collaboration around disasters has become an emergent field, and their integration with appropriate sensor networks presents itself as an effective solution to drive decision making in emergency management.

In this paper, we present a review of existing disaster management systems and their underlying strategies and technologies, and identify the limitations of the tools in which they are implemented. We then propose an architecture for disaster management that integrates the mining of social networks and the use of sensor networks as two complementary technologies to overcome the limitations of the current emergency management tools.

## 1 Introduction

Crisis response activities include undertaking measures to protect lives and properties immediately before, during, and immediately after the occurrence of a disaster. Such activities may span from a few hours to days or even months, depending upon the magnitude of the event. Disaster management using the World Wide Web is an emergent field that uses technology to enhance users collaboration around disasters. While there exist a number of dedicated “disaster portals” [6, 12, 13], large social networks such as Twitter, Facebook, and Google-plus can facilitate the analysis and sharing of a collective intelligence regarding disaster information on a much greater scale. Recent disasters (e.g. Haiti, Australia, Japan, Mexico, etc.) have demonstrated their real potential in providing support to emergency operations for crisis management [38]. Social networks have the potential to increase accessibility to eyewitness’ information, and exploiting their input to gain awareness of an incident is an important research topic. Reaching populations by means of customised and timely alerts through multiple channels (Internet technologies, hand-held devices, and social networks) can help inform those at risk and assure those not at risk with messages that accurately reflect the levels of vulnerability of the target population.

From the earthquake in Haiti in 2010 to the terrorist attack in Boston in 2013, Facebook, Twitter and other social media have shown ability to provide valuable support to civil protection in emergency situations. The earthquake in

Haiti is often referred to as the turning point that changed the way social media can be used during disasters. The size and emotional impact of disasters have created the right motivations for the integration of social media in emergency management. In 2005, when the hurricane Katrina devastated the US coast of the Gulf of Mexico, Facebook was one year old, there was no Twitter, and smart phones were not common yet. In 2012, when hurricane Sandy hit the east coast of the US, social media had become an integral part of the response to disasters: millions of Americans used social media to follow the news, look for persons, and send requests to the authorities. Researchers have now begun to publish reports on the use of social media during disasters, and law and security experts have started to evaluate how to best exploit social media in emergency management.

In 2011, the Australian state of Queensland was impacted by floods and a severe tropical cyclone. While there was significant media coverage, social networks were also inundated with posts related to the floods. A government social media outlet, “@QPSmedia” was created and utilised for community interaction regarding the disaster. In 2012, the Department of Health and Human Services (HHS) in USA sponsored a challenge for software application developers to design a Facebook application called the Personal Emergency Preparedness Plan (PEPP) [34]. Riskr [31] is a low-technological project which applies a Web2.0 [46] solution to creating disaster portals fed by social networking messages; the system has been implemented using Twitter and tested by users to determine the advantages of having interoperability between social networks and disaster portals. Farber *et al.* [31] state that almost all Riskr users have no problem with predefined hash-tags and they are satisfied about the received information. There are some challenges to process posts in social networks. The social networks are not optimised and specialised for emergency management and the posts may include sentimental words, emoticons, links, and personal and untrustable opinions. Extraction of related posts, categorisation of heterogeneous messages, and determination of the trustworthiness of messages are some of the challenges to face when using social media for managing catastrophic events. Efficient methods of handling subjective information, uncertainty, different level of credibility, extraction of exact location (position), and sentiment context should be used to make the utmost use of the data collected through a stream of posts.

In this paper we present a review of existing disaster management systems. In particular, we consider strategies and technologies for the analysis of information collected by mining social networks and information provided by a wireless sensor network, as well as the use of geo-spatial technologies. We claim that the integration of information from these different sources is essential to implement a decision making system aiming at promptly disseminating alerts, efficiently organising rescue activities and providing effective support. Section 2 defines terminology and phases of emergency management. Section 3 reviews existing disaster management systems. In Sect. 4, we propose an architecture for disaster management that integrates the mining of social networks and the use of sensor networks. Finally Sect. 5 clarifies which parts of the proposed architecture have already been implemented.



## 2 Emergency Management

The term *emergency*, or *disaster* or *crisis*, refers to a situation that poses an immediate risk to health, life, property or the environment. Thus *emergency management*, or *disaster management*, is used to encompass all plans defined by national or local agencies, called Emergency Management Agencies (EMAs) and the consequent activities carried out to tackle an emergency situation, or disaster, with the goal of reducing harm to life, property and the environment, and then return to a normal functional condition [26]. Emergency management is an interdisciplinary field of study, which involves intertwined social, political, technological, economic and cultural aspects. Moreover, there is a general consensus in identifying four phases in handling disasters [26]:

- mitigation** often called *prevention* or *risk reduction*, aims to reduce the likelihood or consequence of a hazard risk by defining appropriate measures and procedures before the occurrence of a disaster;
- preparedness** is defined as actions taken in advance of a disaster, as implementation of mitigation measures and procedures, together with roles and responsibility assignments as well as service availability, to ensure adequate response to its impacts and the relief and recovery from its consequences;
- response** when the disaster occurs, all involved actors promptly contribute, according to their roles and within the confines of their limited funding, resources, ability and time, to the search and rescue, with the coordination of the appropriate EMAs that activate and manage measures and procedures;
- recovery** is the process of rebuilding, reconstructing and repairing damages and destruction caused by the disaster, and finally restoring the normal situation.

Mitigation is often considered the “cornerstone of disaster management”. It has been for a long time perceived as a luxury of the wealthy countries, but has now started to gain recognition and practical application also in the developing world. Recently, *prevention* has been distinguished from mitigation to better characterise pro-active measures designed to provide permanent protection [10]. Response is usually subject to extensive media coverage and is therefore the most visible disaster management function at the international level. The effectiveness of the response phase not only depends on the promptness in carrying out search and rescue actions, but also on the definition, during mitigation, of measures and procedures that facilitate data selection, aggregation, integration and availability, and optimal scheduling of actions, as well as on their efficient implementation, during preparedness.

## 3 Literature Review on Disaster Management Systems

Some of the current disaster management solutions are inadequate to help manage disaster due to manual data collection and entry, which result in delayed dissemination of information. Sahana [22] is an open source disaster management system that has been widely used. It has a modular structure for effective

communication and information sharing among various stakeholders including government, NGOs and affected people. However, being a traditional database management system, it requires manual data entry. Global Disaster Information Network (GDIN) [7] is another conventional web based information system that provides effective communication. However, it lacks in the effective management of disaster data. Other notable disaster management systems are Queensland Disaster Management System [6] and Disaster Management Information System (DMIS) [5]. All these systems provide no support for automatic information collection and this is a severe limitation in situations where time is one of the most precious entities and even seconds can save lives.

In this section we review a number of disaster management systems that adopt technologies for automatic information collection. We starts with systems based on the analysis of social networks. Then we consider systems that use wireless sensor networks. We conclude with discussing the use of Geographical Information System (GIS) and Global Positioning System (GPS).

### 3.1 Systems Based on Analysis of Social Networks

Recently, Twitter has played an increasing role as a clearinghouse for information related to emergencies and disasters. Sakaki *et al.* [49] propose an algorithm for tweets monitoring and real-time target detection. To detect a target event, the authors devise a classifier of tweets based on features such as the keywords in a tweet, the number of words, and their context. They then produce a probabilistic spatiotemporal model for the target event that can find the center of the event location. They apply a particle filtering for each Twitter user to achieve a better estimation of target event location. Nguyen *et al.* [45] define an event by five attributes for each Twitter message: actor, action, object, time and location. They build a collective, readable and intelligence-based web ontology tool that understands the meaning of Japanese text messages and extracts semantic data about incidents. The authors propose a novel approach which can automatically build an earthquake semantic network by mining human activities from Twitter. By using this semantic network, computers can recommend suitable action patterns for victims. This approach automatically makes its own training data and uses linear-chain conditional random field as a learning model. In the project conducted by Nguyen *et al.*, the text extractor architecture consists of two modules: “self-supervised learner” and “activity extractor”. Firstly, the learner uses basic Japanese syntax patterns to select analysable activity sentences. Then, it uses deep linguistic parsing to extract activity attributes and relationships between activities in these sentences. Secondly, a “decision module” uses extracted actions and objects to create search keywords for Twitter API.

Although news information is widely available through social media such as Twitter, the credibility of such information may be questionable. To assess the credibility of information propagated through Twitter, Castillo *et al.* [23] propose a mechanism using features from the content of posts and from citations

to external sources. The study conducted by Westerman [53] found a curvilinear pattern between the number of followers and Twitter user's credibility.

### 3.2 Systems Based on Wireless Sensor Networks

Sensor networks have the potential to revolutionise the capture, processing and communication of critical data for use of disaster rescue and early-warning systems. Event detection functionality of wireless sensors can be of great help and importance for real-time detection of, for example, meteorological natural hazards and residential fires. The basic idea of event detection is to define some threshold values and generating an alarm by sensor when input is lower/higher than a pre-defined threshold value. Due to the fact that disasters cannot be detected by simple pre-defined thresholds [43], the new trend in event detection is to use pattern matching or machine learning techniques. Based on the scale of the network, application requirements and constraints, pattern matching have been proposed for use in the base station [55], locally in the sensor nodes [19], or distributed over the network [42]. The RT-HRLE system [32] uses a wireless sensor network for real-time monitoring, tracking of missed people inside buildings and reporting partial or total destruction of buildings to a central database. Liu *et al.* [41] present an architectural approach for wireless sensor networks which can proactively self-adapt to changes and evolution occurring in the provision of search and rescue capabilities in a dynamic environment. The proposed model uses the concepts of dynamic workflow management to enable dynamic service integration for reliable and sustainable provision of rescue capabilities. This approach is able to identify evolution, evaluate the impact of evolution and self-configure services to adapt to evolution.

### 3.3 Systems Based on Geo-Spatial Technologies

Disaster management involves not just crisis-reactive responses to emergencies, but also finding ways to avoid problems in the first place and preparing for those that undoubtedly will occur. Natural disaster management is a complex and critical activity that can be more effectively addressed with the support of geo-spatial technologies and spatial decision support systems [30,48]. In this respect, spatial data and related technologies such as GIS and GPS have been proven crucial for effective disaster management [16,28]. Throughout the first three phases of disaster management, preparedness, mitigation, and response, a GIS can effectively facilitate the integration of spatially distributed information within a decision support system. The effectiveness and growth of GIS and GPS is however dependent on the development of a national disaster management database underlying the varied scope and activities pertaining to national emergency management. A national database provides a common frame of reference for all provincial and local agencies and establishes the framework for managing and organising the data required to support the disaster risk management activities of responsible organisations. Gunes *et al.* [35] aim at building a database in a GIS frame that helps emergency management officers in decision

making, focusing on Douglas County’s preparedness, mitigation, and response efforts for its most common disaster: flooding. The system leads to better flood management by automating the task of determining the probable flood-affected areas and integrating the results with other spatially distributed information. This enables emergency management officers to make more informed decisions before, during, and after a flood situation. Pareta and Pareta [47] address the need, the technical structure and the potential solutions facilitated by the creation of an effective database at a national level and draw upon experience from work in Vietnam and practices from India.

## 4 Towards an Integrated Server Architecture

In this section we propose a comprehensive server architecture, as illustrated in Fig. 1, which consists in a “Command, Control, Communication Computers, and Intelligence” (C4I) [54] unit communicating with the database that provides a common frame of reference for all local agencies (Agency Data) and with sensor networks, social networks, etc. The incorporated components include mechanisms to model event level semantic information, a system for implementing multi-sensor fusion, mechanisms for estimating the veracity of information, data cleaning to reduce uncertainty and enhance accuracy of event detection and notification, and spatiotemporal analyses for pattern and trend analyses for higher level observations. Such a modular architecture makes it possible to upgrade the platform in the future as needs change or new technologies appear. The proposed platform is capable of processing information about various types. Processing can be configured using rules and may include configuration for data loading, pre-processing, aggregating, statistics building, correlating with other events and storing in a database.

The architecture consists of services on which data are ingested, cleaned and analysed to extract information that is customised for emergency services, and hand-held devices on which alerts are visualised. The disaster portal that interfaces the server leverages core features of the platform such as notification and authorisation dialog for citizens. Citizens can register themselves and their own socio-economic situations (having car, address, disabilities, situation of building, etc.) to receive the most appropriate alert messages in pre- and post-catastrophic events. Citizens can also register themselves as volunteers and encourage other citizens to participate. The server can send requests to volunteers in a threatened area and volunteers can send their plans to the server. The portal allows registered users to send their real-time location and situations. Unregistered users can communicate with the server through different channels, e.g. phone call and messages through cellphone. Obviously, analysis and visualisation techniques implemented in such tools need to be customised to the specific disaster management subject domain.

In a disaster management service, deployment scenarios for sensor networks are countless and diverse. For example, sensors may be used for weather forecasting, tsunami detection, pollution detection, and video surveillance. Normally,

a disaster management server allows the operator to query a sensor network and retrieve some resulting data. However, some scenarios may require regular queries to be scheduled and automatically dispatched without external operator intervention. Furthermore, there is a growing need to share resources among diverse network deployments to aid in critical tasks like decision making. For example, a tsunami warning system may rely on water level information from two geographically distributed sets of sensors developed by competing hardware vendors. This presents significant challenges in resource interoperability, fault tolerance and software reliability. We need to implement a set of uniform operations and a standard representation for different entities, sensors data and web services data, which can fulfil the software needs of a network regardless of the deployment scenario. The proposed platform capabilities include:

- monitoring of sensors and social media for the relevant information;
- semantic enrichment of information through multi-modal analysis (tweets, sensor, etc.) to create event level representation;
- integration with other information sources such as the agency database;
- querying targeted sensors and rescue agents for recent updates based on their location;
- making the best decision and generating custom alerts for specific population groups.

Finally, the choice of an adequate data interchange format can have significant consequences on data transmission rates and performance. XML and JSON (JavaScript Object Notation) serialisation have been widely used in the actual development of web applications. Compared to XML, JSON has higher parsing efficiency and the advantages of easy preparation. Since JSON is not just a text format, but a serialised data structure, Resource Description Framework (RDF) libraries can support it [15], not just as a format to parse from or output to, but also internally, as a data structure that can be passed to and returned by functions and methods.

In Sects. 4.1–4.4 we describe the server components, as shown in Fig. 1.

#### 4.1 Event Collection Module

This module gathers information on crisis-related events from a variety of disperse sources including voice, text, image, video, and sensors. It explores integrated distributed systems, data management, and networking systems that enable information to seamlessly flow in real-time from sources to collection points. This is done by periodically querying social networks with different incident related key-words, sending requests to sensors and waiting for voice/message calls. The received data must be pre-processed and stored according to agreed standards to support data sharing with incident management applications. For instance, sensors data may be labeled with sensor ID and location, voice calls may be labeled with the phone number and location. Finally, each type of data is converted to an appropriate low-volume format to facilitate analysis.

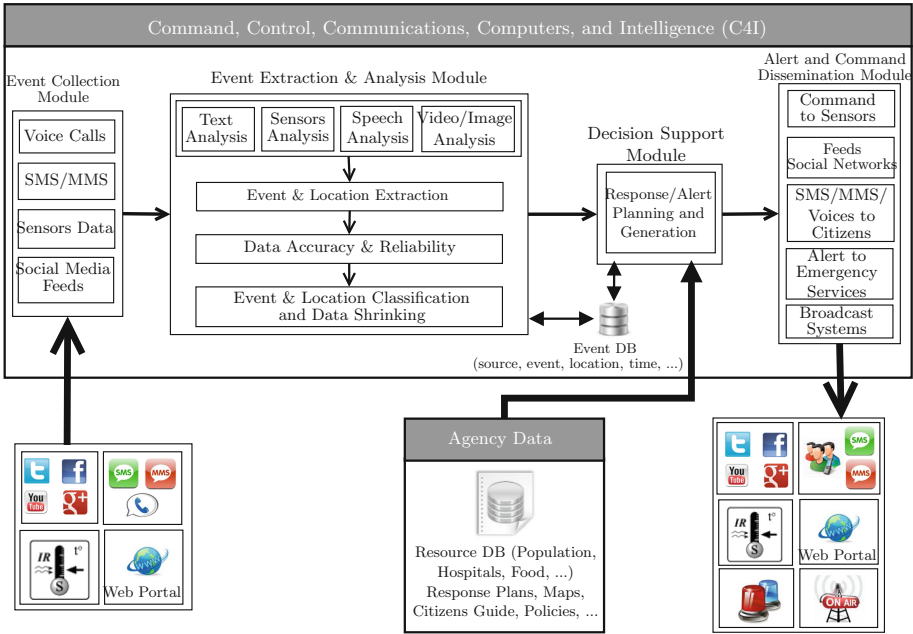


Fig. 1. Server components for data ingestion and enrichment, and alerting tasks.

In geographically spread hazards, such as earthquakes, the amount of resulting generated data is voluminous. Real-time analysis and collecting such data could overwhelm any computing infrastructure. While dynamically acquired cloud computing alleviates some of the overheads, scaling data collection to such big data requires additional techniques. One possible technique is the prioritisation of the collection of diverse data by dynamically optimising the overall situational awareness under resource constraints and source restrictions, e.g. network bandwidth and maximum concurrent queries.

One of the goals in data acquisition is to ensure that the data collected is relevant for the event under consideration and avoid retrieving too many irrelevant data. Precision is obtained by ranking and clustering different keywords and terms relevant for the event. The retrieval challenge is to get as many relevant messages as possible. The precision challenge is instead tackled by applying context-based filtering techniques [36]. In addition, boosting methods [20] are used to collect the initial set of messages related to the event, and then, based on them, select new frequent key-words as query terms. Such query terms could also be chosen using ontologies such as SWEET [4].

The server controls several types of sensors and video-surveillances spatially located in different places. However, such a variety of sensors requires interfacing applications to perform common operations and transformations on sensor data. As sensor data is time-dependent, the user needs to provide, essentially, the desired geographic area, the desired time interval, and the desired properties

to be observed. With the specifications defined through the Sensor Web Enablement (SWE) [8] initiative of the Open Geospatial Consortium (OGC) [9], flexible integration of sensor data is becoming a reality. The NICTA Open Sensor Web Architecture (NOSA) infrastructure [2] is built upon the SWE standard defined by the OGC, which consists of a set of specifications, including sensor model language, observation, measurement, sensor collection service, sensor planning service and web notification service. NOSA adopts a Service Oriented Architecture (SOA) approach to describe, discover and invoke services from a heterogeneous platform using XML and SOAP standards. Services are defined for common operations including data aggregation, scheduling, resource allocation and resource discovery. Each sensor is registered as a web service that can be comfortably discovered. Combining sensors and sensor networks with a SOA is an important step forward in presenting sensors as resources to discover, access and, where applicable, control via the World Wide Web. It offers the opportunity of linking geographically distributed sensors and computational resources into a “sensor-grid”.

The main challenges in sensor networks are the discovery of appropriate sensor information and the real-time fusion of the discovered information [51]. They are key issues in disaster management, where the flow of information is overwhelming and sensor data must be easily accessible for non-experts. By registering every sensor as a web service, sensor discovery and fusion can be carried out by semantically annotating services with terms from a purposed-designed ontology. In doing so, several well known techniques from the GIS and semantic web worlds can be employed. Semantically, annotations of geographically distributed sensors provide an infrastructure with which on-line discovery and integration of sensor data is not more difficult than using standard GIS applications. The service discovery is realised by text search combined with taxonomy browsing [18]. For instance, suppose a number of different types of sensors, e.g. water/air pollution, water level and water temperature, are placed in different point of a river. The highest level of taxonomy will be “river” and “sensor”. The second level connected to river taxonomy will be the different sectors/points of the river and the second level of the sensor taxonomy will be the different types of sensors. The third level of the sensor taxonomy will be the different installed sensors (ID number). Each sensor (ID) is (semantically) connected to a sector/point of the river. The emergency officer can then query the situation of the whole river, or of a specific sector of the river. Then, the server commands the proper sensors, the observed data is integrated/fused together and the result is finally shown on the GIS map.

## 4.2 Event Extraction and Analysis Module

This module further analyses the pre-processed data stored in the databases of an emergency management system to extract meaningful semantic information. The semantic enrichment is the context of location determination and event representation. In addition, the semantic information extracted from multiple data is fused for event classification and disambiguation. In case any ambiguity

is found in the received data, the system can ask more information from citizens to achieve a reliable message. The enriched data is compared with previous similar events stored in the local database. One important challenge is to develop technologies and tools to integrate, analyse and visualise multiple information sources to rapidly assess the nature, composition and pattern of threats, and to address public safety practitioner requirements.

One of the crucial data enrichment challenges is text (tweets, SMS) analysis to extract disaster related information. The sentences retrieved from social media, or received through SMS are complex, often structurally varying, syntactically incorrect, and have many user-defined new words. Thus, extracting activities from these sentences might be very difficult. The Event Extraction and Analysis Module exploits some known platforms for Natural Language Processing (NLP). GATE [33] is one of the most popular platforms for Natural Language Processing (NLP) widely used in industry and academia to extract information from text. The actual processing of the content goes through several steps, starting with tokenisation, sentence splitting and speech tagging. These processing layers are provided by GATE along with grammars and other standard building bricks for obtaining sophisticated information extraction applications. ANNIE (A Nearly New Information Extraction System), a plug-in of GATE, is used to extract disaster information. ANNIE [27] uses PRs (Processing Resources) that have been developed using the JAPE (Java Annotation Pattern Engine) language [52], a pattern/action rule language based on regular expressions.

Another challenge is speech analysis; there have been several recent announcements surrounding the application of speech-to-text analysis in consumer search settings. Google announced its *Political Gadget*, enabling visitors to search the spoken word of content within YouTube Presidential candidate's channels. Adobe plans to include speech-to-text features in future versions of its video authoring applications, such as Premier. Sites such as *WEEI* [1] and *FOX Sports* [3] have been using similar tools to power search and publishing applications for their multimedia archives. The Event Extraction and Analysis Module automatically transcribes speech messages and extract meaningful keywords and then analyses it as a text message.

Detecting the occurrence of an incident-relevant event within a multimedia clip is another task of this module. The representation of an image/video into a set of key-words have been successfully used in many detection and recognition tasks such as object detection [29], scene recognition [39], human action recognition [40] and semantic concept detection [37].

Another crucial analysis technology supported by this module is the geolocalisation of data from sensors, cell-phones and social media sources. This can be done at several levels of complexity, including cellphone location, sensors' location, extraction from tweet, or complex computer vision algorithms that can localise images based on skylines or building facades. Having information about where an event occurs allows for various geo-spatial analyses that can support alert customisation to subscribers. With the advances in location-aware mobile devices, location-based social networking applications have been taking shape



at fast pace. Examples of such applications include Google Buzz Mobile, Loopt, and Microsoft Geo-Life. Potential applications of these systems include the possibility for users to receive nearby geo-tagged messages submitted by friends and to find a new facility within a certain area based on friends' opinions, and completely ignore the social aspect in social networking services. GeoSocialDB [25] provides a holistic framework consisting of three location-based social networking services, namely, location-based news feed, location-based news ranking, and location-based recommendation. Even though GeoSocialDB is not specialised for emergency situations, it can certainly be used to generate queries like: "Send me the  $k$  most relevant messages submitted by victims with tagged locations within  $d$  kilometers of my location", or "Recommend me the best street to go away within  $d$  kilometers of my location based on emergency officers opinions".

Event data management is concerned with providing data management or "database like" capabilities for events. It is important to treat events as objects and provide storage, querying, retrieval and indexing capabilities for them. We are working on several issues in this regard. For example, concerning event modelling, we are developing a semantic data model for events. A large collection of distributed reports are generated during a disaster. In its original form, these data is of limited use, since users can only apply keyword searching to it. It is therefore necessary to extract events and inter-event relationships to produce more structured data, in order to be able to apply most of the existing exploratory and analytical tools.

### 4.3 Decision Support Module

Decision support is an essential functionality of an emergency management system. This module processes the enriched event data to generate targeted alerting. This can be achieved by integrating the structured event representation with other local data such as demographics and resource availability in different areas and organisations. By purposefully utilising collected information, for instance by a data fusion system, the state of some system-relevant environment is adequately assessed to support decision-making. Various multimodal data streams and static environmental information (geo-spatial information) are fused together to produce a refined decision. The GIS-enhanced information can enable decision makers to match on-ground situations and determine alerting requirements in different areas. Pre-defined policies are incorporated into a rule-base to dictate the kind of guide and information to be provided in an alert. This is used to generate messages for specific population groups that are categorised based on location and physical disabilities (elderlies, patients in a hospital). For instance, in the case of an industrial fire, the alert could be sent to people in the neighbourhood and this alert should be different from the one sent to people living at a safe distance from the fire.

Analysis and visualisation are concerned with providing intuitive and visual analysis and querying capabilities for managers of situational information. It is reasonable to expect that managers or field commanders would finally like to see patterns and trends in the information collected and get intuitive and

visual views of the information. In this regard, we propose to develop tools such as a graph based query algebra and language over events. This allows users to query and analyse events in a graphical manner. The resulting graph based semantic network can be stored as either a multi dimensional table or an RDF file. These enable the use of online analytical processing (OLAP) queries [21, 24] and SPARQL (Simple Protocol and RDF Query Language) queries [14], respectively.

An estimation of the current location and number of people in a disaster area and a prediction of the future movement of those people could provide critical information to disaster operation command staff responsible for rescue and evacuation, and also to victims looking for the best way to navigate to a safe place. One important initiative for disaster emergency personnel can be a real-time evacuation planning model, which automatically calculates the evacuation time of a user defined area based upon the transportation network, the population data, and behavioral characteristics to provide emergency planners the ability to effectively plan for and manage evacuations. “People forecasting” and occupancy analysis in real-time are crucial to predict freeway traffic information and (future) event prediction. Occupancy analysis could be carried out by extracting information about human behavior from a variety of sensors such as loop sensors counting cars on a freeway, people counters at doors of buildings and GPS devices on cellphones or cars. A “personal traffic assistant” running on mobile devices can help travelers re-plan their travel when the routes are impacted by failures.

#### 4.4 Alert and Command Dissemination Module

This module focuses on the challenges associated with the timely dissemination of information to entities participating in disaster response activities, to other organisations (e.g. mass media organisations), and to the general public. Empirical social science research has focused on issues related to the dissemination of hazard-related information in both pre- and post-disaster contexts.

A possible strategy for alert dissemination consists in dividing and routing volunteers to different threatened neighbourhoods based on the event propagation, the number of victims, and volunteers’ location. Without proper planning for the route of each volunteer, some places may be visited repeatedly while others may not be visited at all. Furthermore, repeated visits of some places may prolong the response time of exploring the whole disaster area. As a consequence, a major limitation of using social networks and broadcast systems to explore disaster areas is the lack of coordination among volunteers. Google map API, MapQuest [17] and Quantum GIS [11] open source software provide powerful and user friendly tools to find the best GIS platform for spatial management applications.

This module supports customised delivery of alerts to specific sub-populations based on location, the current status of the incident and its expected effects and propagations, status of various locations, needs, etc. A flexible policy definition mechanism allows customisation based on location and geographical information and type of event [44]. The policy determines how often an individual user may be reached as the event evolves and the protective actions change.

## 5 Conclusion and Future Work

In this paper we have reviewed a number of technologies and systems for emergency management. To our knowledge, none of the existing disaster management systems support the integration of data from sensor networks and social networks. We have proposed an architecture that combine a number of existing technologies to achieve such integration.

Some of the functionalities of our proposed architecture have been implemented by the WiLIFE project (Tecnologie WireLess e ICT per un efficiente e integrato sistema per la prevenzione e gestione delle situazioni di crisi e delle Emergenze — <http://www.wilife-project.it/>) during 2012–2015 [50]. With respect to the architecture proposed in Sect. 4, the WiLIFE project produced: an implementation of the Event Collection Module that supports the integration of events from a wireless sensor networks with data from social networks; some functionalities of the Event Extraction and Analysis Module, including some level of geo-localisation of the extracted data; the data presentation aspects of the Decision Support Module; dissemination of commands to public service operators and alerts to Twitter users, also through a mobile app available for Android platforms, which are part of the Alert and Command Dissemination Module. The implementation of further functionalities of the Alert and Command Dissemination Module and the prediction aspects of the Decision Support Module are part of our future work, subject to funding availability.

## References

1. <http://audio.weei.com/>
2. <http://gridbus.csse.unimelb.edu.au/sensorweb/>
3. <http://msn.foxsports.com/topics/>
4. <http://sweet.jpl.nasa.gov/>
5. <https://www-secure.ifrc.org/dmisii/>
6. <http://www.disaster.qld.gov.au/>
7. <http://www.gdin.org/>
8. <http://www.ogcnetwork.net/swe>
9. <http://www.opengeospatial.org/>
10. <http://www.preventionweb.net/english/hyogo>
11. <http://www.qgis.org/>
12. <http://www.un-spider.org/event/6485/2013-05-01/disaster-management>
13. [http://www.undp.org.ir/drm/en/national\\_disaster\\_portal.asp](http://www.undp.org.ir/drm/en/national_disaster_portal.asp)
14. <http://www.w3.org/tr/rdf-sparql-query/>
15. Alexander, K.: RDF in JSON: a specification for serialising RDF in JSON. In: SFSW (2008)
16. Amdahl, G.: Disaster Response: GIS for Public Safety. ESRI, Redlands (2002)
17. M.O.D. APIs and W. Services. <http://developer.mapquest.com/web/products/open>

18. Babitski, G., Bergweiler, S., Hoffmann, J., Schön, D., Stasch, C., Walkowski, A.C.: Ontology-based integration of sensor web services in disaster management. In: Janowicz, K., Raubal, M., Levashkin, S. (eds.) *GeoS 2009*. LNCS, vol. 5892, pp. 103–121. Springer, Heidelberg (2009)
19. Bahrepour, M., Meratnia, N., Havinga, P.: Use of AI techniques for residential fire detection in wireless sensor networks. In: *Artificial Intelligence Applications and Innovations (AIAI)*, vol. 475, Greece, July 2009
20. Becker, H., Naaman, M., Gravano, L.: Selecting quality Twitter content for events. In: *Proceedings of the International Conference on Weblogs and Social Media (ICWSM)*, Barcelona, Spain, July 2011
21. Berson, A., Smith, S.J.: *Data Warehousing, Data Mining, and OLAP*, 1st edn. McGraw-Hill Inc, New York (1997)
22. Careem, M., De Silva, C., De Silva, R., Raschid, L., Weerawarana, S.: Sahana: overview of a disaster management system. In: *International Conference on Information and Automation (ICIA)*, Colombo, Sri Lanka, December 2006
23. Castillo, C., Mendoza, M., Poblete, B.: Information credibility on twitter. In: *International World Wide Web Conference*, Hyderabad, India, March–April 2011
24. Chaudhuri, S., Dayal, U.: An overview of data warehousing and OLAP technology. *SIGMOD Rec.* **26**(1), 65–74 (1997)
25. Chow, C.-Y., Bao, J., Mokbel, M.F.: Towards location-based social networking services. In: *Proceedings of the ACM SIGSPATIAL International Workshop on Location Based Social Networks*, San Jose, California (2012)
26. Coppola, D.P.: *Introduction to International Disaster Management*. Butterworth-Heinemann, Burlington (2011)
27. Cunningham, H., Maynard, D., Bontcheva, K., Tablan, V.: GATE: a framework and graphical development environment for robust NLP tools and applications. In: *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, Philadelphia, PA, USA (2002)
28. Cutter, S.L.: GI science, disasters, and emergency management. *Trans. GIS* **7**(4), 439–446 (2003)
29. Dalal, N., Triggs, B., Schmid, C.: Human detection using oriented histograms of flow and appearance. In: Leonardis, A., Bischof, H., Pinz, A. (eds.) *ECCV 2006*. LNCS, vol. 3952, pp. 428–441. Springer, Heidelberg (2006)
30. Donohue, K.: *Using GIS for all-hazard emergency management* (2002)
31. Farber, J., Myers, T., Trevathan, J., Atkinson, I., Andersen, T.: Riskr: a low-technological Web 2.0 disaster service to monitor and share information. In: *International Conference on Network-Based Information Systems (NBIS)*, Melbourne, Australia, September 2012
32. Fawzy, D., Sahin, Y.: RT-HRLE: a system design for real-time hazards reporting and loss estimation using wireless sensors. In: *International Conference on Education and Management Technology (ICEMT)*, Cairo, Egypt, November 2010
33. General Architecture for Text Engineering. <http://gate.ac.uk/>
34. Greer, M., Ngo, J.: Personal emergency preparedness plan (PEPP) facebook app: using cloud computing, mobile technology, and social networking services to decompress traditional channels of communication during emergencies and disasters. In: *IEEE International Conference on Services Computing (SCC)*, Hawaii, June 2012
35. Gunes, A., Kovel, J.: Using GIS in emergency management operations. *J. Urban Plan. Dev.* **126**(3), 136–149 (2000)

36. Jafarpour, H., Lickfett, J., Kim, K., Xing, B.: A policy driven meta-alert system for crisis communications. In: Incident, Resources, and Supply Chain Management, DHS Workshop on Emergency Management (2009)
37. Jiang, Y.-G., Yang, J., Ngo, C.-W., Hauptmann, A.G.: Representations of keypoint-based semantic concept detection: a comprehensive study. *IEEE Trans. Multimedia* **12**(1), 42–53 (2010)
38. Kongthon, A., Haruechaiyasak, C., Pailai, J., Kongyoung, S.: The role of Twitter during a natural disaster: case study of 2011 Thai flood. In: Proceedings of Technology Management for Emerging Technologies (PICMET), Vancouver, Canada, July-August 2012
39. Lazebnik, S., Schmid, C., Ponce, J.: Beyond bags of features: spatial pyramid matching for recognizing natural scene categories. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), Washington, DC, June 2006
40. Liu, J., Luo, J., Shah, M.: Recognizing realistic actions from videos in the wild. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), Miami, November 2009
41. Liu, L., Webster, D., Xu, J., Wu, K.: Enabling dynamic workflow for disaster monitoring and relief through service-oriented sensor networks. In: International ICST Conference on Communications and Networking in China (CHINACOM), Beijing, China, August 2010
42. Luo, X., Dong, M., Huang, Y.: On distributed fault-tolerant detection in wireless sensor networks. *IEEE Trans. Comput.* **55**(1), 58–70 (2006)
43. Majumdar, K., Majumder, D.: Fuzzy differential inclusions in atmospheric and medical cybernetics. *IEEE Trans. Syst. Man Cybern. Part B Cybern.* **34**(2), 877–887 (2004)
44. McGinley, M., Turk, A., Benet, D.: Design criteria for public emergency warning systems. In: 3rd International ISCRAM Conference, Newark, USA, May 2006
45. Nguyen, T.-M., Koshikawa, K., Kawamura, T., Tahara, Y., Ohsuga, A.: Building earthquake semantic network by mining human activity from Twitter. In: IEEE International Conference on Granular Computing (GrC), Kaohsiung, Taiwan, November 2011
46. Oreilly, T.: What is Web 2.0: design patterns and business models for the next generation of software. *Commun. Strat.* **65**, 17 (2007)
47. Pareta, K., Pareta, U.: Developing a national database framework for natural disaster risk management. In: Proceedings in ESRI International User Conference, San Diego, California (2011)
48. Radke, J., Cova, T., Sheridan, M.F., Troy, A., Mu, L., Johnson, R.: Challenges for GIS in emergency preparedness and response. An ESRI white paper, May 2000
49. Sakaki, T., Okazaki, M., Matsuo, Y.: Tweet analysis for real-time event detection and earthquake reporting system development. *IEEE Trans. Knowl. Data Eng.* **25**(4), 919–931 (2013)
50. Shams, F., Capodici, P., Cerone, A., Fantacci, R., Marabissi, D., Mariotta, G., Sciuto, P., De Nicola, R.: Integration of heterogeneous information sources for an effective emergency management. *Int. J. Emergency Manage.* (2015, in press)
51. Tseng, Y.-C., Kuo, S.-P., Lee, H.-W., Huang, C.-F.: Location tracking in a wireless sensor network by mobile agents and its data fusion strategies. In: Zhao, F., Guibas, L.J. (eds.) *IPSN 2003*. LNCS, vol. 2634, pp. 625–641. Springer, Heidelberg (2003)

52. G.G. Tutorial. <http://gate.ac.uk/sale/thakker-jape-tutorial/gate>
53. Westerman, D., Spence, P.R., Heide, B.V.D.: A social network as information: The effect of system generated reports of connectedness on credibility on Twitter. *Comput. Hum. Behav.* **28**(1), 199–206 (2012)
54. What is C4I? <http://www.c4i.org/whatisc4i.html>
55. Xue, W., Luo, Q., Chen, L., Liu, Y.: Contour map matching for event detection in sensor networks. In: International Conference on Management of Data (COMAD), Chicago, IL, December 2006

# Quantitative Modelling of Residential Smart Grids

Vashti Galpin<sup>(✉)</sup>

Laboratory for Foundations of Computer Science School of Informatics,  
University of Edinburgh, Edinburgh, UK  
Vashti.Galpin@ed.ac.uk

**Abstract.** Generation of electricity has traditionally taken place at a small number of power stations but with advances in generating technology, small-scale generation of energy from wind and sun is now possible at individual buildings. Additionally, the integration of information technology into the generation and consumption process provides the notion of smart grid. Formal modelling of these systems allows for an understanding of their dynamic behaviour without building or interacting with actual systems. This paper reports on using a quantitative process algebra HYPE to model a residential smart grid (microgrid) for a spatially-extensive suburb of houses where energy is generated by wind power at each house and where excess energy can be shared with neighbours and between neighbourhoods. Both demand and wind availability are modelled stochastically, and the goal of the modelling is to understand the behaviour of the system under different redistribution policies that use local knowledge with spatial heterogeneity in wind availability.

**Keywords:** Smart grid · Microgrid · Renewable energy · Process algebra · Quantitative modelling · Stochastic hybrid · Collective adaptive system

## 1 Introduction

The way in which electricity is generated is changing. Until recently, there were a few large producers and many consumers (domestic, commercial and industrial). As it becomes cheaper and easier to install equipment that allows one to generate electricity from sun and wind power on individual buildings, more consumers are becoming generators of energy. Furthermore, the introduction of information technology allows for exchange of information. This paper investigates the possibilities that these changes bring to residential areas consisting of standalone or semi-detached houses. Currently, some countries allow excess renewable energy to be fed back into the grid but this is not the only option and it may be possible to share directly between households. If we consider a large suburb consisting of groups of houses, each supplied by one transformer from the grid (called *neighbourhoods* in this paper) then questions arise about the best way in which to share this energy between neighbourhoods. Location then

becomes important and spatial differences in renewable energy can be investigated. A novel formal model of this scenario is developed in this paper and evaluated through simulation.

A smart grid is an example of a collective adaptive system because it consists of different components that interact and it must adapt to changes in the environment in which it operates. Collective adaptive systems are becoming more common in everyday life, and they are often invisible to users and people affected by them. Hence it is crucial that formal methods are used to reason about their behaviour so that we can obtain a good understanding of how they work and how they may fail. Formal methods can be used to reason about functional properties of systems (such as liveness and correctness with respect to a specification or logical property) as well as nonfunctional properties such as performance. This paper focusses on the quantitative behaviour of smart grids taking into account spatial distribution of the system.

Stochastic HYPE is a quantitative process algebra developed to model systems which include continuous evolution of variables, stochastic behaviour and instantaneous jumps [3, 11]. It has been used to model various systems, artificial and biological and combinations of both [2, 8, 9, 12]. In the case of modelling smart grids, continuous modelling is required for calculating the energy consumption from changing energy rates, stochastic modelling is required for natural phenomena such as wind and instantaneous behaviour occurs when policies change due to events (that themselves may be instantaneous such as a change in the price of electricity). The expressiveness of stochastic HYPE allows for different approaches to modelling spatial aspects. It can model both continuous space and logical space (which is the approach taken here). For the current research, analysis of the model is done by simulation, specifically by considering the averages of variable trajectories over multiple simulation runs [2].

The paper is structured as follows. First, residential smart grids are described, and their behaviour quantified in terms of energy flows. Policies for distributing surplus renewable energy between neighbourhoods are described. Next, stochastic HYPE is introduced and the basic model is presented together with the model parameters. Results of simulation are presented and discussed. Related work is assessed and the paper finishes with conclusions and future work.

## 2 Smart Grids

As mentioned above, energy generation is changing and a number of recent factors have led to this change and will cause greater changes in the future. Amongst these factors are concerns about energy scarcity due to finite quantities of fossil fuels (sustainability), public distrust of nuclear power, desire for sustainability, availability of equipment for small scale generation from renewable resources, and integration of information in the electricity network infrastructure which aids decision making in production and consumption. Thus, some who were historically only the consumers have become producers as well (sometimes referred to “prosumers”). Producers wish to produce energy to cover demand and no



more. Consumers on the other hand, want to pay a reasonable cost for their energy. Information can be used by producer and consumer alike to achieve their goals. For example, smart meters allow consumers to understand consumption, and producers can vary prices to shape demand.

An example residential smart grid consists of 4 to 7 houses served by a single transformer that steps down grid power to domestic voltage. Each house also has photovoltaic cells and a wind turbine [18]. Additionally, each house may have a plug-in electric hybrid vehicle (PHEV) which has a battery which is used both to power the vehicle and to store excess renewable energy. Various types of information can be used. For example, information about the current energy price transmitted from the grid can be used to determine how to use the renewable energy being generated [18], and a limit of the number of vehicles charging from the grid during peak times can be enforced [17].

The specific scenario envisaged in this paper involves small groups of houses (each referred to as a neighbourhood) served by a transformer as described above. There is a wind turbine on each house and no local storage. The focus is on energy sharing (as opposed to reselling) within and between these groups of houses. In such a scenario, sharing energy in a fair way between houses within a neighbourhood where the houses have the same turbine and similar wind speeds is straightforward, as there is an easy argument for fairness under an assumption that demand from each house is similar<sup>1</sup>. In terms of infrastructure, distance between houses in a neighbourhood is assumed to be similar and hence no spatial aspects are introduced within a neighbourhood.

Space is introduced when considering sharing between neighbourhoods. This is more complex, distances vary, and assuming infrastructure for sharing energy that is distinct from the grid, it does not make sense to connect each neighbourhood directly to every other neighbourhood but rather to directly connect neighbourhoods. This paper explores different energy sharing policies between neighbourhoods that use knowledge about the local conditions such demand or wind strength.

## 2.1 Quantifying Smart Grids

We consider  $n$  neighbourhoods where the number of houses in neighbourhood  $N_i$  is  $m_i$ . Each house  $H_{ij}$  has  $a_{ij}$  appliances as well as a background energy profile which is deterministic and distinguishes nighttime when residents are asleep (after 11pm and before sunrise), evening (from sunset to 11pm) and daytime (from sunrise to sunset).

For each point in time, the consumption rate (or demand) within a household can be determined and expressed as  $ld_{ij}(t) = b(t) + \sum_{k=1}^{a_{ij}} o_{ijk}(t) \cdot app_{ijk}$  where  $b(t)$  is the background consumption rate which is assumed to be the same across

<sup>1</sup> Nevertheless one household could get a greater share of the renewable energy by ensuring their appliance use is at different times to the other households, and there are other similar actions that some people would consider unfair.

all houses,  $o_{ijk}$  is an indicator of whether the  $k$ th appliance is on in house  $H_{ij}$  and  $app_{ijk}$  is the energy consumption rate of that appliance.

We define  $lr_i$  as the available renewable energy rate for a household in neighbourhood  $N_i$  (assuming that it is the same for every household in a neighbourhood). Three quantities can be calculated for each house.

$$\begin{aligned} \text{Use of local renewable energy: } lru_{ij}(t) &= \min(ld_{ij}(t), lr_i(t)) \\ \text{Local excess demand: } lxd_{ij}(t) &= ld_{ij}(t) - lru_{ij}(t) \\ \text{Local excess renewable energy: } lxr_{ij}(t) &= lr_i(t) - lru_{ij}(t) \end{aligned}$$

Clearly, if  $lxd_{ij}$  is nonzero at time  $t$  then  $lxr_{ij}(t)$  will be zero at that time point, and if  $lxr_{ij}(t)$  is nonzero at time  $t$ ,  $lxd_{ij}$  will be zero.

It is unnecessary to work at the level of individual houses as long as there is an assumption that surplus renewable energy is allocated maximally between neighbours, in the sense that all energy is allocated and there is no wastage with in a neighbourhood (one possibility for maximal allocation is proportionally with demand). The neighbourhood calculation of demand and renewables are  $nd_i(t) = \sum_{j=1}^{m_i} ld_{ij}(t)$  and  $nr_i(t) = m_i \cdot lr_i(t)$  and from this other values can be calculated, similarly to above.

$$\begin{aligned} \text{Use of neighbourhood renewable energy: } nru_i(t) &= \min(nd_i(t), nr_i(t)) \\ \text{Neighbourhood excess demand: } nxd_i(t) &= nd_i(t) - nru_i(t) \\ \text{Neighbourhood excess renewable energy: } nxr_i(t) &= nr_i(t) - nru_i(t) \end{aligned}$$

Owing to the assumption of maximal allocation, we can conclude that  $nxd_{ij}(t) > 0 \Rightarrow nxr_{ij}(t) = 0$  and  $nxr_{ij}(t) > 0 \Rightarrow nxd_{ij}(t) = 0$ . Without description here (it will be covered in the next section) we assume a policy that allocates some of the surplus to each neighbourhood where the allocation is  $f_i$  (there may be wastage). Again we assume maximal allocation between the houses which have excess demand within a neighbourhood and we can determine a neighbourhood-level excess demand after this allocation that must be satisfied from the grid.

$$\begin{aligned} \text{Use of shared renewable energy: } nsu_i(t) &= \min(nxd_i(t), f_i(t)) \\ \text{Use of energy from the grid: } g_i(t) &= nxd_i(t) - nsu_i(t) \\ \text{Wasted renewable energy: } w_i(t) &= f_i(t) - nsu_i(t) \end{aligned}$$

Again, nonzero  $g_i$  implies zero  $w_i$  and *vice versa*. These equations are then the basic mathematical equations for energy, and describe rates at a point in time. To determine actual quantities over time, further calculation must be done. For example,  $g(t)$  expresses the rate of grid consumption at time  $t$ . To determine the overall consumption of energy, we need to solve the ordinary differential equation (ODE)  $dG(t)/dt = g(t)$  to give the quantity  $G(t)$ , the amount of grid energy consumed up to time  $t$ . Furthermore, the calculations do not include explicitly the losses incurred when converting DC current (from wind turbines) to AC current (for appliances and background consumption) and hence the rate of renewable energy obtained from the wind will have these losses deducted. The issue of losses due to conversion are more important in models that include battery storage.

## 2.2 Policies

As mentioned above, the surplus energy from each neighbourhood  $n_{xr_i}$  is distributed to other neighbourhoods, with neighbourhood  $j$  receiving the quantity  $f_j$ . We consider policies where energy is supplied to all neighbourhoods<sup>2</sup>.

There are two groups of calculations needed to determine what each neighbourhood receives, associated with two decision phases. The first decision is to determine how much to supply in each direction. For a 2-dimensional layout, one can envision “waves” of energy being sent in each direction, considering either the four main compass points (von Neumann neighbourhood) or eight compass points (Moore neighbourhood). Each neighbourhood must determine how its excess is divided up and this division results in expressions denoted by  $tr_{iX}$  where  $tr_{iX}$  represents the transfer from  $N_i$  in direction  $X \in \mathcal{C}$  where  $\mathcal{C}$  is the set of compass points of interest. The value of  $tr_{iX}$  is determined from the excess renewable energy from  $N_i$ ,  $n_{xr_i}$ , and some other factors relating to those neighbourhoods immediately adjacent to  $N_i$  for the directions of interest. Clearly, we require that the amount allocated be less than the amount available hence we require that  $\sum_{X \in \mathcal{C}} tr_{iX} \leq n_{xr_i}$  and a choice that is not equality results in immediate wasted renewable energy. Examples of functions that use local knowledge now follow. Let  $h(\mathcal{C})$  be the set of all adjacent neighbourhoods of  $N_i$  in the directions of interest.

**Split equally between adjacent neighbourhoods:**

$$tr_{iX} = n_{xr_i}/|\mathcal{C}| \text{ for } X \in \mathcal{C}$$

**Split proportionally by demand in adjacent neighbourhoods:**

$$tr_{iX} = n_{xr_i} \cdot n_{xd_j} / (\sum_{N_k \in h(\mathcal{C})} n_{xd_k}) \text{ for } X \in \mathcal{C}$$

**Split by relative wind speed in adjacent neighbourhoods:**

Let  $wl_i$  be the number of adjacent neighbourhoods of  $N_i$  that have wind speed less than  $wind_i$  then, for  $X \in \mathcal{C}$

$$tr_{iX} = \begin{cases} n_{xr_i}/wl_i & wl_i > 0 \wedge wind_j < wind_i \text{ for } N_j \text{ in direction } X \\ 0 & wl_i > 0 \wedge wind_j \geq wind_i \text{ for } N_j \text{ in direction } X \\ n_{xr_i}/|\mathcal{C}| & otherwise \end{cases}$$

The second decision is about what energy each neighbourhood receives. To understand this decision, consider energy being supplied from west to east (or left to right). The leftmost neighbourhood can only pass on its surplus (if any). Then for each neighbourhood as one moves eastward, there are two options; either contribute surplus energy to the supply (if there is excess renewable energy) or to consume some portion of the energy that has arrived (if there is excess demand). The maximum amount that can be consumed is determined

<sup>2</sup> One can consider policies where energy is only supplied to adjacent neighbourhoods but this necessarily seems to result in lower use of renewable energy because fewer neighbourhoods can receive excess renewable energy.

by the excess demand for that neighbourhood, the amount available (the *available* energy), any policy restriction on the amount that can be consumed (giving the *allocated* energy) and what is actually consumed (giving the *actual* energy). In the example that appears later, we will associate with each neighbourhood, a percentage which will describe the proportion of the excess demand that can be satisfied and we will illustrate two ways in which this percentage can be chosen.

Expressions of the form  $tr_{X_i}$  describe the energy from each relevant direction to each neighbourhood. It is not correct to consider the sum of the  $tr_{iX}$  and divide it up because there is directionality and it is necessary to consider the amount available for each neighbourhood, as described above. Assuming a strip of neighbours  $N_1$  to  $N_n$  from west to east,  $N_1$  transfers all its excess energy for the east, hence the amount available at  $N_2$  is  $av_{X_2}(t) = tr_{1X}(t)$ . Assuming that the amount allocated for consumption at  $N_2$  is  $all_{X_2}(t)$  then  $tr_{X_2}(t) = \min(av_{X_2}(t), all_{X_2}(t))$  is the actual amount allocated, and the amount available for  $N_3$  is  $av_{X_3}(t) = av_{X_2} - tr_{X_2}(t) + tr_{2X}(t)$  where one or more of  $tr_{X_2}(t)$  and  $tr_{2X}(t)$  are zero since if energy is required there will have been no excess to pass on, and if there is energy to pass on, there can be no demand. The general definition is then as follows.

$$av_{X_i}(t) = \begin{cases} 0 & i = 1 \\ av_{X_{(i-1)}}(t) - tr_{X_{(i-1)}}(t) + tr_{(i-1)X}(t) & \text{otherwise} \end{cases}$$

$$tr_{X_i}(t) = \begin{cases} av_{X_n}(t) & i = n \\ \min(av_{X_i}(t), all_{X_i}(t)) & \text{otherwise} \end{cases}$$

To keep track of wastage, the last neighbourhood receives all remaining energy regardless of any allocation. For the other directions, the expressions are defined similarly. The total of allocated energy for neighbourhood  $N_i$  is defined by  $f_i = \sum_{X \in C} tr_{X_i}$ . Different allocation policies to determine  $all_{X_i}$  can be used, and the simplest is to allocate the same amount as the demand. A slightly more complex policy is to allocate a proportion of the demand, and other more complex (but not necessarily better) policies can be developed.

### 3 Stochastic HYPE Model

Space limitations prevent an introduction to stochastic HYPE and the reader is referred to [3, 11]. In this section, a flavour of the model is given and a very brief introduction to the semantics of stochastic HYPE are presented. Subcomponents (see Table 1) define the flows that describe the continuous behaviour of the system and they react to events that may change these flows. Underlined events are instantaneous in nature and occur when a Boolean expression (activation condition or guard) becomes true. Overlined events are stochastic and complete after an exponential duration. Each subcomponent must be able respond to the first event init to ensure that the initial behaviour of the subcomponent is defined. The characteristics of a flow are described by the influence triples. The

**Table 1.** Subcomponents, influence mapping, variables, event conditions, controllers/sequencers and the overall system.  $GC$  is a variable which records the current cost of energy from the grid.

$$\begin{aligned}
 Time &= \underline{\text{init}}:(\iota_t, 1, 1).Time \\
 App_{ijk} &= \underline{\text{off}}_{ijk}:(\iota_{ijk}^a, 0, 0).App_{ijk} + \underline{\text{on}}_{ijk}:(\iota_{ijk}^a, app_{ijk}, 1).App_{ijk} + \\
 &\quad \underline{\text{init}}:(\iota_{ijk}^a, 0, 0).App_{ijk} \\
 Back_{ij} &= \underline{\text{night}}:(\iota_{ij}^b, r_n, 1).Back_{ij} + \underline{\text{evening}}:(\iota_{ij}^b, r_e, 1).Back_{ij} + \underline{\text{day}}:(\iota_{ij}^b, r_d, 1).Back_{ij} + \\
 &\quad \underline{\text{init}}:(\iota_{ij}^b, r_n, 1).Back_{ij} \\
 Grid_i &= \underline{\text{init}}:(\iota_i^g, 1, g_i) \quad Shared_i = \underline{\text{init}}:(\iota_i^s, 1, nsu_i) \quad Waste_i = \underline{\text{init}}:(\iota_i^w, 1, w_i) \\
 Cost_i &= \underline{\text{init}}:(\iota_i^k, GC, g_i) \quad Renew_i = \underline{\text{init}}:(\iota_i^r, 1, n\text{rx}_i + nsu_i)
 \end{aligned}$$

$$\begin{aligned}
 iv(\iota_t) &= T & iv(\iota_i^w) &= W & iv(\iota_i^r) &= E & iv(\iota_{ijk}^a) &= iv(\iota_{ij}^b) = D_i \\
 iv(\iota_i^s) &= S_i & iv(\iota_i) &= R_i & iv(\iota_i^g) &= G_i & iv(\iota_i^c) &= C_i
 \end{aligned}$$

$T$  time  
 $W$  total wastage  
 $E$  total renewable energy generated  
 $D_i$  total demand in neighbourhood  $N_i$   
 $S_i$  shared renewable energy usage in neighbourhood  $N_i$   
 $R_i$  total renewable energy usage in neighbourhood  $N_i$   
 $G_i$  grid energy usage in neighbourhood  $N_i$   
 $C_i$  cost of grid energy in neighbourhood  $N_i$

$$\begin{aligned}
 ec(\underline{\text{init}}) &\stackrel{\text{def}}{=} (true, (T' = 0) \wedge (W' = 0) \wedge (P' = 0) \wedge \dots) \\
 ec(\underline{\text{on}}_{ijk}) &\stackrel{\text{def}}{=} (T = T_{on_{ijk}}, T'_{c_{ijk}} = T) & ec(\underline{\text{off}}_{ijk}) &\stackrel{\text{def}}{=} (T = T_{c_{ijk}} + T_{d_{ijk}}, true) \\
 ec(\underline{\text{day}}) &\stackrel{\text{def}}{=} (T \bmod 24 = 6, true) & ec(\underline{\text{peak}}_d) &\stackrel{\text{def}}{=} (T \bmod 24 = 7, GC' = gc_p) \\
 ec(\underline{\text{evening}}) &\stackrel{\text{def}}{=} (T \bmod 24 = 18, true) & ec(\underline{\text{midpeak}}_d) &\stackrel{\text{def}}{=} (T \bmod 24 = 10, GC' = gc_{mp}) \\
 ec(\underline{\text{night}}) &\stackrel{\text{def}}{=} (T \bmod 24 = 22, true) & ec(\underline{\text{peak}}_e) &\stackrel{\text{def}}{=} (T \bmod 24 = 17, GC' = gc_p) \\
 ec(\underline{\text{blow}}) &\stackrel{\text{def}}{=} (r_{blow}, WB' = 1) & ec(\underline{\text{midpeak}}_e) &\stackrel{\text{def}}{=} (T \bmod 24 = 20, GC' = gc_{mp}) \\
 ec(\underline{\text{noblow}}) &\stackrel{\text{def}}{=} (r_{noblow}, WB' = 0) & ec(\underline{\text{offpeak}}) &\stackrel{\text{def}}{=} (T \bmod 24 = 23, GC' = gc_{op})
 \end{aligned}$$

$$\begin{aligned}
 CApp_{ijk} &\stackrel{\text{def}}{=} \underline{\text{on}}_{ijk}.\underline{\text{off}}_{ijk}.CApp_{ijk} \\
 SBack &\stackrel{\text{def}}{=} \underline{\text{day}}.\underline{\text{evening}}.\underline{\text{night}}.SBack \\
 SWind &\stackrel{\text{def}}{=} \underline{\text{blow}}.\underline{\text{noblow}}.SWind \\
 SPeak &\stackrel{\text{def}}{=} \underline{\text{peak}}_d.\underline{\text{midpeak}}_d.\underline{\text{peak}}_e.\underline{\text{midpeak}}_e.\underline{\text{offpeak}}.SPeak
 \end{aligned}$$

$$\begin{aligned}
 RSG &\stackrel{\text{def}}{=} \Sigma \boxtimes_* \underline{\text{init}}.Con \\
 \Sigma &\stackrel{\text{def}}{=} Time \boxtimes_* (\dots \boxtimes_* App_{ijk} \boxtimes_* \dots) \boxtimes_* (\dots \boxtimes_* Back_{ij} \boxtimes_* \dots) \boxtimes_* (\dots \boxtimes_* \\
 &\quad (Grid_i \boxtimes_* Cost_i \boxtimes_* Waste_i \boxtimes_* Energy_i \boxtimes_* Renew_i \boxtimes_* Shared_i) \boxtimes_* \dots) \\
 Con &\stackrel{\text{def}}{=} (\dots \boxtimes_* CApp_{ijk} \boxtimes_* \dots) \boxtimes_* SBack \boxtimes_* SWind \boxtimes_* SPeak
 \end{aligned}$$

first element of a triple is the influence name which identifies the variable that is affected by the influences in this subcomponent through the mapping  $iv$ , as shown in Table 1. The second and third components describe the flow, with the second component representing the strength of the flow as a constant value and

the third element describing a function that introduces variables in the flow definition. A feature of stochastic HYPE is that it allows for multiple flows to affect a single variable. In the example, this is illustrated by the fact that multiple influences are mapped to the variable  $D_i$ , allowing for multiple appliances and the background level of consumption to determine the ODE that describes the value of  $D_i$  over time. The second and third elements are multiplied together in the ODE that is generated for a variable, so their separation in the model is purely a syntactic distinction.

Table 1 also lists event conditions for the model. These consist of an activation condition (in the case of instantaneous events) and a reset of variable values. As is standard in stochastic HYPE models, the `init` event initialises all variables and has the activation condition `true`. It is the first event that happens because of the structure of the overall system *RSG*.

Events that turn appliances on and off are required. The value of  $T_{on_{ijk}}$  is set at the start of the day using a distribution that describes the probability of that type of appliance starting at a particular hour of the day, and a random number of minutes (uniformly chosen from the interval  $[0, 60)$ ). The duration  $T_{d_{ijk}}$  is a fixed value for the type of appliance. There are other events that are dependent on time. There are a number of approaches that could be considered for modelling wind: a constant wind, a wind defined by a stochastic differential equation as in [22] and a stochastic wind that may be present (at a fixed strength) or absent. The latter is most appropriate here since goal of the model is to consider energy sharing, hence the renewable energy provided by the wind may be present or absent. This are determined by two stochastic events `blow` and `noblow` where the first element of the event condition is the rate. The reset determines the value of the variable  $WB$  which in turn determines  $lr_i$  for each neighbourhood.

We also require some controllers and sequencers and these are given in Table 1 with the full model. Both the *SBack* and *SPEak* are somewhat redundant because of the time-based sequencing in the event conditions, however stochastic HYPE requires that all events should appear in controllers, and this explicitness expresses the intent of the model. However, the other controller definitions are required to ensure the correct alternation of events. The semantics of a stochastic HYPE model are defined via structured operational semantics that define a labelled transition system. In this labelled transition system, a function  $\sigma$  is required to record the current values associated with each influence name, hence the operational semantics are defined over pairs  $\langle P, \sigma \rangle$ . The labelled transition system can then be mapped to transition-driven stochastic hybrid automata [5], a subset of piecewise deterministic Markov processes [6]. The states of the labelled transition system become the modes of the stochastic hybrid automata, and the ODEs for a mode are defined in terms of the values associated with each influence name in that mode.

To illustrate the ODEs that are obtained from the model above, the ODE that defines the demand for neighbourhood  $N_i$  is as follows in the case where there are five houses in  $N_i$ , one appliance is on in the third house, two appliances are on in the fifth house, and it is before 07:00 in the morning, then the equation

is  $dD_i/dt = 5r_n + app_{i31} + app_{i51} + app_{i52}$ . For the total waste, we have the ODE  $dW/dt = \sum_{i=1}^n waste_i$  and for total renewable energy generated, we have  $dE/dt = \sum_{i=1}^n grid_i$ .

The transition-driven stochastic hybrid automaton has the following structure and associated behaviour.

- Modes and their associated ODEs describe how variable values change.
- Continuous evolution of variable values are determined by the current mode.
- Switching between modes occurs when
  1. activation conditions (guards) of instantaneous events become true
  2. durations of stochastic events expire
 with possible jumps in variable values determined by resets.

A trace of an automaton consists of a continuous trajectory for each variable interspersed with non-continuous changes in values. The behaviour of stochastic HYPE models can be explored using the stochastic hybrid simulator described in [2] and this simulator was used for the results reported here.

### 3.1 Model Parameters

**Appliances:** There are two per house, one washing machine (consumption 0.82 kWh, cycle length 1 h) and one dishwasher (consumption 2.46 kWh, cycle length 1 and a half hours) [18]. Distributions for the probability of being on in a specific hour are used to determine the starting hour of the appliance [18].

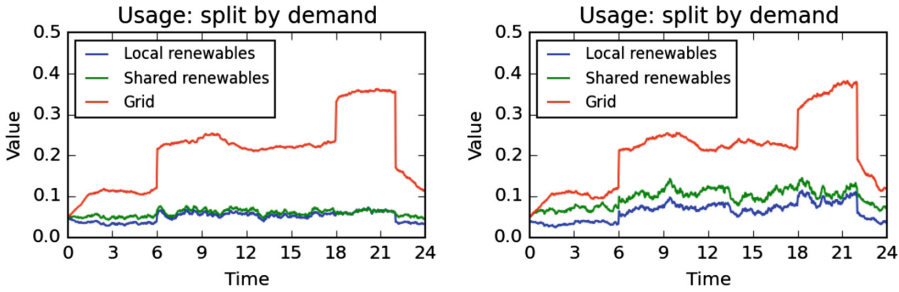
**Background Consumption:** Using the figures from Fig. 10 in [25] as a guide, the daytime figure is 0.3 kWh, the evening figure is 0.5 kWh and the nighttime figure is 0.1 kWh.

**Wind:** It has been argued that on average in the UK, at any specific point, the probability of there being wind sufficiently strong to drive a turbine is 80% [21]. As mentioned above, to explore the issue of sharing renewable energy, it is necessary for the wind to be stochastic, and two exponential distributions are used, one for wind presence and one for wind absence. We consider various possibilities including that from [21]. The average generation capacity of the wind in the UK has been calculated to be somewhere between 25% and 35% [21]. This means a turbine with a rating of  $x$  kWh will give that percent of its rated power.

**Electricity Cost:** Here we follow [18], so at peak times, the cost is 0.272 £/kWh, mid-peak cost is 0.194 £/kWh, and off-peak is 0.107 £/kWh.

## 4 Results

A number of different experiments were considered. The two main wind patterns that were investigated are as follows.



**Fig. 1.** Stacked graphs of the average instantaneous local renewable usage, shared renewable usage and grid usage over one day (right: one wind scenario, left: two wind scenario).

**One wind scenario:** Seven neighbourhoods in a strip were considered, with  $N_1$  to the west and  $N_7$  to the east.  $N_1$  and  $N_2$  had the full strength of the wind,  $N_3$  and  $N_4$  had half strength wind, and  $N_5$ ,  $N_6$  and  $N_7$  had quarter strength wind.

**Two wind scenario:** This had the same strip layout with no wind in the central neighbourhood  $N_4$  and one wind at full strength in  $N_1$ , half strength in  $N_2$  and a quarter strength in  $N_3$ . The second wind was available in  $N_7$  at full strength,  $N_6$  at half strength and  $N_5$  with quarter strength.

A number of policies were investigated and are described by the following abbreviations.

**eq100** Split equally in each direction, allowing 100% of demand to be satisfied

**wn100** Split by relative wind speed, allowing 100% of demand to be satisfied

**dm100** Split proportionally by demand, allowing 100% of demand to be satisfied

**dminc** Split proportionally by demand, with proportion of demand to be satisfied increasing in the direction of supply

**dmwnd** Split proportionally by demand, with proportion of demand to be satisfied determined by wind level.

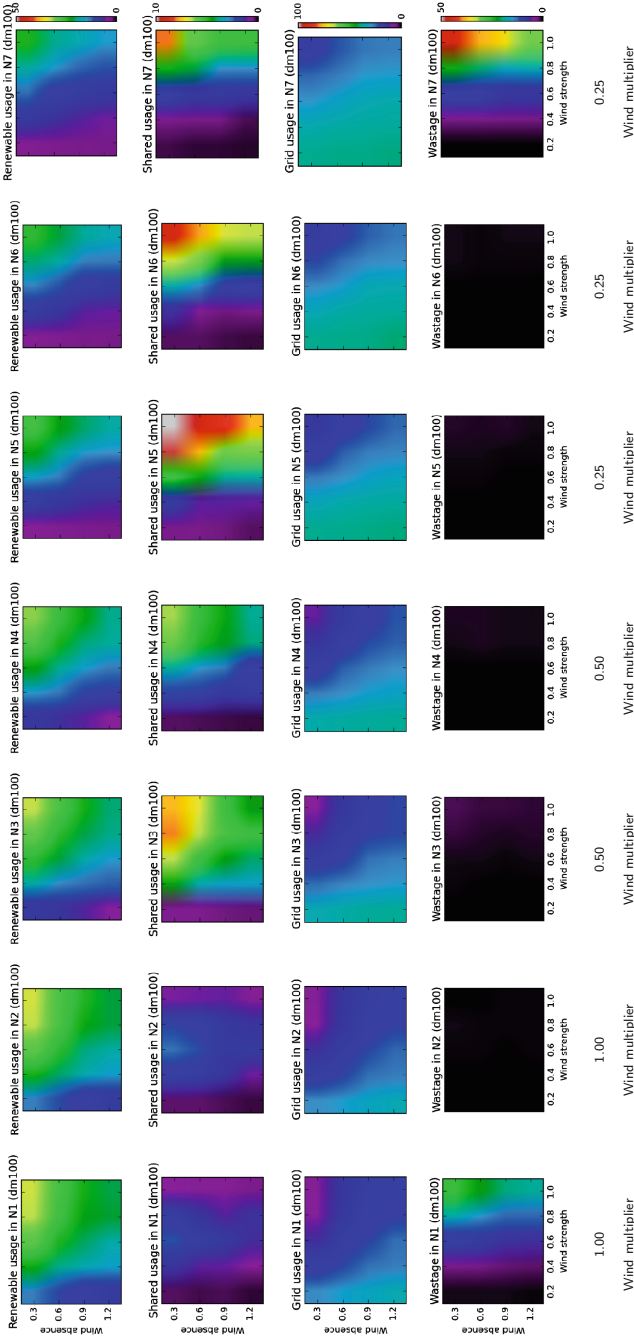
**dw100** Split proportionally by demand with a weighting factor to favour higher demand, allowing 100% of demand to be satisfied

**da100** Direction of highest demand receives all excess, allowing 100% of demand to be satisfied

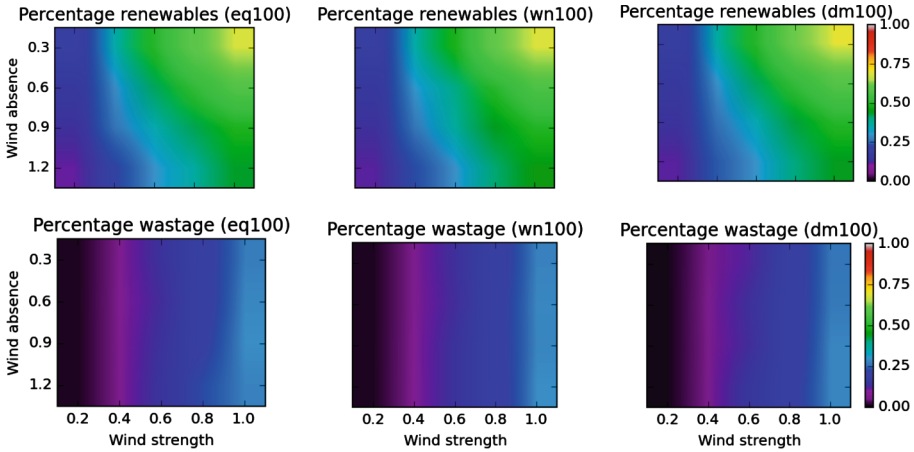
Figure 1 shows the instantaneous energy consumption across the day (as a stacked graph). There is a peak in the early evening capturing the higher background level at that time and the higher likelihood of appliance being used. There is more use of shared renewables at night because there is a greater likelihood of excess renewables due to lower consumption then. The right hand graph shows that there is more scope for sharing of renewables in the two wind scenario.

Figure 2 provides heatmaps across the neighbourhoods of the parameter space for wind strength and wind absence (under the dm100 policy in the one wind





**Fig. 2.** Heatmaps for renewable usage, shared usage, grid usage and wastage across neighbourhoods for different maximum wind strengths and wind absence rate (one wind scenario over 24h) for policy dm100. As the wind decreases over neighbourhoods from left to right, renewable usage decreases and grid usage increases. Shared energy increase is dependent on the wind multiplier of adjacent neighbourhoods. Within a neighbourhood, renewable usage increases and grid usage decreases as the wind absence rate decreases and the wind strength increases. Shared usage shows the same pattern except in  $N_1$  and  $N_2$  where shared usage is very low. Wastage only shows this pattern in  $N_1$  and  $N_7$  as this is where unused renewable energy collects.  $N_7$  has higher wastage because the demand driven policy is likely to allocate more shared energy to the neighbourhoods with lower wind.



**Fig. 3.** Heatmaps for percentage of energy used that is renewable (out of total energy consumed) and percentage renewable energy wastage (out of total renewable available) for different maximum wind strengths and wind absence rate (one wind scenario over 24 h) considering different policies.

scenario). The wind strength varies from 0.2 to 1 (and is adjusted by the wind multiplier for the region). The average wind presence rate is 1.2 h and the average wind absence varies from 0.3 h (in line with the 80 % presence of [21]) to 1.2 h (giving a 50 % presence). The heat maps show how the figures vary across regions and across parameters. Most wastage occurs at the extreme neighbourhoods since these are supplied with all excess energy not yet allocated.

Heatmaps can also be used to compare policies and Fig. 3 shows different heatmaps for three policies in the one wind scenario. Note that percentage wastage appears to only depend on windspeed and be independent of average wind absence. This occurs because wastage only occurs when the wind is present, and hence variations in how long the wind is absence have no effect.

For all policies in the one wind scenario, there are no obvious differences, and thus it appears that using different types of local knowledge have no impact. These policies were also investigated for a four by four grid of neighbourhoods with wind multipliers that decreased from the north-west corner to the south-east corner, and again no major differences were found. This suggests that in the case of a single wind at the strengths and absences investigated, the different policies do not make a major difference in how much energy is supplied to other neighbourhoods and hence no difference is seen. This can be explained by the fact that most wastage occurs at night when there is low background usage and a lower chance of appliance use (as illustrated by Fig. 1). Thus when there is high demand, all available renewable energy is used, regardless of policy; and when there is low demand, there is sufficient energy to allocate it all (again regardless of policy). This could be investigated further by reducing the amount of renewable energy to much lower levels so that all is needed and hence differences in policies

**Table 2.** Cost per day for different policies across neighbourhoods with mean and variance, grid consumption, percentage waste and percentage renewable use (two wind scenario over 24h), ordered by average cost.

	$N_1$	$N_2$	$N_3$	$N_4$	$N_5$	$N_6$	$N_7$	Mean	Variance	Grid	W%	R%
da100	1.09	1.16	1.19	1.46	1.18	1.13	1.11	1.19	0.0130	159.3	15.9 %	47.4 %
wn100	1.11	1.14	1.22	1.37	1.21	1.16	1.16	1.20	0.0064	158.4	16.2 %	47.5 %
dw100	1.10	1.14	1.22	1.43	1.20	1.13	1.15	1.20	0.0110	158.6	17.1 %	47.6 %
eq100	1.15	1.13	1.25	1.44	1.22	1.13	1.13	1.21	0.0111	160.9	18.6 %	46.7 %
dm100	1.13	1.15	1.28	1.47	1.20	1.19	1.13	1.22	0.0129	163.7	16.8 %	45.9 %
dmdec	1.07	1.21	1.31	1.48	1.29	1.17	1.06	1.23	0.0192	165.0	19.2 %	45.3 %
dmdwn	1.07	1.30	1.32	1.30	1.32	1.28	1.10	1.24	0.0101	165.6	19.6 %	45.2 %

may be demonstrated. Furthermore, introducing the ability to store renewable energy and use it later might lead to more significant differences in policies.

Note that all policies described above lead to a greater use of renewable energy and lower grid usage when compared to the situation where no energy is shared between neighbourhoods. On average, the amount of renewable energy consumed (as a percentage of total energy demand) increases to 70 % from 55 % when neighbourhood sharing is introduced, and the wastage of renewable energy (as a percentage of all renewable energy produced) drops from 57 % to 27 % through sharing between neighbourhoods.

Next, we consider the two wind scenario and the various policies described above. The results are shown in Table 2. The first nine columns of this table considers the cost of electricity per day. Although the differences are not large, these can accumulate over a year. The results suggest that using a more extreme policy where knowledge of demand is used to allocate all surplus in one direction or the other leads to the lowest average cost per day. However, this policy does not lead to the lowest variance in cost suggesting that is it not as fair as the wind based policy which does have the lowest variance. The policies that allocate less than 100 % of demand, dmdec and dmdwn, seem to be poor in terms of average cost, although dmdwn leads to a low variance. Note however, that these results are specific to a particular scenario and hence cannot be generalised without further experimentation.

## 5 Related Work

A recent survey of modelling smart grids considers smart grids as complex systems with emergent behaviour and identifies multi-agent systems as an existing modelling tool [13]. In [20], a multi-agent systems approach is taken to controlling a smart grid and in [15], agents act as elements of the smartgrid. There are limitations to what can be achieved using this method, because the size of models is limited computationally. Pretopology and percolation theory applied to complex systems may be able to successfully model large smart grids [13].

Modelling of smart grids has been done at two main levels: either very detailed, focussing on modelling the electrical components such as wind turbines and inverters in terms of their performance [14, 19, 26], or at the level of a group of houses with various sources of renewable energy [7, 10, 17, 18, 22]. Both of these levels differ from the approach taken here where the focus is on spatial aspects of redistribution of renewable energy.

Fine-grained models of residential consumption have been developed [1, 16, 24, 25]. The goal is to build up realistic profiles of consumption using various data sources about individual human behaviour and from these models estimate demand. To ensure the stochastic HYPE model presented here has a reasonable size, this level of detail has not been used. However, the general profile generated in [25] has been used to provide parameters for the stochastic HYPE model as mentioned in Sect. 3.1.

## 6 Conclusions and Further Work

To conclude, a stochastic hybrid model has been developed of smart grid generation of power to consider spatial aspects of sharing between neighbourhoods. Different knowledge-based policies for splitting surplus renewable energy appear to have little effect when there is only one wind but the two wind scenario does lead to differences. In general, sharing of energy significantly increases the amount of renewable energy used and reduces costs. Further exploration of wind patterns and the effect on policies is warranted as it is not possible to generalise from a single pattern.

This is ongoing research as part of a project on quantified modelling of collective adaptive systems (see [www.quanticol.eu](http://www.quanticol.eu)). In this paper, we have developed a model at an appropriate level for reasoning about distribution of energy throughout a suburb which can be explored through simulation. However, simulation is an expensive technique, and we wish to develop scalable approximation techniques that allow us to reason about these systems, similar to various fluid and mean-field techniques such as [4, 23].

**Acknowledgements.** This work is supported by the EU project QUANTICOL, 600708.

## References

1. Ardakanian, O., Keshav, S., Rosenberg, C.: Markovian models for home electricity consumption. In: Second ACM SIGCOMM Workshop on Green Networking, pp. 31–36. ACM (2011)
2. Bortolussi, L., Galpin, V., Hillston, J.: Hybrid performance modelling of opportunistic networks. In: QAPL 2012, EPTCS 85, pp. 106–121 (2012)
3. Bortolussi, L., Galpin, V., Hillston, J.: Stochastic HYPE: flow-based modelling of stochastic hybrid systems. CoRR abs/1411.4433 (2014)

4. Bortolussi, L., Hillston, J., Latella, D., Massink, M.: Continuous approximation of collective systems behaviour: a tutorial. *Perform. Eval.* **70**, 317–349 (2013)
5. Bortolussi, L., Policriti, A.: Hybrid dynamics of stochastic programs. *Theor. Comput. Sci.* **411**, 2052–2077 (2010)
6. Davis, M.: *Markov Models and Optimization*. Chapman & Hall, London (1993)
7. Delamare, J., Bitachon, B., Peng, Z., Wang, Y., Haverkort, B., Jongerden, M.: Development of a smart grid simulation environment. In: *UKPEW 2014* (2014)
8. Feng, C.: Patch-based hybrid modelling of spatially distributed systems by using stochastic HYPE - ZebraNet as an example. In: *QAPL 2014, EPTCS 154* (2014)
9. Galpin, V.: Modelling a circadian clock with HYPE. In: *PASTA 2010*, pp. 92–98 (2010)
10. Galpin, V.: Modelling residential smart energy schemes. In: *FoCAS@SASO14* (2014)
11. Galpin, V., Bortolussi, L., Hillston, J.: HYPE: hybrid modelling by composition of flows. *Formal Aspects Comput.* **25**, 503–541 (2013)
12. Galpin, V., Hillston, J., Bortolussi, L.: HYPE applied to the modelling of hybrid biological systems. *ENTCS* **218**, 33–51 (2008)
13. Guérard, G., Ben Amor, S., Bui, A.: Survey on smart grid modelling. *Int. J. Syst. Control Commun.* **4**, 262–279 (2012)
14. Kariniotakis, G., Soutanis, N., Tsouchnikas, A., Papathanasiou, S., Hatziaargyriou, N.: Dynamic modeling of microgrids. In: *International Conference on Future Power Systems*, p. 7. IEEE (2005)
15. Kremers, E., Viejo, P., Barambones, O., Gonzalez de Durana, J.: A complex systems modelling approach for decentralised simulation of electrical microgrids. In: *ICECCS 2010*, pp. 302–311. IEEE (2010)
16. McQueen, D., Hyland, P., Watson, S.: Monte Carlo simulation of residential electricity demand for forecasting maximum demand on distribution networks. *IEEE Trans. Power Syst.* **19**, 1685–1689 (2004)
17. Oviedo, R.M., Fan, Z., Gormus, S., Kulkarni, P.: A residential PHEV load coordination mechanism with renewable sources in smart grids. *Int. J. Electr. Power Energy Syst.* **55**, 511–521 (2014)
18. Oviedo, R., Fan, Z., Gormus, S., Kulkarni, P., Kaleshi, D.: Residential energy demand management in smart grids. *IEEE PES T&D* **2012**, 1–8 (2012)
19. Panigrahi, T., Saha, A., Chowdhury, S., Chowdhury, S., Chakraborty, N., Song, Y., Byabortta, S.: A Simulink-based microgrid modelling & operational analysis using distributed generators. In: *UPEC 2006*, pp. 222–226. IEEE (2006)
20. Pipattanasomporn, M., Feroze, H., Rahman, S.: Multi-agent systems in a distributed smart grid: design and implementation. In: *IEEE/PES PSCE 2009*, pp. 1–8. IEEE (2009)
21. Sinden, G.: Characteristics of the UK wind resource: long-term patterns and relationship to electricity demand. *Energy Policy* **35**, 112–127 (2007)
22. Štřelec, M., Macek, K., Abate, A.: Modeling and simulation of a microgrid as a stochastic hybrid system. In: *IEEE PES ISGT Europe*, pp. 1–9. IEEE (2012)
23. Tribastone, M., Gilmore, S., Hillston, J.: Scalable differential analysis of process algebra models. *IEEE Trans. Softw. Eng.* **38**, 205–219 (2012)
24. Widén, J., Wäckelgård, E.: A high-resolution stochastic model of domestic activity patterns and electricity demand. *Appl. Energy* **87**, 1880–1892 (2010)
25. Yao, R., Steemers, K.: A method of formulating energy load profile for domestic buildings in the UK. *Energy Buildings* **37**, 663–671 (2005)
26. Yubing, D., Yulei, G., Qingmin, L., Hui, W.: Modelling and simulation of the microsources within a microgrid. In: *ICEMS 2008*, pp. 2667–2671. IEEE (2008)

# Attributed Probabilistic P Systems and Their Application to the Modelling of Social Interactions in Primates

Roberto Barbuti<sup>(✉)</sup>, Alessandro Bompadre, Pasquale Bove, Paolo Milazzo,  
and Giovanni Pardini

Dipartimento di Informatica, Università di Pisa,  
Largo B. Pontecorvo 3, 56127 Pisa, Italy  
{barbuti,bovepas,milazzo,pardinig}@di.unipi.it,  
alessandro.bompadre@gmail.com

**Abstract.** We propose a variant of probabilistic P Systems, Attributed Probabilistic P systems (APP systems), in which objects are annotated with attributes. We use APP systems for modelling social behaviours of some species of primates. In this context attributes can represent position of the animals in the environment, age of the animal, dominance level, aggressiveness, etc. As in standard P systems, the dynamics of the system is described by multiset rewrite rules that are applied in a maximally parallel way. Probabilities of rule application, in a maximal step, are computed according to weight functions associated to rules. As an application, we develop models to compare despotic and egalitarian behaviours on different species of primates.

## 1 Introduction

P systems [29] were introduced as distributed parallel computing devices inspired by the structure and the functioning of a living cell. A P system consists of a hierarchy of membranes, each of them containing a multiset of objects, representing molecules, a set of evolution rules, representing chemical reactions, and possibly other membranes. For each evolution rule there are two multisets of objects, describing the reactants and the products of the chemical reaction. Evolution rules can be applied more than once to different objects, with maximal parallelism, namely it cannot happen that some evolution rule is not applied when the objects needed for its triggering are available and not consumed by the application of any other rule.

Many variants of P systems exist that include features to increase their expressiveness or which are based on different evolution strategies [30]. In this paper we define a variant of P systems (Attributed Probabilistic P systems, APP systems) which includes features for the description of population dynamics. APP systems are actually the extension of Minimal Probabilistic P Systems [2] with the possibility of enriching objects with attributes. In the modelling of populations, such attributes can be used for describing characteristics of the individuals (position, age, etc.). APP systems include also the probabilistic choice

of the rules to be applied in each maximally parallel step, and the possibility to include rule promoters to enable/disable rules in different phases of evolution.

We show how APP systems can be used for describing real ecological systems, such as self-organizing populations of animals. Such models have been successfully applied for understanding the behaviour of schools of fishes or flocks of birds [14, 19]. Modelling social interactions in primates is an interesting research field which has been usually tackled by means of agent-based models [17, 18, 20, 21, 25, 26, 34]. Such models have been often criticized because, in many cases, they were so poorly documented that the models could not be evaluated. For these reasons protocols have been defined for creating a standard structure by which all the agent-based models could be documented [15, 16]. Such protocols document a model by providing a check list of questions which must be answered by the authors of the model. For example, questions of that check list include: “Who (i.e. what entity) does what, and in what order?”, “When are state variables updated?”, “What kind of entities are in the model?”, “By what state variables, or attributes, are those entities characterized?”.

We present an APP system model of social interactions in primates. The model is composed of a few unambiguous rules. Such rules can be seen as an implementation of the rules used in agent-based models which, however, are usually programmed “ad hoc” and their effect needs to be documented separately. On the contrary, the rules of our APP-system-based model are almost self-explanatory, showing the ease of use of the formalisms for modelling real-world systems. Simulations of our model are also presented. We show that the obtained results are compatible with the results of agent-based models described in the literature.

From a theoretical point of view, the computational expressiveness of APP systems is the same as that of P systems, since in principle each attributed object could be replaced with a new symbol by embedding the values of the attributes in the symbol itself (thus having one symbol for each combination of attribute values). Nevertheless, the use of APP systems for modelling real systems provides important advantages, mainly with respect to readability of the models, and the compact and unambiguous descriptions which can be obtained.

As related work we mention studies in which formal notations are used to model and simulate population dynamics and ecosystems. In [32, 33] a process algebra is proposed and used to model population dynamics by taking spatial distribution of the individuals into account. In [11] the Bio-PEPA process algebra is used to describe epidemiological problems, again by including a notion of spatiality. In [9] a variant of P systems is used to model the dynamics of some endangered species in the Pyrenees. In [23, 24] the BlenX and LIME languages are used to compositionally construct models of ecosystems. In [5, 6] Spatial P Systems are proposed and used to model the schooling behaviour of fish. In [12, 31] formal notations are used to model and simulate the population dynamics of *Solea solea* in the Adriatic Sea by taking the effect of fishing into account.

The feature that mainly makes a difference between APP systems and other proposals is the use of maximal parallelism for the application of rules. This

feature is particularly suitable for the modelling of populations that evolve by stages (e.g. reproductive stages or stages related with seasons). The usefulness of maximal parallelism in the context of ecosystems modelling is confirmed by the recent proposal of S-PALPS [36], an extension of PALPS that incorporates maximal parallelism by means of a synchronous parallel composition operator. The difference between S-PALPS and APP systems is in the modelling approaches, which are process algebraic and rewrite-based, respectively.

As regards the use of attributes, related works are [8, 13, 22]. The first proposes an extension of the  $\pi$ -calculus process algebra with attributed processes and attribute-dependent synchronization. The second defines Stochastic Concurrent Constraint Programming, in which the constraint programming constructs can be used to perform operations similar to those that can be done on attributes. The third proposes an extension of the Kappa language with annotations representing geometric information.

As regards spatial formalisms we mention also Spatial CLS [4] and Grid Systems [3, 35]. In particular, the latter is a rich extension of P systems aimed at the modelling of ecosystems with a focus on population dynamics driven by properties of the environment and environmental events.

The paper is organized as follows: Sect. 2 introduces the Attributed Probabilistic P systems; Sect. 3 presents a model of social behaviour in primates; Sect. 4 shows the results of the experiments; Sect. 5 concludes the paper.

## 2 Attributed Probabilistic P Systems

We denote with  $\{a_1, \dots, a_n\}$  the set of objects  $a_1, \dots, a_n$ , and with  $\{\{a_1, \dots, a_n\}\}$  the multiset of objects  $a_1, \dots, a_n$ . Moreover, we denote with  $|w|$  the size (number of elements) of the multiset  $w$ , and with  $-$  and  $+$  the difference and the union of multisets, respectively.

**Definition 1 (APP System).** *An Attributed Probabilistic P system,  ${}^aP$ , is a tuple  $\langle A, \text{arity}, D_{a_1}, \dots, D_{a_n}, w_0, R \rangle$  where:*

- $A$  is an ordered finite alphabet of symbols,  $\{a_1, \dots, a_n\}$ ;
- $\text{arity} : A \rightarrow \mathbb{N}$  is a function which for each  $a_i \in A$  gives the arity of  $D_{a_i}$ ;
- each  $D_{a_i}$  is a set of tuples,  $D_{a_i} = I_1 \times \dots \times I_{\text{arity}(a_i)}$ , where each  $I_j$  is a (possibly infinite) set of unstructured values; the set  $D_{a_i}$  is called the set of attributes of  $a_i$ ;
- $w_0$  is a multiset of values in  $\Sigma = \{\langle a_i, d_i \rangle \mid a_i \in A, d_i \in D_{a_i}\}$  describing the initial state of the system, where  $\Sigma$  is called the set of objects of  $P$ . In the following we will write  $w_0 \in \Sigma^*$ .
- given a set of variables  $V$ ,  $R$  is a finite set of evolution rules having the form

$$u_V \xrightarrow{f} v_V \mid_{pr_V}$$

where  $u_V, pr_V \in \Sigma_V^*$  are multisets (often denoted without brackets) of objects and variables denoting reactants and promoters, respectively;  $v_V \in \Sigma_{EV}^*$  is



a multiset of objects and expressions with variables denoting products; and  $f : \Sigma^* \mapsto \mathbb{R}^{\geq 0}$  is a weight function. Precisely:

$$\Sigma_V = \{(a_i, d_i) \mid a_i \in A, d_i \in D_{a_i}^V\} \quad \Sigma_{EV} = \{(a_i, e_i) \mid a_i \in A, e_i \in E_{a_i}^V\}$$

where  $D_{a_i}^V = (V \cup I_1) \times \dots \times (V \cup I_{arity(a_i)})$ ; and  $E_{a_i}^V = Exp(V, I_1) \times \dots \times Exp(V, I_{arity(a_i)})$ , with  $Exp(V, I)$  denoting the set of well-typed expressions built from operators, variables  $V$ , and values of  $I$ . Moreover, we have  $Vars(v_V) \subseteq Vars(u_V) \cup Vars(pr_V)$ , where  $Vars(t)$  denotes the set of variables occurring in  $t$ . Rules without variables are called ground rules.

In what follows we will denote an (attributed) object  $\langle a, d \rangle$  as  $a_{(d)}$ . A state (or configuration) of an APP system is a multiset of objects in  $\Sigma^*$ . By definition, the initial state is  $w_0$ , and we denote a generic state as  $w$ .

The evolution of an APP system is a sequence of probabilistic maximally parallel steps. We formally define the semantics of APP systems as a transition relation in the style of [7]. In each step a maximal multiset of evolution rule instances is selected and applied as described by the following semantic rules:

$$\begin{array}{l}
 \text{(rule application)} \quad \frac{r_i = u \xrightarrow{k} v \in R \quad u \subseteq w'}{K = \{k' \mid u' \xrightarrow{k'} v' \in R, u' \subseteq w'\} \quad p = k / \sum_{k' \in K} k'} \\
 \text{(single rule sequence)} \quad \frac{(w', \bar{w}') \xrightarrow{r_i, p} (w'', \bar{w}'')}{(w', \bar{w}') \xrightarrow{[r_i], p^+} (w'', \bar{w}'')} \\
 \text{(multiple rules sequence)} \quad \frac{(w', \bar{w}') \xrightarrow{r_i, p_i} (w'', \bar{w}'') \quad (w'', \bar{w}'') \xrightarrow{\bar{r}, \bar{p}^+} (w''', \bar{w}''')}{(w', \bar{w}') \xrightarrow{\bar{r} @ [r_i], p_i \cdot \bar{p}^+} (w''', \bar{w}''')} \\
 \text{(step rule)} \quad \frac{(w, \emptyset) \xrightarrow{\bar{r}, \bar{p}^+} (w', \bar{w}') \quad (w', \bar{w}') \xrightarrow{\bar{r}, \bar{p}^+} (w', \bar{w}')}{w \xrightarrow{\bar{r}, \bar{p}} w' + \bar{w}'}
 \end{array}$$

where  $[r_i]$  denotes the sequence composed of the single element  $r_i$ , and  $@$  denotes the concatenation of sequences.

Given a system state,  $w$ , the (step rule) describes the evolution in a new state by the  $\xrightarrow{\bar{r}, \bar{p}}$  relation, where  $\bar{p}$  is the probability of the transition, and  $\bar{r}$  is the sequence of applied ground rules. (step rule) invokes  $(w, \emptyset) \xrightarrow{\bar{r}, \bar{p}^+} (w', \bar{w}')$  where  $R(w)$  is the set of applicable ground rules in the state  $w$ , with their weights, namely:

$$R(w) = \left\{ u_V \sigma \xrightarrow{f(w)} v_V \sigma \mid u_V \xrightarrow{f} v_V \mid_{pr_V} \in R, \exists \sigma. u_V \sigma \subseteq w \wedge pr_V \sigma \subseteq w \right\}$$

where (i)  $\sigma : V \rightarrow flat(D_{a_1}) \cup \dots \cup flat(D_{a_n})$ , with  $flat(D_{a_i}) = I_1 \cup \dots \cup I_{arity(a_i)}$ , for all  $a_i \in A$ ; (ii)  $u_V \sigma$  ( $u_V \in \Sigma_V^*$ ) is the well-typed multiset obtained by substituting values for variables in  $u_V$  according to  $\sigma$ ; and (iii)  $v_V \sigma$  ( $v_V \in \Sigma_{EV}^*$ )

${}^aP_{\text{prot}} = (A, \text{arity}, D_p, D_{\text{move}}, D_{\text{repr}}, D_{\text{aging}}, w_0, R)$  where:

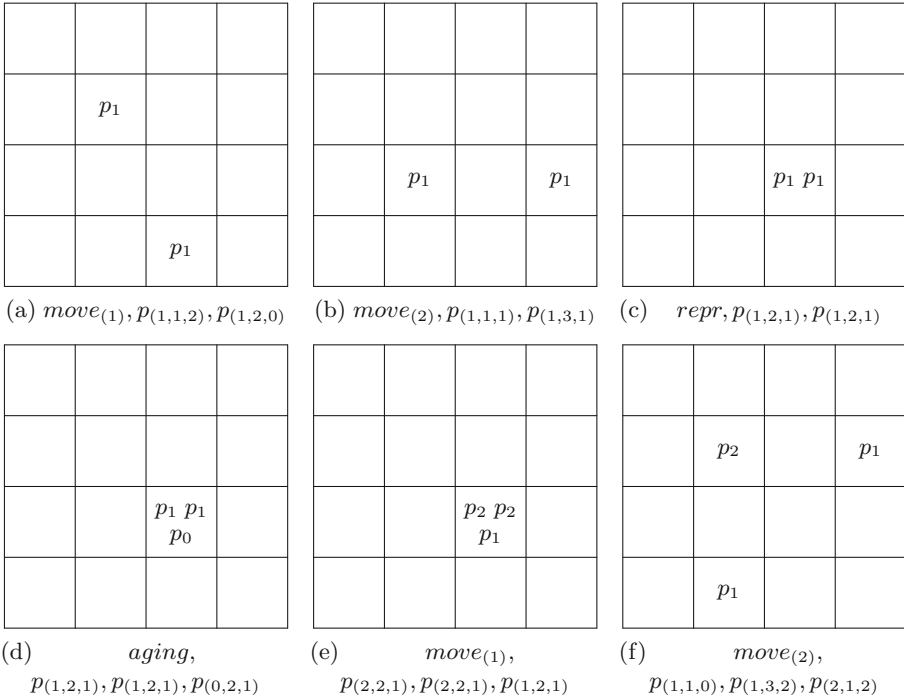
$$A = \{p, \text{move}, \text{repr}, \text{aging}\} \quad \text{arity} = \{p \mapsto 3, \text{move} \mapsto 1, \text{repr} \mapsto 0, \text{aging} \mapsto 0\}$$

$$D_p = \{0, 1, 2\} \times \{0, 1, 2, 3\} \times \{0, 1, 2, 3\};$$

$$D_{\text{move}} = \{1, 2\}; \quad D_{\text{move}2} = D_{\text{repr}} = D_{\text{aging}} = \emptyset \quad w_0 = \{\text{move}_{(1)}, p_{(1,1,2)}, p_{(1,2,0)}\}$$

$$R = \left\{ \begin{array}{ll} r_1 : p_{(a,x,y)} \xrightarrow{1} p_{(a,x+1,y)} \mid \text{move}_{(n)} & \text{if } x < 3 \\ r_2 : p_{(a,x,y)} \xrightarrow{1} p_{(a,x,y+1)} \mid \text{move}_{(n)} & \text{if } y < 3 \\ r_3 : p_{(a,x,y)} \xrightarrow{1} p_{(a,x-1,y)} \mid \text{move}_{(n)} & \text{if } x > 0 \\ \vdots & \\ r_8 : p_{(a,x,y)} \xrightarrow{1} p_{(a,x-1,y-1)} \mid \text{move}_{(n)} & \text{if } x > 0 \wedge y > 0 \\ r_9 : p_{(a_1,x,y)}, p_{(a_2,x,y)} \xrightarrow{1} p_{(a_1,x,y)}, p_{(a_2,x,y)}, p_{(0,x,y)} \mid \text{repr} & \\ r_{10} : p_{(a,x,y)} \xrightarrow{1} p_{(a+1,x,y)} \mid \text{aging} & \text{if } a < 2 \\ r_{11} : p_{(2,x,y)} \xrightarrow{1} \mid \text{aging} & \\ r_{12} : \text{move}_{(1)} \xrightarrow{1} \text{move}_{(2)} & r_{13} : \text{move}_{(2)} \xrightarrow{1} \text{repr} \\ r_{14} : \text{repr} \xrightarrow{1} \text{aging} & r_{15} : \text{aging} \xrightarrow{1} \text{move}_{(1)} \end{array} \right\}$$

**Fig. 1.** APP system modelling protozoans.



**Fig. 2.** Example of the evolution of the protozoans model, with (a)–(f) representing a possible sequence of states reached by the system. The caption of each figure contains the complete multiset of objects present in the system. (For compactness, only the age attributes are shown for the objects in the figures, that is,  $p_{(a,x,y)}$  is depicted as  $p_a$ .)

is the well-typed multiset obtained by evaluating the expressions in  $v_V$  under the substitution  $\sigma$ . Transition relation  $\xrightarrow{R}_{\bar{r}, \bar{p}}^+$  is the transitive closure of  $\xrightarrow{R}_{r, p}$ .

A transition  $(w', \bar{w}') \xrightarrow{r_i, p}_R (w' - u, \bar{w}' + v)$  corresponds to the application of a single rule. When a rule is selected, its application consists in removing its reactants from  $w'$  and adding its products to  $\bar{w}'$ . The  $\bar{w}'$  multiset will collect all products of all applied rules. Note that  $R(w)$  takes into account that each rule is applied with respect to the weights of the rules computed in the initial state  $w$ . Moreover,  $R(w)$  contains only the ground rules the promoters of which are present in the initial state  $w$  ( $pr_V\sigma \subseteq w$ ). Once objects in  $w'$  are such that no further rule in  $R(w)$  can be applied to them, by (step rule) the new system state is  $w' + \bar{w}'$  (where  $w'$  are the unused objects and  $\bar{w}'$  are the new products).

Intuitively, the semantic definition states that all the rules to be applied are selected in a probabilistic way from the set of applicable rules, their reactant are removed for the available reactants,  $w'$ , and their product are added to a suspended multiset  $\bar{w}'$ . When no further rule can be applied to  $w'$  the new state, which is composed by the unused objects in  $w'$  plus the suspended products in  $\bar{w}'$ , is produced. Finally we give the probability of a transition between two states by means of the following rule:

(state transition prob.)	$PR = \{(\bar{r}, \bar{p}) \mid w \xrightarrow{\bar{r}, \bar{p}}_R w'\} \quad p = \frac{\sum_{(\bar{r}, \bar{p}) \in PR} \bar{p}}{w \xrightarrow{p}_R w'}$
--------------------------	--

*Example 1.* We consider a population of sexually reproducing protozoans in which two individuals are necessary for producing an offspring. Protozoans are free ranging on a laboratory Petri dish. The Petri dish is abstracted by a  $n \times n$  grid and protozoans can move one step at a time on the grid. They can reproduce if they meet in the same entry of the grid. They have a finite lifespan, at the end of which they die. Each individual is represented by an attributed object  $p_{(a,x,y)}$  in which  $p$  stands for “protozoan”, attribute  $a$  is an integer representing the age of the individual, and attributes  $x$  and  $y$  are the coordinates of the position of the individual on the Petri dish. The evolution cycles among three phases: a *movement* phase, a *reproduction* phase, and an *aging* phase. Each phase is represented by a different symbol, namely *move*, *repr*, and *aging*, respectively, which are used as promoters to enable different sets of rules for each phase. The movement phase has a duration of two maximally-parallel steps, hence the *move* symbol has an attribute taking values from the set  $\{1, 2\}$  to allow modelling it.

For the sake of simplicity we assume a  $4 \times 4$  grid and a lifespan of 3 age units. We consider an initial configuration in *move*<sub>(1)</sub> phase with two individuals, of age 1, in positions (1, 2) and (2, 0), respectively. The APP system is shown in Fig. 1. Rules  $r_1 - r_8$  model the movement phase, with a different rule for each possible direction of movement: east for rule  $r_1$ , north for rule  $r_2$ , and so on, also allowing diagonal movement as exemplified by rule  $r_8$ . Rule  $r_9$  handles the reproduction phase, while rules  $r_{10}, r_{11}$  model the aging phase. Finally, rules  $r_{12} - r_{15}$  are used to switch phases. Note that all the weights associated with the rules are constant

and equal to 1, thus for each phase all (and only) the rules specific for that phase can be applied, and such rules are equiprobable. An example of evolution of the system is shown in Fig. 2.

### 3 Modelling Social Interactions in Primates

In this model, we describe the behaviour of male monkeys and how it changes when a female monkey enters the oestrus. The model is inspired by the social behaviours of species of prosimians as described in [10, 27, 28]. The population is dispersed in an environment, which is modelled as a continuous 2D space, hence each individual is associated with coordinates  $(x, y)$ . Male and female monkeys are represented by symbols MMonkey and FMonkey, respectively, and both have attributes in the domain  $\mathbb{R}^2 \times \mathbb{N}$ . Beyond the actual position, we also keep track of the *dominance* level of each individual, which is used to derive the likeliness of a individual to win (or just engage in) a fight against another individual.

At the beginning, the population is composed only of male monkeys having a dominance level of 1500. All the male monkeys alternate between two phases: a *movement* phase, represented by the special symbol MOV, in which they wander around slowly and move towards other individuals in order to keep the population compact; and a *fight* phase, represented by FGT, in which they chase other individuals to fight, yielding to variations in their levels of dominance. Females alternate between a *normal* phase, denoted by the symbol NORMAL, and an *oestrus* phase, denoted by OEST. In this model, for simplicity, there is only one female monkey, which is explicitly represented only during the oestrus phase. In other words, the individual FMonkey appears only at the beginning of the oestrus phase, and is removed from the model at the end of the phase.

Formally, the model is composed of 6 symbols  $A = \{\text{MMonkey}, \text{FMonkey}, \text{MOV}, \text{FGT}, \text{NORMAL}, \text{OEST}\}$ , having a corresponding set of attributes defined as  $D = \{(\mathbb{R}^2 \times \mathbb{N}), (\mathbb{R}^2 \times \mathbb{N}), \mathbb{N}, \mathbb{N}, \mathbb{N}, \mathbb{N}\}$ . As regards symbols MOV, FGT, NORMAL, OEST, an attribute from  $\mathbb{N}$  is associated with each of them, denoting the length of those phases in terms of the steps taken by the Attributed P system. The initial state of the system is the following:

$$w_0 = \{\text{MMonkey}_{(x_1, y_1, 1500)}, \text{MMonkey}_{(x_2, y_2, 1500)}, \text{MMonkey}_{(x_3, y_3, 1500)}, \\ \text{MMonkey}_{(x_4, y_4, 1500)}, \text{MMonkey}_{(x_5, y_5, 1500)}, \text{MMonkey}_{(x_6, y_6, 1500)}, \\ \text{MMonkey}_{(x_7, y_7, 1500)}, \text{MMonkey}_{(x_8, y_8, 1500)}, \text{MOV}_{(1)}, \text{NORMAL}_{(Snl)}\}$$

where the positions of the individuals  $(x_i, y_i) \in \mathbb{R}^2$  are randomly generated within a square area of *cage*  $\in \mathbb{R}$  side length. Parameter *Snl* denotes the duration of the “normal” phase for females.

The evolution rules of the model are as follows.

$$\begin{aligned}
 r_1 &: \text{MOV}_{(n)} \xrightarrow{1} \text{MOV}_{(n-1)} && \forall n > 1 \\
 r_2 &: \text{MOV}_{(1)} \xrightarrow{1} \text{FGT}_{(Nfl)} | \text{NORMAL}_{(n)} \\
 r_3 &: \text{MOV}_{(1)} \xrightarrow{1} \text{FGT}_{(OfI)} | \text{OEST}_{(n)} \\
 r_4 &: \text{FGT}_{(n)} \xrightarrow{1} \text{FGT}_{(n-1)} && \forall n > 1 \\
 r_5 &: \text{FGT}_{(1)} \xrightarrow{1} \text{MOV}_{(Msn)}
 \end{aligned}$$

Rules  $r_1$ – $r_5$  model the alternation between “movement” and “fight” phases for males. For both MOV and FGT, their attributes decrease to keep track of the number of steps passed, until they reach 1 and a phase switch occurs. In particular, in the switch from MOV and FGT, the number of steps that the fight phase lasts depend on the phase of the female; namely it is either  $Nfl$  or  $OfI$ , if the female is currently either in the normal phase (NORMAL) or the oestrus phase (OEST), respectively. The movement phase lasts  $Msn$  steps.

$$\begin{aligned}
 r_6 &: \text{NORMAL}_{(n)} \xrightarrow{1} \text{NORMAL}_{(n-1)} && \forall n > 1 \\
 r_7 &: \text{NORMAL}_{(1)} \xrightarrow{1} \\
 &\quad \text{OEST}_{(Sol)}, \text{FMonkey}_{(FemaleInitX, FemaleInitY, FemaleInitDom)} \\
 r_8 &: \text{OEST}_{(n)} \xrightarrow{1} \text{OEST}_{(n-1)} && \forall n > 1 \\
 r_9 &: \text{OEST}_{(1)}, \text{FMonkey}_{(x,y,dom)} \xrightarrow{1} \text{NORMAL}_{(Snl)}
 \end{aligned}$$

Rules  $r_6$ – $r_9$  model the alternation between “normal” and “oestrus” phases for the female monkey. The durations of the oestrus phase is modelled by the parameter  $Sol$ . The initial coordinates of the female monkey are denoted by the parameters  $FemaleInitX$  and  $FemaleInitY$ , while  $FemaleInitDom$  denotes its initial dominance level.

$$\begin{aligned}
 r_{10} &: \text{MMonkey}_{(x',y',dom')} \xrightarrow{f_{10}} \text{MMonkey}_{(\text{move}(x',x'',SMMA), \text{move}(y',y'',SMMA), dom')} \\
 &\quad | \text{MOV}_{(n)}, \text{NORMAL}_{(m)}, \text{MMonkey}_{(x'',y'',dom'')} \\
 r_{11} &: \text{MMonkey}_{(x',y',dom')} \xrightarrow{f_{11}} \text{MMonkey}_{(\text{move}(x',x'',SMMA), \text{move}(y',y'',SMMA), dom')} \\
 &\quad | \text{MOV}_{(n)}, \text{OEST}_{(m)}, \text{MMonkey}_{(x'',y'',dom'')}
 \end{aligned}$$

Rules  $r_{10}$ – $r_{11}$  handle the movement of males during either the “normal” or “oestrus” phase for the female. In both cases, a male (in a position  $x', y'$ ) is allowed to move towards any other male (in position  $x'', y''$ ). The resulting position of the male which moves is computed as  $(\text{move}(x', x'', SMMA), \text{move}(y', y'', SMMA))$ , where  $\text{move}$  is a function to move the coordinates  $(x', y')$  towards coordinates  $(x'', y'')$  with a given speed factor described by the parameter  $SMMA$  (Speed Male-Male approach). Formally, this function is defined as:

$$\text{move}(a, b, \gamma) = a + (b - a)/\gamma$$

The most important difference between rules  $r_{10}$  and  $r_{11}$  lies in the weight functions, which are defined as:

$$f_{10} = \begin{cases} 1 & \text{if } SD/NT < dist((x', y'), (x'', y'')) < SD; \\ 0 & \text{otherwise;} \end{cases}$$

$$f_{11} = \begin{cases} 1 & \text{if } SD/OT < dist((x', y'), (x'', y'')) < SD; \\ 0 & \text{otherwise.} \end{cases}$$

where  $dist((x', y'), (x'', y''))$  is a function giving the euclidean distance between two points, parameter  $SD$  (Spot Distance) denotes the maximum visibility distance of a monkey, and parameters  $NT$  (Normal Tolerance) and  $OT$  (Oestrus Tolerance) are used to derive the minimum distance allowed between two monkeys to enable the relative movement of one towards the other. In particular, such a minimum distance depends on the phase of the female, and it is either  $SD/NT$  during the NORMAL phase, and  $SD/OT$  during the OEST phase. In this manner, during the oestrus phase, males are allowed to come closer one another, hence increasing the possibility to engage in a fight.

$$r_{12} : \text{MMonkey}_{(x', y', dom')} \xrightarrow{f_{12}} \text{MMonkey}_{(\text{move}(x', x'', SMFA), \text{move}(y', y'', SMFA), dom')} \\ \mid \text{MOV}_{(n)}, \text{FMonkey}_{(x'', y'', dom'')}$$

Rule  $r_{12}$  models the movement of a male monkey towards the female. In this case, the speed of the male is denoted by the parameter  $SMFA$ . The corresponding weight function is:

$$f_{12} = \begin{cases} Pff & \text{if } dist((x', y'), (x'', y'')) < SD \text{ and } dom' + FT > dom''; \\ 0 & \text{otherwise;} \end{cases}$$

which enables the movement only if both (i) their relative distance is less than  $SD$ , and (ii) the dominance level of the male, plus a tolerance value  $FT$  (Female Tolerance), is greater than that of the female. The actual weight used is denoted by the parameter  $Pff$  (Preference for Female).

$$r_{13} : \text{MMonkey}_{(x', y', dom')}, \text{MMonkey}_{(x'', y'', dom'')} \xrightarrow{elo\_rating(dom', dom'')} \\ \text{MMonkey}_{(\text{chase}(x', x'', CSN), \text{chase}(y', y'', CSN), elo\_low(dom', dom''))}, \\ \text{MMonkey}_{(\text{flee}(x', x'', FSN), \text{flee}(y', y'', FSN), elo\_low(dom'', dom'))} \mid \text{FGT}_{(n)}, \text{NORMAL}_{(m)}$$

with  $dom' \geq dom''$ ,  $dist((x', y'), (x'', y'')) \leq NAD$ , and  $dom' - dom'' \leq AN$ , where  $NAD$  (Normal Aggression Distance) and  $AN$  (Avoidance Normal) and model parameters representing the minimum distance and the maximum difference in dominance that enable an aggression when the female is in normal condition. In this rule the monkey in position  $(x', y')$  has a dominance that is higher or equal to that of the other monkey. The probability that the first monkey wins the fight is given by the standard Elo rating method, originally defined

for applications to games, and then used for the modelling of social interactions. Such a method is based on a table that gives the probability of success in a fight depending on the difference of rating (or dominance) of the involved individual. The Elo rating table we consider for this model is in [1]. The function  $elo\_rating(\Delta dom)$  looks in the table and gives as result the probability of the victory of the stronger monkey over the weaker one.

Function  $chase$  gives the new position of the winner of the fight; function  $flee$  gives the new position of the loser of the fight;  $elow$  gives the new dominance of the winner of the fight following the Elo rating table and method;  $elol$  the new dominance of the loser of the fight. These functions are defined as follows:

$$\begin{aligned} chase(a, b, \rho) &= a + \rho \cdot (b - a) & flee(a, b, \rho) &= b + \rho \cdot (b - a) \\ elow(d', d'') &= d' + \begin{cases} (1 - elo\_rating(\Delta dom)) \cdot stepness & \text{if } d' > d'' \\ elo\_rating(\Delta dom) \cdot stepness & \text{if } d' < d'' \end{cases} \\ elol(d', d'') &= d' - \begin{cases} elo\_rating(\Delta dom) \cdot stepness & \text{if } d' > d'' \\ 1 - (elo\_rating(\Delta dom)) \cdot stepness & \text{if } d' < d'' \end{cases} \end{aligned}$$

where  $\Delta dom = |d' - d''|$ , and  $stepness$  is a parameter representing the maximum increase/decrease of dominance. The parameters  $CSN$  (Chase Speed Normal) and  $FSN$  (Flee Speed Normal) used in rule  $r_{13}$  describe how fast the monkeys move.

$$\begin{aligned} r_{14} : & \text{MMonkey}_{(x', y', dom')}, \text{MMonkey}_{(x'', y'', dom'')} \xrightarrow{1 - elo\_rating(dom', dom'')} \\ & \text{MMonkey}_{(flee(x', x'', FSN), flee(y', y'', FSN), elol(dom', dom''))}, \\ & \text{MMonkey}_{(chase(x', x'', CSN), chase(y', y'', CSN), elow(dom'', dom'))} \Big|_{\text{FGT}_{(n)} \text{NORMAL}_{(m)}} \end{aligned}$$

with  $dom' > dom''$ ,  $dist((x', y'), (x'', y'')) \leq NAD$ , and  $|dom' - dom''| \leq AN$ .

Rule  $r_{14}$  is analogous to rule  $r_{13}$ , but describes the case in which the winner is the weaker monkey.

$$\begin{aligned} r_{15} : & \text{MMonkey}_{(x', y', dom')}, \text{MMonkey}_{(x'', y'', dom'')} \xrightarrow{elo\_rating(dom', dom'')} \\ & \text{MMonkey}_{(chase(x', x'', CSO), chase(y', y'', CSO), elow(dom', dom''))}, \\ & \text{MMonkey}_{(flee(x', x'', FSO), flee(y', y'', FSO), elol(dom'', dom''))} \Big|_{\text{FGT}_{(n)}, \text{OEST}_{(m)}} \end{aligned}$$

with  $dom' \geq dom''$ ,  $dist((x', y'), (x'', y'')) \leq OAD$ , and  $dom' - dom'' \leq AO$ .

$$\begin{aligned} r_{16} : & \text{MMonkey}_{(x', y', dom')}, \text{MMonkey}_{(x'', y'', dom'')} \xrightarrow{1 - elo\_rating(dom', dom'')} \\ & \text{MMonkey}_{(flee(x', x'', FSO), flee(y', y'', FSO), elol(dom', dom''))}, \\ & \text{MMonkey}_{(chase(x', x'', CSO), chase(y', y'', CSO), elow(dom'', dom'))} \Big|_{\text{FGT}_{(n)}, \text{OEST}_{(m)}} \end{aligned}$$

with  $dom' > dom''$ ,  $dist((x', y'), (x'', y'')) \leq OAD$ , and  $|dom' - dom''| \leq AO$ .

Rules  $r_{15}$  and  $r_{16}$  is analogous to  $r_{13}$  and  $r_{14}$ , respectively, but describe the case in which the female is in oestrus state. Parameters  $OAD$  (Oestrus

Aggression Distance), AO (Avoidance Oestrus), *CSO* (Chase Speed Oestrus) and *FSO* (Flee Speed Oestrus) of these rules are analogous to the corresponding ones of rules  $r_{13}$  and  $r_{14}$ , but with values that depend on the fact that the female is in oestrus state.

$$r_{17} : \text{MMonkey}_{(x',y',dom')}, \text{MMonkey}_{(x'',y'',dom'')} \xrightarrow{1} \\ \text{MMonkey}_{(\text{chase}(x',x'',CSN),\text{chase}(y',y'',CSN),dom')}, \\ \text{MMonkey}_{(\text{flee}(x',x'',FSN),\text{flee}(y',y'',FSN),dom'')} | \text{FGT}_{(n)}, \text{NORMAL}_{(m)}$$

with  $dom' > dom''$ ,  $\text{dist}((x',y'),(x'',y'')) \leq NAD$  and  $|dom' - dom''| > AN$ .

$$r_{18} : \text{MMonkey}_{(x',y',dom')}, \text{MMonkey}_{(x'',y'',dom'')} \xrightarrow{1} \\ \text{MMonkey}_{(\text{chase}(x',x'',CSO),\text{chase}(y',y'',CSO),dom')}, \\ \text{MMonkey}_{(\text{flee}(x',x'',FSO),\text{flee}(y',y'',FSO),dom'')} | \text{FGT}_{(n)}, \text{OEST}_{(m)}$$

with  $dom' > dom''$ ,  $\text{dist}((x',y'),(x'',y'')) \leq OAD$  and  $|dom' - dom''| > AO$ .

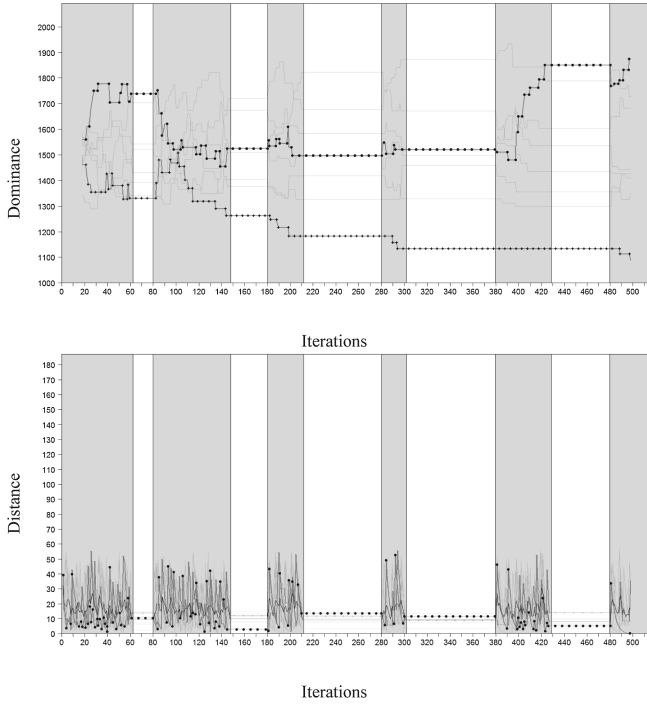
Rules  $r_{17}$  and  $r_{18}$  describe the interaction between two individuals when the difference in dominance is too high to motivate a fight (greater than parameters AN and AO for the normal and oestrus cases, respectively). In these cases the monkeys move but do not fight, so there is no change in dominance levels.

## 4 Experimental Results

We studied the dynamics of the APP model described in the previous section by running simulations. In particular, we implemented an APP systems interpreter in C# that allows attributed objects and evolution rules to be represented as instantiations of specific C# classes. Once an APP systems model is specified, the interpreter simulates it by performing a number of iterations to be given as a parameter. In each iteration, a maximally parallel step is performed according to the APP systems semantics. The result of a simulation is the sequence of configurations reached by the interpreter at each iteration. In order for the interesting measurements to be easily readable, we processed the simulation results and produced graphical representations by using the statistical framework R.

In Fig. 3 and in Fig. 4 we show the dynamics of two groups of monkeys. In particular, Fig. 3 refers to a group with a low level of aggressiveness (egalitarian), while Fig. 4 describes a group with a higher level of aggressiveness (despotic). The upper part of both figures shows the dominance level of each male in the group during the simulation, while the lower part shows the distance of each male either from the center of the group or from the female (when present). In the figures we put in evidence the lines corresponding to both the monkey with highest dominance level at the end of the simulation (line marked with  $\bullet$ ) and the one with the lowest one (marked with  $+$ ). The main model parameters (that are different in the two cases) are reported in the figures. The other parameters have, in both cases, the following values: iteration = 498, Sol = 20, Snl = 80,



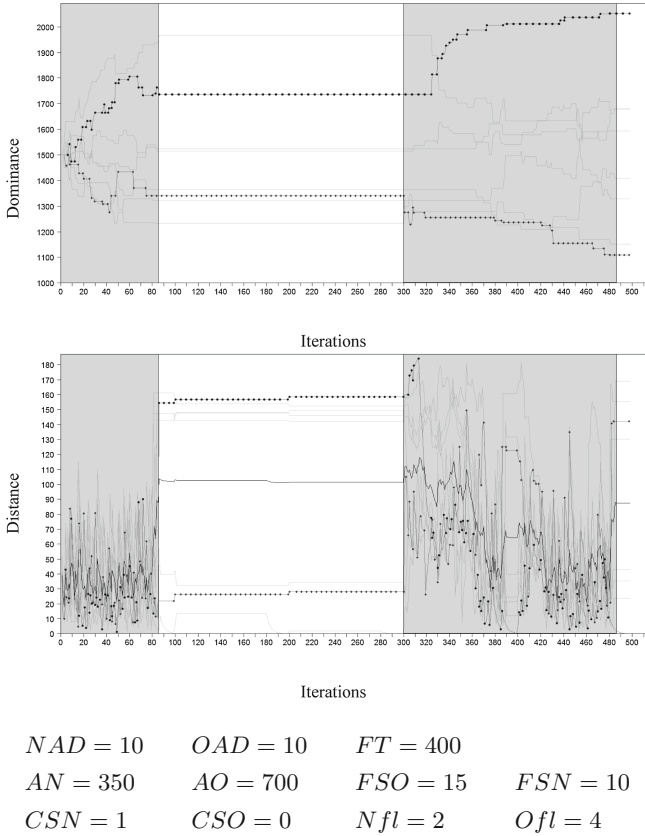


$NAD = 5$	$OAD = 5$	$FT = 600$
$AN = 200$	$AO = 400$	$FSO = 12$ $FSN = 8$
$CSN = 0$	$CSO = 0$	$Nfl = 1$ $Ofl = 2$

**Fig. 3.** Simulation of an egalitarian group.

$SMMA = 0.25$ ,  $SD = 100$ ,  $NT = 3$ ,  $OT = 2$ ,  $SMFA = 0.25$ ,  $PfF = 8$ ,  $Msn = 1$  and  $Stepness = 100$ . Note that we ran the simulations for 498 iterations. The time corresponding to an iteration is, in the real world, about a few hours. Actually, such a time could vary among the different phases described by the model. However, this is not a problem since we are not interested in precise description of timing aspects. As regards the other model parameters they have been estimated from the descriptions in [10,27,28]. The results we obtain are compatible with the behaviour of prosimians, as described in the above studies.

As regards Fig. 3, initially all the males have the same dominance level. From the beginning and up to about 210 iterations, the group of monkeys struggles to define a clear dominance among individuals. Between iteration 210 and 390, the dotted line is not the dominant of the group, thus its position is not the closest neither to the group center nor to the female. At the end of the simulation, when the monkey with dotted line becomes the most dominant (alpha male), we can observe that it gains a central position in the group.



**Fig. 4.** Simulation of a despotic group.

Figure 4 shows the dynamics of a group with a more rigid hierarchy, due to the higher level of aggressiveness. The first phase, up to iteration 90, in which the males establish a first hierarchy, is followed by a phase (up to iteration 320) in which the hierarchy becomes very stable. From iteration 320 onwards, the monkey with the dotted line becomes the alpha male, it gains the position which is closer to the center of the group and to the female, and it does not allow any other monkey to come close.

Normal and oestrus periods last respectively 80 and 20 iterations. In normal phase monkeys are less willing to fight and they keep a distance from each other in order to avoid unnecessary conflicts; this corresponds to the more linear parts of the graphs. In oestrus periods males have the female as the pole of attraction and they are more willing to fight. Fights can change the dominance levels of males, thus oestrus periods correspond to more “hectic” parts of the graphs, where, often, the ranking of dominance changes. When the dominance levels change, the topology of the group changes accordingly. The results of

the simulations agree with the behaviour of different species of prosimians as described in [10,27,28]

## 5 Conclusions

We proposed an extension of probabilistic P Systems, called Attributed Probabilistic P systems (APP systems), in which objects are annotated with attributes. APP systems are intended to be used to model the dynamics of populations and ecosystems. In this context, attributes can be used to represent characteristics of the population individuals such as age, position, and so on. Apart from attributes, the feature that mainly makes a difference between APP systems and other proposals is the use of maximal parallelism for the application of rules. This feature is particularly suitable for the modelling of populations that evolve by stages (e.g. reproductive stages or stages related with seasons).

We used APP systems for modelling social behaviours of some species of primates. In particular, as an application we developed a model to compare despotic and egalitarian behaviours of different species of primates. Such kinds of social systems are usually approached by means of agent-based models that are often poorly documented and ambiguous. On the contrary, since both the syntax and the semantics of APP systems are formally defined, the model based on APP systems is unambiguous.

The model has been inspired by the behaviour of species of prosimians. We plan to adapt our general model to the modelling of the behaviour of particular species of primates by changing the values of the parameters.

## References

1. Albers, P.C., de Vries, H.: Elo-rating as a tool in the sequential estimation of dominance strengths. *Anim. Behav.* **61**(2), 489–495 (2001)
2. Barbuti, R., Bove, P., Schettini, A.M., Milazzo, P., Pardini, G.: A computational formal model of the invasiveness of eastern species in European water frog populations. In: Counsell, S., Núñez, M. (eds.) SEFM 2013. LNCS, vol. 8368, pp. 329–344. Springer, Heidelberg (2014)
3. Barbuti, R., Cerone, A., Maggiolo-Schettini, A., Milazzo, P., Setiawan, S.: Modelling population dynamics using grid systems. In: Cerone, A., Persico, D., Fernandes, S., Garcia-Perez, A., Katsaros, P., Ahmed Shaikh, S., Stamelos, I. (eds.) SEFM 2012 Satellite Events. LNCS, vol. 7991, pp. 172–189. Springer, Heidelberg (2014)
4. Barbuti, R., Maggiolo-Schettini, A., Milazzo, P., Pardini, G.: Spatial calculus of looping sequences. *Theor. Comput. Sci.* **412**(43), 5976–6001 (2011)
5. Barbuti, R., Maggiolo-Schettini, A., Milazzo, P., Pardini, G.: Simulation of spatial P system models. *Theore. Comput. Sci.* **529**, 11–45 (2014)
6. Barbuti, R., Maggiolo-Schettini, A., Milazzo, P., Pardini, G., Tesei, L.: Spatial P systems. *Nat. Comput.* **10**(1), 3–16 (2011)
7. Barbuti, R., Maggiolo-Schettini, A., Milazzo, P., Tini, S.: An overview on operational semantics in membrane computing. *Int. J. Found. Comput. Sci.* **22**(01), 119–131 (2011)

8. Bortolussi, L., Policriti, A.: Modeling biological systems in stochastic concurrent constraint programming. *Constraints* **13**(1–2), 66–90 (2008)
9. Cardona, M., Colomer, M.A., Margalida, A., Palau, A., Pérez-Hurtado, I., Pérez-Jiménez, M.J., Sanuy, D.: A computational modeling for real ecosystems based on P systems. *Nat. Comput.* **10**(1), 39–53 (2011)
10. Cavigelli, S.A., Pereira, M.E.: Mating season aggression and fecal testosterone levels in male ring-tailed lemurs (*Lemur catta*). *Horm. Behav.* **37**(3), 246–255 (2000)
11. Ciocchetta, F., Hillston, J.: Bio-pepa for epidemiological models. *Electron. Notes Theor. Comput. Sci.* **261**, 43–69 (2010)
12. Nieto Coria, C.A., Tesei, L., Scarcella, G., Russo, T., Merelli, E.: Sea-scale agent-based simulator of solea solea in the Adriatic sea. In: Canal, C., Idani, A. (eds.) SEFM 2014 Workshops. LNCS, vol. 8938, pp. 259–275. Springer, Heidelberg (2015)
13. Danos, V., Honorato-Zimmer, R., Jaramillo-Riveri, S., Stucki, S.: Rigid geometric constraints for Kappa models. *Electron. Notes Theor. Comput. Sci.* **313**, 23–46 (2015)
14. Gautrais, J., Ginelli, F., Fournier, R., Blanco, S., Soria, M., Chaté, H., Theraulaz, G.: Deciphering interactions in moving animal groups. *Plos Comput. Biol.* **8**(9), e1002678 (2012)
15. Grimm, V., Berger, U., Bastiansen, F., Eliassen, S., Ginot, V., Giske, J., Goss-Custard, J., Grand, T., Heinz, S.K., Huse, G., et al.: A standard protocol for describing individual-based and agent-based models. *Ecol. Model.* **198**(1), 115–126 (2006)
16. Grimm, V., Berger, U., DeAngelis, D.L., Polhill, J.G., Giske, J., Railsback, S.F.: The odd protocol: a review and first update. *Ecol. Model.* **221**(23), 2760–2768 (2010)
17. Hemelrijk, C.K.: Spatial centrality of dominants without positional preference. *Artif. Life VI* **6**, 307–315 (1998)
18. Hemelrijk, C.K.: An individual-orientated model of the emergence of despotic and egalitarian societies. *Proc. R. Soc. Lond. B Biol. Sci.* **266**(1417), 361–369 (1999)
19. Hemelrijk, C.K., Hildenbrandt, H.: Schools of fish and flocks of birds: their shape and internal structure by self-organization. *Interface Focus* **2**(6), 726–737 (2012)
20. Hemelrijk, C.K., Puga-Gonzalez, I.: An individual-oriented model on the emergence of support in fights, its reciprocation and exchange. *PLoS One* **7**(5), e37271 (2012)
21. Hemelrijk, C.: Self-organization and natural selection in the evolution of complex despotic societies. *Biol. Bull.* **202**(3), 283–288 (2002)
22. John, M., Lhoussaine, C., Niehren, J., Uhrmacher, A.M.: The attributed pi calculus. In: Heiner, M., Uhrmacher, A.M. (eds.) CMSB 2008. LNCS (LNBI), vol. 5307, pp. 83–102. Springer, Heidelberg (2008)
23. Jordán, F., Scotti, M., Priami, C.: Process algebra-based computational tools in ecological modelling. *Ecol. Complex.* **8**(4), 357–363 (2011)
24. Kahramanoğulları, O., Lynch, J.F., Priami, C.: Algorithmic systems ecology: experiments on multiple interaction types and patches. In: Cerone, A., Persico, D., Fernandes, S., Garcia-Perez, A., Katsaros, P., Ahmed Shaikh, S., Stamelos, I. (eds.) SEFM 2012 Satellite Events. LNCS, vol. 7991, pp. 154–171. Springer, Heidelberg (2014)
25. Macal, C.M., North, M.J.: Tutorial on agent-based modelling and simulation. *J. Simul.* **4**(3), 151–162 (2010)
26. McLane, A.J., Semeniuk, C., McDermid, G.J., Marceau, D.J.: The role of agent-based models in wildlife ecology and management. *Ecol. Model.* **222**(8), 1544–1556 (2011)

27. Nakamichi, M., Koyama, N.: Social relationships among ring-tailed lemurs (*Lemur catta*) in two free-ranging troops at Berenty Reserve. *Madagascar Int. J. Primatol.* **18**(1), 73–93 (1997)
28. Palagi, E., Paoli, T., Tarli, S.B.: Aggression and reconciliation in two captive groups of *Lemur catta*. *Int. J. Primatol.* **26**(2), 279–294 (2005)
29. Păun, G.: Computing with membranes. *J. Comput. Syst. Sci.* **61**(1), 108–143 (2000)
30. Paun, G., Rozenberg, G., Salomaa, A.: *The Oxford Handbook of Membrane Computing*. Oxford University Press, Inc., New York (2010)
31. Penna, P., Paoletti, N., Scarcella, G., Tesei, L., Marini, M., Merelli, E.: DISPAS: an agent-based tool for the management of fishing effort. In: Counsell, S., Núñez, M. (eds.) SEFM 2013. LNCS, vol. 8368, pp. 362–367. Springer, Heidelberg (2014)
32. Philippou, A., Toro, M.: Process ordering in a process calculus for spatially-explicit ecological models. In: Counsell, S., Núñez, M. (eds.) SEFM 2013. LNCS, vol. 8368, pp. 345–361. Springer, Heidelberg (2014)
33. Philippou, A., Toro, M., Antonaki, M.: Simulation and verification in a process calculus for spatially-explicit ecological models. *Sci. Ann. Comp. Sci.* **23**(1), 119–167 (2013)
34. Puga-Gonzalez, I., Hildenbrandt, H., Hemelrijk, C.K.: Emergent patterns of social affiliation in primates, a model. *PLoS Comput. Biol.* **5**(12), e1000630 (2009)
35. Setiawan, S., Cerone, A.: Stochastic modelling of seasonal migration using rewriting systems with spatiality. In: Counsell, S., Núñez, M. (eds.) SEFM 2013. LNCS, vol. 8368, pp. 313–328. Springer, Heidelberg (2014)
36. Toro, M., Philippou, A., Kassara, C., Sfenthourakis, S.: Synchronous parallel composition in a process calculus for ecological models. In: Ciobanu, G., Méry, D. (eds.) ICTAC 2014. LNCS, vol. 8687, pp. 424–441. Springer, Heidelberg (2014)

# Probabilistic Modelling and Analysis of a Fish Population

Chiara Cini<sup>4</sup>, Luca Tesei<sup>1,3</sup>(✉), Giuseppe Scarcella<sup>2</sup>, Cesar A. Nieto Coria<sup>1,3</sup>,  
and Emanuela Merelli<sup>1,3</sup>

<sup>1</sup> School of Science and Technology, University of Camerino, Via del Bastione, 1,  
62032 Camerino, Italy

{luca.tesei, cesar.nietocoria, emanuela.merelli}@unicam.it

<sup>2</sup> National Research Council - Institute of Marine Sciences Ancona,  
Largo Fiera della Pesca, 2, 60125 Ancona, Italy

giuseppe.scarcella@an.ismar.cnr.it

<sup>3</sup> CINFAI, Consorzio Interuniversitario Nazionale per la Fisica Delle Atmosfere  
e delle Idrosfere, Sezione di Camerino, Via del Bastione, 1, 62032 Camerino, Italy

<sup>4</sup> e-Lios (e-Linking Online Systems) S.r.l., Via Le Mosse, 22, 62032 Camerino, Italy  
chiara.cini@e-lios.eu

**Abstract.** The fish stock of the common sole in the Adriatic Sea has been analysed by agent-based modelling and simulation techniques as an integration of other classical stock assessment models. In this work we start investigating also about the formal probabilistic modelling of our case study in order to extract valuable biological information from available formal verification techniques. In particular, a PRISM model for the common sole is developed and some initial results are discussed.

**Keywords:** Fish stock assessment · Common sole · Adriatic Sea · Probabilistic models · PRISM model checker

## 1 Introduction

One of the most critical aspects about marine ecosystems and fish population nowadays is the sustainability of the fisheries. Fish stocks, even if they are renewable, are not unlimited. In a lot of cases, where the fishing activity is not controlled, fish stocks are overfished. In 2002 the World Summit on Sustainable Development (WSSD) gave guidelines on how to handle the marine ecosystem in the future [13]. In the European continent the management of fisheries is regulated by the Common Fisheries Policy (CFP) whose most recent version took effect on 1st January 2014 [14]. It stipulates that, between 2015 and 2020, catches should achieve a sustainable level for maintaining fish stocks in the long term. Until now, the policy did not have a big impact on the exploitation and its goal is still far to be achieved, especially in the Mediterranean Sea [15].

---

This work has been supported by the RITMARE Flagship Project and by the PRIN Project CINA (prot. 2010LHT4KM), both funded by the Italian MIUR.

Some of the authors have been working on models for fish stock evaluation since 2010. The starting idea was to introduce agent-based models as an alternative for the ODE/PDE-based deterministic mathematical models usually employed in the field [2]. The use of agent-based models or, more generally, of individual-based, compositional and stochastic models was, and still is, a promising approach in the context of ecological modelling [4]. The fish stock on which we have concentrated our efforts is the one of the common sole, *Solea solea* (Linnaeus 1758), in the northern and central Adriatic Sea. Independent scientific surveys in this area have been performed since 2005 by the Solemon Project [3, 10], which allows us to have a suitable collection of data for validation.

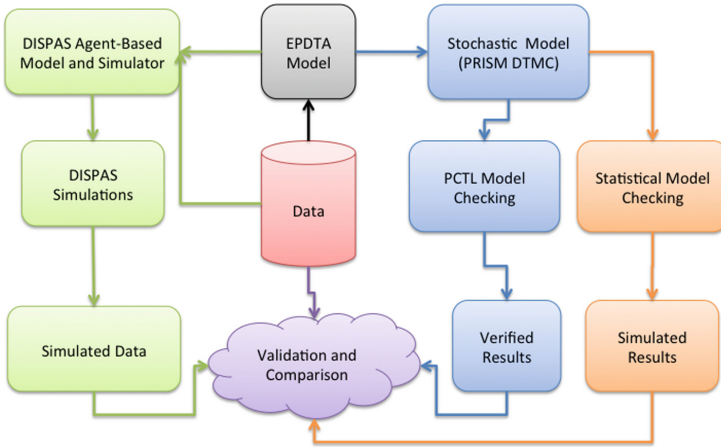
In [1] an automata-based formalism was introduced, namely Extended Probabilistic Discrete Timed Automata (EPDTA). This formalism permitted to easily model the time-dependent and probabilistic basic behaviour of a common sole and to translate it directly into a Markov decision process. This EPDTA model was used as a basis for defining the behaviour of an individual agent, representing a sole, in the development and validation of the version 1.0 of the Demersal fish Stock Probabilistic Agent-based Simulator (DISPAS) [8, 9]. The version 2.0 of the simulator, which scales from an average square kilometre of Adriatic Sea to the whole surface of the sea, was presented in [7] and is under development.

Figure 1 shows the workflow of the studies that we performed so far, which can be considered an example framework for non-marine biologists who are interested in a combination of methods to “simulate and test”. Starting from data, the EPDTA model is derived. On the left (green) side we used it to derive the probabilistic agent-based model of DISPAS, which required additional data for handling newborns. The simulated data were statistically analysed and compared with real data for validation [8, 9]. The validated model can be used to get new information by simulating different future scenarios with different levels of fishing efforts or different environmental conditions.

In this work we present the initial results of the right (blue) part of the flow. Starting from the same EPDTA model, a more analytical approach is followed. The model is translated into a Discrete Time Markov Chain (DTMC), or another stochastic formalism, and is given as input to a model checker in order to be verified against formally expressed properties. The results of this kind of analysis can be validated with the available data and can also be compared with those obtained using the simulation-based approach. In particular, in this work we report on the creation of a DTMC, derived from our initial EPDTA model, that can be given as input to the PRISM model checker and we report some initial results obtained verifying properties expressed in Probabilistic Computation Tree Logic (PCTL) [5]. The far right (orange) part of the flow shows a planned future work in which the stochastic model is used to verify formal properties by statistical model checking [6] using tools, among which PRISM itself, that are now sufficient mature in this relatively new area of semi-formal analysis.

## 2 Sole Behaviour as DTMC

Figure 2 shows a part of an EPDTA representing the probabilistic and timed behaviour of a sole [8, 9]. Soles are divided in classes according to their length,

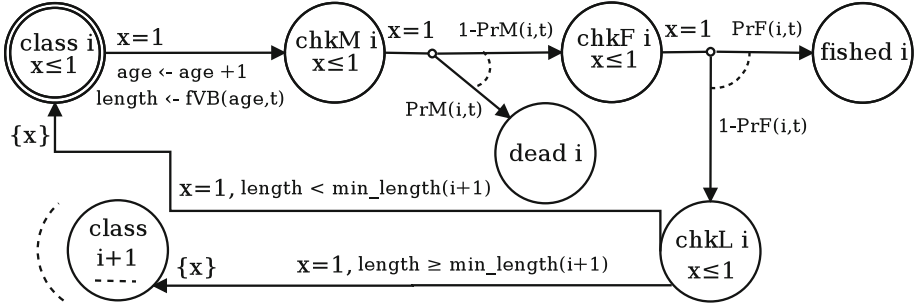


**Fig. 1.** Workflow of the studies on the common sole stock: in green the agent-based modelling and simulation approach (past and present), in blue the model checking approach (present and future), in orange the statistical model checking approach (future) (Color figure online).

assuming that individuals in the same class are subject to the same probability of being fished or of dying for natural mortality. The clock  $x$  is discrete-time and measures time in months. The length of the sole is updated each month according to its age and to environmental conditions summarised by a constant  $k$  in the von Bertalanffy growth function [12]. Then, in state CHKM  $i$ , the sole can die for natural mortality (predators, availability of food, environmental conditions) with a certain probability  $PrM(i, t)$  depending on the class  $i$  and on the current month  $t$ . If the sole survives, in state CHKF  $i$  the sole can be fished with a certain probability  $PrF(i, t)$ . If the sole survives then it proceeds to the next month possibly changing class if a given threshold on the length is reached.

The crucial point of this relatively simple model is the estimation of the probabilities for each class. In order to do so we used the natural mortality index  $M$  and the fishing effort index  $F$  calculated in the framework of specific working groups for the assessment of demersal stocks [10]. These indices represent the annual exponential decay,  $e^{-(M+F)}$ , of the population of a class according to the two causes of mortality. We took the indexes for each class from year 2006 to year 2013 and calculated the annual fishing mortality probability of each year  $y$  as  $PrF(i, y) = 1 - e^{-F_y}$ . To distribute the probability among each month, according to the Markovian characteristic of the model, we simply divided the annual probability by 12, i.e.,  $PrF(i, t) = PrF(i, y)/12$ . It must be mentioned that, according to information on fishery patterns or on fishing bans, this distribution over months can be refined considerably. The probability that quantifies natural mortality per year was calculated in the same way but, for the sake of simplicity, it was considered constant over the years. The following table shows the values used in the model. Column PrM reports the annual constant mortality probabilities while the other columns report the values of annual PrF.





**Fig. 2.** Part of an EPDTA representing the behaviour of a sole in class  $i$ . The double circled state is the initial one when  $i = 0$ . From state **chkL  $i$**  the automaton goes to the next class  $i + 1$  if the length of the sole is sufficient to be considered in the new class.

Cl	PrM	2006	2007	2008	2009	2010	2011	2012	2013
0	0.50341	0.18149	0.15970	0.14053	0.20768	0.15274	0.13822	0.15597	0.07672
1	0.29531	0.62506	0.56932	0.52251	0.67468	0.55412	0.52237	0.57209	0.32942
2	0.24421	0.36292	0.32834	0.27957	0.41247	0.31358	0.30154	0.30531	0.15760
3	0.22119	0.33458	0.30895	0.23496	0.38797	0.29187	0.31524	0.25808	0.13110
4	0.20546	0.28084	0.25701	0.20232	0.32702	0.24283	0.25029	0.22054	0.11068
5+	0.19748	0.24615	0.22363	0.18184	0.28713	0.21144	0.20794	0.19685	0.09805

In order to translate the EPDTA model into the PRISM model we proceeded ideally as formally specified in the semantics of EPDTA [1], with some optimisations. In particular, we used PRISM integer variables `month` and `year` to simulate the discrete clock  $x$  in the EPDTA model and we simulated the passage of time by inserting periodical transitions suitably incrementing these variables. Moreover, in the particular EPDTA model of the sole behaviour, non-determinism does not arise, thus yielding a DTMC instead of a Markov decision process, which is, in general, the formal semantics of an EPDTA. All the probabilities in the PRISM model were expressed as constants and additional states were added to allow the time elapse after 2013 or after the death of the sole<sup>1</sup>.

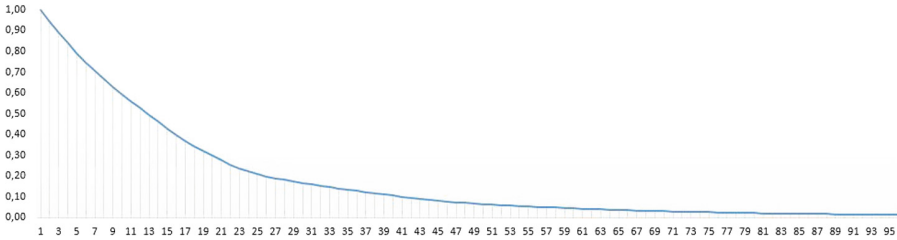
### 3 Discussion and Conclusions

The first formal analysis done on the model was to determine the survival trend of the sole during the whole period (from 2006 to 2013). This was done by exploiting the PRISM capability of calculating the probability of a given PCTL formula. In particular, the request

$$P = ? [F \text{!state} = 4 \ \& \ \text{!state} = 3 \ \& \ \text{year} = y \ \& \ \text{month} = m]$$

<sup>1</sup> The full PRISM model together with some variants with which we experimented is available at <https://dl.dropboxusercontent.com/u/33462615/Soles.zip>.

makes PRISM calculate the probability that a state in which the sole is alive, i.e. not fished (state 3) and not naturally dead (state 4), is reachable when the current year is  $y$  and the current month is  $m$ . Figure 3 shows the values calculated making the variables  $y$  and  $m$  vary on the period 2006–2013.



**Fig. 3.** Plot of the survival probabilities calculated from the PRISM model. On the  $x$  axis the months over the period 2006–2013.

A first interesting evaluation of this result is that the probability of survival becomes less than 0.5 just after 12 months and becomes less than 0.25 after 22 months. This is in accordance to the evaluation of fishery biologists that the stock under consideration is not only overfished, but also there is a huge pressure on juveniles which compromises the stock capacity of replenishment.

Another comparison of this result with the information coming from the same data but from other models is about *longevity*. Longevity is the maximum age reported in years that individuals of a given population would reach. Following [11], it can be calculated as the age at 95% of the asymptotic length of the fish, a parameter of the von Bertalanffy growth equation. Using Solemon data and this evaluation model, the longevity of the sole in the considered stock was estimated to be 6.36 years, i.e. about 76 months. However, the probability of survival at month 76 calculated by the model is 0.023946507, which means that only 2% of the whole population reaches the age of 76 months. Thus, the longevity of the stock seems overestimated by the classical model. As a future work, a probabilistic-based model for longevity could be devised using our approach.

Finally, an experiment that we performed was to determine how the fishing mortality could be reduced in order to have a survival probability greater than a certain value after a given number of months. This can be done by changing the probability constants and re-running the model on the period of interest. For instance, we found that a reduction of fishing mortality probabilities of 75% after 1 year and after 2 years (fishing probabilities in year 0 are already low because of the size of the fish) together with a reduction of 50% after 3 years, makes the survival probability after 4 years remain above 0.15. These kinds of experiments are relatively easy to do on the model and have the potential of revealing new information regarding the impact of fishing regulations.

As future work we plan to derive more information both from the classical and the statistical model checking of our model. Moreover, we are working on the problem of how to correctly compare the different results that we get from the three approaches followed in Fig. 1.

## References

1. Buti, F., Corradini, F., Merelli, E., Paschini, E., Penna, P., Tesei, L.: An individual-based probabilistic model for fish stock simulation. *Elect. Proc. Theor. Comput. Sci.* **33**, 37–55 (2010)
2. Christensen, V., Walters, C.J.: Ecopath with Ecosim: methods, capabilities and limitations. *Ecol. Model.* **172**(2–4), 109–139 (2004)
3. Grati, F., Scarcella, G., Polidori, P., Domenichetti, F., Bolognini, L., et al.: Multi-annual investigation of the spatial distributions of juvenile and adult sole (*Solea solea*, L.) in the Adriatic Sea (Northern Mediterranean). *J. Sea Res.* **84**, 122–132 (2013)
4. Jordán, F., Scotti, M., Priami, C.: Process algebra-based computational tools in ecological modelling. *Ecol. Complex.* **8**(4), 357–363 (2011)
5. Kwiatkowska, M., Norman, G., Parker, D.: PRISM: probabilistic symbolic model checker. In: Field, T., Harrison, P.G., Bradley, J., Harder, U. (eds.) *TOOLS 2002*. LNCS, vol. 2324, pp. 200–204. Springer, Heidelberg (2002)
6. Legay, A., Delahaye, B., Bensalem, S.: Statistical model checking: an overview. In: Barringer, H., et al. (eds.) *RV 2010*. LNCS, vol. 6418, pp. 122–135. Springer, Heidelberg (2010)
7. Nieto Coria, C.A., Tesei, L., Scarcella, G., Russo, T., Merelli, E.: Sea-scale agent-based simulator of *Solea solea* in the Adriatic Sea. In: Canal, C., Idani, A. (eds.) *SEFM 2014 Workshops*. LNCS, vol. 8938, pp. 259–275. Springer, Heidelberg (2015)
8. Penna, P.: *DISPAS: individual-based modelling and simulation for demersal fish population dynamics*. Ph.D. thesis, School of Advanced Studies, Doctoral course in Information science and complex systems, University of Camerino (2014)
9. Penna, P., Paoletti, N., Scarcella, G., Tesei, L., Marini, M., Merelli, E.: *DISPAS: an agent-based tool for the management of fishing effort*. In: Counsell, S., Núñez, M. (eds.) *SEFM 2013*. LNCS, vol. 8368, pp. 362–367. Springer, Heidelberg (2014)
10. Scarcella, G., Leoni, S., Grati, F., Polidori, P., Pellini, G., et al.: Stock assessment of common sole in GSA 17. Technical report, Stock assessemtn form for FAO-GFCM (2014)
11. Taylor, C.C.: Cod growth and temperature. *J. Conseil* **23**(3), 366–370 (1958)
12. von Bertalanffy, L.: A quantitative theory of organic growth (inquiries on growth laws II). *Hum. Biol.* **10**(2), 181–213 (1938)
13. VV. AA.: Report of the world summit on sustainable development. Technical report, United Nations (2002). [http://www.un.org/jsummit/html/documents/summit\\_docs/131302\\_wssd\\_report\\_reissued.pdf](http://www.un.org/jsummit/html/documents/summit_docs/131302_wssd_report_reissued.pdf)
14. VV. AA.: Common Fisheries policy. Technical report, European Commission (2014). [http://ec.europa.eu/fisheries/reform/index\\_en.htm](http://ec.europa.eu/fisheries/reform/index_en.htm)
15. VV. AA.: Fish stocks in Northeast Atlantic recover, whilst serious overfishing in Mediterranean: commission sets out plans for 2015 fishing opportunities. Technical report, European Commission (2014). [http://europa.eu/rapid/press-release\\_IP-14-724.en.htm](http://europa.eu/rapid/press-release_IP-14-724.en.htm)

# A Tool for the Modelling and Simulation of Ecological Systems Based on Grid Systems

Suryana Setiawan<sup>1,3</sup>, Antonio Cerone<sup>1,2</sup>(✉), and Paolo Milazzo<sup>1</sup>

<sup>1</sup> University of Pisa, Pisa, Italy

{setiawan,cerone,milazzo}@di.unipi.it, setiawan@cs.ui.ac.id

<sup>2</sup> IMT Institute for Advanced Studies, Lucca, Italy

antonio.cerone@imtlucca.it

<sup>3</sup> University of Indonesia, Jakarta, Indonesia

**Abstract.** Grid Systems is a formalism for modelling population and ecosystem dynamics that combines features of membrane computing, such as rewrite rules and maximal parallelism, with a representation of space similar to that of Cellular Automata. Moreover, Grid Systems include features for the description of environmental events and of events that can be associated with frequencies and durations that can be either deterministic or stochastic. The combination of all of these features makes Grid Systems a comprehensive formalism for the modelling and analysis of ecosystems.

This tool paper describes the implementation and the features of a simulator for Grid Systems. The simulator is equipped with a graphical user interface for defining and editing models of populations, and for simulating population dynamics and movement. The aim of this tool is to allow modellers to construct and analyse models based on a comprehensive and rigorous formalism such as Grid Systems with a friendly interface.

## 1 Introduction

In order to better understand how to preserve highly endangered species and plan actions of biodiversity conservation in complex ecological communities, social scientist need the support of effective tools, equipped with graphical user interfaces (GUI) that facilitate visual animation and visual presentation of the output [9]. Tools would help in modelling the link between local and global processes, simulating density dependence [7] and dealing with several other challenges of ecology.

Ecologists emphasised the importance of modelling demographic and environmental stochasticity in metapopulation dynamics [8], investigated fluctuations affecting the densities of populations in communities as a consequence of environmental variability [18], and analysed the effects of random perturbations on cyclic population dynamics [12]. We developed a simulator aiming to address these important needs by implementing the Grid Systems, a formal notation for modelling population dynamics, which was inspired by the concepts

of membrane computing, as in P systems [17], and spatiality dynamics, as in Cellular Automata (CA) [3] and spatial P systems [5, 6].

A Grid System consists of an envelop compartment (the outer membrane) containing a grid of adjacent inner compartments (inner membranes, also called cells). Each membrane, characterised by its position in the grid, may represent a distinct part of the environment with specific parameters and behavioural rules. Rules can move objects across membranes, thus describing the migration of resources or individuals.

Our simulator was developed using the Java Netbeans IDE. The latest version of the tool was updated to Java 7.6. A GUI was developed using JSwing and, for the graphical output, AWT. The models created by the user are stored in XML format. Details on the XML encoding can be found in the simulator documentation [20]. The simulator was tested on two case studies: population dynamics of a species of mosquitoes (*Aedes albopictus*) [4], and seasonal migration of a wildebeest species in the Serengeti National Park [21]. Such case studies will be used in this paper to describe the tool functionalities and to include screenshots of the main tool windows.

As regards related (and competitor) tools, we mention CoSBI Lab LIME [11], Ecopath with Ecosim (EwE) [15], DISPAS [16], and more general tools such as NetLogo [1] and tools for system dynamics [2]. All of these tools can be used to model and simulate ecosystems, with different levels of detail on the description of the individuals and of the environment. The Grid Systems simulator presented in this paper is a prototype tool based on a formalism that aims at making the description of events that may happen in the ecosystem unambiguous and easy. Moreover, the modelling language is designed with the aim of allowing many aspects of the ecosystem dynamics to be dealt with, such as, in particular, environmental events and spatial dynamics.

The rest of this paper is organised as follows. Section 2 introduces Grid Systems [4, 19], the formal notation on which the simulator is based. Section 3 describes the implementation by sketching the evolution algorithm that represents the engine of the tool and briefly presents the software architecture. Section 4 briefly introduces the two case studies used to show the tool features in action. Section 5 illustrates the features of the GUI, which include model definition and editing windows, presentation of the simulation output, visual animation and debugging. Section 6 concludes the paper.

## 2 Grid Systems

A grid system models space as a two dimensional grid of cells adopting the idea of cells in CA. Each cell, also called membrane, is addressed by an ordered pair of natural numbers: row number and column number. The term “membrane” comes from P Systems and the term “cell” comes from CA. Objects are represented by a set of unique symbols  $\Sigma$  and the state of a cell is represented by a multiset over  $\Sigma$ . Similar to CA and P Systems, the behaviour of Grid Systems is described by reaction rules, which are defined as in P Systems, with the additional features of

having a duration and modelling spatially dynamic behaviour. In order to model spatially dynamic behaviour, the applicability of the rules is defined by means of a set of associations of rules with membranes.

**Definition 1.** A Grid System  $G = (\Sigma, R, A, C^{(0)})$  is defined as follows:

- $G$  is the Grid System name;
- $\Sigma$  is a finite set of symbols representing the alphabet of object types;
- $R$  is a finite set of reaction rules (see Definition 2);
- $A = \{(\rho, \gamma) \mid \rho \in R, \gamma \in \{G_{i,j} \mid i, j \geq 0\} \cup \{G_E\}\}$  is the set of associations of the rules with the membranes, where:
  - $G_{i,j}$  denotes the cell at position  $(i, j)$ , called local membrane;
  - $G_E$  is the global membrane surrounding the cells;
- $C^{(0)}$  is the initial configuration of the Grid System (see Definition 3).

## 2.1 Reaction Rules

The behaviour of a Grid System is defined using reaction rules that rewrite a given multiset of objects into a new multiset of objects.

**Definition 2.** Let  $\Sigma$  be the alphabet in a Grid System. A rule  $\rho$  is a relation of multiset  $\alpha$  giving multiset  $\beta$  under conditions given by parameters  $\psi, \chi, c, d, X$ , and written as

$$\rho : \alpha \xrightarrow[d, X]{c} \beta \ [\psi \mid \chi]$$

where

- $\rho$  is the unique identifier of the rule;
- $\alpha$  is a non-empty multiset of reactants,  $\alpha$  is a multiset over  $\Sigma$  and  $\alpha \neq \lambda$ ;
- $\beta, \psi$  and  $\chi$  are multisets over  $\Sigma$  of products, promoters and inhibitors, respectively. Elements of these multisetes are possibly associated with coordinates of other membranes
- $c \in \mathbb{R}^+$  is the rate with which the rule may be applied to perform a reaction;
- $d \in \mathbb{R}^+$  is the (exact or mean) duration of the reaction;
- $X \in \{‘D’, ‘M’\}$  is a marking to the rule; ‘D’ indicates that the duration of the reaction will take exactly  $d$  time units when it is applied; and ‘M’ indicates that the duration time is an exponentially distributed random variable that has mean value  $d$  time units.

## 2.2 Configurations and Evolution Algorithm

A configuration  $C^{(t)}$  is the state of the system at time point  $t$ . It consists of two parts: current objects and current ongoing reactions. The current objects are represented as the multisets of existing objects in each membrane. The ongoing reactions are the reactions that have been applied but due to their durations they have not been accomplished. For its semantic purpose, the list of ongoing reactions is sorted according to reactions’ termination time.

**Definition 3.** A Configuration  $C^{(t)}$  is a pair

$$(\{C^{(t,m)} \mid m \in \{G_{i,j} \mid i, j \geq 0\} \cup \{G_E\}\}, \Omega^{(t)})$$

where

- $C^{(t,m)}$  is the multiset over objects that exist inside membrane  $m$  at time  $t$ ;
- $\Omega^{(t)} = \{(r_k, t_k, m_k) \mid k = 1, \dots, n \text{ and } t \leq t_1 \leq \dots \leq t_n\}$  is the set of ongoing reactions where each  $(r_k, t_k, m_k)$ ,  $k = 1, \dots, n$ , denotes ongoing reactions that instantiate rule  $r_k$  in membrane  $m_k$  and will terminate at time  $t_k$ .

The semantics of Grid Systems is described through an algorithm called Evolution Algorithm of Grid Systems. The algorithm starts from a given initial configuration  $C^{(0)}$  in which  $\Omega^{(0)}$  is assumed to be an empty set. The configuration at time point  $t_k$  is denoted by  $C^{(t_k)}$ . It contains all the objects and the membranes where they are located, and a list of on-going reactions:  $C^{(t_k,m)}$  represents the objects in membrane  $m$  and  $\Omega^{t_k}$  represents the list of ongoing reactions. Moreover, configuration  $C^{(t_k,m)}$  contains two multisets over objects:  $Avail^{(t,m)}$  and  $Committed^{(t,m)}$ . The former represents the objects that are available for the next reactions and the latter represents the objects that are already involved in some ongoing reactions. In addition to these dynamic aspects, the configuration also contains some static entities: the list of objects  $\Sigma$ , the list of reactions rules  $R$  and the association list  $A$ .

### 2.3 Links

Living species have been given by nature the ability to sense and follow the pathways for movements. Pathways can either result from physical perceptions (salmons sense the geomagnetic field [14] and sperm cells sense chemotaxes to locate the ovum [13]) or cognitively created by the individual (wood ants memorize snapshot views and landmarks [10]). In Grid Systems pathways are modelled using links. A link is defined as a special object that carries pointers. A pointer is a piece of information that provides a dynamic addressing of a destination membrane. The pointers can be used by rules in referring to the objects in another cell. Different pointers carried by a link introduce further non-determinism into the system. In order to resolve this form of non-determinism, a decision is made stochastically based on the weight of each pointer. Weights are real numbers between 0 and 1.0, and the total weight in the same cell is 1.0. Like ordinary objects, the number of links in a cell can be increased or decreased by applying its related rule.

**Definition 4.** A pointer is an ordered pair of integers. There are two types of pointers: relative pointers and absolute pointers. For the relative pointer the pair of integers is marked by curved brackets, as “ $(a, b)$ ”, where  $a, b \in \mathbb{Z}$ . For the absolute pointer the pair of integers is marked by squared parentheses, as “ $[r, c]$ ”, where  $r, c \in \mathbb{N}$ .

Links are definitely objects in Grid Systems. Objects are called links when they carry at least one pointer. When they carry several pointers, the pointers are equipped with weights that represent a form of priority.

**Definition 5.** *G is a Grid System extended with links when any object in  $\Sigma$  can carry pointers. A link is an object that carries at least one pointer.*

The use of links requires rule replication, that is, each rule has to be instantiated by an actual rule in which the link is replaced by its pointer. When a link of the rule has more than one pointer, the rule is instantiated by one actual rule for each pointer. Furthermore, one rule may have several different links. Replications for such a rule are based on the combinations of the pointers of each link.

### 3 Implementation

The semantics of Grid Systems is described operationally in terms of the Evolution Algorithm of Grid Systems. Such operational semantics has been the basis for our implementation. However, since the operational semantics is given as a recursive mathematical function, we need to identify the key aspects that have to be reworked to implement it using an imperative programming language. First, we need to categorise Grid Systems in terms of the types of reaction rules used:

- *L0 (Stepwise Rule) Grid Systems*, whose reaction rules are in the form

$$\rho : \alpha \xrightarrow{D} \beta [\psi \mid \chi].$$

The absence of values for  $c$  and  $d$  on the arrow denotes that rate and duration take default value 1.

- *L1 (0-1 Rule) Grid Systems* include L0 Grid Systems and the ones whose rules are in forms

$$\rho : \alpha \xrightarrow[1,D]{0} \beta [\psi \mid \chi].$$

Each rule has duration of either 0 or 1 time unit.

- *L2 (Stochastic) Grid Systems* have all possible types of reaction rules as in Definition 2.

Second, we have to distinguish between *Grid Systems without links* and *Grid Systems with links*.

Third, although Grid Systems are unbounded, as for Definition 2, in our implementation we consider the dimension of Grid Systems bounded to  $N \times M$  cells. Therefore, we will use *bounded Grid Systems*

$$G = (N, M, \Sigma, R, A, C^{(0)}),$$

where  $N, M \in \mathbb{N}$ , such that the number of membranes is  $N \times M$  and the association set is  $A = \{(\rho, \gamma) \mid \rho \in R, \gamma \in \{G_{i,j} \mid 0 \leq i < N, 0 \leq j < M\} \cup \{G_E\}\}$ .

Finally multisets of the configurations are represented as tables. For example,  $\Omega$  is represented by  $\Omega$ -table.

In Sect. 3.1 we sketch the implementation of the evolution algorithm. In Sect. 3.2 we outline the software architecture of the simulator. Full details of the implementation are available in the first author's PhD thesis [19].



### 3.1 Evolution Algorithm Implementation

Let  $C^{(t_k)}$  be the current configuration where  $Avail^{(t_k, m)}$  is the multiset of the objects that are available for the next reactions and  $Committed^{(t_k, m)}$  is the multiset of the objects that are already involved in some ongoing reactions. The following steps are performed iteratively.

#### 1. Find Reaction Candidates

For each membrane  $m$ , if  $Avail^{(t_k, m)} = \emptyset$ , then the algorithm continues on the next membrane. Otherwise, the multiset  $Cand^{(t_k, m)}$  of the candidate reactions is calculated as follows.

1. for each rule  $r$ , calculate the number of times  $\text{mult}(r, m)$  that rule  $r$  is applicable in  $m$  with maximum parallelism;
2.  $C0 := \{(r, \text{mult}(r, m)) \mid r \in \text{assoc}(m) \text{ and } \text{mult}(r, m) > 0\}$ ;
3. partition  $C0$  into multiset  $Cn$  of the rules that are involved in non-determinism and multiset  $Cd$  of the other rules;
4. calculate multiset  $Cs$  of the selected rules by resolving non-determinism in  $Cn$ , using a stochastic rule selection adapted from Gillespie's Stochastic Simulation Algorithm;
5.  $Cand^{(t_k, m)} := Cd \cup Cs$ .

*Grid Systems with links* require some additional calculations.

#### 2. Add New Ongoing Reactions to $\Omega$ table

For each rule  $r$  and membrane  $m$  such that  $(r, \text{mult}(r, m)) \in Cand^{(t_k, m)}$ , a number  $\text{mult}(r, m)$  of elapse times  $\text{elapse}(r, m, i)$ , with  $i = 1, \dots, \text{mult}(r, m)$ , is calculated as follows:

- for *L0 Grid Systems*,  $\text{elapse}(r, m, i) := 1$ ;
- for *L1 Grid Systems*,  $\text{elapse}(r, m, i)$  equals the duration of rule  $r$  (0 or 1);
- for *L2 Grid Systems*, if  $r$  is a deterministic duration time rule, then  $\text{elapse}(r, m, i)$  returns the duration of  $r$ , otherwise ( $r$  has an exponentially distributed duration time)

$$\text{elapse}(r, m, i) = -\text{duration}(r) \ln X_i$$

where  $X_i \sim U[0, 1]$  ( $X$  is a uniformly distributed random variable).

For each  $(r, n) \in Cand^{(t_k, m)}$  and  $i = 1, \dots, n$ , build the multiset  $\Omega^+$  of triples  $(r, t_k + \text{elapse}(r, m, i), m)$ . Update the table of ongoing reactions  $\Omega$ -table by inserting the triples from  $\Omega^+$ .

For *Grid Systems with links*, rules are replicated as described in Sect. 2.3.

#### 3. Determine the Earliest Reactions to Remove from $\Omega$ -table

For *L0 Grid Systems*,  $t_{k+1} := t_k$ .

For *L1 Grid System*, if there is a new reaction with duration 0, then  $t_{k+1} := t_k$ , otherwise  $t_{k+1} := t_k + 1$ .

For *L2 Grid System*,  $t_{k+1}$  is the minimum  $t$  such that  $(r, t, m)$  has been added to  $\Omega$ -table.

Calculate the multiset  $\Omega^-$  of the triple  $(r, t, m)$  in  $\Omega$ -table such that  $t \leq t_{k+1}$ .

#### 4. Generate Products, Remove Reactants and Terminated Reactions

For each membrane  $m$ , calculate

- the new multiset  $Avail^{(t_{k+1},m)}$  from  $Avail^{(t_k,m)}$  by removing all reactants of reactions in  $\Omega^+$  and adding all products of reactions in  $\Omega^-$ ;
- the new multiset  $Committed^{(t_{k+1},m)}$  from  $Committed^{(t_k,m)}$  by removing all reactants of reactions in  $\Omega^-$  and adding all reactants of reactions in  $\Omega^+$

Update the table of ongoing reactions  $\Omega$ -table by removing the triples in  $\Omega^-$ . If  $\Omega$ -table is empty and, for each membrane  $m$ , no rule is applicable in  $C^{(t_{k+1})}$ , then the algorithm terminates. Otherwise, the next iteration of the algorithm starts from **Step 1**.

In terms of efficiency, for *L2 Grid Systems*, the  $\Omega$ -table is implemented by using a priority queue (heap tree) data structure, which supports insertion and removal in logarithmic time. In this way the performance is still acceptable in presence of a large number of ongoing rules.

### 3.2 Software Architecture

Models are encoded in XML packages. The simulator also uses two internal data representations: raw data tables, for editing functionalities, and actual data table, for running the simulation. The software architecture of the simulator consists of three modules:

**XML Data Handler Module.** It is the layer that accesses XML packages.

It provides methods for loading the data tables from the XML structures, updating the structures and load from or save to files.

**Editor Module.** It contains editing interfaces and methods for verifying the input. In general, each interface is specialised to handle the editing of a specific part of the model. Then, when the interface is closed and the update is accepted, the editor module accesses the XML data handler for updating that part. Finally, it updates related tables and sends the update to the XML data handler module. The interfaces implemented by this module define the four editing modes described in Sect. 5.1

**Simulation Module.** It implements the Evolution Algorithm of Grid Systems and visualises the results. There are three simulation modes, which are described in Sect. 5.2

## 4 Case Studies

In order to describe, in Sect. 5, the model definition and simulation functionalities, we consider two case studies of population dynamics that have been modelled by means of Grid Systems in our previous work [4, 21].

The first case study [4] consists in modelling the dynamics of a population of a species of mosquitoes, *Aedes albopictus*. The model includes objects that represent population individuals at different development phases: egg ( $E$ ), immature ( $I$ ), i.e. pupa/larva, and adult ( $A$ ). Development takes place in small

water containers, and it is influenced by water level and environmental temperature. Hence, the model considers three types of external events, temperature change, rainfall, and desiccation, which change the behaviour of the species either directly or indirectly. The ecosystem is modelled as a  $3 \times 3$  grid. Temperature is represented by the number of objects  $T$  in the global membrane, while the water level in containers is modelled by the number of objects  $W$  in the central cell of the grid. Reaction rules of this model include rules for the movement of adults into adjacent cells, rules for oviposition, rules for development ( $E$  into  $I$ , and  $I$  into  $A$ ) and rules for death.

The second case study [21] consists in the modelling of the seasonal migration of a wildebeest species in the Serengeti National Park, Tanzania. The area of the park is modelled as a  $50 \times 50$  grid. In order to precisely describe the shape of the territory, dummy objects  $Z$  are inserted in grid cells that are not included in the representation of the park. In the other cells, the instances of object  $G$  represent the quantity of available grass, objects  $A1, \dots, A9$  and  $B1, \dots, B9$  represent wildebeests at different stages and in different states (movable/resting). Instances of object  $C$  are used (for modelling purposes) to maintain the total number of individuals in the population.

The model relies on the observations that wildebeest migration is driven by the search for grazing areas and water resources, and individuals tend to follow movements of other individuals. Assuming the existence of dynamic guiding paths, which could be representations of individual or communal memory of wildebeests, or physical tracks marking the land, we model movement by rewritings between adjacent cells driven by conditions in the origin and destination cells. As conditions we consider number of individuals, grass available, and dynamic paths. Paths are initialised with the patterns of movements observed in reality, but dynamically change depending on variation of movement caused by other conditions. In the grid systems model, paths are represented by objects *path* carrying links that are dynamically created by reaction rules, and that are used by other rules to determine the movement of wildebeests. Reaction rules of the model include rules for birth and death of individuals, for grass growth (that uses auxiliary objects  $R$  and  $H$  to describe grass growth phases), for feeding of individuals, for change of stage and state of individuals and for movement according to grass availability and paths. Moreover, reaction rules for the update of paths are present.

## 5 Features of the GUI

### 5.1 Model Definition and Editing

In this section we briefly illustrate how to define and edit a model by showing screenshots that refer to the *Aedes albopictus* case study [4]. Full details are available in the simulator documentation [19]. There are four editing modes that allow the user to define and modify a model.

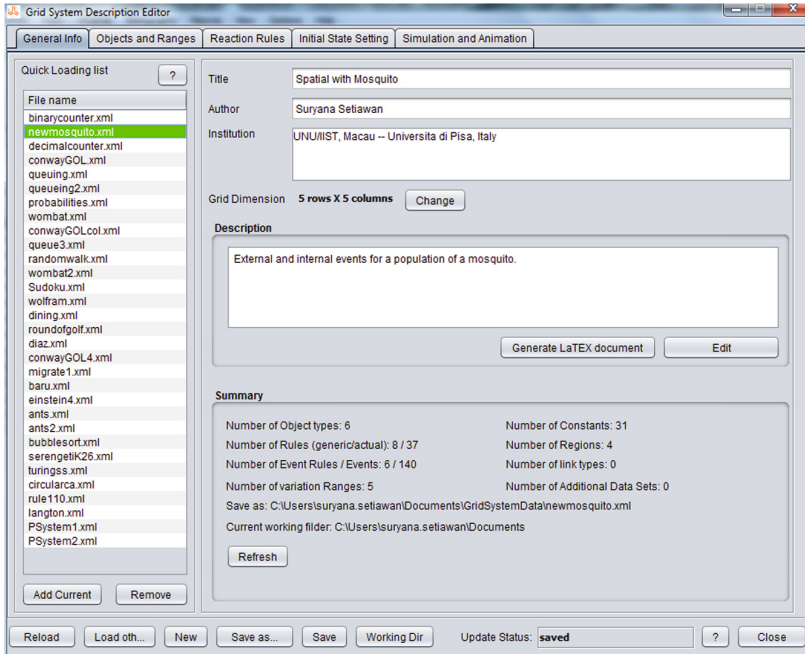


Fig. 1. The Dashboard: “General Info” Mode

**“General Info” Mode.** This mode allows the user to create a new model or select the model to be analysed from a list of models. The user can access and edit the model general information. As shown in Fig. 1, the list of models is on the left side while the editable information about the selected model is visualised on the right side. It is also possible to generate a  $\text{\LaTeX}$  description of the model.

**“Objects and Ranges” Mode.** This mode allows the user to define the objects and the topology of the model. In particular, the following entities can be created and edited:

*object* which defines a species or a developmental stage of a species (e.g. Adult Mosquito, Egg, Pupa/larva), an environmental condition (e.g. Temperature level and Water level) or a movement constraint (e.g. Mosquito movement blocker);

*constant* which describes fixed values such as thresholds and can be an expression containing other defined constants and the values associated with the grid, such as number of rows (NumRows) and number of columns (NumCol);

*region* which is interactively defined by means of a dialog box by selecting cells and incorporating already defined regions over the canvas that represents the grid;

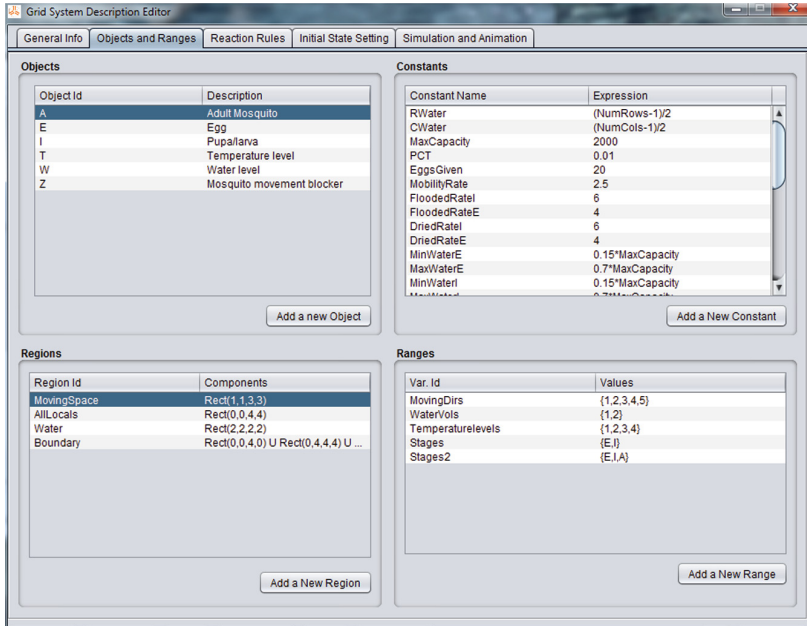


Fig. 2. The Dashboard: “Objects and Ranges” Mode

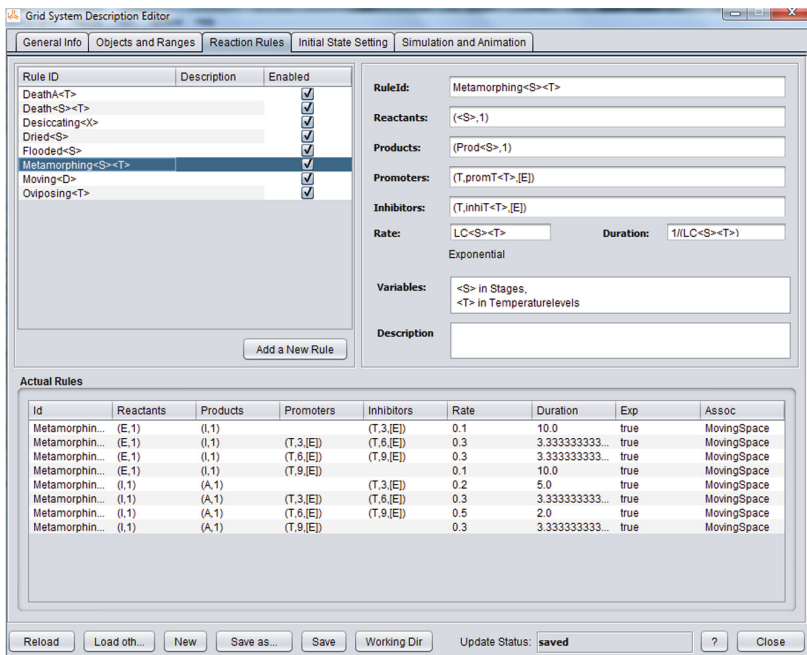


Fig. 3. The Dashboard: “Reaction Rules” Mode

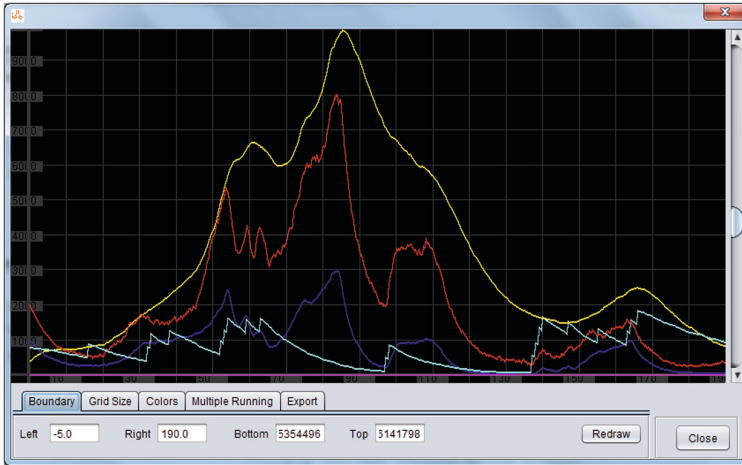


Fig. 4. Output as Chart

*range* which is defined as a set of discrete values associated to variables.

The dashboard of this mode is shown in Fig. 2.

**“Reaction Rules” Mode.** This mode allows the user to define reaction rules.

As shown in Fig. 3 the dashboard supports the definition of the rules according to Definition 2.1, the selection of the rules to be used in the simulation and visualises, in a tabular form, all actual rules corresponding to a highlighted template rule.

**“Initial State Setting” Mode.** This mode allows the user to define the list of datasets for different initial objects, links, external events and rules for events.

## 5.2 Simulation and Animation

Simulation can be performed using three different modes as follows.

**“Growth” Mode.** This mode is intended for observing how the configuration changes during simulation. There are two different ways to observe such changes: (1) as textual output, and (2) as a chart. The chart presents a graphical output using different colours for different objects as shown in Fig. 4 for a mosquito population (adults in yellow, immatures in dark blue and eggs in red) and its environment (water level in light blue) [4].

**“Animation” Mode.** Animating the simulation is essential to observe spatial aspect. This mode supports the visualisation of object changes in each membrane. It is illustrated on a model of the seasonal migration of a wildebeest species in the Serengeti National Park, Tanzania [21]. Figure 5 shows a map of the Serengeti National Park, with the white part, consisting of tiny dots, showing the density of the individuals of the species.

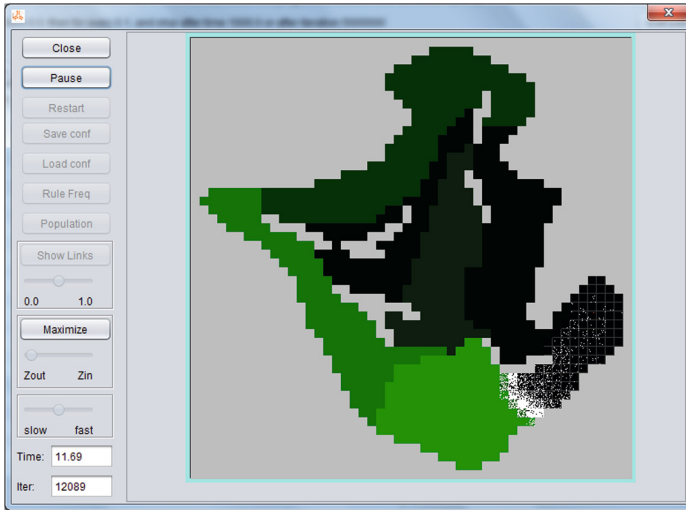


Fig. 5. Animation

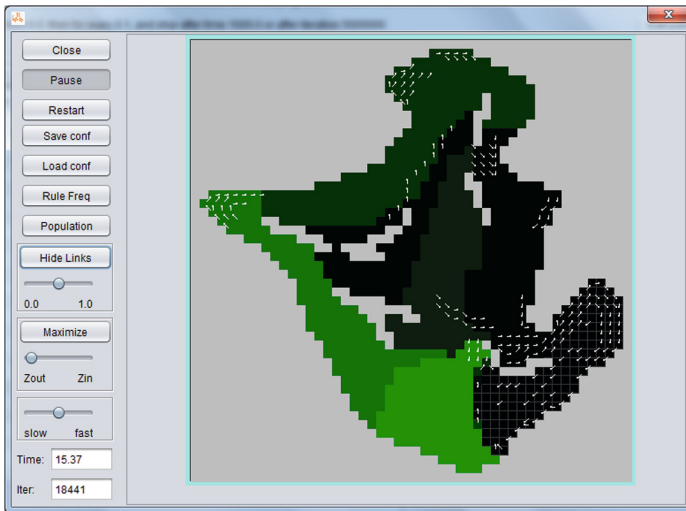


Fig. 6. Displaying the Links

The animation can be paused with button “Pause” (as in Fig. 6) to observe an instantaneous state, possibly saving the current configuration in a file (button “Save conf”) in order to continue at a later stage (button “Load conf”), and can be restarted from the initial configuration (button “Restart”). When the animation is paused it is possible to inspect rule frequency (for both template and actual rules) with button “Rule Freq” and object numbers (available, committed and total) with button “Population”.

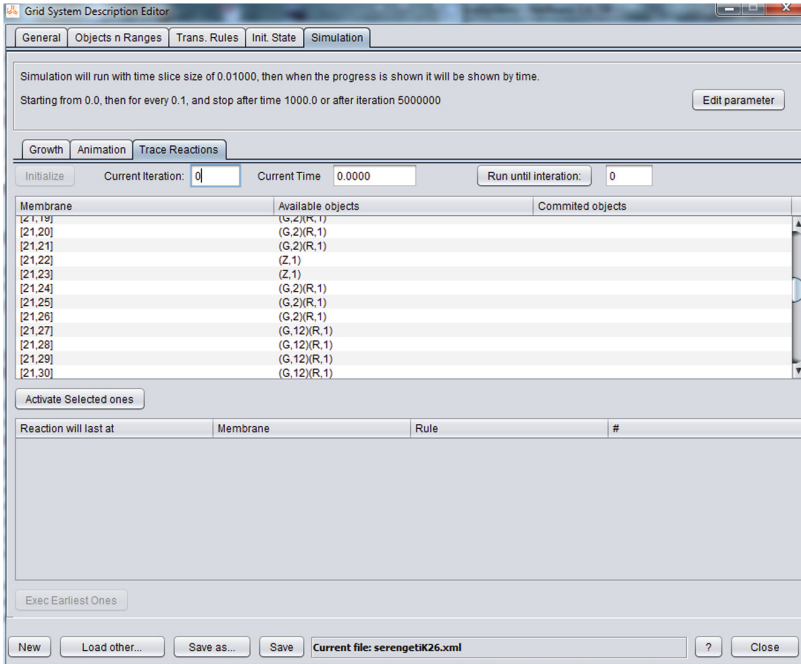


Fig. 7. The Dashboard: “Trace Reactions” Mode

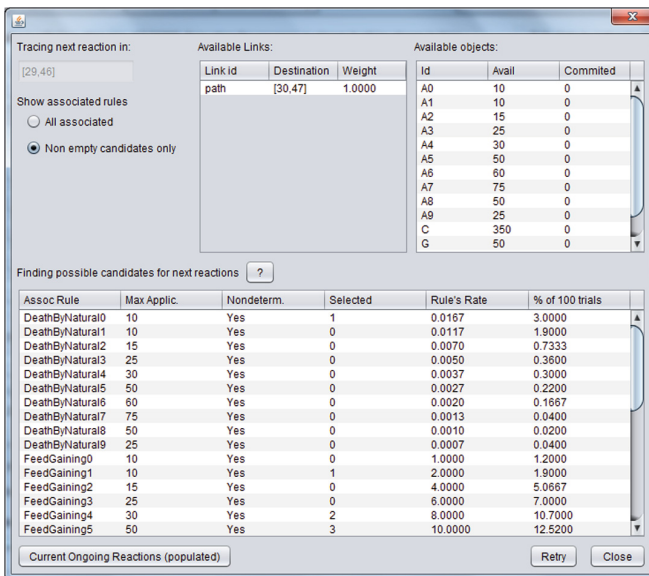


Fig. 8. State of a Membrane



It is also possible to visualise the current links as arrows on the canvas (buttons “Show Links” and “Hide Links”), possibly with the objects hidden (button “Only Links”, which appears after pushing “Show Links”). Since links have different directions and each direction has its own weight, the interface provides a sliding bar to set the weight limits so that only directions with weights above the limits (high priority links) are visualised. Visualisation of links with objects hidden is shown in Fig. 6 for the wildebeest case study. Finally, frames of animation can be captured and stored in files.

**“Trace Reactions” Mode.** This debugging mode supports the modeller in observing the behaviour of the rules step-by-step during the simulation. As shown in Fig. 7, in this mode, the initial dashboard shows, for each membrane, the available and committed objects it contains. After selecting a membrane, it is possible to inspect its state, represented in textual format organised in tables. The state of a membrane (for rules with available reactants, as shown in Fig. 8, or for all associated reaction rules) includes:

- the available and committed objects;
- the current paths, including their destinations and weights;
- for each considered rule associated with the membrane;
  - the identifier of the rule for that reaction;
  - the maximum number of times the rule can be applied;
  - the existence of other rules competing for the same reaction;
  - the number of reactions resulting from the stochastic selection process applied to the rule;
  - the rule rate;
  - the percentage of trials, that is, the empirical likelihood that the rule is selected out of 100 trials;

## 6 Conclusion and Future Work

We have presented a simulator for ecological systems that is equipped with a GUI that supports visual animation and visual presentation of the output. We have illustrated the modelling, editing and analytical functionalities of the tool presenting screenshots from two case studies: the dynamics of a population of a species of mosquitoes, *Aedes albopictus*, and the seasonal migration of a wildebeest species in the Serengeti National Park, Tanzania. Simulator and documentation can be downloaded at <http://www.di.unipi.it/msvbio/wiki/GridSystems/>.

In our future work we plan to apply the simulator to other case studies in ecology and aim to extend it with further analytical functionalities based on model checking and statistical model checking.

## References

1. NetLogo web site. <https://ccl.northwestern.edu/netlogo/>
2. Tools for system dynamics web site. <http://tools.systemdynamics.org/>
3. Adamatzky, A.: Identification of Cellular Automata. Taylor & Francis, London (1994)
4. Barbuti, R., Cerone, A., Maggiolo-Schettini, A., Milazzo, P., Setiawan, S.: Modelling population dynamics using grid systems. In: Cerone, A., Persico, D., Fernandes, S., Garcia-Perez, A., Katsaros, P., Ahmed Shaikh, S., Stamelos, I. (eds.) SEFM 2012 Satellite Events. LNCS, vol. 7991, pp. 172–189. Springer, Heidelberg (2014)
5. Barbuti, R., Maggiolo-Schettini, A., Milazzo, P., Pardini, G.: Simulation of spatial P system models. *Theor. Comp. Sci.* **529**, 11–45 (2014)
6. Barbuti, R., Maggiolo-Schettini, A., Milazzo, P., Pardini, G., Tesei, L.: Spatial P systems. *Nat. Comput.* **10**(1), 3–16 (2011)
7. Björnstad, O.N., Fromentin, J.M., Stenseth, N.C., Gjøsæter, J.: Cycles and trends in cod populations. *Proc. Nat. Acad. Sci. U.S.A* **96**, 5066–5071 (2009)
8. Bonsall, M.B., Hastings, A.: Demographic and environmental stochasticity in predator-prey metapopulation dynamics. *J. Anim. Ecol.* **73**, 1043–1055 (2004)
9. Cerone, A., Scotti, M.: Research challenges in modelling ecosystems. In: Canal, C., Idani, A. (eds.) SEFM 2014 Workshops. LNCS, vol. 8938, pp. 276–293. Springer, Heidelberg (2015)
10. Durier, V., Graham, P., Collett, T.S.: Snapshot memories and landmark guidance in wood ants. *Curr. Biol.* **13**, 1614–1618 (2003). Elsevier Science Ltd
11. Kahramanoğulları, O., Jordán, F., Lynch, J.: Cosbilab lime: a language interface for stochastic dynamical modelling in ecology. *Environ. Model. Softw.* **26**(5), 685–687 (2011)
12. Kaitala, V., Ranta, E., Lindstroem, J.: Cyclic population dynamics and random perturbations. *J. Anim. Ecol.* **65**, 249–251 (1996)
13. Kaupp, U.B., Kashikar, N.D., Weyand, I.: Mechanism of sperm chemotaxis. *Annu. Rev. Physiol.* **70**, 93–117 (2008)
14. Lohmann, K.J., Putman, N.F., Lohmann, C.M.F.: Geomagnetic imprinting: a unifying hypothesis of long-distance natal homing in salmon and sea turtles. *Proc. Nat. Acad. Sci.* **105**(49), 19096–19101 (2008)
15. Pauly, D., Christensen, V., Walters, C.: Ecopath, ecosim, and ecospace as tools for evaluating ecosystem impact of fisheries. *ICES J. Mar. Sci.* **57**(3), 697 (2000)
16. Penna, P., Paoletti, N., Scarcella, G., Tesei, L., Marini, M., Merelli, E.: DISPAS: an agent-based tool for the management of fishing effort. In: Counsell, S., Núñez, M. (eds.) SEFM 2013. LNCS, vol. 8368, pp. 362–367. Springer, Heidelberg (2014)
17. Păun, G.: Computing with membranes. *J. Comput. Syst. Sci.* **61**(1), 108–143 (2000)
18. Ripa, J., Ives, A.R.: Food web dynamics in correlated and autocorrelated environments. *Theor. Popul. Biol.* **64**, 369–384 (2003)
19. Setiawan, S.: Formal modelling for population dynamics. Ph.D thesis, Department of Computer Science, University of Pisa, May 2015
20. Setiawan, S.: The grid systems simulator (version date: 3 June, 2015) 2015. <http://www.di.unipi.it/msvbio/wiki/GridSystems/>
21. Setiawan, S., Cerone, A.: Stochastic modelling of seasonal migration using rewriting systems with spatiality. In: Counsell, S., Núñez, M. (eds.) SEFM 2013 Workshops. LNCS, vol. 8368, pp. 313–328. Springer, Heidelberg (2014)

**VERY\*SCART 2015**

# Distributed Coordinated Adaptation of Cloud-Based Applications

Luciano Baresi, Sam Guinea, and Giovanni Quattrocchi<sup>(✉)</sup>

Politecnico di Milano Dipartimento di Elettronica, Informazione e Bioingegneria,  
Piazza L. da Vinci, 32, 20133 Milano, Italy  
{luciano.baresi,sam.guinea,giovanni.quattrocchi}@polimi.it

**Abstract.** Steering modern Internet applications in the Cloud, given a set of functional and non-functional requirements, is a complex task. System maintainers need to have a holistic view of the application; they need to understand the intricate horizontal and vertical dependencies that exist between the infrastructure, platform, and software constituents. In this paper we advocate that MAPE control loops can help, and we focus on coordination of multiple adaptation actions. To this end we have developed a simple language for describing the adaptation capabilities of an Internet application. We then use this description to understand the dependencies that exist among the different adaptations we want to execute. Finally, we provide a distributed framework that, given a complex adaptation plan, helps our actuators collaborate in a decentralized fashion. We have validated our approach on an on-line auction application, deployed onto a mix of physical servers and Amazon EC2 virtual machines.

## 1 Introduction

Dynamic resource allocation techniques in the Cloud have eased the creation of Internet applications that scale on demand, according to real needs. *Horizontal scalability* allow us to add new computational resources in a relative short amount of time, allowing applications to grow with the illusion of infinite scalability [12].

Although Infrastructure as a Service (IaaS) providers, such as Amazon AWS and OpenStack, offer some automated tools for achieving horizontal scalability, runtime adaptation remains a complex problem. The current state of practice is to focus on increasing or reducing the number of computational resources dedicated to an application (e.g., AWS Autoscaling Groups); this is, at best, a partial solution. We also need to consider the scaling capabilities of the middleware and software components that are deployed onto these computational resources and the dependencies that may exist between them. Platform as a Service (PaaS) providers deal with these issues by offering a completely automated

---

The work presented in this paper has been partially supported by project EEB - Edificio A Zero Consumo Energetico In Distretti Urbani Intelligenti (Italian Technology Cluster For Smart Communities) - CTN01.00034.594053.

platform, but they usually lock-in the users with proprietary technologies and greatly reduce the configuration and personalization of the product.

In this paper we present a general-purpose solution for adapting applications that are hosted on private, public, and hybrid clouds. The work we present is part of our ongoing ECoWare initiative. ECoWare is a framework for enriching Internet and Service-based applications with self-\* capabilities; it is based on the classical notion of MAPE (Monitoring, Analysis, Planning, and Execution) control loops [9]. The main peculiarity of ECoWare is that it provides a holistic approach: it considers all the different parts of the application to be intimately interconnected, meaning it considers all the vertical and horizontal dependencies that may exist therein. Vertical dependencies concern the relationships between software components of a single service (e.g. a JBoss server depends on the JVM); horizontal dependencies refer to the interconnections between different services (e.g. a JBoss server has to be connected to a MySQL database). ECoWare is aware of the design-time description of the application, how it is deployed, and how it evolves at runtime.

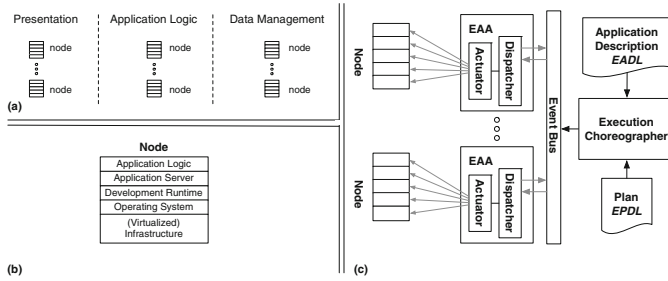
After presenting the monitoring and analysis capabilities of ECoWare in [5], and its planning capabilities in [13], here we concentrate on the Execution phase. This phase starts when we receive an adaptation plan that consist of a list of adaptation actions that we want to perform on the application. These actions can refer to the infrastructural resources (e.g., the VMs, OS containers, etc.), the platform resources (e.g., application servers, database management systems, etc.), and/or the actual deployed business logic.

The main contributions of this paper thus are (i) a language for describing the application and its adaptation capabilities called EADL (EcoWare Application Description Language), (ii) a language for specifying adaptation strategies called EPDL (EcoWare Plan Description Language), and (iii) a supporting distributed and decentralized runtime adaptation framework. We evaluated our solution in the context of an online auction site that was deployed onto a hybrid cloud consisting of a mix of 6 physical servers and 300 Amazon EC2 virtual machines.

The rest of this paper is organized as follows. Section 2 introduces our solution. Section 3 presents both the EADL and the EPDL languages that we have developed. Section 4 explains the algorithm that we use to create a distributed adaptation choreography, given a specific EPDL plan, and presents the runtime framework we have developed for executing the choreographies. Section 5 describes the experiments we ran to evaluate our solution. Section 6 surveys the state of the art, and Sect. 7 concludes the paper.

## 2 Solution Overview

Before we start discussing our solution, we need to clarify the kind of applications we focus on. Modern Internet applications are often built by following a multi-tier architecture. **Tiers** allow us to logically and physically separate software components that deal with different functional aspects of an application. Figure 1(a) illustrates a common and simple example in which we have three tiers: a presentation tier, an application logic tier, and a data management tier.



**Fig. 1.** (a) A typical 3-tier application; (b) The multi-level nature of a node; (c) Coordination of distributed adaptations

Each tier in our application contains at least one **node**; depending on the workload that we experience at runtime, we can dynamically add, or remove, nodes to or from a tier. A node is a computing instance (e.g., a VM) running software components (e.g., the JVM and an application server) that enable the execution of app-specific code (e.g., a JEE application) or the management of application data. An example of node is showed in Fig. 1(b). For the sake of simplicity, we require that nodes within the same tier have the same adaptation capabilities. Finally, the nodes that exist in the application, across all tiers and at any given time, make up what we call the application’s **topology**.

Since adaptation actions are provided by technology- and application-specific actuators, which are deployed directly onto the system, we enact the adaptation plan in a decentralized fashion, i.e., through a distributed *adaptation choreography*. This allows us to avoid having a central point of failure, and to optimize network usage by favoring local and autonomous adaptation as much as possible. In general, centralized solutions are not desirable in the presence of different organizational boundaries, which is quite common in hybrid and public clouds.

Figure 1(c) provides an overview of our coordinated adaptation framework. At the center of our solution lies the **Execution Choreographer (EC)**. It is responsible for receiving a plan, written in EPDL, and for creating an adaptation choreography that can be executed by a distributed set of **ECoWare Adaptation Agents (EAAs)**. The plan it receives could be manually generated by a domain expert, or passed to the EC by ECoWare planing component [13]. In order to understand the dependencies that exist among the various adaptation actions, and to inform the EAAs, the EC is pre-configured with a description of the application, written in EADL.

The adaptation choreography is created by using the **Choreography Creation Algorithm (CCA)** discussed in Sect. 4.1. The EAA wraps the application and technology-specific **Actuator** that we associate with a specific node. In our solution we have one EAA per node. The main responsibility of the EAA is to coordinate its adaptation activities with those performed by other agents. This means sharing information, and synchronizing adaptations with other EAAs. This is achieved through the **Dispatcher** which is connected to the distributed **Event Bus**.

**Listing 1.1.** Example App Description.

```

app:
  tier Loadbalancer: minq: 1 maxq: 1
  actions:
    alloc(): requirements:
      all AppServer where started==true
    updateServerList(): requirements:
      all AppServer where started==true
    setAlgorithm(String): critical
  tier AppServer: minq: 1 maxq: 10
  attributes: port
  actions:
    alloc(): requirements:
      any DB where started==true
    changeDatabase(DB): critical
    setMaxThreads(Num): critical
    setDBPoolSize(Num, Num): critical
  tier DB: minq: 1 maxq: 1
  attributes: port, user, password
  actions:
    alloc():
      modifies: port, user, password
    setMaxConnection(Num) : critical
    setMemLock(Boolean): critical

```

### 3 Models and Languages

In this section we will present ECoWare Application Description Language (EADL), our language for describing adaptation capabilities of an application, and ECoWare Plan Description Language (EPDL), our language for defining an adaptation plan.

#### 3.1 ECoWare Application Description Language

Our application description language was inspired by the OASIS Topology and Orchestration Specification for Cloud Applications (TOSCA) [6]. TOSCA focuses on providing interoperable descriptions of application and infrastructure cloud services, of their relationships, and of their operational behaviors. TOSCA, unfortunately, does not focus on runtime adaptation; in particular, it does not help define how multiple actions need to collaborate to reach global adaptation. As a result we kept some of its main (architectural) ideas and developed EADL, to focus on the dependencies that exist among different adaptation capabilities.

We use EADL to identify the adaptation capabilities of the nodes of our application. Listing 1.1 provides the complete EADL description for a 3-tier application example.

The first thing we need to clarify when defining a tier is the minimum (**minq**) and maximum (**maxq**) number of nodes that it can handle. For instance, in our example we can say we want between one and ten application servers.

Second, we need to define the attributes that are publicly exposed by the nodes in a tier. An **attribute** is a piece of information that is shared by a node to support the distributed adaptation. Attributes are typically application- and/or technology-specific, and are defined by the system maintainer. However, all nodes also expose four pre-defined attributes. **id** (the unique identifier of a node), **ip** (the IP address of a node), **started** that tells us whether the node's application logic is running or not, and **configuration** that provides a description of the node's infrastructural resources (e.g. numbers of cores and memory size). Attribute **configuration** can be used to change the internal configuration of software components (e.g., the heap size of the JVM).

In our running example, the DB tier introduces three technology-specific attributes: **port**, **username**, and **password**. These attributes represent the data that the application servers need to know to successfully connect to the DBMS.

Third, we define the adaptation actions that can be performed on the nodes in a tier. Once again, actions are typically application- and/or technology-specific. Actions can be targeted at any one of the node's software components, and can modify the values of the node's exposed attributes. The attributes that are modified by an action are listed in the action's **modifies** clause. All nodes also have five pre-defined actions, which can be overridden by the maintainer if required. The five predefined actions are: **alloc** for allocating a node (it assigns a new **ip** and sets **started** to false), **start** for starting the application logic (it sets **started** to true), **new** for combining **alloc** and **start** into a single more convenient action, **stop** for stopping the application logic (it sets **started** to false), and **remove** for removing a node from the topology.

Actions support various types of input parameters. A parameter can have a primitive type (i.e., a string, a number, or a boolean value), or it can be of the type **node**. Type **node** is defined by the tier the node belongs to; it aggregates all the attributes defined for that tier using a map data structure. Input parameters can be either **explicit** or **implicit**. Explicit parameters are specified in the adaptation plan that is given to the Execution Choreographer. Implicit parameters, on the other hand, are discovered at run time, while the choreography is in execution. They are defined by, what we call, action **requirements**.

A requirement defines a dependency between an adaptation action and the attributes of a specific node. We support two kinds of requirements: **all** requirements and **any** requirements both contain a filtering **expression**. **all** requirements require that all the nodes, in a specific tier, that satisfy the expression, pass their attributes to the action, under the form of a node parameter. Note that it is acceptable to receive zero nodes if no node in the tier satisfies the expression. **any** requirements require that one, and only one, of the nodes in the tier satisfy the expression and pass its attributes to the action. Which of these nodes is actually chosen is irrelevant. Note that we require that at least one node satisfy the expression. In our example, action **alloc** exposed by the tier **Loadbalancer** requires the attributes of all the application servers that have their attribute **started** set to true.

Adaptation actions can also be identified as **delayed** or **critical**. **delayed** actions are actions that need to be advertised to the topology for safety reasons, before they can be executed. Therefore the actual action execution starts after a configurable amount of time. A good example of this is when we want to **stop** an application server in our tier **AppServer**. We need to advertise our intention to do so, so that the loadbalancer in tier **Loadbalancer** can avoid sending requests to that specific application server. If not, we could lose important requests. **critical** actions, on the other hand, require that the business logic, or the underlying platform, be stopped momentarily to perform the adaptation. A good example of this is when we want to change a configuration parameter of an application server that is only read during the startup.



**Listing 1.2.** Example Plan.

```

LoadBalancer1 updateServerList();           DB1 setMemLock(true);
LoadBalancer1 setAlgorithm("roundrobin");   AppServer alloc();
let newAppServer = AppServer new();         AppServer new();
newAppServer setMaxThreads(250)

```

### 3.2 ECoWare Plan Description Language

Given an application description in EADL, a plan is simply a list of the EADL actions that need to be performed on the application's topology. When adding an action to our plan, we usually want to specify which node should execute it. This can be done by specifying the node's unique id. However, there are cases in which the node's id is unknown, since it has yet to be created. Actions `alloc` and `new` are good examples of this, since no node exists when they are first executed. In this case we simply identify the tier in which we want the new node to be created. The classical *let* notation can be used to bind a variable to a node. Actions can be placed within the plan in any order, since the Execution Choreographer will sort out the dependencies, given the EADL description of the application. If there are no dependencies, the actions are executed in parallel for greater efficiency.

Listing 1.2 illustrates a plan taken from our running example that requests to add three new application servers and to change the configuration of node `Loadbalancer1`, `DB1`, and one of the new application server (using the *let* notation).

## 4 Runtime Adaptation

In this section we focus on the Choreography Creation Algorithm (CCA) and on the design and implementation of the EcoWare Adaptation Agents (EAAs).

### 4.1 Choreography Creation Algorithm

The CCA is activated when the EC (see Fig. 2(a)) receives a new plan, and is achieved by the EC's `Choreographer` over five steps.

In **step 1** the `Choreographer` retrieves the application's current topology from the `Topology Manager` (TM). The TM is a component that is responsible for keeping track of the unique ids of the nodes that exist in the topology, and of whether the nodes are on or off. In **step 2** the `Choreographer` identifies whether the plan can be achieved given the topology's current status and the tier quantity thresholds described in the application's EADL. In **step 3** the `Choreographer` creates and/or removes nodes from the application's tiers, depending on the `new`, `alloc`, and `remove` actions that are in the plan. These actions are performed directly by the EC, through its `Allocation/Deallocation Manager` (ADM). The ADM provides a generic interface for interacting with cloud vendors; this is a design decision that we made to support multi-cloud adaptation in the future.

In **step 4** the Choreographer creates the distributed choreography by assigning the actions in the plan to the EAAs that will be responsible for performing them. In this phase the Choreographer also analyzes the synchronization and data sharing that the EAAs will need to achieve during the actual execution. Finally, in **step 5** the Choreographer sends out the choreography directives to the application's EAAs. This is achieved by sending out messages **Prepare** and **Start Plan**.

Step 4 is the most complex step. Its main objective is to produce efficient sub-plans for the EAAs. This goal requires that the EC understand the dependencies that exist among the adaptation actions in the plan. If no dependency occurs, each node can execute its sub-plan in parallel. We have a dependency when, in order to execute, an action must first know the “finalized” attributes of one or more nodes in the topology. We say that an attribute has been finalized if its value will no longer change during the execution of the adaptation plan. Indirectly, this means that we require that the actions on the interested nodes be completed before we execute our action (note that we do not support direct or indirect cyclic dependencies). Given an action  $a$ , this means dealing with the following three cases.

**arguments** – If action  $a$  has arguments that are related to nodes that have to perform actions in the plan, we know that the values of these attributes might still change. Therefore, we create a dependency between action  $a$  and the interested nodes. This way we are sure that action  $a$  will be executed when the nodes' attributes have been finalized.

**all requirement** – If we have a requirement  $p$  of type **all**, we want  $a$  to receive the attributes of all the nodes that satisfy the filtering expression, regardless of whether they are involved in the adaptation plan or not. For the nodes that are involved in the plan we introduce a dependency. In this case, the filtering expression is evaluated once they have finished executing all their adaptation actions.

**any requirement** – If we have a requirement  $p$  of type **any**, and there is already a node  $n$  that can satisfy  $p$ , and  $n$  is not involved in the plan, there is no need to introduce a dependency, and action  $a$  can proceed immediately, since  $n$ 's attributes will not change. If this is not the case we need to look at the nodes  $n_i$  that are involved in the plan. We introduce a dependency between action  $a$  and all these nodes. As the nodes progressively finish executing their actions we evaluate the filtering expression that is associated with the requirement. As soon as one of the nodes satisfies the expression, all the dependencies for action  $a$  are removed. If this never happens, the distributed plan fails and must be rolled back.

## 4.2 ECoWare Adaptation Agent

When the EAA (see Fig. 2(b)) is initialized it receives a **Tier Info** message from the EC containing all the informations required to be a good citizen in a tier. This includes the node's unique id, the tier's name, the lists of **critical** and

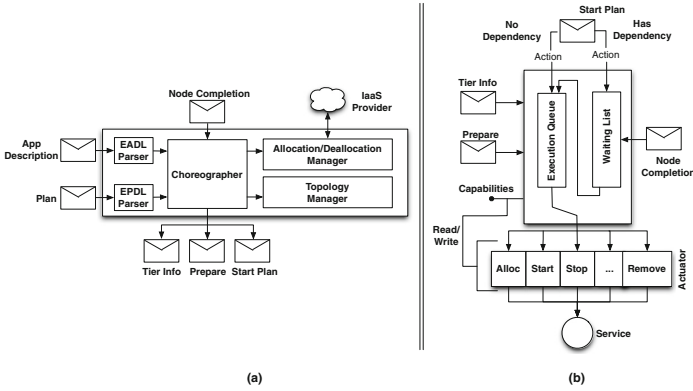


Fig. 2. Architectures of the EC (a), and of the EAA (b).

delayed actions that it is capable of performing, and all the requirements for the actions provided by the EAA’s **Actuator**.

Once the choreography has been created, the EAA is sent a **Prepare** message by the EC. This message contains the list of actions that it should perform, together with a map containing the actions’ arguments and dependencies.

When the EC is ready to launch the actual plan execution, it sends the EAA a **Start Plan** message. This causes two internal data structures to be filled. The first is the **Execution Queue**; it contains all the actions from the plan that have no dependencies. The actions are queued to ensure consistent access to the EAA’s attributes. Furthermore, critical actions are reordered and grouped together to minimize the amount of restarts that need to be performed. The second is the **Waiting List**; it contains all the actions from the plan that have dependencies, and that need other EAAs to complete their actions before being moved to the **Execution Queue**. They are moved when the EAA receives a **Node Completion** message, which signals that a certain node in the topology has completed all the actions in its sub-plan.

When the EAA executes a concrete action from the **Execution Queue** it needs to interact with the concrete adaptations provided by the EAA’s **Actuator**. Before interacting with the **Actuator**, however, the EAA may need to query other nodes for attributes or to satisfy the action’s requirements. If the requirements cannot be satisfied the execution is terminated and the EC is sent a **Failure** message. If the requirements are satisfied, the EAA sends a **Warning** message to the bus if the action is **delayed**. This way the interested EAAs can react accordingly. After a configurable delay the EAA checks whether the action is **critical**. If it is, and the service is running, the EAA executes a **stop** action. At this point, the EAA proceeds to interact with the **Actuator** to perform the adaptation.

Once the action has been completed, the EAA performs the following checks. If all the critical actions have been executed, and the node needs to be restarted, the EAA performs a **start** action. If not, it keeps the service in the “off” state,

so that additional critical actions can be performed. The EAA also sends out another **Warning** message, if needed, to communicate that it has finished executing the **delayed** action.

Finally, once the EAA has completed the execution of all the actions in its plan, it sends a **Completion** message to the EC. The EC proceeds to initiate a Two-Phase Commit (2PC) protocol to reach a distributed consensus on whether the plan should be committed or rolled back.

## 5 Evaluation

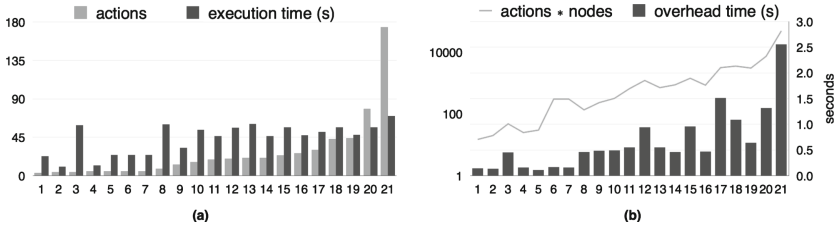
To evaluate our work we used RUBiS [4], a well-known Internet application benchmark that simulates an online auction site. Our RUBiS deployment consisted of four tiers. The first tier was a loadbalancer tier, responsible for routing user requests to either our second or third tier. The second tier was an Apache tier, containing nodes that served static content (e.g., HTML files, images, etc.). The third tier was a JBoss tier, containing nodes that served dynamic content (e.g., servlets). The fourth tier was used for a MySQL database server. We deployed RUBiS on a hybrid cloud. We used six physical servers equipped with Intel Xeon processors running Ubuntu 12.04; and adopted Amazon EC2 for public scaling. In particular we used EC2 to horizontally scale the JBoss tier, as it was the tier that was most stressed when the client workload increased. ECoWare’s components, and in particular the Execution Choreographer, were installed on a seventh private server. The same server was also used to run the RabbitMQ enterprise service bus that we use for inter-EAA communication.

Our experiments consisted in running 21 different randomly generated plans, with varying numbers and types of adaptation actions involved. The topology varied from 6 physical servers to a hybrid cloud of 306 nodes, thanks to AWS EC2 integration. 300 is the maximum number of EC2 instances that we were allowed to instantiate at the same time, given the nature of our contract with AWS.

During our experiments we used the unix tools **ps** and **top** to measure the resource consumption of our framework. We observed that both the EAA and the EC had a negligible impact on the node’s CPU and memory. Since our approach is distributed we also considered the impact that message exchange had on the network resources. The number of messages per execution was linear with respect to the number of nodes in the topology, and to the number of node arguments and requirements that needed to be satisfied; the size of these messages varied from around a hundred bytes to tens of kilobytes, depending on the type of data being communicated.

The Choreography Creation Algorithm also produced an overhead between receiving the plan and initiating its execution. The complexity of our algorithm is  $O(n*m*p*t)$  where  $n$  is the number of nodes in the topology,  $m$  is the number of actions for each node,  $p$  is the number of requirements for each action, and  $t$  is the number of nodes contained in the tier mentioned in the requirement.

The results of our tests have shown that, on average, 56.2% of the total time was spent allocating and deallocating AWS EC2 instances, 35.2% of the time was



**Fig. 3.** (a) Execution time; (b) Overhead introduced by the Choreography Creation Algorithm

spent executing the choreography, 8.1 % of the time was spent discovering the initial topology and executing the two-phase commit step, and 0.5 % of the time was spent creating the choreography. The time spent allocating and deallocating EC2 instances is a variable that is out of our control, and would be the same for any competing approach that also used AWS EC2. Its amount could change drastically if we were to adopt a different IaaS cloud provider.

Figure 3a illustrates how the number of actions in the plan impacts the execution time of the actual choreography. Because of the distributed approach of our processing model, each EAA can execute its part of the plan in parallel to others (unless one of the actions contains a dependency). For this reason, depending on the type of the actions, the execution times of plans containing from 3 to 174 actions oscillate between 10 and 70 seconds, demonstrating the sub-linear relationship between the two factors.

Figure 3b demonstrates how the overhead produced by the Coreography Creation Algorithm is influenced by different components. The continuous line refers to the left y-axis (measured with a logarithmic scale), and represents the product of the number of actions in the plan and the number of nodes in the topology. The actual overhead is shown by the vertical bars and is measured in seconds. The two sets of values have similar trends; however, in some experiments, this is not true (e.g., the eighth experiment). This is related to the presence of requirements that cause more bus communication, which in turn can be heavily affected by networking time, especially if the communication needs to be done on the public Internet. To satisfy a requirement the CCA needs to start a conversation with the involved EAAs. Therefore, requirements are a key aspect to consider when evaluating the overall impact on performance. So, even though the number of actions and nodes were lower in experiment eight (with respect to experiments six and seven), we saw a higher overhead; this was due to the fact that its actions had more requirements.

## 6 Related Work

The automated runtime management tools provided by IaaS cloud vendors have always been quite primitive, focusing mainly on how to auto-scale infrastructural resources. Only recently Amazon AWS has begun providing OpsWorks [1].

Internally OpsWorks makes use of Chef recipes [2]. Chef, however, does not allow one to configure the actual virtualized or hardware resources he or she uses. Cloudify [3] takes application life-cycle automation one step further, by allowing us to setup the virtualized or hardware resources. Cloudify supports a primitive monitoring system, and therefore can only easily drive infrastructure auto-scaling. Instead, with our approach we can drive true multi-layer adaptation, coordinate multiple distributed actuators, and provide both vertical and horizontal scalability.

As for research initiatives, Zeginis et al. [14] propose a framework for the proactive cross-layer adaptation of service-based applications called ECMAF. They consider the business process management (BPM) level, the service composition and coordination (SCC) level, and the service infrastructure (SI) level. The framework is event-based, and uses Astro [11] and Nagios [8] for service and infrastructure monitoring, respectively. It uses complex event processing to identify problems, and to initiate pre-defined adaptation strategies expressed as BPEL orchestrations. A similar approach is provided by Popescu et al. [10]. They also define adaptation using BPEL, and they activate adaptation when certain semantically annotated events that are collected at run time. Their processes are called templates, and they focus on a single layer; however, they can invoke WSDL operations exposed by other templates, effectively making the approach multi-layered. Both approaches chose BPEL to perform adaptation, and are therefore orchestrations. This means they do not have the advantages of a decentralized approach. Moreover, BPEL uses SOAP messages, which have a higher impact on network consumption with respect to our approach. Inzinger et al. [7] present MONINA, a domain-specific language for defining cloud monitoring and adaptation in an integrated fashion. Much like ourselves, one of their main goals is to minimize reaction times and overhead by deploying control operators onto the running system in a distributed fashion. Adaptation is managed using production rule engines. Unfortunately, they do not clarify how and if the distributed control operators can share information and synchronize their actions. Zengin et al. [15] propose the Cross Layer Adaptation Manager (CLAM). CLAM uses adaptation trees to calculate adaptation strategies. The nodes in the tree represent application configurations. The root represents the initial configuration, while the other nodes are reached by applying specific adaptation actions. Leaf nodes represent final configurations, in which the desired end-result has been reached or no further adaptation is possible. Their work is complementary to ours, in the sense that they do not focus on how the discovered strategy can actually be executed onto the running system.

## 7 Conclusions and Future Work

This paper presents a solution for implementing distributed adaptation choreographies for cloud-based applications. We introduce EADL for defining the adaptation capabilities of an application, and EPDL for creating multi-node adaptation plans. Furthermore, we discuss our distributed coordination model, and evaluated our work with experiments in a hybrid-cloud environment.

In the future we will continue to evaluate our approach with a stronger focus on multi-cloud applications. Technically this will require enriching EADL and EPDL and extending the QN-based planner we developed in [13], to include a decision algorithm for selecting the most convenient way to perform horizontal scalability.

Plan execution is completely distributed. Nevertheless, we use a centralized node to generate the choreography. To prevent this node from becoming a single point of failure, we have designed and implemented a leader election mechanism through which we can dynamically select any node in the topology to act as the choreographer.

ECoWare currently requires that nodes in the same tier be the same. Although we do not believe this to be a strong limitation, since we can manage as many tiers as needed, this solution might not always be ideal. If we have a master DB and multiple slave DBs, their adaptation would need to be managed through separate tiers. We will further assess the tradeoffs that would be introduced by allowing different kinds of nodes within the same tier.

## References

1. AWS OpsWorks. <http://aws.amazon.com/opsworks/>
2. Chef. Automation Platform for the new IT. <http://www.getchef.com>
3. Cloudify: Cloud Orchestration and Automation Made Easy. <http://getcloudify.org>
4. RUBiS: The Rice University Bidding System. <http://rubis.ow2.org>
5. Baresi, L., Guinea, S.: Event-based multi-level service monitoring. In: Proceedings of the 20th International Conference on Web Services, ICWS, pp. 83–90 (2013)
6. Binz, T., Breiter, G., Leyman, F., Spatzier, T.: Portable cloud services using TOSCA. *IEEE Internet Comput.* **16**(3), 80–85 (2012)
7. Inzinger, C., Hummer, W., Satzger, B., Leitner, P., Dustdar, S.: Generic event-based monitoring and adaptation methodology for heterogeneous distributed systems. *Softw. Pract. Experience* **44**(7), 805–822 (2014)
8. Josephsen, D.: Building a monitoring infrastructure with Nagios. Prentice Hall, Upper Saddle River (2007)
9. Kephart, J.O., Chess, D.M.: The vision of autonomic computing. *Computer* **36**(1), 41–50 (2003)
10. Popescu, R., Staikopoulos, A., Liu, P., Brogi, A., Clarke, S.: Taxonomy-driven adaptation of multi-layer applications using templates. In: Proceedings of the 4th IEEE International Conference on Self-Adaptive and Self-Organizing Systems, SASO, pp. 213–222 (2010)
11. Raik, H., Bucchiarone, A., Khurshid, N., Marconi, A., Pistore, M.: ASTRO-CAPtEvo: dynamic context-aware adaptation for service-based systems. In: Proceedings of the 8th IEEE World Congress on Services, SERVICES, pp. 385–392 (2012)
12. Rochwerger, B., Breitgand, D., Levy, E., Galis, A., Nagin, K., Llorente, I., Montero, R., Wolfsthal, Y., Elmroth, E., Caceres, J., Ben-Yehuda, M., Emmerich, W., Galan, F.: The reservoir model and architecture for open federated cloud computing. *IBM J. Res. Develop.* **53**(4), 4:1–4:11 (2009)

13. Seracini, F., Menarini, M., Krueger, I., Baresi, L., Guinea, S., Quattrocchi, G.: A comprehensive resource management solution for web-based systems. In: Proceedings of the 11th International Conference on Autonomic Computing, ICAC (2014, to appear)
14. Zeginis, C., Konsolaki, K., Kritikos, K., Plexousakis, D.: Towards proactive cross-layer service adaptation. In: Wang, X., Cruz, I., Delis, A., Huang, G. (eds.) WISE 2012. LNCS, vol. 7651, pp. 704–711. Springer, Heidelberg (2012)
15. Zengin, A., Kazhamiakin, R., Pistore, M.: CLAM: cross-layer management of adaptation decisions for service-based applications. In: Proceedings of the 18th International Conference on Web Services, ICWS, pp. 698–699 (2011)



# Fuzzy Description Logics for Component Selection in Software Design

Tommaso Di Noia<sup>1</sup>, Marina Mongiello<sup>1(✉)</sup>, and Umberto Straccia<sup>2</sup>

<sup>1</sup> Politecnico di Bari, Via E. Orabona, 4, 70125 Bari, Italy

{[tommaso.di.noia](mailto:tommaso.di.noia@poliba.it),[marina.mongiello](mailto:marina.mongiello@poliba.it)}@poliba.it

<sup>2</sup> ISTI-CNR, Via G. Moruzzi, 1, 56124 Pisa, Italy

[umberto.straccia@isti.cnr.it](mailto:umberto.straccia@isti.cnr.it)

**Abstract.** In the Future Internet era, the way software will be produced and used will more and more depend on the new challenges deriving from the virtually infinite number of software services that can be composed to build new applications. The integration and composition of existing software, components and services is now gaining a crucial role in the software modeling and production and encompasses several aspects ranging from theoretical issues like modeling and analysis, to practical and implementation ones like run-time management and integration. In the wide set of issues concerning software composition, in this position paper we propose a formalization via a Fuzzy Description Logic for modeling architectural aspects of a software system.

The formalism models architectural patterns and non-functional requirements about quality attributes where both the relationships among patterns and the set non-functional requirements are modelled together with their mutual interactions. The declarative approach proposed here would make possible to formally represent and maintain the above mentioned knowledge by keeping the flexibility and fuzziness of modeling thanks to the use of fuzzy concepts as high, low, fair, etc. We also identify the need for a reasoning task able to exploit the fuzzy nature of the adopted logic to retrieve a ranked list of set of patterns covering given user requirements represented in terms of NFRs and families of patterns.

## 1 Introduction

The way software will be produced in the next Future Internet era —according to given goals and by integration and compositions of existing services and components— calls for new formally grounded and formalized aspects and methods to support the conceptual modeling of system specifications.

Important issues concerning software architectures, design decisions, quality and goals evaluations are closely linked, anyway any formal definition is available to get relevant information and support in software modeling based on this set of features. In defining and modeling software systems a set of related but complex issues must be considered when composing pieces of reusable artifacts

through design or architectural patterns driven by non functional requirements satisfaction.

In this paper we propose a formalization via a Fuzzy Description Logic for modeling architectural aspects of a software system. The formalism models architectural patterns and non-functional requirements about quality attributes where both the relationships among patterns and the set non-functional requirements are modeled together with their interactions. The framework we propose enables to represent and reason on mutual relationships among non functional requirements by means of fuzzy Description Logics (DL). The fuzzy version of DLs is needed in order to represent both ontological relations and factual ones. In the former case we may model that “*portability* and *adaptability* are directly proportionate” while “*stability* and *adaptability* are inversely proportionate”. For the latter we may represent that “the Adapter pattern has a *high portability*”. We see that the combination of ontological and factual knowledge allows a reasoning procedure to infer new information about a pattern. With reference to the previous example, we may infer that “the Adapter pattern has a *high adaptability* and a *low stability*”. In the framework we propose here we are allowed also to formally define that “a *high adaptability* implies a *medium maintainability*<sup>1</sup>”. Once the knowledge about non functional requirements has been modeled via a formal language and encoded in a knowledge base, we need a tool to query and retrieve data related to a specific task. In our case, the task we propose to solve is the following: given a set of non functional requirements  $\mathcal{R} = \{r_1, \dots, r_n\}$ , retrieve the minimal subset of patterns that better satisfies them. This means that we prefer patterns with a high value of a specific functional requirement  $r_i$  to those with a medium or low one. If there is no pattern with high value for  $r_i$ , than we prefer patterns with a medium value to those with a low one. In such patterns, fuzziness is evident: in fact, terms such as *high*, *medium* and *low* can be defined in terms of fuzzy sets [29].

The remaining of this paper is organized as follows. In the next section we motivate our proposal. Section 3 describes the approach we use to model the ontology and defines a theoretical algorithm. Section 4 presents a case study to explain the proposed idea. Conclusions and future works close the paper.

## 2 Motivation

Since Anton Jansen and Jan Bosch [15] gave a modern definition of Software Architecture, several important issues concerning software architectures, design decisions, quality and goals evaluations have been dominating the scientific literature in this field as comprehensively and systematically provided by Tofan et al. [27]. Designing the software architecture of non-trivial systems belonging to several application domains, namely industrial automation, defense and telecommunication financial services, and so on, is not an easy task, and requires highly

<sup>1</sup> In this case we do not have a high maintainability as the system adapts its configuration to context (or requirements) variations. At the same time the maintainability is not low as it has its own internal logic.

skilled and experienced people. Beyond these, new challenges in the design and in architectural models are derived from self-managing and self-adaptive capabilities that are typical of many modern and emerging software systems, including the industrial internet of things, cyber-physical systems, cloud computing, and mobile computing. The satisfaction of quality requirements and the appropriate options for future changes are among the major goals of software architectures, even more important than functional requirements. Quality goals often compete or even conflict with each other and with functional requirements. In defining and modeling software architecture through patterns, a challenging issue is also concerned with the number of different available decisions depending on the fact that patterns can cooperate, are composable, are complementary or exclusive with respect to a given problem [9, 19]. To solve the challenging problem of choosing a set of patterns, some structures have been proposed supported by pattern languages with a given syntax and style [5].

In self-adaptable models but also in classical software architectures, the link between architecture and design-time features modeling and the relationship between non-functional requirements, patterns and design decisions should be made more flexible. The idea is to provide an approach to more faithfully reproduce the existing relationships in order to formalize the extent to which they are guaranteed in the design of software architecture.

### 3 Problem Statement and Approach

In software design domain, a typical problem to solve is the following:

*“Given a set of requirements define the software design that (better) models the given requirements”*

In order to solve such a problem, adopted empirical approaches generally depend on the designer’s know-how and experience.

According to modern software production and modeling, mainly based on component integration and/or composition and according to the modern definition of software architecture [15], the above problem is that of finding the architectural model as a solution to a decision making problem. The architectural model can hence be defined using design or architectural patterns selected according to Non-functional Requirements (NFRs) satisfaction. The selection of the right NFRs may result crucial in the initial design of a software system. It may happen that the designer is looking for the best design solution given a set of non functional requirements and some problem areas and/or pattern families related to the system. Design patterns give proven solutions to recurrent problems based on typical situations. Therefore, they are a first attempt to formalize the knowledge about NFRs and to give a somewhat structured approach to their compliance in the software design [5, 6, 11].

With respect to the general problem stated at the beginning of this section, we restrict our interest to the connection among patterns, problem areas/families and NFRs. More in detail we are aiming to finding a solution to the following design problem:

“Given a software design to model, a set of NFRs and the problem areas the software refers to, which are the components/patterns that best fit them?”

The task is non-trivial as: NFRs may be disjoint with each other and cannot be satisfied at the same time; some families may not contain patterns satisfying some of the NFRs. Moreover, the designer may not be aware of all the patterns available given a NFR or given a pattern family.

To the best of our knowledge, the software design theory misses a formal superstructure to integrate and relate the elements of the given sets and implicit or explicit relationships between elements. The approach we propose here, exploits a Fuzzy Description Logic and related reasoning tasks in order both (i) to provide a formal representation of the relations intercurring among design areas, NFRs and design patterns and (ii) to reason with such a representation to help the designer during the selection of the right set of patterns that best match the initial requirements. Full and exhaustive background in Description Logics and Fuzzy Description Logic are available in literature ([2] for DLs and [22–24] for fuzzy DLs).

We will illustrate how to encode the information by leveraging on:

- Fuzzy DL statements to have a high level model of the domain we are dealing with and to represent relations among non-functional properties;
- Fuzzy DL reasoning to infer new knowledge about NFRs mutual relations and to retrieve sets of patterns satisfying specific requirements;

We next describe the role played by each of the above indicated technologies in the decision process.

*Fuzzy DL statements.* In order to encode all the information related to NFRs, patterns and corresponding families, we need a formalization of the domain knowledge. The ontology we use to cope with this task can be seen as composed by two main modules: the one describing, at a high level, the connections between patterns and families and between patterns and NFRs; the other one modeling the relations intercurring among NFRs. The formal definition of the ontology is encoded in Fuzzy DL as:

$$\begin{aligned} \exists \text{isInFamily} &\sqsubseteq \text{SoftwareDesignPattern} \\ \exists \text{nFR} &\sqsubseteq \text{SoftwareDesignPattern} \\ \top &\sqsubseteq \forall \text{isInFamily.Families} \\ \top &\sqsubseteq \forall \text{nFR.NonFunctionalRequirement} \end{aligned}$$

The first two statements represent *domain* restrictions while the last two represent range ones. In other words we say that the role `isInFamily` connects instances of the concept `SoftwareDesignPattern` to instances of the concept `Families` while the role `isInFamily` relates instances of `SoftwareDesignPattern` to instances of `NonFunctionalRequirement`.

Please note that the structure of the high level ontology we model makes it possible to easily extend it to deal also with other elements, such as Functional Requirements.

Given the ontology, we can state explicit facts about the description of a pattern in terms of pattern family it belongs to and NFRs it guarantees. These statements form the ABox of our knowledge base. Specifically, let us consider the following Fuzzy DL assertions:

```

proxyPattern:SoftwareDesignPattern
resourceManagement:Families
reliability:NonFunctionalRequirement
loadBalancing:NonFunctionalRequirement
reusability:NonFunctionalRequirement
(proxyPattern,resourceManagement):isInFamily

```

That is, we introduced the pattern `proxyPattern`, the family `resource Management` and the non-functional requirements `reliability`, `loadBalancing`, `reusability` as instances/individuals of the classes `SoftwareDesignPattern`, `Families` and `NonFunctionalRequirement` respectively.

Based on these individuals and properties we may wish to state that the *Proxy Pattern* assures a high *Load Balancing* and a high *Reliability*. In order to formally represent such constraints we need to introduce new datatype properties, together with the corresponding fuzzy sets, and axioms related to the non functional requirements we just introduced. In the following we will always refer to them for every datatype property and we will use  $\mathbf{R}$  (for rating) to represent the interval [very bad, bad, medium, good, very good]. The set of axioms we are going to define are needed in order to exploit the full potential of the fuzzy DL reasoning. It is noteworthy that in a production scenario, they can be automatically added to the knowledge base in a straight way. These are:

$$\begin{aligned} \exists \text{nFR}.\{\text{reliability}\} &\equiv \exists \text{reliabilityRate}.\in_{\mathbf{R}} \\ \exists \text{nFR}.\{\text{loadBalancing}\} &\equiv \exists \text{loadBalancingRate}.\in_{\mathbf{R}} \\ \exists \text{nFR}.\{\text{reusability}\} &\equiv \exists \text{reusabilityRate}.\in_{\mathbf{R}} \end{aligned}$$

In the previous statements we say that whenever we have a pattern with an associated non functional requirement we will always have a corresponding degree and vice versa. Based on the datatype properties just introduced we can state, for instance, that

$$\begin{aligned} \text{proxyPattern}:\exists \text{loadBalancingRate} &=_{\text{good}} \\ \text{proxyPattern}:\exists \text{reliabilityRate} &=_{\text{good}} \end{aligned}$$

Besides the modeling of the ABox relations, we use Fuzzy DL also to explicitly model relations intercurring between NFRs. Consider the non-functional requirements *load balancing*, *reliability*, *reusability* previously defined as instances (individuals) of the class `NonFunctionalRequirement`. An example of mutual relation between NFRs is the one between *load balancing* and *reliability*. Indeed, they are directly proportionate, i.e., if the *loadBalancing* increases (decreases) the same happens for the *reliability*. With reference to our ontology, such a relation can be written in Fuzzy DL as

$$\begin{aligned} \exists \text{loadBalancingRate.High} &\sqsubseteq \exists \text{reliabilityRate.High} \\ \exists \text{loadBalancingRate.Fair} &\sqsubseteq \exists \text{reliabilityRate.Fair} \\ \exists \text{loadBalancingRate.Low} &\sqsubseteq \exists \text{reliabilityRate.Low} \end{aligned}$$

We also know that a system cannot be *reliable* and *reusable* at the same time. That is, the two non functional requirements are inversely proportionate. Hence, if a pattern guarantees *reliability* it cannot guarantee also *reusability*. We may encode such disjoint relations with the following statement:

$$\exists \text{reusabilityRate.High} \sqsubseteq \exists \text{reliabilityRate.Low}$$

From the two relations explicitly stated before, we may imply that as the *Proxy Pattern* guarantees a high degree of *load balancing*, it cannot guarantee a high degree of *reusability*.

By using automated reasoning over Fuzzy DL knowledge bases we automatically infer all these kind of implicit relations thus providing better results while looking for a design solution. It is noteworthy that the Fuzzy DL we are targeting is allowed to represent also statements like: “a system with a high adaptability has a fair maintainability”. Indeed, as the system adapts its own configuration to context o requirements variations it may not be highly maintainable, i.e.,

$$\exists \text{adaptabilityRate.High} \sqsubseteq \exists \text{maintainabilityRate.Fair}$$

## 4 Use Case Scenario

We next illustrate how to apply the proposed framework by means of a use case scenario. We model the use case of a *Cloud-Social-Adaptable System*. In the cloud environment, let us think of an application in social domain in which the various applications (apps) share data distributed over different clusters or data centres.

The system is allowed to dynamically and extensively load external applications depending on variations in the context or depending on changes in the behavior of the user. For example, if the user is travelling for a week-end or on holiday, an app arranges all stored material related to the destination of the trip and creates albums, photo collections with captions, stories etc. Other context-dependent conditions set in the application, enable dynamic loading of different apps. Dynamically loaded applications might compromise properties of the entire system, then it is of crucial importance to provide mechanisms working at run-time and able to check and guarantee the preservation of properties of interest. All nodes in the application are started exploiting the Cloud virtualization, i.e. the physical hardware is shared between all services but the software environment is independent, ensuring low coupling between the virtual nodes. The architecture is flexible since every consumer can access a public service with the available resources, with a saving in terms of costs. Moreover, being the various virtual machines unconnected, the fault of one of them does not compromise all others, thus ensuring a good fault-tolerance in the overall architecture. Virtual machines are made up and launched directly from the middleware just as a

consumer requests. The failure of a virtual machine does not affect the others. The availability of content is ensured by a content delivery network. The request for more machines avoids the single point of failure.

The implementation of the scenario previously described requires patterns satisfying characteristics derived both from the running environment and from the main features of the context. We see that such patterns should belong to families that manage *Cloud* features, but also must solve problems about process communication and middleware. They have to ensure adaptability being the system social-adaptable, hence it would be desirable they belong to the *Adaptation and Extension* family. Actually, the patterns we are looking for could also be in the *Application Control* family. Indeed, given the specific environment of the system, it should be necessary to separate the interface from the applications core functionalities. As for the Non Functional Requirements, the architectural model must ensure *adaptability*, being the system a social adaptable and *elasticity* since it will work in a cloud environment and will manage a large amount of data. Also *fault tolerance* is a requirement to be satisfied in order to ensure a dependable system. A low level of *coupling* is also required being the system implemented in a cloud environment. All these requirements can be summarized as:

- belonging families: *Cloud Patterns*, *Application Control*, *Adaptation and Extension*, *Distribution Infrastructure*;
- non functional requirements: high *adaptability*, high *elasticity*, high *fault tolerance*, low *coupling*.

Now suppose we have defined in a knowledge base  $\mathcal{K}$  (ABox and TBox) information about a set of patterns, families and NFRs. That is, the ABox contains

adaptationAndExtension:Families , cloud:Families , distributionInfrastructure:Families  
applicationControl:Families

adaptability:NonFunctionalRequirement , stability:NonFunctionalRequirement  
maintainability:NonFunctionalRequirement , simplicity:NonFunctionalRequirement  
dependability:NonFunctionalRequirement , redundancy:NonFunctionalRequirement  
performance:NonFunctionalRequirement , reliability:NonFunctionalRequirement  
elasticity:NonFunctionalRequirement , faultTolerance:NonFunctionalRequirement  
loadBalancing:NonFunctionalRequirement , security:NonFunctionalRequirement  
flexibility:NonFunctionalRequirement , scalability:NonFunctionalRequirement  
coupling:NonFunctionalRequirement , robustness:NonFunctionalRequirement  
resilience:NonFunctionalRequirement

reflection:SoftwareDesignPattern , strictConsistency:SoftwareDesignPattern  
hypervisor:SoftwareDesignPattern , observer:SoftwareDesignPattern  
broker:SoftwareDesignPattern

(reflection,adaptationAndExtension):isInFamily , (strictConsistency,cloud):isInFamily  
(hypervisor,cloud):isInFamily , (observer,applicationControl):isInFamily  
(broker,distributionInfrastructure):isInFamily

reflection: $\exists$ adaptabilityRate. =<sub>very good</sub> , reflection: $\exists$ stabilityRate. =<sub>bad</sub>  
reflection: $\exists$ maintainabilityRate. =<sub>medium</sub> , reflection: $\exists$ simplcityRate. =<sub>bad</sub>

$\text{strictConsistency}:\exists\text{dependabilityRate.} =_{\text{very good}}$  ,  $\text{strictConsistency}:\exists\text{redundancyRate.} =_{\text{medium}}$   
 $\text{strictConsistency}:\exists\text{performanceRate.} =_{\text{medium}}$  ,  $\text{strictConsistency}:\exists\text{reliabilityRate.} =_{\text{good}}$   
 $\text{hypervisor}:\exists\text{elasticityRate.} =_{\text{very good}}$  ,  $\text{hypervisor}:\exists\text{redundancyRate.} =_{\text{medium}}$   
 $\text{hypervisor}:\exists\text{faultToleranceRate.} =_{\text{good}}$  ,  $\text{hypervisor}:\exists\text{loadBalancingRate.} =_{\text{good}}$   
 $\text{hypervisor}:\exists\text{securityRate.} =_{\text{good}}$   
 $\text{observer}:\exists\text{flexibilityRate.} =_{\text{good}}$  ,  $\text{observer}:\exists\text{scalabilityRate.} =_{\text{medium}}$   
 $\text{observer}:\exists\text{adaptabilityRate.} =_{\text{medium}}$   
 $\text{broker}:\exists\text{loadBalancingRate.} =_{\text{very good}}$  ,  $\text{broker}:\exists\text{robustnessRate.} =_{\text{medium}}$   
 $\text{broker}:\exists\text{faultToleranceRate.} =_{\text{bad}}$  ,  $\text{broker}:\exists\text{performanceRate.} =_{\text{medium}}$   
 $\text{broker}:\exists\text{reliabilityRate.} =_{\text{good}}$  ,  $\text{broker}:\exists\text{resilienceRate.} =_{\text{good}}$

Eventually, the TBox contains

$\exists\text{flexibilityRate.High} \sqsubseteq \exists\text{couplingRate.Low}$   
 $\exists\text{elasticityRate.High} \sqsubseteq \exists\text{adaptabilityRate.High}$   
 $\exists\text{elasticityRate.Fair} \sqsubseteq \exists\text{adaptabilityRate.Fair}$   
 $\exists\text{elasticityRate.Low} \sqsubseteq \exists\text{adaptabilityRate.Low}$   
 $\exists\text{robustnessRate.High} \sqsubseteq \exists\text{faultToleranceRate.High}$   
 $\exists\text{robustnessRate.Medium} \sqsubseteq \exists\text{faultToleranceRate.Medium}$   
 $\exists\text{robustnessRate.Low} \sqsubseteq \exists\text{faultToleranceRate.Low}$   
 $\exists\text{faultToleranceRate.High} \sqsubseteq \exists\text{reliabilityRate.High}$   
 $\exists\text{faultToleranceRate.Medium} \sqsubseteq \exists\text{reliabilityRate.Medium}$   
 $\exists\text{faultToleranceRate.Low} \sqsubseteq \exists\text{reliabilityRate.Low}$   
 $\exists\text{reliabilityRate.High} \sqsubseteq \exists\text{dependabilityRate.High}$   
 $\exists\text{reliabilityRate.Medium} \sqsubseteq \exists\text{dependabilityRate.Medium}$   
 $\exists\text{reliabilityRate.Low} \sqsubseteq \exists\text{dependabilityRate.Low}$   
 $\exists\text{loadBalancingRate.High} \sqsubseteq \exists\text{elasticityRate.High}$   
 $\exists\text{loadBalancingRate.Medium} \sqsubseteq \exists\text{elasticityRate.Medium}$   
 $\exists\text{loadBalancingRate.Low} \sqsubseteq \exists\text{elasticityRate.Low}$

In order to get the best set of patterns satisfying the requirements needed to solve the task, we need a reasoning method for retrieving more suitable patterns according to given requirements.

The set of retrived pattern should be incrementally built considering subsets of the original requirements:

Specifically, let us start with the subset

- belonging families: *Adaptation and Extension*;
- non functional requirements: *adaptability*.

The formula modeling the previous requirements is thus:

$$C' = \exists\text{isInFamily.}\{\text{adaptationAndExtension}\} \sqcap \exists\text{adaptabilityRate.High.}$$

Therefore, the retrieved ranked list of top-3 patterns satisfying  $C'$  would be:  $\langle\text{reflection, strictConsistency, hypervisor}\rangle$  When we defined  $C'$  we assumed the patterns we were looking for belonged only to the *Adaptation and Extension* family.



Actually, we may look also for patterns in other families which satisfy the *adaptability* NFR. We then extend the previous subset with

- belonging families: *Cloud Patterns, Application Control, Adaptation and Extension, Distribution Infrastructure*;
- non functional requirements: *adaptability*.

In order to model such a more relaxed requirement we modify  $C'$  in  $C_1$  as

$$C_1 = (\exists \text{InFamily.}\{\text{adaptationAndExtension}\} \sqcup \exists \text{InFamily.}\{\text{cloud}\} \sqcup \\ \exists \text{InFamily.}\{\text{applicationControl}\} \sqcup \exists \text{InFamily.}\{\text{distributionInfrastructure}\}) \sqcap \\ \exists \text{adaptabilityRate.High.}$$

With respect to the newly stated requirements, the retrieved ranked list of patterns is:  $\langle \text{hypervisor, reflection, observer} \rangle$  with **hypervisor** and **reflection** in the same ranking position, and **observer** that can be equivalently substituted by **broker**.

We may also define  $C_2$ ,  $C_3$  and  $C_4$

$$C_2 = (\exists \text{InFamily.}\{\text{adaptationAndExtension}\} \sqcup \exists \text{InFamily.}\{\text{cloud}\} \sqcup \\ \exists \text{InFamily.}\{\text{applicationControl}\} \sqcup \exists \text{InFamily.}\{\text{distributionInfrastructure}\}) \sqcap \\ \exists \text{elasticityRate.High.}$$

$$C_3 = (\exists \text{InFamily.}\{\text{adaptationAndExtension}\} \sqcup \exists \text{InFamily.}\{\text{cloud}\} \sqcup \\ \exists \text{InFamily.}\{\text{applicationControl}\} \sqcup \exists \text{InFamily.}\{\text{distributionInfrastructure}\}) \sqcap \\ \exists \text{faultToleranceRate.High.}$$

$$C_4 = (\exists \text{InFamily.}\{\text{adaptationAndExtension}\} \sqcup \exists \text{InFamily.}\{\text{cloud}\} \sqcup \\ \exists \text{InFamily.}\{\text{applicationControl}\} \sqcup \exists \text{InFamily.}\{\text{distributionInfrastructure}\}) \sqcap \\ \exists \text{couplingRate.Low.}$$

Hence, the best solution is to adopt both **broker** and **hypervisor**, then we have the pair **reflection, broker** and eventually **observer, broker**.

## 5 Related Work

In this section we briefly review the literature on knowledge representation approaches architectural concerns such as nonfunctional requirements, design pattern and service composition.

QoS aspects of Service composition are studied in [7, 18]. While Architectural concerns about services and future Internet applications are considered in [3, 14, 16]. Temporal behavior of design patterns are modeled in [20] using formal methods. A Balanced Pattern Specification Language (BPSL) is modeled in [25] based on a subset of First Order Logic and Temporal Logic of Action to specify both structural and behavioral aspects of patterns [26]. Relationships among design pattern as architectural tactics and architecture are studied in [13]. Pattern languages connect patterns from a variety of different sources into a single pattern network [17]; they are supported by trees or directed graphs and

enable typical search algorithms of depth-first or breadth-first. Anyway the modeling of relationships and sequences of patterns and the taxonomy of the several issues concerned with their use needs a more structured approach with respect to data structures such as trees or directed graphs, due to the complex semantics they convey. A state of the art on the treatment of non-functional requirements is in [8]. Definitions of non-functional requirements are surveyed in [10, 12]. In [4] a model of non-functional requirements is given.

Mylopoulos et al. propose two complementary approaches for using non-functional information: process-oriented and product oriented, [21]. State of the art of Model Driven Development approaches with respect to non-functional requirements is in [1], where also a framework for integrating NFRs in the core of Model Driven Development process is given [28].

## 6 Conclusion and Future Work

In this paper we proposed to use a Fuzzy Description Logic (DL) to model the knowledge related to NFRs, patterns and related families in order to ease and support software components integration and composition in architectural decision making problems.

Although the overall framework has been presented to deal with NFRs, thanks to its declarative nature, it can be easily extended to other characteristics such as Functional Requirements.

We are currently working on how to take into account also information related to temporal interaction among software components thus extending the Fuzzy DL we propose with temporal operators. We are also performing extensive experiments on a structured benchmark to test the framework functionalities and performance on simple and on composable schemes also in more advanced architectural environments.

## References

1. Ameller, D., Franch, X., Cabot, J.: Dealing with non-functional requirements in model-driven development. In: 2010 18th IEEE International Requirements Engineering Conference (RE), pp. 189–198. IEEE (2010)
2. Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F. (eds.): *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, Cambridge (2003)
3. Baresi, L., Caporuscio, M., Ghezzi, C., Guinea, S.: Model-driven management of services. In: 2010 IEEE 8th European Conference on Web Services (ECOWS), pp. 147–154. IEEE (2010)
4. Botella, P., Burgues, X., Franch, X., Huerta, M., Salazar, G.: Modeling non-functional requirements. In: *Proceedings of Jornadas de Ingenieria de Requisitos Aplicada JIRA* (2001)
5. Buschmann, F., Henney, K., Schmidt, D.C.: *Pattern-Oriented Software Architecture: A Pattern Language for Distributed Computing*, vol. 4 (2007)

6. Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P., Stal, M.: *Pattern-Oriented Software Architecture: A System of Patterns*. John Wiley & Sons Inc, New York, NY, USA (1996)
7. Caporuscio, M., Mirandola, R., Trubiani, C.: Qos-based feedback for service compositions. In: *Proceedings of the 11th International ACM SIGSOFT Conference on Quality of Software Architectures*, pp. 37–42. ACM (2015)
8. Chung, L., do Prado Leite, J.C.S.: On non-functional requirements in software engineering. In: Borgida, A.T., Chaudhri, V.K., Giorgini, P., Yu, E.S. (eds.) *Conceptual Modeling: Foundations and Applications*. LNCS, vol. 5600, pp. 363–379. Springer, Heidelberg (2009)
9. Egyed, A., Grunbacher, P.: Identifying requirements conflicts and cooperation: how quality attributes and automated traceability can help. *IEEE Softw.* **21**(6), 50–58 (2004)
10. Franch, X.: Systematic formulation of non-functional characteristics of software. In: *1998 Third International Conference on Requirements Engineering*, pp. 174–181. IEEE (1998)
11. Gamma, E., Helm, R., Johnson, R., Vlissides, J.: *Design Patterns: Elements of Reusable Object-Oriented Software*. Pearson Education, Massachusetts (1994)
12. Glinz, M.: On non-functional requirements. In: *15th IEEE International Requirements Engineering Conference, RE 2007*, pp. 21–26. IEEE (2007)
13. Harrison, N.B., Avgeriou, P.: How do architecture patterns and tactics interact? a model and annotation. *J. Syst. Softw.* **83**(10), 1735–1758 (2010)
14. Issarny, V., Georgantas, N., Hachem, S., Zarras, A., Vassiliadist, P., Autili, M., Gerosa, M.A., Hamida, A.B.: Service-oriented middleware for the future internet: state of the art and research directions. *J. Internet Serv. Appl.* **2**(1), 23–45 (2011)
15. Jansen, A., Bosch, J.: Software architecture as a set of architectural design decisions. In: *5th Working IEEE/IFIP Conference on Software Architecture, WICSA 2005*, pp. 109–120. IEEE (2005)
16. Johnson, K., Calinescu, R.: Efficient re-resolution of smt specifications for evolving software architectures. In: *Proceedings of the 10th International ACM Sigsoft Conference on Quality of Software Architectures*, pp. 93–102. ACM (2014)
17. Li, F.-L., Horkoff, J., Mylopoulos, J., Liu, L., Borgida, A.: Non-functional requirements revisited. In: *iStar*, pp. 109–114. Citeseer (2013)
18. Maiden, N., Lockertbie, J., Zachos, K., Bertolino, A., De Angelis, G., Lonetti, F.: A requirements-led approach for specifying qos-aware service choreographies: an experience report. In: Weerd, I., Salinesi, C. (eds.) *REFSQ 2014*. LNCS, vol. 8396, pp. 239–253. Springer, Heidelberg (2014)
19. Mairiza, D., Zowghi, D., Nurmuliani, N.: *Managing conflicts among non-functional requirements* (2009)
20. Mikkonen, T.: Formalizing design patterns. In: *Proceedings of the 20th International Conference on Software Engineering*, pp. 115–124. IEEE Computer Society (1998)
21. Mylopoulos, J., Chung, L., Nixon, B.: Representing and using nonfunctional requirements: a process-oriented approach. *IEEE Trans. Softw. Eng.* **18**(6), 483–497 (1992)
22. Straccia, U.: Reasoning within fuzzy description logics. *J. Artif. Intell. Res.* **14**, 137–166 (2001)
23. Straccia, U.: A fuzzy description logic for the semantic web. In: Sanchez, E., (ed.) *Capturing Intelligence: Fuzzy Logic and the Semantic Web*, Chapter 4, pp. 73–90. Elsevier (2006)

24. Straccia, U.: *Foundations of Fuzzy Logic and Semantic Web Languages*. CRC Studies in Informatics Series. Chapman & Hall, Boca Raton (2013)
25. Taibi, T., Ngo, D.C.L.: Formal specification of design patterns - a balanced approach. *J. Object Tech.* **2**(4), 127–140 (2003)
26. Tichy, W.F.: A catalogue of general-purpose software design patterns. In: *Proceedings on Technology of Object-Oriented Languages and Systems, TOOLS 23*, pp. 330–339. IEEE (1997)
27. Tofan, D., Galster, M., Avgeriou, P., Schuitema, W.: Past and future of software architectural decisions-a systematic mapping study. *Inf. Softw. Tech.* **56**(8), 850–872 (2014)
28. Xu, L., Ziv, H., Richardson, D., Liu, Z.: Towards modeling non-functional requirements in software architecture. In: *Early Aspects* (2005)
29. Zadeh, L.A.: Fuzzy sets. *Inf. Control* **8**(3), 338–353 (1965)

# Towards Adapting Choreography-Based Service Compositions Through Enterprise Integration Patterns

Amleto Di Salle, Francesco Gallo, and Alexander Perucci<sup>(✉)</sup>

University of L'Aquila, Via Vetoio, 67100 L'Aquila, Italy  
{amleto.disalle,francesco.gallo}@univaq.it,  
alexander.perucci@graduate.univaq.it  
<http://www.univaq.it>

**Abstract.** The Future Internet is becoming a reality, providing a large-scale computing environments where a virtually infinite number of available services can be composed so as to fit users' needs. Modern service-oriented applications will be more and more often built by reusing and assembling distributed services. A key enabler for this vision is then the ability to automatically compose and dynamically coordinate software services. Service choreographies are an emergent Service Engineering (SE) approach to compose together and coordinate services in a distributed way. When mismatching third-party services are to be composed, obtaining the distributed coordination and adaptation logic required to suitably realize a choreography is a non-trivial and error prone task. Automatic support is then needed. In this direction, this paper leverages previous work on the automatic synthesis of choreography-based systems, and describes our preliminary steps towards exploiting Enterprise Integration Patterns to deal with a form of choreography adaptation.

## 1 Introduction

The Future Internet promotes a distributed computing environment that will be increasingly surrounded by a large number of software services, which can be composed to meet user needs. The Future Internet of Services paradigm emerges from the convergence of the Future Internet (FI) and the Service-Oriented Computing (SOC) paradigm [1]. Services play a central role in this vision as effective means to achieve interoperability between heterogeneous parties of a business process, and new value added service-based systems can be built as a *choreography* of services available in the FI. Service choreography is a decentralized approach, which provides a loose way to design service composition by specifying the participants (i.e., roles) and the (message-based) interaction protocol between them, by decoupling the participant tasks from the services that only later will be bound to the specified roles.

The need for service choreography was recognized in the Business Process Modeling Notation version 2.0<sup>1</sup> (BPMN2), which introduced *Choreography*

<sup>1</sup> <http://www.omg.org/spec/BPMN/2.0/>.

*Diagrams* to offer choreography-specific modeling constructs. A choreography diagram models peer-to-peer communication by defining a multi-party protocol that, when put in place by the cooperating parties, will permit to reach the overall choreography objectives in a fully distributed way. In this sense, service choreographies are quite different from service orchestrations in which a single stakeholder centrally plans and decides how an objective should be reached through the cooperation with other services.

In this paper we leverage the experience on choreography development that we have been doing so far within the EU CHOReOS project<sup>2</sup>. Then, being supported by the EU CHOReVOLUTION (follow-up) project<sup>3</sup>, we report on the novel idea we are currently investigating to achieve choreography adaptation and evolution to face the challenges posed by the heterogeneity of FI services.

In this direction, we propose a way to enhance the previous CHOReOS approach to the automatic synthesis of choreography-based systems [2–5], and describes the preliminary steps we are undertaking within CHOReVOLUTION towards exploiting Enterprise Integration Patterns (EIP) so as to deal with a form of choreography adaptation. The novel contributions can be summarized as follow: (i) adoption of EIP to deal with a form of adaptation for choreography-based systems; (ii) enhancement of our synthesis process by introducing an adapters generator; (iii) enhancement of the architectural style for including adapters.

The paper is structured as follow. Section 2 sets the context of our work, and Sect. 3 introduces an explanatory example. Then, Sect. 4 describes how the synthesis process can be enhanced to deal with choreography adaptation and evolution through protocol coordination, protocol adaptation and related complex data mappings, and Sect. 5 describes the proposed enhancement at work on the explanatory example. Related work is discussed in Sect. 6, and conclusions are given in Sect. 7.

## 2 Setting the Context

This section sets the context of our work by describing the problem we want to address in Sect. 2.1, and the idea underlying the proposed solution in Sect. 2.2. Then, Sect. 2.3 provides basic notions of the Enterprise Integration Patterns (EIP) [6] that we propose to exploit to deal with adaptation issues.

### 2.1 The Problem Space

When considering choreography-based service-oriented systems, the following problems are mainly considered:

- (i) *realizability check* - checks whether the choreography can be realized by implementing each participant so that it conforms to the played role;

<sup>2</sup> <http://www.choreos.eu/>.

<sup>3</sup> <http://www.chorevolution.eu/>.

- (ii) *conformance check* - checks whether the set of services satisfies the choreography specification;
- (iii) *automatic realizability enforcement* - given a choreography specification and a set of existing services, externally coordinate and adapt their interaction so as to fulfill the collaboration prescribed by the choreography specification.

In the literature, the approaches proposed in [7–19] address the problems (i) and (ii); the approaches proposed in [2,3,5,20] address the problem (iii). In this paper we concentrate on the automatic realizability enforcement problem. Specifically, starting from previous work in [2–5], we propose the following enhancement to deal with a form of choreography adaptation that exploits EIP to built service adapters.

## 2.2 The Solution Space

Addressing the automatic realizability enforcement problem calls for solving both coordination issues and adaptation issues.

Coordination issues are addressed in previous work [2,5], where we propose an automatic approach to synthesize the global coordination logic to be then distributed and enforced among the considered services. Preliminary ideas towards addressing adaptation issues are described in [3,4], where we propose the use of adapters for solving interaction protocol mismatches deriving from the heterogeneity of services not born to be directly composed together.

In this paper we describe the initial steps we have done towards exploiting Enterprise Integration Patterns so as to deal with a form of choreography adaptation that, in addition to interaction protocol mismatches, also account for I/O data mismatches. Our mid-term goal within the CHOReVOLUTION project is to achieve automated data-flow coordination and adaptation, which means effectively coping with heterogeneous service interfaces and dealing with as much EIPs [6] as possible in a automatic way. In particular, the idea is to automatically generate adapters by combining different EIPs based on a notion of protocol mediation and data similarity.

## 2.3 Exploiting Enterprise Integration Patterns

From a technical point of view, achieving the above calls for dealing with *mismatching service signatures* and *interaction protocols*. In particular, to achieve adaptation, the operations signature and the interaction protocol of the concrete services may need to be adapted to the roles to be played in the input choreography model. This requires to implement a suitable notion of matching between protocols by means of *complex data mappings* over both operation names and I/O messages. Protocol refinement techniques must be developed to bridge the gap between the abstract protocol of the choreography participant roles and the protocol of the concrete services. These techniques, together with the ability

of dealing with, e.g., appearing and disappearing services at run-time, would permit to achieve evolution through on-the-fly service binding.

EIPs offer more than one approach for integrating applications, i.e., *File Transfer*, *Shared Database*, *Remote Procedure Invocation*, and *Messaging* [6]. We focus on the Messaging approach since we consider Web Services (WSs) as possible choreography participants, and WSs communicate through messages passing (e.g., request/response or one-way operation types).

The Messaging approach uses the “pipes-and-filters” architectural style [21] as base for connecting applications. The Endpoints (Filters) are connected with one another via Channels (Pipes). The producing endpoint sends messages to the channel, and the messages are retrieved by the consuming endpoint. There are different types of pipes and filters patterns, each one of them dedicated to solve a particular integration aspect.

For the purposes of this paper, we consider: *Message Transformation* that converts a message from a format to another one; *Message Aggregator* that receives multiple messages and combines them into a single message.

### 3 Explanatory Example

The explanatory example introduced in this section is a very small portion of an *In-store Marketing and Sale* choreography that was used by the EU CHOReOS project to demonstrate an *Adaptive Customer Relationship Booster* system. The whole choreography was aimed at monitoring the activity of a client inside the shop in order to propose him/her tailored shopping offers and/or advertisements according to the user information (preferences, current shopping list, etc.) held by a shopping assistant application service.

Figure 1 reports a simplified choreography diagram realized by using the Eclipse BPMN2 modeler plugin<sup>4</sup>. The diagrams also shows the input and output messages of each choreography task. Within the Eclipse BPMN2 modeler, messages are specified by using the XML schema, which is the default language for specifying BPMN2 messages.

The choreography is triggered by the **Client** entering the shop. A **Shop Entrance** service (not shown in the figure) detects the presence of a specific **Client** inside the store and assigns him a virtual cart. Once subscribed to the cart, the **Client** can add and remove products to and from it. Once the **Client** finishes shopping, the **Smart Cart** service allows for executing the payment by interacting with the a **Self Check-out Machine**.

### 4 Method Description

In this section we describe the proposed method by distinguishing between *protocol coordination* and *protocol adaptation*.

<sup>4</sup> <http://www.eclipse.org/bpmn2-modeler/>.



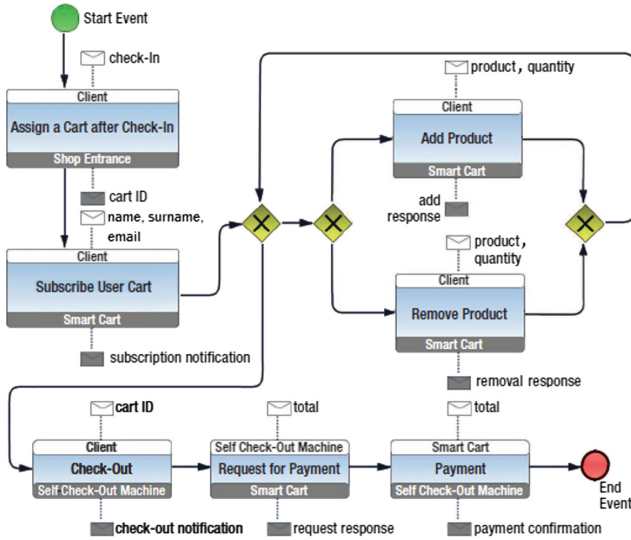


Fig. 1. In-store marketing and sale choreography

Protocol coordination allows for preventing undesired interactions among (possibly adapted) services. That is, interactions not allowed by the choreography specification can happen when the services collaborate in an uncontrolled way. To deal with this problem, additional software entities, called *Coordination Delegates* (CDs), are generated and interposed among the services participating in the specified choreography in order to prevent possible undesired interactions. Thus, the intent of CDs is to coordinate the interaction of the participant services in a way that the resulting collaboration correctly realizes the specified choreography. For instance, the *Client* is allowed to perform the *Add Product* task to add products to the *Smart Cart* (see the top of the Fig. 1). However, after paying and before check-out, an undesired interaction can happen since the *Client* might try to add products (see the top-most tasks just before the End Event), thus avoiding paying for them.

Protocol adaptation allows for dealing with services that do not exactly fit the choreography roles. That is, adapters are automatically synthesized to mediate the interaction service-to-CD and CD-to-service according to the choreography roles (see Fig. 2). Each Adapter is generated so as to bridge/mediate the concrete service interaction protocol in order to exactly match the abstract participant interaction protocol. In other words, Adapters realize correct service-role binding by solving possible interoperability issues (e.g., signature and protocol mismatches) between concrete services and abstract participants. By leveraging a sufficiently accurate notion of behavioral interface refinement, Adapters enforce service-role similarity, hence binding the concrete services to the abstract roles defined by the choreography. The synthesized Adapters enforce exact similarity through complex data mappings and complex protocol mediation patterns. For instance, Adapters are able to map message data types, or reorder/merge/split the sequence of operation calls and/or related I/O messages.

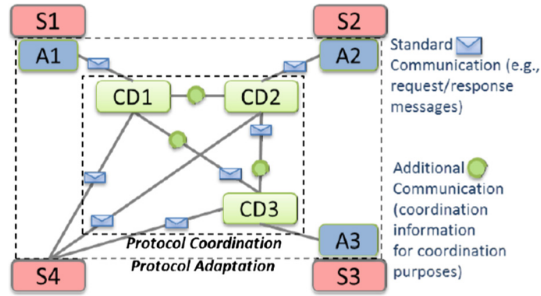


Fig. 2. Architectural style with adapters

Coordination and adaptation software entities are synthesized in order to proxy and control the participant services' interaction. When interposed among the services, according to the architectural style shown in Fig. 2, *coordination entities* still guarantee the collaboration specified by the choreography specification through protocol coordination; *adaptation entities* mediate the interaction of the participant services so as to fit the choreography roles.

An important aspect here is that the coordination logic performed by the CDs is *service-independent* since it is based on the expected behavior of the participants as specified by the choreography, rather than on the actual concrete services to be binded and coordinated. In this way *separation of concerns* is realized by separating pure coordination issues (i.e., undesired interactions) from adaptation/mediation ones (e.g., operation signature mismatches and data incompatibilities at the service interface level, and behavior mismatches). For example, the latter can arise whenever a service discovered as a participant does not exactly match the role to be played.

In order to automatically synthesize adaptation software entities we propose an extension of our CHOReOSynt tool [22] introducing a new RESTful service called **Synthesis Adapter Generator** (see Fig. 3).

By taking as input a BPMN 2.0 specification of the choreography, the extension we propose allows for deriving service Adapters in addition to CDs (Fig. 3). To this end, model transformations are employed and interoperability with the **Service Discovery** is required (out of the scope of this paper). Both CDs and Adapters, when deployed by the **Enactment Engine** (out of the scope of this paper), allow for enacting the choreography by realizing the distributed coordination logic between the discovered services.

The tool consists of the following RESTful services, and a set of Eclipse plugins that have been developed to interact with such services.

**M2M Transformator** – The Model-to-Model (M2M) Transformator offers a set of model transformations.

**Synthesis Discovery Manager** – The Synthesis process and the Discovery process interact each other to retrieve, from the service base, those candidate services that are suitable for playing the participant roles required by the chore-

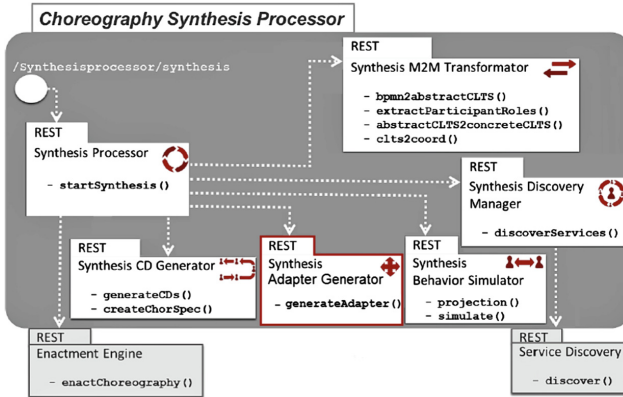


Fig. 3. REST architecture of the extended synthesis processor

ography specification, and hence, those services whose (offered and required) operations and behavior are compatible with the expected behavior as extracted from the choreography through projection.

**Behavior Simulator** – Once a set of concrete candidate services has been discovered, the synthesis process has to select them by checking, for each participant, if its expected behavior can be simulated by some candidate service. Note that, for a given participant, behavioral simulation is required since, although the discovered candidate services for it are able to offer and require (at least) the operations needed to play the role of the participant, one cannot be sure that the candidate services are able to support the operations flow as expected by the choreography.

**Coordination Delegate and Adapter Generators** – Once the services have been selected for all the choreography participants, the synthesis processor can generate the needed CDs and Adapters through the operations `generateCD()` and `generateAdapter()`, respectively.

In the following we introduce an example in the marketing and sale domain that will be then used in Sect. 5 to describe our method at work.

## 5 Method at Work

This section describes the proposed enhancement at work on the explanatory example introduced in Sect. 3. There are several frameworks and/or systems that implement/use EIPs in order to integrate applications. We have chosen Spring Integration framework<sup>5</sup> since it implements most of the EIPs, and it is well integrated with the Spring ecosystem. In particular, it is integrated with the Spring Web Services project<sup>6</sup>.

<sup>5</sup> <http://projects.spring.io/spring-integration/>.

<sup>6</sup> <http://projects.spring.io/spring-ws/>.

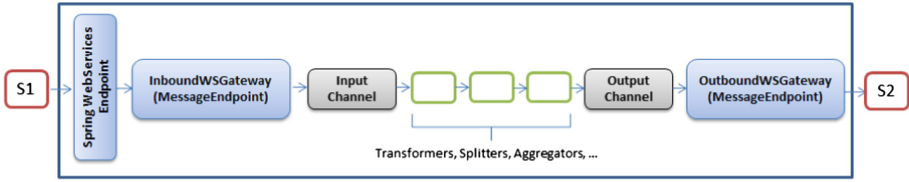


Fig. 4. Adapter architecture

Figure 4 describes the architecture of the generated adapters by using Spring Web Services and Spring Integration. In particular, the Spring Web Services Endpoint is the Web Service that mediates the interaction of the Service S1 and the Service S2. When the Service S1 calls an operation *op1* by sending a message *m1*, the Endpoint receives the operation and put the message into the input channel by using Inbound Web Service Gateways. The chain of EIPs, from the Input Channel to the Output Channel, is generated by the synthesis processor depending of the found interoperability issues (e.g., signature and protocol mismatches). The chain is made of one or more EIPs handlers to, e.g., *Message Transformers*, used to convert a message from one format to another one; *Message Routers*, used to decouple a message source from the ultimate destination of the message, and so on. *Message Routers* patterns can be, e.g., *Splitter*, *Aggregator*, *Resequencer* [6].

Referring to the explanatory example in Fig. 1, we focus on the **Subscribe User Card** and **Add Product** Choreography Tasks.

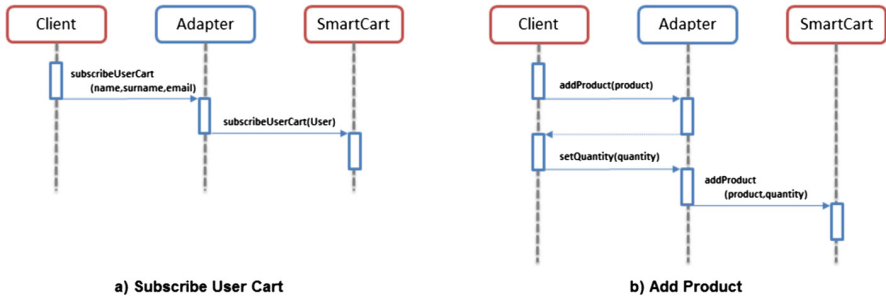


Fig. 5. Adapter example

Concerning **Subscribe User Card** choreography task, let us suppose that the **Client** service is able to invoke a **subscribeUserCard** operation expecting as input message three string elements, one for **name**, one for **surname**, and one for **email**. The XSD schema codifying the input message is shown in Listing 1.1. Let us also suppose that the **SmartCard** service offers a **subscribeUserCard** operation expecting as input message only one **User** element. As shown in Listing 1.2, this element is a complex type encapsulating the following string elements: **firstname**, **lastname**, and **mail**.

In order to let the `Client` and the `SmartCart` services to communicate, the processor generates an adapter that offers the operation `subscribeUserCard` (`name`, `surname`, `email`) so that when the `Client` invokes `subscribeUserCard` (`name`, `surname`, `email`) operation, the adapter transforms the first message (Listing 1.1) into the second one (Listing 1.2). This is done by generating an ad-hoc *Message Transformer* handler and adding it to the chain. At the end, the adapter invokes the `subscribeUserCard(User)` operation offered by the `Smart Cart` service. This behavior is shown in Fig. 5a.

Listing 1.1. input parameters of `subscribeUserCard` operation

```

1 <xsd:schema version="1.0" targetNamespace="http://choreosynth.disim.univaq.it/">
2   <xsd:element name="name" type="xsd:string"></xsd:element>
3   <xsd:element name="surname" type="xsd:string"></xsd:element>
4   <xsd:element name="email" type="xsd:string"></xsd:element>
5 </xsd:schema>

```

Listing 1.2. input parameters of `subscribeUserCard` operation discovered

```

1 <xsd:schema version="1.0" targetNamespace="http://choreosynth.disim.univaq.it/">
2   <xsd:complexType name="User">
3     <xsd:sequence>
4       <xsd:element name="firstname" type="xsd:string"></xsd:element>
5       <xsd:element name="lastname" type="xsd:string"></xsd:element>
6       <xsd:element name="mail" type="xsd:string" minOccurs="0" maxOccurs="unbounded">
7         ↪</xsd:element>
8     </xsd:sequence>
9   </xsd:complexType>
10 </xsd:schema>

```

Concerning the `Add Product` choreography task let us suppose that the `Client` service invokes two operations, `addProduct`(`product`) and `setQuantity` (`quantity`). Let us also suppose that the `SmartCart` service offers a `addProduct` (`product`,`quantity`) operation. Differently from the previous case, the adapter is now generated by using the *Message Router Aggregator* pattern. This pattern allows for accumulating the two messages (i.e., `product` and `quantity`) received from the `Client`, and subsequently invokes the `addProduct` (`product`, `quantity`) operation offered by `SmartCart` (as shown in the Fig. 5b).

The method for generating adapters exemplified above requires automated synthesis of I/O data mappings. To this end, the idea is to exploits a slightly modified version of the *Strawberry* tool [23] that allows for automatically inferring data mappings between different messages of two different Web services, i.e., `Client` and `SmartCart` in our case. *Strawberry* exploits (i) static data type analysis to analyze the type structure<sup>7</sup> of the two different messages; (ii) testing check if the two messages are also semantically correlated (since in general, considering the messages' type structure only is not sufficient). Efforts in this direction will be part of future work.

<sup>7</sup> E.g., the type structure of the XML Schema types of the messages in the WSDL of the considered services.

## 6 Related Work

The mediation/adaptation of protocols have received attention since the early days of networking. Indeed many efforts have been done in several directions including for example formal approaches to protocol conversion, like in [24, 25].

Recently, with the emergence of web services and advocated universal interoperability, the research community has been studying solutions to the automatic mediation of business processes [26, 27]. However, most solutions are discussed informally, making it difficult to assess their respective advantages and drawbacks.

Spitznagel and Garlan present an approach for formally specifying adapter wrappers as protocol transformations, modularizing them, and reasoning about their properties, with the aim to resolve component mismatches [28]. Although this formalizations supports modularization, automated synthesis is not treated at all hence keeping the focus only on adapter design and specification.

Passerone et al. use a game theoretic approach for checking whether incompatible component interfaces can be made compatible by inserting a converter between them which satisfies specified requirements. This approach is able to automatically synthesize the converter [29]. In contrast to our method, their method needs as input a deadlock-free specification of the requirements that should be satisfied by the adapter, hence delegating to the user a non-trivial specification task.

Recently, Bennaceur and Issarny presented an approach that, exploiting ontology reasoning and constraint programming, allows for automatically inferring mappings between components interfaces [30]. Importantly, these mappings guarantee semantic compatibility between the operations and data.

Rahm et al. propose a catalog of criteria for documenting the evaluations of schema matching systems [31]. In particular, the authors discuss various aspects that contribute to the match quality obtained as the result of an evaluation. In [32, 33] the authors present a generic schema match system called COMA, which provides an extensible library of simple and hybrid match algorithms and supports a powerful framework for combining match results. This framework can be used for systematically evaluate different aspects of match processing, match direction, match candidate selection, and computation of combined similarity, and different matcher usages.

Paolucci et al. propose a base algorithm [34] for semantic matching between service advertisements and service requests based on DAML-S, a DAML-based language for service description. The algorithm proposed differentiate between four degrees of matching and can be used for automatic dynamic discovery, selection and inter-operation of web services.

## 7 Conclusion and Future Works

In this paper, we propose a way to enhance the previous CHOReOS approach to the automatic synthesis of choreography-based systems, and we report on the

novel idea we are currently investigating within CHOReVOLUTION to achieve choreography adaptation and evolution. In particular, the idea is to automatically generate adapters by combining different EIPs depending on a notion of protocol mediation and data similarity. In order to automatically synthesize the adapters we propose an extension to our CHOReOSynt tool by introducing a new RESTful service called **Synthesis Adapter Generator**. Furthermore, we propose a pipe-and-filter-based architecture of the generated adapters by using Spring Web Services and Spring Integration frameworks.

An explanatory example has been used to show two types of adaptation based on the Message Transformation pattern and the Message Aggregator pattern. The former plays a very important role by allowing the mediation of loose-coupling Message Producers and Message Consumers, which do not agree on a common data format. The latter is a type of Message Endpoint that receives multiple Messages and combines them into a single Message.

As future work, our plan is to fully implement the proposed extension and validate it on the case studies of the CHOReVOLUTION project.

Moreover, in order to achieve even more ambitious objectives within the CHOReVOLUTION project and to improve the applicability of the approach, we plan to extend it so as to deal with security aspects of the choreographies. This would allow for dealing with multiple services that belong to different security domains governed by different authorities, and use different identity attributes. This can be achieved by integrating EIPs with Security Patterns [35].

**Acknowledgment.** This research work has been supported by the Ministry of Education, Universities and Research, prot. 2012E47TM2 (project IDEAS - Integrated Design and Evolution of Adaptive Systems), by the European Union's H2020 Programme under grant agreement number 644178 (project CHOReVOLUTION - Automated Synthesis of Dynamic and Secured Choreographies for the Future Internet), and by the Ministry of Economy and Finance, Cipe resolution n. 135/2012 (project INCIPICT - INnovating CIty Planning through Information and Communication Technologies).

## References

1. European Commission: Digital Agenda for Europe - Future Internet Research and Experimentation (FIRE) initiative (2015)
2. Autili, M., Di Ruscio, D., Di Salle, A., Inverardi, P., Tivoli, M.: A model-based synthesis process for choreography realizability enforcement. In: Cortellesa, V., Varró, D. (eds.) FASE 2013 (ETAPS 2013). LNCS, vol. 7793, pp. 37–52. Springer, Heidelberg (2013)
3. Autili, M., Di Salle, A., Tivoli, M.: Synthesis of resilient choreographies. In: Gorbenko, A., Romanovsky, A., Kharchenko, V. (eds.) SERENE 2013. LNCS, vol. 8166, pp. 94–108. Springer, Heidelberg (2013)
4. Salle, A.D., Inverardi, P., Perucci, A.: Towards adaptable and evolving service choreography in the future Internet. In: IEEE Services, pp. 333–337 (2014)
5. Autili, M., Inverardi, P., Tivoli, M.: Automated synthesis of service choreographies. IEEE Softw. **32**(1), 50–57 (2015)

6. Hohpe, G., Woolf, B.: *Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions* - Printing 2011. Addison-Wesley Longman, Boston (2004)
7. Basu, S., Bultan, T.: Choreography conformance via synchronizability. In: *Proceedings of WWW* (2011)
8. Calvanese, D., Giacomo, G.D., Lenzerini, M., Mecella, M., Patrizi, F.: Automatic service composition and synthesis: the roman model. *IEEE Data Eng. Bull.* **31**(3), 18–22 (2008)
9. Hallé, S., Bultan, T.: Realizability analysis for message-based interactions using shared-state projections. In: *Proceedings of FSE*, pp. 27–36 (2010)
10. Pathak, J., Lutz, R., Honavar, V.: Moscoe: an approach for composing web services through iterative reformulation of functional specifications. *Int. J. Artif. Intell. Tools* **17**, 109–138 (2008)
11. Salaün, G.: Generation of service wrapper protocols from choreography specifications. In: *Proceedings of SEFM* (2008)
12. Poizat, P., Salaün, G.: Checking the realizability of BPMN 2.0 choreographies. In: *Proceedings of SAC 2012* (2012)
13. Gössler, G., Salaün, G.: Realizability of choreographies for services interacting asynchronously. In: *Arbab, F., Ölveczky, P.C. (eds.) FACS 2011. LNCS, vol. 7253*, pp. 151–167. Springer, Heidelberg (2012)
14. Basu, S., Bultan, T., Ouederni, M.: Deciding choreography realizability. In: *Proceedings of POPL. ACM* (2012)
15. Gudemann, M., Poizat, P., Salaün, G., Dumont, A.: VerChor: a framework for verifying choreographies. In: *Cortellessa, V., Varró, D. (eds.) FASE 2013 (ETAPS 2013). LNCS, vol. 7793*, pp. 226–230. Springer, Heidelberg (2013)
16. Salaün, G., Bultan, T., Roohi, N.: Realizability of choreographies using process algebra encodings. *IEEE TSC* **5**(3), 290–304 (2012)
17. Ouederni, M., Salaün, G., Bultan, T.: Compatibility checking for asynchronously communicating software. In: *Fiadeiro, J.L., Liu, Z., Xue, J. (eds.) FACS 2013. LNCS, vol. 8348*, pp. 310–328. Springer, Heidelberg (2014)
18. Basu, S., Bultan, T.: Automatic verification of interactions in asynchronous systems with unbounded buffers. In: *Proceedings of ASE*, pp. 743–754 (2014)
19. Gudemann, M., Poizat, P., Salaün, G., Dumont, A.: VerChor: a framework for verifying choreographies. In: *Cortellessa, V., Varró, D. (eds.) FASE 2013 (ETAPS 2013). LNCS, vol. 7793*, pp. 226–230. Springer, Heidelberg (2013)
20. Gudemann, M., Salaün, G., Ouederni, M.: Counterexample guided synthesis of monitors for realizability enforcement. In: *Chakraborty, S., Mukund, M. (eds.) ATVA 2012. LNCS, vol. 7561*, pp. 238–253. Springer, Heidelberg (2012)
21. Shaw, M., Garlan, D.: *Software Architecture - Perspectives on an Emerging Discipline*. Prentice Hall, Upper Saddle River (1996)
22. Autili, M., Ruscio, D.D., Salle, A.D., Perucci, A.: Choreosynt: enforcing choreography realizability in the future Internet. In: *Proceedings of FSE*, pp. 723–726 (2014)
23. Bertolino, A., Inverardi, P., Pelliccione, P., Tivoli, M.: Automatic synthesis of behavior protocols for composable web-services. In: *Proceedings of ESEC/FSE* (2009)
24. Calvert, K.L., Lam, S.S.: Formal methods for protocol conversion. *IEEE J. Sel. Areas Commun.* **8**(1), 16 (1990)
25. Lam, S.S.: Correction to “protocol conversion”. *IEEE TSE* **14**(9), 1376 (1988)
26. Vaculín, R., Sycara, K.: Towards automatic mediation of OWL-S process models. In: *Proceedings of IEEE Web Services* (2007)



27. Vaculín, R., Neruda, R., Sycara, K.: An agent for asymmetric process mediation in open environments. In: Kowalczyk, R., Huhns, M.N., Klusch, M., Maamar, Z., Vo, Q.B. (eds.) *Service-Oriented Computing: Agents, Semantics, and Engineering*. LNCS, vol. 5006, pp. 104–117. Springer, Heidelberg (2008)
28. Spitznagel, B., Garlan, D.: A compositional formalization of connector wrappers. In: *Proceedings of ICSE* (2003)
29. Passerone, R., Alfaro, L.D., Henzinger, T.A., Sangiovanni-Vincentelli, A.L.: Convertibility verification and converter synthesis: two faces of the same coin. In: *Proceedings of ICCAD* (2002)
30. Bennaceur, A., Issarny, V.: Automated synthesis of mediators to support component interoperability. *IEEE TSE* **41**(3), 221–240 (2015)
31. Do, H.H., Melnik, S., Rahm, E.: Comparison of schema matching evaluations. In: *Web, Web-Services, and Database Systems*, pp. 221–237 (2002)
32. Do, H.H., Rahm, E.: COMA - a system for flexible combination of schema matching approaches. In: *Proceedings of VLDB*, pp. 610–621 (2002)
33. Massmann, S., Engmann, D., Rahm, E.: COMA++: results for the ontology alignment contest OAEI 2006. In: *Proceedings of OM/ISWC* (2006)
34. Paolucci, M., Kawamura, T., Payne, T.R., Sycara, K.: Semantic matching of web services capabilities. In: Horrocks, I., Hendler, J. (eds.) *ISWC 2002*. LNCS, vol. 2342, pp. 333–347. Springer, Heidelberg (2002)
35. Schumacher, M., Fernandez-Buglioni, E., Hybertson, D., Buschmann, F., Sommerlad, P.: *Security Patterns Integrating Security and Systems Engineering*. Wiley, Verlag (2005)

# An Experimental Evaluation on Runtime Verification of Self-adaptive Systems in the Presence of Uncertain Transition Probabilities

Kento Ogawa<sup>(✉)</sup>, Hiroyuki Nakagawa, and Tatsuhiro Tsuchiya

Graduate School of Information Science and Technology, Osaka University,  
1-5 Yamadaoka, Suita, Osaka 565-0871, Japan  
{o-kento,nakgawa,t-tutiya}@ist.osaka-u.ac.jp

**Abstract.** Self-adaptive systems can deal with environmental changes by changing their own behaviors. Since self-adaptive systems modify their own behaviors dynamically, runtime verification is necessary to guarantee the correctness of the systems' behaviors. Discrete time Markov chain model checking is a promising approach for implementing runtime verification; however, the computational cost of the current model checking approach increases as the number of parameterized transition probabilities increases. In this study, we conduct experiments on various instances of Markov chain models and demonstrate that repeated application of Laplace expansion leads to the large computational cost. The results suggest that an approach to reducing the number of times of performing Laplace expansion should be developed.

## 1 Introduction

Software systems are required to maintain high performance of the systems under high reliability. However, the external environment of the systems changes over time. Since the behaviors of the systems may become inappropriate in the environment that has changed in some cases, some changes of the environment prevent the systems from maintaining high performance. Therefore, the implementation of self-adaptive systems, which change their behaviors to adapt environmental changes, has been strongly desired in recent years [1, 2].

Self-adaptive systems modify their behaviors to adapt to environmental changes. Since the new behaviors that the systems have modified lead to bad results in some cases, the systems that have modified the behaviors do not satisfy system requirements. Therefore, self-adaptive systems are required to execute *model checking* [3] dynamically, which is the method for verifying whether systems satisfy system requirements or not, that is, *runtime model checking* [4, 5]. Since runtime model checking for self-adaptive systems is executed while the systems are running, the computation for runtime model checking is required to be efficient. Various studies about self-adaptive systems are conducted, but

the implementation of self-adaptive systems still needs to address the problem of efficiency in runtime model checking, such as described in [1].

Model checking is the method for verifying whether a system satisfies system requirements expressed in a formal language. For example, a system is modeled as a discrete time Markov Chain (DTMC) [6], which is a state-transition diagram that has state transitions represented as probabilities. The requirements are expressed in probabilistic computational tree logic (PCTL), which can be used to describe the properties of the system. Filieri et al. [7] proposed an efficient approach to verify whether DTMC models satisfy the requirements expressed by PCTL. The advantage of this approach is that it can shift the computational cost from run time to design time. This approach separates the model checking activity in two steps. The process of the first step is executed at design time. In this step, systems are modeled by using DTMC. Uncertain transition probabilities are represented as variables. Then, an expression is generated using the known values and the variables. The process of the second step is executed at runtime. The values of the unknown parameters at design time are determined by observation. These values are substituted for the parameters in the expression. The satisfaction of the requirements is verified by evaluating the expression. The computation for model checking becomes efficient by shifting the large part of the computation from run time to design time.

In this method, a high number of variables lead to a large computational cost. DTMC models are represented as matrices, and model checking requires the computation of the *determinants*. The computation of the determinants is performed by using *Laplace expansion* and *LU-decomposition*. Although the process of Laplace expansions and LU-decomposition are usually executed at design time, these processes have to be executed at runtime in some cases. In these cases, the process of generating an expression also should be efficient since the expression has to be generated at runtime. In this paper, we conduct experiments on various instance of DTMC models to find factors that largely affect the computational cost. We demonstrate that the large number of times of Laplace expansion leads to the large computational time. We will present a possible approach to reduce the computational cost.

This paper is organized as follows: Sect. 2 describes the overview of general model checking and the approach proposed in Filieri et al. [7]. Sections 3 and 4 explain DTMC and PCTL, respectively. Section 5 provides a detailed explanation of the existing approach. Sections 6 and 7 describe the experimental results, the discussion and research direction in the future. Section 8 concludes this paper.

## 2 Overview of Existing Approach

This section briefly describes how the existing approach described in [7] is applied to running systems. Figure 1 shows a flow of general model checking [8,9]. A system is monitored, and parameters of the system are collected while the system is running. The system is modeled by using the parameters. Model checking is executed by using the system model and system requirements. We verify whether

the system model satisfies the requirements. Since self-adaptive systems deal with the unknown parameters, such a model checking process should be executed at runtime. However, the computational cost of the model checking is large if the size of the systems is large.

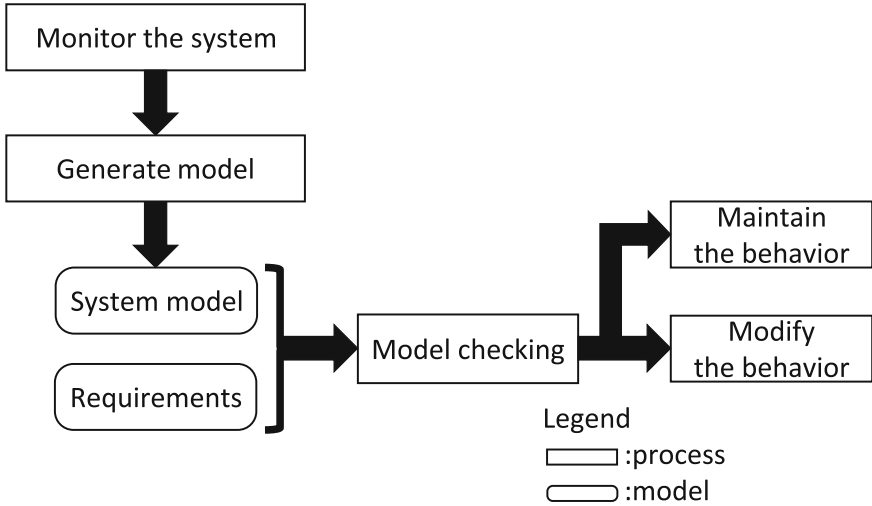


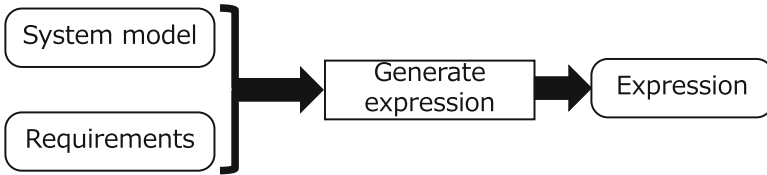
Fig. 1. Runtime model checking.

On the other hand, Fig. 2 shows the approach proposed in Filieri et al. [7]. In order to realize requirements at runtime, this approach separates model checking activity in two steps, executed at design time and runtime. In the step at design time, expressions are generated from a system model represented in DTMC and requirements expressed in PCTL; unknown parameters in the DTMC model at design time are represented as variables. In the second step at runtime, by monitoring the system, the values of the unknown parameters are collected, and the collected values are substituted for the variables. As a result, we verify whether the system model satisfies the requirements or not.

### 3 Discrete Time Markov Chain

In the existing approach described in [7], a system is modeled using Discrete Time Model Chain (DTMC) [6]. A DTMC model is a state transition diagram that has state transitions represented as probabilities. *States* represent possible configurations of the system. Transitions among states occur at discrete time and have associated probabilities. DTMC is a random process that has transitions from one state to another state on a state space. DTMC must respect the following property of Markov chains: the probability distribution of the next

■ Before the system runs



■ While the system is running

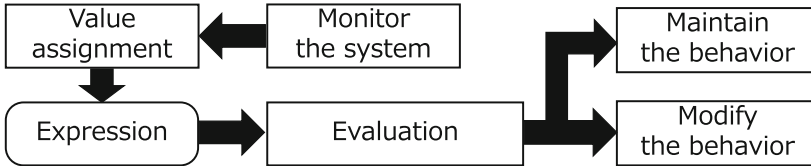


Fig. 2. Existing approach.

state depends only on the current state and not on the sequence of events that preceded the current state.

Formally, a DTMC is represented as a tuple  $(S, S_0, P, L)$  where

- $S$  is a set of states
- $S_0 (\subseteq S)$  is a set of initial states
- $P : S \times S \rightarrow [0, 1]$  is a stochastic matrix. An element of  $P(s_i, s_j)$  represents the probability that the next state of the process will be  $s_j$  given the current state is  $s_i$ .
- $L : S \rightarrow 2^{AP}$  is a labeling function, where  $AP$  is a set of atomic proposition that represents a basic property.

We call a state  $s \in S$  with  $P(s, s) = 1$  *absorbing state*, which is also used in [7]. If a DTMC model contains at least one absorbing state, such a DTMC model is called an *absorbing DTMC*. A state that is not absorbing state is called *transient state*.

We show an example of a DTMC model in Fig. 3. This model represents a self-adaptive system model. The transition probabilities of a self-adaptive system change due to the behavioral changes. Some of the transition probabilities cannot be determined at design time. In Fig. 3, transition probabilities that cannot be determined at design time are represented as variables to use in model checking. For example, a cleaning robot detects obstacles by using sensors that the robot has and decides a direction to move by using data collected by the sensors. The robot moves different directions, such as forward, turning and backward. Transition probabilities of moving are unknown before deciding the path; therefore, the probabilities are represented as variables.

A DTMC model can be represented as a matrix. An absorbing DTMC that has  $r$  absorbing states and  $t$  transient states is represented by the matrix  $P$  that is in the following canonical form:

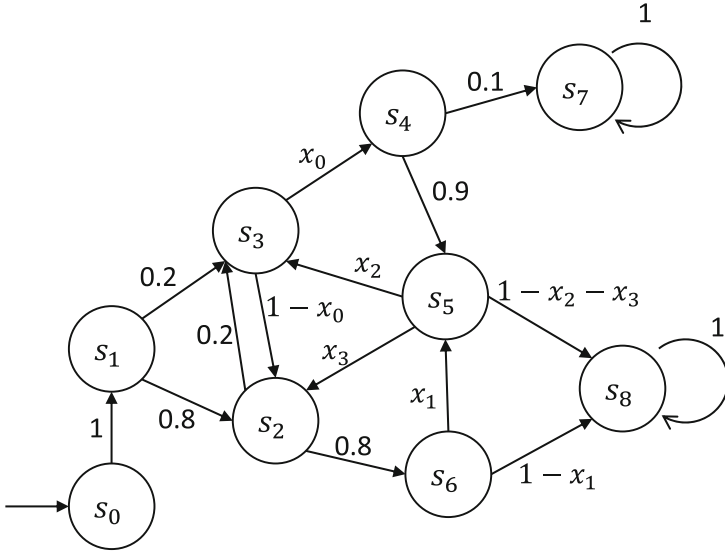


Fig. 3. An example of DTMC: Cleaning Robot.

$$P = \begin{pmatrix} Q & R \\ O & I \end{pmatrix} \tag{1}$$

where  $I$  is an  $r$  by  $r$  identity matrix,  $O$  is an  $r$  by  $t$  zero matrix,  $R$  is a  $t$  by  $r$  nonzero matrix and  $Q$  is a  $t$  by  $t$  matrix. The element  $q_{ij}$  of the matrix  $Q$  is the transition probability from the transient state  $s_i$  to the transient state  $s_j$ . The element  $r_{ik}$  of the matrix  $R$  is the transition probability from the transient state  $s_i$  to the absorbing state  $s_k$ . An absorbing state has the probability 1 from the absorbing state to itself. Therefore, the probabilities of absorbing states to themselves are represented as the identity matrix  $I$ . Since transition probability from an absorbing state to a transient state is 0,  $O$  is a zero matrix.

The following matrix represents the DTMC model illustrated in Fig. 3.

$$P = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.8 & 0.2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.2 & 0 & 0 & 0.8 & 0 & 0 \\ 0 & 0 & 1-x_0 & 0 & x_0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.9 & 0 & 0.1 & 0 \\ 0 & 0 & x_3 & x_2 & 0 & 0 & 0 & 0 & 1-x_2-x_3 \\ 0 & 0 & 0 & 0 & 0 & 0 & x_1 & 0 & 1-x_1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \tag{2}$$

The matrices  $Q$ ,  $R$ ,  $O$  and  $I$  for the matrix  $P$  are shown in Fig. 4.

The transition probability from the transient state  $s_i$  to the transient state  $s_j$  in two steps is calculated by  $\sum_{s_x \in S} P(s_i, s_x) \cdot P(s_x, s_j) = \sum_{s_x \in S} Q(s_i, s_x) \cdot$

$Q(s_x, s_j)$ . The  $k$ -step transition probability, with which the state  $s_j$  is reached from the state  $s_i$  in  $k$  steps, is the element  $q_{ij}^k$  of the matrix  $Q^k$ . Since the element  $q_{ij}^0$  of the matrix  $Q^0$  represents the transition probability from the state  $s_i$  to the state  $s_j$  in zero steps, this matrix  $Q^0$  is a  $t$  by  $t$  identity matrix. Due to the fact that  $R$  must be a nonzero matrix,  $Q$  has uniform-norm strictly less than 1, thus  $Q^n \rightarrow 0$  as  $n \rightarrow \infty$ , which implies that eventually the process will be absorbed with probability 1.

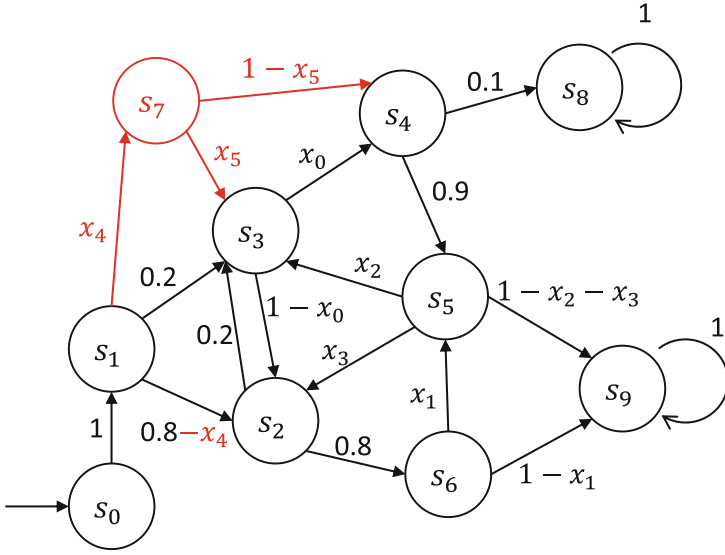
$$P = \left( \begin{array}{cccc|cccc}
 & & & \begin{matrix} \uparrow \\ Q \\ \downarrow \end{matrix} & & & & & \begin{matrix} \uparrow \\ R \\ \downarrow \end{matrix} & & \\
 \hline
 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 0 & 0 & 0.8 & 0.2 & 0 & 0 & 0 & 0 & 0 & 0 \\
 0 & 0 & 0 & 0.2 & 0 & 0 & 0.8 & 0 & 0 & 0 \\
 0 & 0 & 1-x_0 & 0 & x_0 & 0 & 0 & 0 & 0 & 0 \\
 0 & 0 & 0 & 0 & 0 & 0.9 & 0 & 0.1 & 0 & 0 \\
 0 & 0 & x_3 & x_2 & 0 & 0 & 0 & 0 & 1-x_2-x_3 & 0 \\
 \hline
 0 & 0 & 0 & 0 & 0 & 0 & x_1 & 0 & 1-x_1 & 0 \\
 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
 \hline
 & & & \begin{matrix} \downarrow \\ O \\ \uparrow \end{matrix} & & & & & \begin{matrix} \downarrow \\ I \\ \uparrow \end{matrix} & & \\
 \hline
 \end{array} \right)$$

**Fig. 4.** Matrices  $Q$ ,  $R$ ,  $O$ , and  $I$  of the matrix  $P$  illustrated in Exp. (2)

Figure 5 shows a model after adding state and transitions to the model illustrated in Fig. 3. This model represents the model of self-adaptive systems that change their behaviors with adding states and transitions. Here, to match the matrix of this model with Expression (1), we add the new state  $s_7$ , illustrated in Fig. 5. Since the transition probabilities from the state  $s_1$  to the state  $s_7$  and from the state  $s_7$  to the states  $s_3$  and  $s_4$  are unknown, the values of the probabilities are represented by the variables. This model has more variables than the model in Fig. 3. The transition matrix of this model is as follows:

$$P = \left( \begin{array}{cccccccccc}
 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 0 & 0 & 0.8 & 0.2 & 0 & 0 & 0 & x_4 & 0 & 0 \\
 0 & 0 & 0 & 0.2 & 0 & 0 & 0.8 & 0 & 0 & 0 \\
 0 & 0 & 1-x_0 & 0 & x_0 & 0 & 0 & 0 & 0 & 0 \\
 0 & 0 & 0 & 0 & 0 & 0.9 & 0 & 0 & 0.1 & 0 \\
 0 & 0 & x_3 & x_2 & 0 & 0 & 0 & 0 & 0 & 1-x_2-x_3 \\
 0 & 0 & 0 & 0 & 0 & 0 & x_1 & 0 & 0 & 1-x_1 \\
 0 & 0 & 0 & x_5 & 1-x_5 & 0 & 0 & 0 & 0 & 0 \\
 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
 \end{array} \right) \tag{3}$$

As shown in Expression (3), the new model has five variables. We study how computational cost depends on the number of variables in Sect. 6.



**Fig. 5.** The model added a new state into. To match the matrix of this model with Expression (1), the state numbers in Fig. 3 do not correspond to the numbers in Fig. 5.

### 4 Probabilistic Computational Tree Logic

Filieri et al. [7] describe system requirements that the DTMC model should satisfy by using Probabilistic Computational Tree Logic (PCTL), which express reliability property. PCTL is an extension of Computational Tree Logic (CTL) that allows for probabilistic quantification of described properties. CTL is a branching temporal logic and can express branching directly to describe the specification of the systems for model checking. PCTL is a stochastic time logic that introduces probabilistic concept to CLT and can describe quantitative formality related to possibility of branching.

PCTL is expressed as  $\mathcal{P}_{\bowtie p}(\cdot)$ . This expression represents whether the expressions in parentheses satisfied the constraints  $\bowtie p$  or not:  $p \in [0, 1]$ ,  $\bowtie p \in \{<, <, \geq, >\}$ . PCTL is defined by the following syntax:

$$\Phi ::= true \mid a \mid \Phi \vee \Phi \mid \Phi \wedge \Phi \mid \neg \Phi \mid \mathcal{P}_{\bowtie p}(\varphi) \tag{4}$$

$$\varphi ::= X \Phi \mid \Phi U \Phi \mid \Phi U^{\leq t} \Phi \tag{5}$$

The operators included in Expressions (4) and (5) are as follows:

- $X$ : “next”
- $U$ : “until”
- $F$ : “eventually” (sometimes called “future”)
- $G$ : “always” (sometimes called “globally”)

$X$  and  $U$ , which appear in path formulae, represent “next” and “until”, respectively.  $P_{<0.01}(X s = 1)$  means that the reachability probability from the current



state to the next state  $s_1$  is less than 0.01.  $F$  means that the statement is satisfied in the future. For example,  $P_{>0.98}(F s = 4)$  means the probability of eventually reaching state  $s_4$  is more than 0.98.  $G$  means “always (globally)”.  $P_{<0.99}(G s < 5)$  means the probability that the state number continues to be less than 5 is less than 0.05.

We conduct an experiment to compare the computational time for the determination of reachability property. The reachability property states that a target state is eventually reached from a given initial state. Since many safety properties are represented as reachability properties, we focus on reachability in the experiment. In most cases, the state to be reached is an absorbing state. Below is an example of a PCTL formula that represents a reachability property.

$$P_{\leq 0.001}(\text{true } U s = 9) = P_{\leq 0.001}(F s = 9) \quad (6)$$

Expression (6) indicates that the probability from an initial state to the absorbing state  $s_9$  has to be less than 0.001. In Sect. 6, we verify the requirement of the reachability property.

## 5 Existing Approach

As described in Sect. 2, system models generated by modeling self-adaptive systems by DTMC are represented as matrices. In this section, we describe how to determine a reachability property expressed by PCTL by using the computation of matrices.

The probability of moving from the transient state  $s_i$  to the absorbing state  $s_k$  is represented as the element  $p_{ik}$  of the matrix  $P$ . Note that this is the probability of moving in one step. In the case of more than 1 steps, we have to compute the probability of the transition from a transient state to another transient state and the transition from the transient state to an absorbing state. Similarly as for the matrix  $P$ , the probability from the transient state  $s_i$  to the transient state  $s_j$  in one step is the element  $q_{ij}$  of the matrix  $Q$ . The transition probability in two steps is the element  $q_{ij}^2$  of the matrix  $Q^2 (= Q \times Q)$ , and the transition probability in  $k$  steps is the element  $q_{ij}^k$  of the matrix  $Q^k$ . Therefore, the probability from the transient state  $s_i$  to the transient state  $s_j$  is the  $(i, j)$ -th element of the matrix  $I + Q + Q^2 + Q^3 + \dots = \sum_{k=0}^{\infty} Q^k$ . Let  $N$  denote this matrix. Since the other states cannot be reached from absorbing states, the probability from the transient state  $s_j$  to the absorbing state  $s_k$  is the element  $r_{jk}$  of the matrix  $R$ . The probability from the transient state  $s_i$  to the absorbing state  $s_k$  in some steps is defined as the element  $b_{ik}$  of the matrix  $B$ :

$$B = N \times R \quad (7)$$

The computation for the matrix  $B$  require the computation for the matrix  $N$ . The following expressions hold from  $N = I + Q + Q^2 + Q^3 + \dots = \sum_{k=0}^{\infty} Q^k$ .

$$\begin{aligned}
N &= I + Q + Q^2 + Q^3 + \dots \\
&= \sum_{k=0}^{\infty} Q^k \\
&= (I - Q)^{-1}
\end{aligned} \tag{8}$$

Since the matrix  $N$  is the inverse of  $(I - Q)$ , which  $(I - Q)^{-1}$  is defined as  $N$ , and which is computed by means of Gauss-Jordan elimination algorithm [10], the element  $n_{ij}$  of the matrix  $N$  is as follows:

$$n_{ij} = \frac{1}{\det(W)} \cdot \alpha_{ji}(W) \tag{9}$$

where  $\alpha$  is the cofactor obtained by multiplying  $(-1)^{(i+j)}$  by the determinant of the matrix with the  $i$ -th row and the  $j$ -th column removed. Due to Expressions (7) and (9), the element  $b_{ij}$  of the matrix is calculated as follows:

$$\begin{aligned}
b_{ik} &= \sum_{x \in 0 \dots t-1} n_{ix} \cdot r_{xj} \\
&= \frac{1}{\det(W)} \sum_{x \in 0 \dots t-1} \alpha_{xi}(W) \cdot r_{xj}
\end{aligned} \tag{10}$$

Here, the computation of  $t$  determinants with size  $t - 1$  is required. The computation of the determinants is executed by using Laplace expansion and LU decomposition. Laplace expansion removes variables from a matrix of a model. LU-decomposition computes determinants of the matrix containing no variables. The determinant  $|A|$  of the  $n \times n$  matrix  $A$  is the sum of the product of the element  $a_{ij}$  by each determinant of  $n$  sub-matrices of  $A$  with size  $(n - 1) \times (n - 1)$  (Fig. 6), where the expression of  $|A|$  in the case expanded in the  $i$ -th row is as follows:

$$|A| (= \det A) = \sum_{j=0}^n (-1)^{i+j} a_{ij} A'_{ij} \tag{11}$$

$A'_{ij}$  is the  $(n - 1) \times (n - 1)$  sub-matrix generated by removing the  $i$ -th row and the  $j$ -th column of the matrix  $A$ .

The computation of determinants is required to compute reachable probability. The computation of determinants is executed by using LU-decomposition and Laplace expansion; while the former requires the small computational cost, the latter requires the large computational cost. The rows that include variables representing unknown parameters is expanded by using Laplace expansion. The determinant of the sub-matrix that includes no variables by using Laplace expansion on several occasions is computed by using LU decomposition. The high number of times of Laplace expansions leads to the huge computational cost. However, the models of self-adaptive systems that have their behavioral changes including state and transition addition include have many variables that are represented as unknown parameters. The large number of variables leads to

$$\begin{array}{c}
 \begin{array}{|ccccccc|}
 \hline
 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
 0 & 0 & 0.8 & 0.2 & 0 & 0 & 0 \\
 0 & 0 & 0 & 0.2 & 0 & 0 & 0.8 \\
 \hline
 0 & 0 & 1-x_0 & 0 & x_0 & 0 & 0 \\
 0 & 0 & 0 & 0 & 0 & 0.9 & 0 \\
 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 0 & 0 & 0 & 0 & 0 & 0 & x_1 \\
 \hline
 \end{array} \\
 \\
 = (1-x_0) \begin{array}{|cccccc|}
 \hline
 0 & 1 & 0 & 0 & 0 & 0 \\
 0 & 0 & 0.2 & 0 & 0 & 0 \\
 0 & 0 & 0.2 & 0 & 0 & 0.8 \\
 0 & 0 & 0 & 0 & 0.9 & 0 \\
 0 & 0 & x_2 & 0 & 0 & 0 \\
 0 & 0 & 0 & 0 & 0 & x_1 \\
 \hline
 \end{array} + x_0 \begin{array}{|cccccc|}
 \hline
 0 & 1 & 0 & 0 & 0 & 0 \\
 0 & 0 & 0.8 & 0 & 0 & 0 \\
 0 & 0 & 0 & 0 & 0 & 0.8 \\
 0 & 0 & 0 & 0 & 0.9 & 0 \\
 0 & 0 & x_3 & 0 & 0 & 0 \\
 0 & 0 & 0 & 0 & 0 & x_1 \\
 \hline
 \end{array}
 \end{array}$$

Fig. 6. An example of Laplace expansion.

the large number of times of Laplace expansions and the large computational time (Fig. 7). Moreover, when the behavioral models of self-adaptive systems are largely changed, the pre-computed expressions at the design time no longer correspond to the new behaviors and therefore cannot be applied. In this case, the systems require to execute the process of pre-computation at runtime. Therefore, the computational cost of Laplace expansions should be reduced.

## 6 Experiment

In the previous sections, we pointed out a problem with runtime model checking of self-adaptive systems in the presence of state and transition addition: a rapid increase in computation time due to a large number of times of Laplace expansion. We also claimed that the computation cost rapidly increases when there are many variables in DTMC models of self-adaptive systems. In this section we evaluate the strength of the relationship between the expression generated at design time and the computation time. We implemented the existing approach in Java for the experiments. As problem instances, we randomly generated DTMC models with the size ranging from 10 to 150 states including

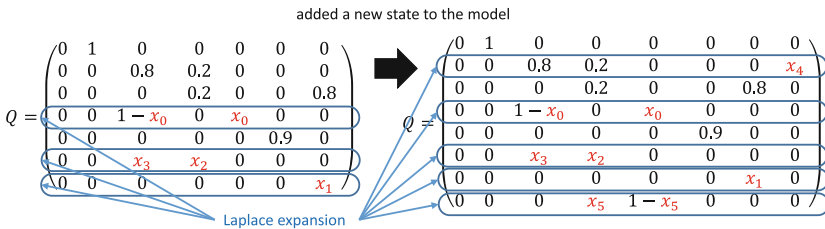
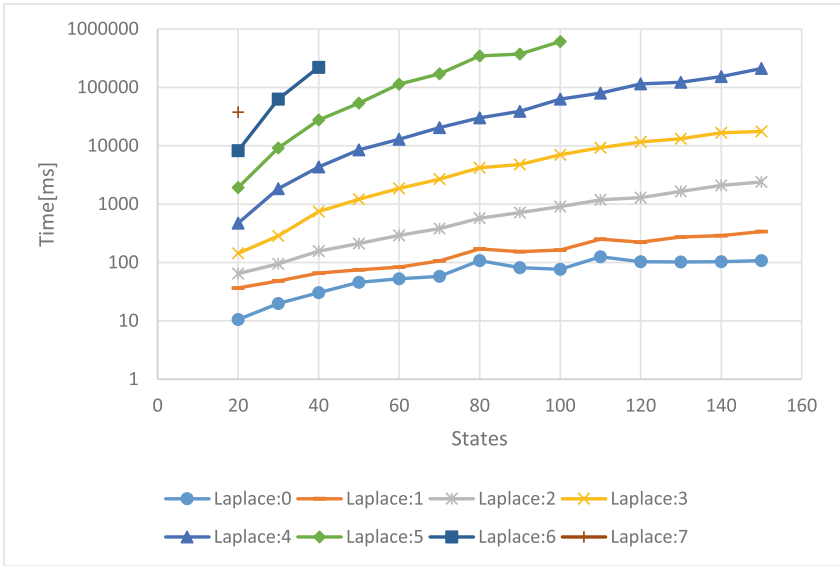


Fig. 7. The number of Laplace expansion (left:3, right:5).

four absorbing states. In these DTMC models, the number of outgoing transitions of each state follows a Gaussian distribution with mean 10 and standard deviation 2. Each row of the model has one variable. The experiment was conducted as follows. For each problem instance, Laplace expansion was executed  $k$  times. The value of  $k$  ranged from two to eight. We compare the averages of 10 runs for each configuration. In each run, we verified the requirements specification in PCTL, which represented reachability property to an absorbing state:  $P_{\leq 0.001}(true \ U \ s = MAX\_STATE) = P_{\leq 0.001}(F \ s = MAX\_STATE)$ , where MAX.STATE is the maximum number of states in DTMC.



**Fig. 8.** Scalability analysis.

Figure 8 shows the results of the experiment. In this figure, the x-axis represents the number of states in the DTMC models, while the y-axis represents the computation time required for runtime model checking in logarithmic scale. The different curves in the graph correspond to different numbers of times of Laplace expansion. As for the number of times of Laplace expansion greater than four, the model checking could not finish when the number of states increased. The results exactly show that the number of times of Laplace expansion and the number of states greatly affect computation time.

## 7 Discussion and Direction for Future Work

The results of the experiments described in Sect. 6 showed that the computational time is large if the number of times of Laplace expansion is large. The

behaviors of the self-adaptive systems, which we described in this paper, change according to environmental changes. In some cases, the models of the systems are largely changed, and as a result, pre-computation, which is usually conducted at design time, has to be performed at runtime. Therefore, it is important to construct DTMC models with a small number of parameters.

Our approach is *caching*, such as [11] for continuous-time Markov chain (CTMC), to decrease the computational time of generating an expression. When we deal with a self-adaptive system, which largely changes its behavior, the system has to generate an expression of the new model at runtime. In the process of caching, the system stores the intermediate expressions during generating an expressions in the pre-computation, and use the stored expressions corresponding to matrices in Laplace expansion at runtime. Our approach still cope with the problem of long time computation. If the number of the stored expressions increases, the time of searching the intermediate expression at runtime.

## 8 Conclusion

In this paper, we experimentally evaluated an existing approach of runtime model checking for self-adaptive systems. This approach makes runtime model checking efficient; however, self-adaptive systems have to execute the process of generating expressions at runtime when the systems largely change their behaviors. We conducted experiments and demonstrated that the computational cost rapidly increased according to the increase of the number of times of Laplace expansion. To minimize the computational cost, we first need to decline the number of times of Laplace expansion.

The current approach requires the large computational cost when it is applied to the self-adaptive systems that largely change their behaviors. In the future, we plan to construct an algorithm for minimizing the computational time by caching, in which the intermediate expressions of pre-computation at design time is used when generating the expressions of the new model after changing the behavior largely at runtime.

**Acknowledgments.** This work was supported by JSPS Grants-in-Aid for Scientific Research (No. 15K00097).

## References

1. Iftikhar, M.U., Weyns, D.: Activforms: active formal models for self-adaptation. In: Proceedings of the 9th International Symposium on Software Engineering for Adaptive and Self-Managing Systems, SEAMS 2014, pp. 125–134. ACM, New York (2014)
2. Zhang, J., Cheng, B.H.C.: Model-based development of dynamically adaptive software. In: Proceedings of the 28th International Conference on Software Engineering, ICSE 2006, pp. 371–380. ACM, New York (2006)

3. Baier, C., Katoen, J.P.: Principles of Model Checking (Representation and Mind Series). MIT Press, Cambridge (2008)
4. Blair, G., Bencomo, N., France, R.: Models@ run.time. *Computer* **42**(10), 22–27 (2009)
5. Morin, B., Barais, O., Jezequel, J.M., Fleurey, F., Solberg, A.: Models@ run.time to support dynamic adaptation. *Computer* **42**(10), 44–51 (2009)
6. Goseva-Popstojanova, K., Trivedi, K.S.: Architecture-based approach to reliability assessment of software systems. *Perform. Eval.* **45**(2–3), 179–204 (2001)
7. Filieri, A., Ghezzi, C., Tamburrelli, G.: Run-time efficient probabilistic model checking. In: Proceedings of the 33rd International Conference on Software Engineering, ICSE 2011, pp. 341–350. ACM, New York (2011)
8. Courcoubetis, C., Yannakakis, M.: The complexity of probabilistic verification. *J. ACM* **42**(4), 857–907 (1995)
9. Epifani, I., Ghezzi, C., Miranda, R., Tamburrelli, G.: Model evolution by run-time parameter adaptation. In: Proceedings of the 31st International Conference on Software Engineering, ICSE 2009, pp. 111–121. IEEE Computer Society, Washington, D.C. (2009)
10. Althoen, S.C., McLaughlin, R.: Gauss-Jordan reduction: a brief history. *Am. Math. Monthly* **94**(2), 130–142 (1987)
11. Gerasimou, S., Calinescu, R., Banks, A.: Efficient runtime quantitative verification using caching, lookahead, and nearly-optimal reconfiguration. In: Proceedings of the 9th International Symposium on Software Engineering for Adaptive and Self-Managing Systems, SEAMS 2014, pp. 115–124. ACM, New York (2014)

# Towards Automatic Decision Support for Bike-Sharing System Design

Maurice H. ter Beek<sup>(✉)</sup>, Stefania Gnesi, Diego Latella, and Mieke Massink

ISTI-CNR, Via G. Moruzzi 1, Pisa, Italy  
{terbeek,gnesi,latella,massink}@isti.cnr.it

**Abstract.** Public bike-sharing systems are a popular means of sustainable urban mobility, but their successful introduction in a city stands or falls with their specific designs. What kind of bikes and docking stations are needed, how many and where to install them? How to avoid as much as possible that stations are completely empty or full for some period? Hence, a bike-sharing system can be seen both as a highly (re)configurable system and as a collective adaptive system. In this paper, we present two complementary strategies for the evaluation of bike-sharing system designs by means of automated tool support. We use the *Clafer* toolset to perform multi-objective optimisation of attributed feature models known from software product line engineering and the recently developed mean field model checker *FlyFast* to assess performance and user satisfaction aspects of variants of large-scale bike-sharing systems. The combined use of these analysis approaches is a preliminary step in the direction of automatic decision support for the initial design of a bike-sharing system as well as its successive adaptations and recon-figurations that considers both qualitative and performance aspects.

## 1 Introduction

More and more cities are deploying public bike-sharing systems (BSS) as a sustainable urban mode of transportation [22]. The concept is simple: a user arrives at a docking station, rents a bike, uses it for a while and returns it to a station close to their destination; payment is either per trip or by subscription. BSS potentially offer multiple benefits, among which the reduction of vehicular traffic, pollution, noise and energy consumption. To improve the efficiency and user satisfaction of BSS, the load between different stations should be balanced, e.g. by using incentive schemes that influence the behaviour of users but also by efficient redistribution of bikes among stations.

The current third generation technology-based BSS are very different from the first generation free BSS introduced in Amsterdam roughly half a century ago. *Bicing*, the well-known and successful BSS of the city of Barcelona, currently consists of over 6,000 bikes and 420 stations. There are now similar BSS

---

Research partly supported by the EU FP7-ICT FET-Proactive project QUANTICOL (600708) and by the Italian MIUR project CINA (PRIN 2010LHT4KM).

in more than 500 cities worldwide. The largest can be found in China with upto 90,000 bikes and over 2,000 stations, one every 100m. Fourth generation BSS are already being developed. These include movable and solar-powered stations, electric bikes and smartphone real-time availability applications [22]. In the context of QUANTICOL ([www.quanticol.eu](http://www.quanticol.eu)) we collaborate with PisaMo S.p.A., an in-house public mobility company of the Municipality of Pisa, which introduced the BSS *CicloPi* in Pisa two years ago. This BSS, which currently consists of roughly 140 bikes and 15 stations, was supplied by Bicincittà S.r.l.

The design of a BSS is multi-faceted and complex. First of all, BSS are composed of many components, among which bikes and stations, but also human users. The latter form an intrinsic part of the BSS and their individual patterns of behaviour have a decisive impact on the collective usability and performance of a BSS, which is highly dynamic. Furthermore, there are questions concerning costs of installing and running a BSS, maintenance, specific user preferences and needs and specificities of the city architecture. Hence, BSS can be seen as highly (re)configurable systems and collective adaptive systems<sup>1</sup> (CAS). Our long-term goal is to be able to provide automatic decision support for the initial design of a BSS to be deployed in a city, as well as for successive adaptations and reconfigurations that consider both qualitative and performance aspects. In this context, it is important to realise that the design and behaviour of the individual entities from which a BSS is composed, may exhibit variability not only in the kind of features but also in the quantitative characteristics of features.

In [3–6], we studied product lines of BSS and their bikes, respectively. In this paper, we make use of these product lines to analyse different configurations. Software product line engineering (SPLE) is an engineering approach aimed at cost-effectively developing a variety of (software-intensive) products from a common reference model or architecture, i.e. that together form a (software) product line. Commonalities and differences are defined in terms of *features* and variability models encode exactly those combinations of features forming valid products. Actual product configuration during application engineering is thus reduced to selecting desired options in the variability model. We extend the product lines from [3, 4, 6] by explicitly taking feature attributes and feature cardinalities into account. The former enrich variability models with quantitative constraints, while the latter allow us to explicitly distinguish BSS configurations by the specific number of stations and bikes of each variant that are used.

We will first perform multi-objective optimisation of a BSS variability model with the recently developed toolset Clafer [1], after which we will study the use of the recently developed on-the-fly mean field model checker FlyFast [18, 20] to analyse behavioural and performance aspects of BSS configurations. Clafer-MOO(Visualizer) [23] allows to compare system configurations (variants) with respect to various quality dimensions (e.g. cost), select the most desirable one and analyse the impact of reconfigurations on a variant's quality dimensions.

<sup>1</sup> These are systems consisting of a large number of spatially distributed heterogeneous entities with decentralised control and varying degrees of complex autonomous behaviour able to adapt to changing circumstances.



Model checking is a widely used, powerful approach to the automatic verification of concurrent, distributed systems, including performance aspects. It is an efficient procedure that, given an abstract system model  $\mathcal{M}$ , decides whether  $\mathcal{M}$  satisfies a (temporal) logic formula  $\Phi$ . Currently, the integration of mean field and fluid approximation techniques with model-checking [8, 9, 17, 18, 20] is receiving increased attention as a way to obtain highly scalable formal methods supporting the design of *large-scale* CAS for which performance aspects are essential to their desired behaviour. Mean field approximation techniques originate in statistical physics and biochemistry where they are used to analyse large-scale phenomena like particle interaction and chemical reactions between molecules of different substances. The key idea of such approximation techniques is to replace the actual stochastic or probabilistic interactions in a system, which often lead to a combinatorial explosion of possibilities, by an approximation of the average system behaviour over time in terms of the numbers (fractions) of elements present in a population. In our setting, we assume that the elements have a small number of local states, and we consider the number (fraction) of elements (agents) that are in a particular local state [16, 21]. The change over time of these fractions can be defined as the solution of a set of ordinary differential equations or, in our case, difference equations, given an initial state of the overall system, and approximate its evolution over time. Informally speaking, the larger the populations in the system, the better the quality of the approximation. Typically their size is in the order of hundreds, thousands or even better, millions.

A more technical introduction to mean field and fluid approximation can be found in [10]. For what concerns mean field model checking, the idea is to analyse the properties of the behaviour of a single agent in the context of the overall system behaviour (approximated as described before). The difficulty in model-checking the properties of such an agent is that the probabilities involved in its behaviour are not constant but may, e.g., depend on the changing fractions of specific agents in the system over time. So the probabilities in the model of the agent under study are time-dependent or time-inhomogeneous, or more formally time-inhomogeneous discrete time Markov chains (IDTMC). In [20], we analysed performance aspects of a BSS with homogeneously distributed resources.

An important characteristic of BSS is the probability that stations are empty or full, since these situations generate distress among users and discourage them to use the BSS. In an online survey conducted by Froehlich [13] on the *Bicing* BSS in Barcelona in 2009 about the experience of users with bike sharing, it turned out that 75% of the users stated commuting as a motivation to sign up for membership. Moreover, the same users identified “finding an available bike and a parking slot” as the two most important problems encountered (76% and 66% of the 212 respondents, respectively). These problems should therefore be addressed in the best possible way within the obvious budget constraints of cities, and at the same time keeping the number of kilometres made by vans that are involved in the (unavoidable) redistribution of bikes as small as possible.

We thus use the on-the-fly mean field model checker FlyFast [18,20] for the analysis of properties of selected stations in a BSS and for a mean field analysis of the BSS [21]. This prototypical tool was developed for the analysis of discrete time Markov population models. This differs from fluid model-checking approaches developed for the analysis of *continuous time* Markov population models as in [8,9,17] which moreover use a global model-checking approach, i.e. an approach in which a formula is analysed for *all* states. The latter approach always requires a full state space search, whereas an on-the-fly approach generates only as much of the state space as necessary to analyse the property. Especially in case of *conditional* reachability properties this may be much more efficient. On-the-fly mean field model checking on discrete time Markov population models can also be used to approximate global fluid model checking of continuous Markov population models under certain conditions (see [19] for further details).

The paper is organised as follows. In Sect. 2 we define a variability model of a BSS and analyse it with ClaferMOOVisualizer in Sect. 3. In Sect. 4 we capture BSS behaviour in a Markov population model and analyse it with FlyFast in Sect. 5. Section 6 outlines how to combine these approaches in a future decision support system for BSS design.

## 2 Attributed Feature Model of BSS

The de facto standard variability model in SPLE is a *feature model* [24]. A *feature* characterises a stakeholder visible piece of functionality of a product or system and a feature model provides a compact representation of all possible products of a product line or configurable system in terms of their features (behaviour is not captured). However, there may be hundreds of features or configurable options, which easily leads to superfluous or contradictory variability information (e.g. ‘false’ optional or ‘dead’ features). There is a lot of work on computer-aided analyses of variability models to extract valid products and detect anomalies [7].

Graphically, features are nodes of a rooted tree and relations between them regulate their presence in products: *optional* features may be present provided their parent is; *mandatory* features must be present provided their parent is; exactly one *alternative* feature must be present provided their parent is; at least one *or* feature must be present whenever their parent is; a *requires* constraint indicates that the presence of a feature requires that of another; an *excludes* constraint indicates that two features are mutually exclusive. We identify a product  $\mathcal{P}$  from the product line with a non-empty subset  $\mathcal{P}_{\mathcal{F}}$  of the set  $\mathcal{F}$  of features. Deciding whether a product satisfies a feature model can be reduced to Boolean satisfiability (SAT), which can be effectively computed with SAT solvers [2].

In this paper we consider a BSS with three types of stations. We assume stations with capacity 15 to be located in the city centre (C), those with capacity 5 in the periphery (P) and those with capacity 10 in between (M, for middle). We define configuration 1 of a size in the order of that of the BSS in Barcelona in 2009 [13]: 330 stations of which 100 of type P, 150 of type M and 80 of type C.

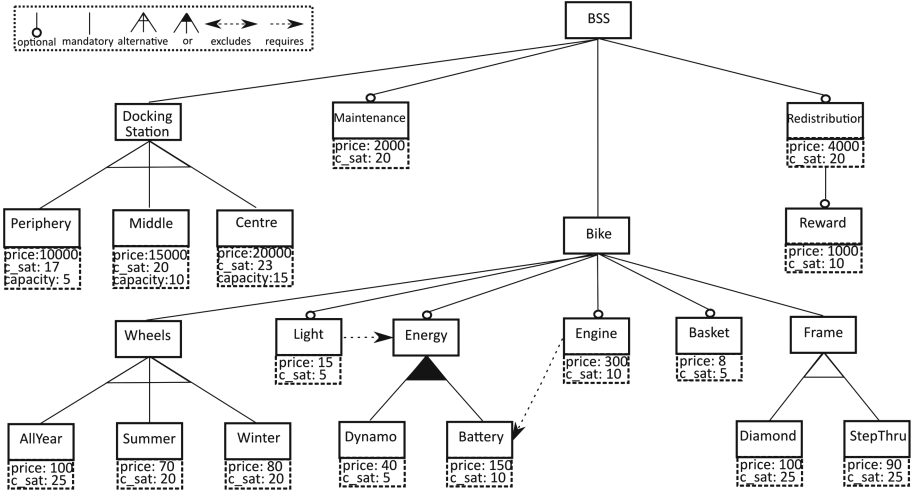


Fig. 1. Attributed feature model of a BSS.

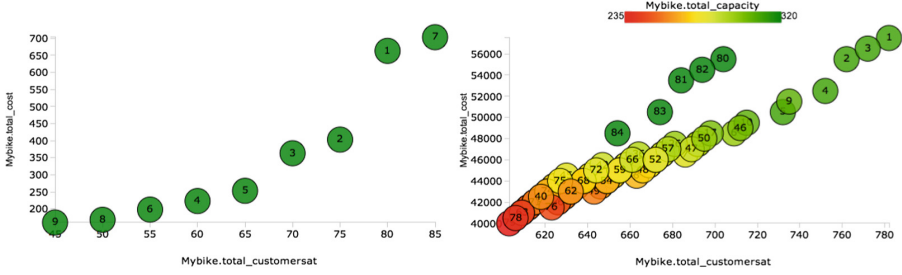
Subsequently, we present analyses of reconfiguring it into 397 stations of which 200 of type P, 150 of type M and 47 of type C (defined as configuration 2).

Once we equip features with attributes (e.g. `capacity(Centre) = 15`) we obtain an *attributed feature model* [7]. Now a product  $\mathcal{P}$  is a non-empty subset  $\mathcal{P}_{\mathcal{F}} \subseteq \mathcal{F}$  that moreover satisfies the additional quantitative constraints over feature attributes (e.g. `capacity(DockingStation) ≤ 10`). Complex quantitative constraints require satisfiability modulo theories (SMT) solvers like Z3 [11].

We consider the attributed feature model of a BSS depicted in Fig. 1, which we obtained by adding attributes to the feature model of the bike-sharing product line of [3, 4] (ignoring, mainly, the user feature) and replacing its `Bike` feature with the attributed feature model of the bikes product line of [6] (ignoring the computational unit feature). We extracted all features and values for the cost (in euros) and capacity attributes from documents received from Bicincittà. The values for customer satisfaction (`c_sat`) instead are estimates based on discussions with PisaMo and Bicincittà. Our research aims to replace them by more realistic values obtained from performance analyses like those we perform in Sect. 5.

### 3 Variability Analysis of BSS

In this section, we present initial variability analyses over the above BSS model. We use the attributed feature modelling capabilities of the lightweight textual SPL modelling language Clafer [1] and its extension ClaferMOO(Visualizer) [23]. Each feature can have attributes and quality constraints can be specified globally or in the context of a feature. Hence we can associate a price to each feature and a global constraint that only allows products (feature configurations) whose total



**Fig. 2.** Bubble graphs of Pareto fronts for Bike configurations (left) and the BSS (right).

costs remain within a predefined threshold value. If more than one attribute is associated with a feature this may result in multiple such optimisation objectives.

The *ClaferMOO* extension of *Clafer* was specifically introduced to support attributed feature models with complex multi-objective optimisation goals. The latter have a set of solutions, known as the Pareto front, that represents the trade-offs between several conflicting objectives. Intuitively, a Pareto-optimal solution is such that no objective can be improved without worsening another. A set of Pareto-optimal variants generated by *ClaferMOO* can be visualised (as a multi-dimensional space of optimal variants) and explored in the interactive online tool *ClaferMOOVisualizer*, which was specifically designed for SPL scenarios.

We first focus on the bike feature in Fig. 1 in isolation. Figure 2(left) depicts the result of optimising it by minimising the cost of a product while at the same time maximising customer satisfaction. If we inspect the nine resulting variants in detail, we see that variants 1 and 7 are very costly, since these concern electric bikes. We conclude that variants 2–5 offer reasonable customer satisfaction at an affordable cost. These are exactly the variants without **Engine** but with **Energy**.

Now that we selected the bike configuration(s), we turn to the rest of the BSS in Fig. 1. Recall that we are interested in comparing BSS configurations with different distributions of docking stations over a city. Therefore, we make use of the possibility to add feature cardinalities to a feature model specified in *Clafer*. We allow between 33 and 39 docking stations,<sup>2</sup> upto 10 stations of type P, 15 of type M and 8 of type C for the first configuration, upto 20 stations of type P, 15 of type M and 4 of type C for the second configuration and we use additional constraints to prohibit mixing the two configurations of docking stations. For this to work, we actually clone the features **Periphery**, **Middle** and **Centre** thus allowing us to distinguish the stations used in one configuration from those used in the other. This is a novel use of feature cardinalities and feature cloning.

Figure 2(right) depicts the result of optimising the model just described by minimising a product’s cost while at the same time maximising both customer satisfaction and capacity. This results in 84 variant configurations, coming from

<sup>2</sup> There is no technical limitation, but for ease of presentation and to speed up the computation we divided the number of stations and all costs by a factor 10.

the three optional features **Maintenance**, **Redistribution** and **Reward** and all possibilities to reconfigure configuration 1 (33 stations: 10 of type P, 15 of type M, 8 of type C) into a variant configuration with upto 39 stations, of which upto 20 of type P, upto 15 of type M and upto 4 of type C, thus including configuration 2. Variants 1–5 all concern configuration 2, while variants 80–84 all concern configuration 1. If we inspect these variants in detail, it turns out that configuration 2 has slightly less capacity than configuration 1 (310 vs. 320), offers much higher user satisfaction (732–782 vs. 654–704) at a generally higher cost (50500–57500 vs. 48500–55500). Note that the actual cost of a BSS with configuration 1 or 2 must be multiplied by 10 and, moreover, the cost of the bikes must be added. For capacities of 3100–3200 parking slots, some 1550–1600 bikes are needed [12]. Bike variants 2–5 cost between 223 and 403 euro per bike.

### 4 BSS as a Markov Population Model

For the analysis of performance-related aspects we now define the BSS as a Markov population model. Our model is inspired by the bike-sharing model of Fricker and Gast [12], which is a continuous time model based on homogeneous space where all locations are equally accessible to the users. We however use *discrete time* variants of this model. In the simple case, the model consists of  $N$  stations, each with a capacity of  $K$  parking slots. The number of bikes in the system is constant at  $s$  bikes per station on average, amounting to  $sN$  bikes in total. In every time step each station has the same probability that a user requests a bike. The probability that a bike is returned to a station depends on the number of bikes in circulation (i.e. not parked). Figure 3 shows a graphical representation of the model of a single bike station with  $K$  parking slots.

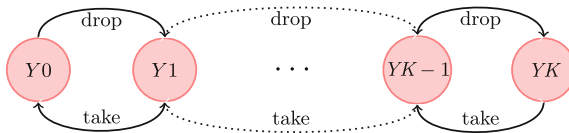


Fig. 3. A bike station.

Following [12], we denote the stochastic process composed of  $N$  stations in parallel by  $Y^N(i) = (Y_0^N(i), \dots, Y_K^N(i))$ , but in our case this denotes a discrete time Markov process where  $i$  denotes the discrete time step. Each element  $Y_k^N(i)$  in the vector denotes a random variable that gives the fraction of the total number of stations that have  $k$  bikes parked in it at time step  $i$ . In the following we address the request and return rates, assuming that we reached the system state  $(y_0, y_1, \dots, y_K)$ .

*Bikes Requested.* Assume there is one request of a bike per time unit per station on average, and for simplicity, assume that one time unit corresponds to 1 h. Then we have in total, for the whole system, on average,  $N$  requests for bikes per hour. The arrival of requests to the set of stations in which exactly  $k$  bikes are parked is then  $Ny_k$ . If such a request indeed arrives at a station with  $k$  parked bikes, then the fraction of stations  $y_k$  decreases with  $1/N$  and the fraction of stations  $y_{k-1}$  increases with  $1/N$ . For technical reasons a time unit of 1 h is too long to obtain accurate results using a mean field approximation in discrete time in the context of this BSS model. In what follows we therefore rescale the time units to smaller ones, adjusting the probabilities of occurrences of requests per time unit accordingly. In particular, we use  $\lambda$  to denote the number of requests per hour and assume time units of two minutes, obtaining the probability of a request for a bike in a time unit as  $\lambda/30$ .

*Bikes Returned.* The number of bikes that can be returned depends on the number of bikes in use, which can be obtained as the total number of bikes minus those that are parked in the stations. The total number of bikes is  $sN$ , where  $s$  is the average number of bikes per station and  $N$  the total number of stations. The average number of bikes parked in the stations is  $\sum_{k=1}^N ky_k$ , so the number of bikes in transit can be expressed as  $N(s - \sum_{k=1}^N ky_k)$ . We further assume that the average travel times, i.e. the time during which a bike is used for one trip, is  $1/\mu$  and that stations to which bikes are returned are selected at random. Under these assumptions, assuming for simplicity that  $\mu = 1$ , a bike is returned to a station with  $k \leq K - 1$  bikes with rate  $y_k \mu N (s - \sum_{k=1}^N ky_k)$  per hour. If such a bike is indeed returned to a station with  $k \leq K - 1$  bikes, then  $y_k$  is decreased by  $1/N$  and  $y_{k+1}$  is increased by  $1/N$ . As before, we will work with time steps of two minutes, which means that we work with  $\mu/30$  and that the probability to return a bike to a station with  $k$  bikes in one time step is  $y_k (\mu/30) N (s - \sum_{k=1}^N ky_k)$ . In all experiments in the sequel, one time unit corresponds to  $1/30$ th of an hour (i.e. 2 min).

We extend this basic model by introducing the populations P, M and C of stations, as anticipated in Sect. 2. Recall that next to a different capacity, each also has a different location in the city. These locations are characterised by different usage patterns of bikes by commuters. Such patterns were observed in data about usage in real BSS such as the one in Barcelona [13]. Most commuters live in the suburbs and take the bike in the morning to go to their work or to school in the centre and go homewards somewhere in the afternoon or towards the evening. This leads to clearly distinguishable usage patterns that show complementary behaviour: stations in the periphery have a high request rate in the morning whereas those in the centre, with some delay, have a high return rate. These flows are reversed in the afternoon and towards the evening. The stations in the middle show a more stable pattern as requests and returns are more balanced.

We model the flow of commuters between the periphery and the centre during daytime by a combination of oscillating functions. The latter are modelled

as populations within the same specification framework<sup>3</sup>. Their definition is inspired by the oscillatory process defined in [15]. Figure 4(left) shows three periods of 12 h each, reflecting periods during which the BSS is most used (we omit night time). The early morning of the first day period starts around time step 100. The curves show the change in request and return rates in the periphery and centre stations over the period, with a clear peak in the morning and a smaller, more distributed peak during the afternoon and evening hours. The rates oscillate around the average request rate  $\lambda = 1$  and return rate  $\mu = 4$ . For request rate 1 this means 330(=  $N$ ) requests per hour for the total system. When modulated by the oscillations, the number of requests varies between  $0.5 \times N = 115$  and  $1.3 \times N = 440$  per hour, assuming that each request corresponds to finding a bike. In data about the BSS in Barcelona (which has twice as many bikes and total capacity than in our model) the number of requests goes up to 1000 per hour in the morning. So given the capacity and number of bikes used in our model, these numbers seem in line with those in the literature.

Besides the modulation of request and returns during day in the various locations, also the number of stations located in the various parts of the city is of importance. Recall that we defined two configurations in Sect. 2. We furthermore assume that initially each station of type P is filled with 3 bikes, those of type M with 5 bikes, and those of type C with 7 bikes, leading to an average number of  $s = \frac{3 \times 100 + 5 \times 150 + 80 \times 7}{330} = \frac{1610}{330} \approx 4.8788$  bikes per station.

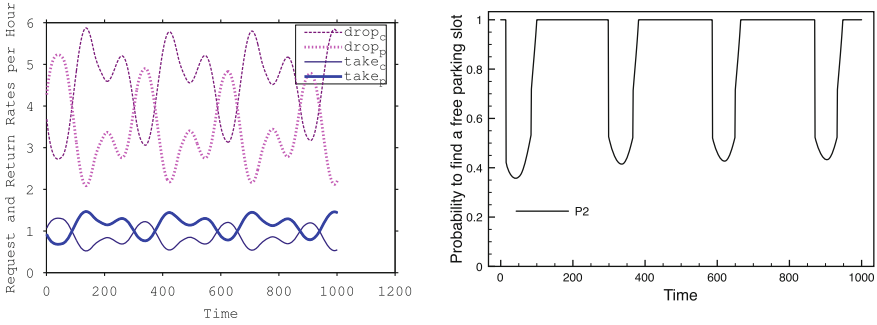
Figure 5 shows the variation of the fraction of stations of type P with 0, 1, 2, 3, 4 and 5 bikes parked, respectively, over a time horizon of 300 time units, comparing the average of 500 stochastic simulation runs in Fig. 5(left) with a mean field approximation of the model in Fig. 5(right). The results show a good correspondence. For the precise conditions of the model under which this happens we refer to [21]. In the case of the BSS model these conditions are satisfied as long as the population sizes used in the model are sufficiently large<sup>4</sup>. Recall that mean field models provide an approximative result on the *average* behaviour of a system. Individual simulations, in particular of models with relatively small populations, may show varying behaviour due to stochastic variability.

## 5 Examples of Performance Features of BSS

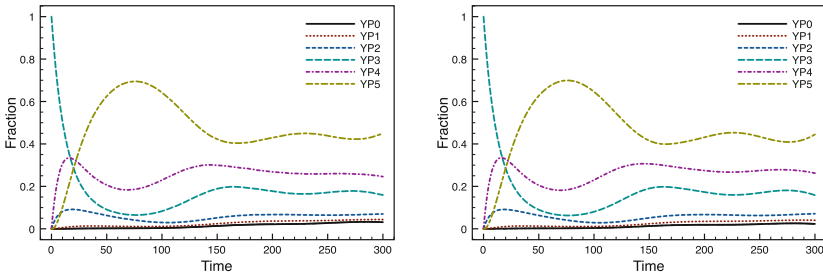
This section contains exemplary performance analyses over the BSS model of Sect. 4.

<sup>3</sup> The input language of the FlyFast model-checker used for the analysis is described in [18, 20] and consists of a simple high level language for Markov population models from which the mathematical structures on which the model-checking algorithms are based are automatically generated in an on-the-fly fashion.

<sup>4</sup> In the simulation we used population sizes multiplied by a factor 10, so considering 3300 stations instead of 330. This has nothing to do with the factor 10 scaling in Sect. 3.



**Fig. 4.** (left) Request and return rates per hour for stations of types P and C for three cycles of one day (night omitted) and (right) time dependent probability of property P2 for a model with  $\lambda = 1$  and  $\mu = 4$  per hour and of type configuration 1, i.e. with  $N = 330$  stations, of which 100 with capacity 5 in P and each holding 3 bikes initially; 150 with capacity 10 in M and 5 bikes; 80 with capacity 15 in C and 7 bikes (cf. Sect. 5.2).



**Fig. 5.** Fractions of stations of type P with 0 to 5 bikes, indicated respectively by YP0 to YP5: (left) simulation average over 500 runs and (right) mean field approximation.

### 5.1 Normalised Activity/Bicycle Data

One way to characterise usage patterns of stations is by their average filling degree over time. Froehlich et al. call this normalised activity/bicycle (NAB) data [13]. These filling degrees are usually collected from data observed in real systems, but we obtain them from the mean field analysis of the model as follows:

$$NAB = \frac{\sum_{k=1}^{K_i} y_{i_k} k}{K_i} \times \frac{N}{N_i}$$

where  $i \in \{P, M, C\}$  denotes the type of stations,  $K_i$  the capacity of stations of type  $i$ ,  $y_{i_k}$  the fraction of type- $i$  stations with  $k$  bikes parked in it of the total number of stations  $N$  and  $N_i$  the number of stations of type  $i$ . The multiplication with  $\frac{N}{N_i}$  is because we are interested in the filling degree of type- $i$  stations and not in the total number of stations  $N$ , whereas  $y_{i_k}$  gives the fraction with respect to the *total* number of stations  $N$ . For instance, let  $YP1, \dots, YP5$  denote the

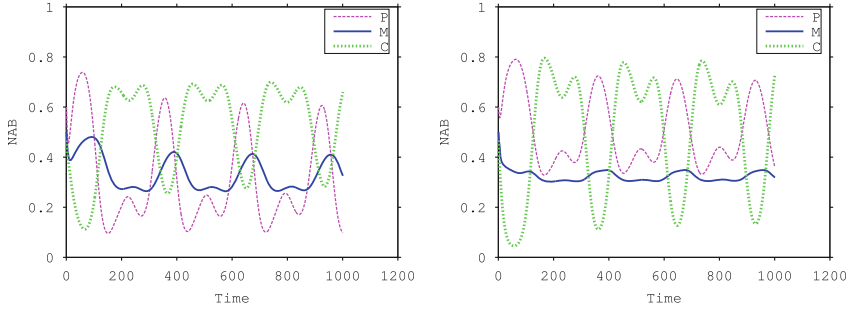


fractions of the total number of stations  $N$  of stations of type P with 1, 2, 3, 4 and 5 bikes, respectively, where  $K_p = 5$  is the capacity of such stations. Then  $BP = YP1 \times N + 2 \times YP2 \times N + 3 \times YP3 \times N + 4 \times YP4 \times N + 5 \times YP5 \times N$  is the total number of bikes parked in stations of type P. This gives on average  $ABP = BP/N_P$  bikes parked per station, where  $N_P$  is the number of stations of type P, and  $ABP/K_p$  is the filling degree of such a station of type P.

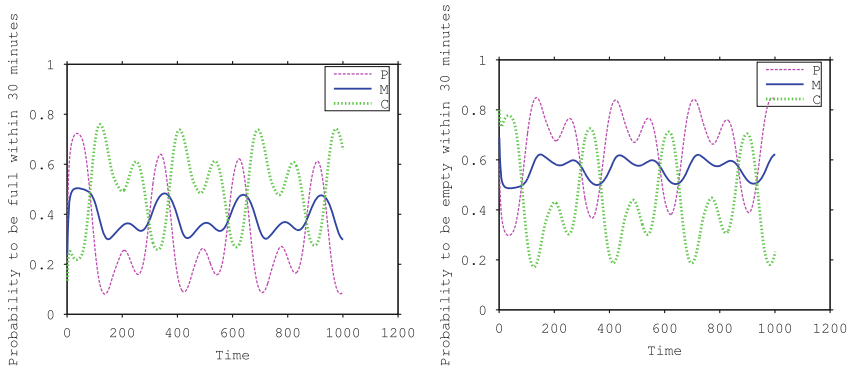
We now compare the NAB of configurations 1 and 2 defined in Sect. 2. The NAB of configuration 1, for an average request rate of one bike per station per hour and an average trip duration of 15 min, is shown in Fig. 6(left) for a period of approximately 3 days of 12 h each (night time is omitted). We clearly see the pattern of fluctuations of request rates also appear in the fluctuations in the filling degrees of the stations. Furthermore, a decrease in filling degree in the stations in the periphery in the morning corresponds to a slightly delayed increase in filling degree in the stations in the centre. The delay is due to the time it takes to cycle from the periphery to the centre. The reverse is happening in the evening when commuters go back home. Interestingly, the stations in the middle area show a more stable filling degree than those in the centre and in the periphery. Both patterns have been observed also in real data from the BSS in Barcelona [13].

The NAB of configuration 2 is shown in Fig. 6(right). In order to compare the configurations under the same assumptions on bike requests we have to correct for this fact in the model of configuration 2 because the request of bikes in the model are expressed per station. So we divide the requests (and returns) in the periphery for each station by 2, and double the requests (and returns) per station for those in the centre. Under these assumptions, the figure shows that in configuration 1 the stations in the periphery are in general less empty, but that the filling degree of those in the centre now fluctuates much more. Larger fluctuations might require more frequent interventions from BSS operators to rebalance the distribution of bikes. The situation in the periphery seems improved, but perhaps there are now more stations and bikes than really necessary. Many other BSS configurations and situations can be investigated. For example the effect on the average filling degree of an increase in requests for bikes when the density and number of stations and bikes is increased or the effect on resource usage and related implications on user satisfaction and costs of different distributions of stations over the various areas of the city. We limited ourselves here to only some examples due to space limitations.

Mean field based models provide an approximation of the average behaviour in a computationally efficient way compared to a simulation based approach. Moreover, the analysis time is independent of the number of stations in the BSS, meaning that the approach easily scales to BSS with a size in the order of those found in Chinese cities. The approximations are actually better for BSS with more stations because this reduces the stochastic variability of system behaviour. Just as an indication of the evaluation times we would like to mention that the analysis in Fig. 6 takes less than a second to perform.



**Fig. 6.** NAB for station types P (pink dashes), M (blue line) and C (green dots), with  $\lambda = 1$  and  $\mu = 4$  per hour: (left) for configuration 1 (see Sect. 2), with P-stations each holding 3 bikes initially; M-stations 5 bikes; and C-stations 7 bikes, and (right) for configuration 2 (see Sect. 2), with the same initial distribution of bikes over stations (Color figure online).



**Fig. 7.** Time dependent probability to be full (left) or empty (right) within 30 min for the station types P (pink dashes), M (blue line) and C (green dots), with  $\lambda = 1$  and  $\mu = 4$  per hour, for configuration 1 (Color figure online).

## 5.2 Properties of Individual Stations in the Context of the System

In the previous section, we analysed performance aspects of the BSS as a whole. However, also the behaviour of individual stations in the context of the overall BSS can provide valuable insight into the level of service that is provided to its customers. We provide some examples of properties of individual stations that may be relevant and that have been obtained by applying the mean field model checker FlyFast, developed by some of the authors of the current paper [18, 20].

**Probability of a Station Getting Full or Empty Within the Time of Arrival.** An example of a property of an individual station, operating in the context of the global system, is the question of how likely it is that a station gets full (or empty, respectively) within approximately 30 min (corresponding to

15 time steps) corresponding to the usual maximal free allowance in a BSS, i.e. no fee is due for trips shorter than 30 min. In other words, this property could provide some insight in the likelihood that a station selected as destination by a user gets full between the time the user departs and the maximal free allowance. For a station in the periphery with a capacity of 5 parking slots, this property can be formalised in PCTL [14] as

$$\mathcal{P}_{=?}(tt \ U^{\leq 15} \ YP5) \quad (P1)$$

where  $YP5$  a periphery station that is full, i.e. having 5 bikes parked. A similar property can be formulated for a station getting empty replacing  $YP5$  by  $YP0$ . Figure 7(left) shows the results for a selected destination station in state  $YP3$ , so there are still  $5 - 3 = 2$  parking slots free at the destination at the time the user departs. Note that this probability depends on the time at which property P1 is evaluated because of the evolution of the system over time and that the figure shows its evaluation at times ranging from 0 to 1000 time units. The interested reader is referred to [20] for further details on the mean field model-checking algorithms for such properties. Similarly, Fig. 7(right) shows the probabilities for a destination station to get *empty* within 30 min when it currently has two bikes parked in its slots.

A more sophisticated nested property, P2, considers the situation in which, within 30 min a destination station in the periphery: (i) gets full, but then, with high probability ( $\geq 0.99$ ) has a free slot within 6 min, or (ii) does not get full, but then, with low probability ( $\leq 0.01$ ) gets full within 6 min.

This property can be formalised in PCTL as follows:

$$\mathcal{P}_{=?}(tt \ U^{\leq 15} \ ((YP5 \wedge \mathcal{P}_{\geq 0.99}(YP5 \ U^{\leq 3} \ \neg YP5)) \vee (\neg YP5 \wedge \mathcal{P}_{\leq 0.01}(\neg YP5 \ U^{\leq 3} \ YP5)))) \quad (P2)$$

The result of checking P2 against a BSS with an initial state of the selected individual destination station and which has two empty parking slots available (i.e. the local state is  $YP3$ ) is shown in Fig. 4(right). Property P2 is evaluated at times ranging from 0 to 1000 time units. We clearly see that there are periods during which the likelihood to find a free slot at the destination station in the periphery is 1, which thus indicates a very good level of service indeed. But there are also periods during which this likelihood reduces considerably. These are peak times when the level of service for what concerns finding free slots to park a bike in the periphery drops to a much lower level. Note that P2 formalises the probability with which a user can expect to find a free parking slot at a station in the periphery within 30 min from departure. Similar properties can be verified for stations in other parts of the city. Property P2 took just a few seconds to be analysed, which is a very good result given the size of the system and the nested form of the formula.

## 6 Discussion and Future Work

In this paper, we presented two approaches for the evaluation of BSS designs by means of automated tools in a rather orthogonal way. We performed

variability analysis (by multi-objective optimisation) on system configurations and performance analysis (by mean field model checking) on behavioural models. We focussed on a simplistic comparison of two different BSS configurations with respect to their cost, user satisfaction and capacity (in terms of parking slots) to illustrate the ideas. In the future, we intend to strengthen the integration of these two approaches and use the outcome of performance analyses as input for variability modeling. Think, e.g., of measuring the user satisfaction for specific configurations and feeding the resulting values into the variability model. Clearly, the probability of finding an empty (or full) docking station, based on the capacity of a BSS configuration, may directly impact user satisfaction. The resulting combined use of analysis approaches can trigger the development of automatic decision support for the design of BSS as well as for successive adaptations and reconfigurations. This is not merely an utopia, since we showed that the use of recently developed scalable performance analysis methods (i.e. on-the-fly mean field model checking in discrete time) proposed in this paper easily scale to BSS of realistic size.

**Acknowledgements.** We thank Michele Loreti, who is the developer of FlyFast. We also thank Bicincittà S.r.l. and Marco Bertini from PisaMo S.p.A. for generously sharing with us relevant information concerning bike-sharing systems in general and Pisa's *CicloPi* bike-sharing system in particular.

## References

1. Bał, K., Diskin, Z., Antkiewicz, M., Czarnecki, K., Wąsowski, A.: Clafer: unifying class and feature modeling. *Softw. Syst. Model.* 1–35 (2015). doi:[10.1007/s10270-014-0441-1](https://doi.org/10.1007/s10270-014-0441-1)
2. Batory, D.: Feature models, grammars, and propositional formulas. In: Obbink, H., Pohl, K. (eds.) *SPLC 2005*. LNCS, vol. 3714, pp. 7–20. Springer, Heidelberg (2005)
3. ter Beek, M.H., Fantechi, A., Gnesi, S.: Challenges in modelling and analyzing quantitative aspects of bike-sharing systems. In: Margaria, T., Steffen, B. (eds.) *ISoLA 2014, Part I*. LNCS, vol. 8802, pp. 351–367. Springer, Heidelberg (2014)
4. ter Beek, M.H., Fantechi, A., Gnesi, S.: Applying the product lines paradigm to the quantitative analysis of collective adaptive systems. In: *Proceedings 19th International Conference on Software Product Lines (SPLC 2015)*, pp. 321–326. ACM (2015)
5. ter Beek, M.H., Gnesi, S., Mazzanti, F.: Model checking value-passing modal specifications. In: Voronkov, A., Virbitskaite, I. (eds.) *PSI 2014*. LNCS, vol. 8974, pp. 304–319. Springer, Heidelberg (2015)
6. ter Beek, M.H., Legay, A., Lluch Lafuente, A., Vandin, A.: Statistical analysis of probabilistic models of software product lines with quantitative constraints. In: *Proceedings of 19th Software Product Line Conference (SPLC 2015)*, pp. 11–15. ACM (2015)
7. Benavides, D., Segura, S., Ruiz-Cortés, A.: Automated analysis of feature models 20 years later: a literature review. *Inf. Syst.* **35**(6), 615–636 (2010)
8. Bortolussi, L., Hillston, J.: Fluid model checking. In: Koutny, M., Ulidowski, I. (eds.) *CONCUR 2012*. LNCS, vol. 7454, pp. 333–347. Springer, Heidelberg (2012)

9. Bortolussi, L., Hillston, J.: Fluid Model Checking (2013). [arXiv:1203.0920v2](https://arxiv.org/abs/1203.0920v2)
10. Bortolussi, L., Hillston, J., Latella, D., Massink, M.: Continuous approximation of collective system behaviour: A tutorial. *Perf. Eval.* **70**, 317–349 (2013)
11. de Moura, L., Bjørner, N.S.: Z3: an efficient SMT solver. In: Ramakrishnan, C.R., Rehof, J. (eds.) TACAS 2008. LNCS, vol. 4963, pp. 337–340. Springer, Heidelberg (2008)
12. Fricker, C., Gast, N.: Incentives and redistribution in homogeneous bike-sharing systems with stations of finite capacity. *EURO J. Transp. Logistics*, 1–31 (2014). doi:[10.1007/s13676-014-0053-5](https://doi.org/10.1007/s13676-014-0053-5)
13. Froehlich, J., Neumann, J., Oliver, N.: Sensing and predicting the pulse of the city through shared bicycling. In: Boutilier, C. (ed.) Proceedings of 21st International Joint Conference on Artificial Intelligence (IJCAI 2009), pp. 1420–1426. Morgan Kaufmann, San Francisco (2009)
14. Hansson, H., Jonsson, B.: A logic for reasoning about time and reliability. *Formal Aspects Comput.* **6**, 512–535 (1994)
15. Hayden, R.: Scalable Performance Analysis of Massively Parallel Stochastic Systems. Ph.D. thesis, Imperial College London, April 2011
16. Hillston, J.: Fluid flow approximation of PEPA models. In: Proceedings of 2nd Conference on the Quantitative Evaluation of Systems (QEST 2005), pp. 33–43. IEEE (2005)
17. Kolesnichenko, A., de Boer, P.-T., Remke, A., Haverkort, B.R.: A logic for model-checking mean-field models. In: Proceedings of 43rd Conference on Dependable Systems and Networks (DSN 2013), pp. 1–12. IEEE (2013)
18. Latella, D., Loretì, M., Massink, M.: On-the-fly fast mean-field model-checking. In: Abadi, M., Lluch Lafuente, A. (eds.) TGC 2013. LNCS, vol. 8358, pp. 297–314. Springer, Heidelberg (2014)
19. Latella, D., Loretì, M., Massink, M.: On-the-fly fluid model checking via discrete time population models. In: Beltrán, M., Knottenbelt, W., Bradley, J. (eds.) EPEW 2015. LNCS, vol. 9272, pp. 193–207. Springer, Heidelberg (2016)
20. Latella, D., Loretì, M., Massink, M.: On-the-fly PCTL fast mean-field approximated model-checking for self-organising coordination. *Sci. Comput. Prog.* **110**, 23–50 (2015)
21. Le Boudec, J.-Y., McDonald, D., Munding, J.: A generic mean field convergence result for systems of interacting objects. In: Proceedings of 4th Conference on the Quantitative Evaluation of Systems (QEST 2007), pp. 3–18. IEEE (2007)
22. Midgley, P.: Bicycle-Sharing Schemes: Enhancing Sustainable Mobility in Urban Areas. Background Paper CSD19/2011/BP8, Commission on Sustainable Development, United Nations Department of Economic and Social Affairs, May 2011
23. Murashkin, A., Antkiewicz, M., Rayside, D., Czarnecki, K.: Visualization and exploration of optimal variants in product line engineering. In: Proceedings of 17th International Software Product Line Conference (SPLC 2013), pp. 111–115. ACM (2013)
24. Schobbens, P., Heymans, P., Trigaux, J.: Feature diagrams: a survey and a formal semantics. In: Proceedings of 14th Conference on Requirements Engineering (RE 2006), pp. 136–145. IEEE (2006)

# Automated Synthesis of Protocol Converters with BALM-II

Giovanni Castagnetti<sup>1</sup>, Matteo Piccolo<sup>1</sup>, Tiziano Villa<sup>1</sup> (✉),  
Nina Yevtushenko<sup>2</sup>, Robert Brayton<sup>3</sup>, and Alan Mishchenko<sup>3</sup>

<sup>1</sup> Dipartimento d'Informatica, Università di Verona, Verona, Italy  
tiziano.villa@univr.it

<sup>2</sup> Computer Science Laboratory, Tomsk State University, Tomsk, Russia

<sup>3</sup> Department of EECS, University of California, Berkeley, CA, USA

**Abstract.** We address the problem of the automatic design of automata to translate between different protocols, and we reduce it to the solution of equations defined over regular languages and finite automata (FA)/finite state machines (FSMs). The largest solution of the defined language equations includes all protocol converters that solve the problem; this is a strong advantage over computational techniques that deliver only one or a few solutions, which might lead to suboptimal implementations (e.g., as sequential circuits). Our model is versatile, because it can handle different topologies and constraints on the solutions. We propose a fully automatic procedure implemented inside a software package BALM-II which solves language equations. For illustration we show examples of setting up and solving language equations for classical protocol mismatch problems, aiming at the design of protocol converters to interface an alternating-bit (AB) sender and a non-sequenced (NS) receiver. Our automatic converter synthesis procedure yields a complete solution for automata and FSMs, and may serve as a core engine to embed into any full-fledged interface synthesis tool.

## 1 Introduction

In electronic system level design, an important step is the implementation of communication, e.g., consider a design scenario based on the composition of reusable Intellectual Property (IP) cores that realize different communication protocols (as in [1]). This problem may be addressed either by interface template instantiation (customization of a set of templates from functional and performance requirements), or by interface synthesis [2]. The latter aims to adapt incompatible transmitters and receivers of data, by designing a converter (if it exists) that makes them compatible within the given constraints of the implementation.

For instance, consider the example (from [2, 3]) of a producer and a consumer communicating complex data each partitioned into two parts. The producer (handshake protocol) can wait an unbounded amount of time between sending the two parts, whereas the consumer (serial protocol) must receive the second part just after the first one with no intermediate wait. The further specification is

that data - to be transmitted in the same order - are neither lost nor duplicated. The goal is to synthesize a converter that takes the first part of a datum sent by the producer and delivers it to the consumer, only after it receives also the second part of the datum.

In this paper we address automatic interface synthesis when the protocols and specification are expressed by automata and regular languages, and obey an FSM semantics to be specified according to the composition operators: given protocols  $P$  and  $Q$ , synthesize the most general converter  $X$  such that the composition of  $P$ ,  $Q$  and  $X$  behaves like a required specification  $C$ .

We model the problem of synthesizing a converter as follows: given a module  $A$  (the context, i.e., the product of the two protocols  $P$  and  $Q$ ) and a specification module  $C$ , synthesize the unknown converter  $X$  such that the composition of  $A$  and  $X$  conforms to or refines  $C$ . We allow any communication topology between the protocols and the converter.

The main contributions of this paper are:

1. We reduce the problem to solving inequalities and equations over regular languages/automata (FA) and finite state machines (FSMs), for which there are closed-form solutions, with respect to the synchronous and parallel composition operators (see [4–9]).
2. We describe a new package - BALM-II (see [10, 11]) - for solving inequalities and equations over FA/FSMs, which we developed to extend to equations over parallel composition the previous software package BALM (see [12]) that was restricted to equations over synchronous composition.

In Sect. 2 we survey briefly the previous work, whereas in Sect. 3 we outline the theory and practice of solving inequalities and equations over FA/FSMs with BALM-II, a software package that solves synchronous and parallel equations. In Sect. 4 we apply this synthesis technique to an example of protocol mismatch problem modeled with parallel equations. We conclude in Sect. 5 by summarizing a comparison with other approaches and mentioning future work.

## 2 Previous Work

The problem of protocol converter synthesis received recently a lot of attention in the literature (for instance, see the references [1, 2, 13–23]).

In [15–18], Passerone *et al.* considered as composition any binary operator that satisfies appropriate algebraic properties; the proposed definition includes parallel and synchronous composition of automata as special cases (see Sect. 3 for their formal definition). Moreover, they considered various preorder conformance relations of which containment is the simplest case (it is only required that the composition is monotonic). To synthesize a converter, they introduced a mirror function that is not unique for an agent, and instead satisfies appropriate properties depending on the composition operator and conformance relation.

This general setting reduces to our model when protocols are described by automata. For instance, for parallel inequalities/equations over automata with

respect to the containment relation, the complement operator appearing in the closed-form  $X = \overline{A \diamond \overline{C}}$  is exactly the mirror function, and so both approaches capture all solutions.

Further requirements on the synthesized converter are mentioned in [17]: one is the extraction of a deterministic solution for the serial topology (for which it is known that selecting a complete submachine is sufficient), and the other is about enforcing progressiveness. About the latter, defined as the property that each action defined by the specification should be provided by the composition, it is suggested to enforce it by deleting transitions from the largest solution; however, there is no discussion that in general (in the rectification topology, for example) this trimming operation is sufficient only when the largest solution is represented by a perfect automaton (see [24,25] for details).

Jiang and Jin studied in [14] a variant where the composition of the given protocols and of the converter behaves like a FIFO with a buffer of size  $k$ .

In [20], Bhaduri and Ramesh solved the problem of converter synthesis for interface automata with respect to an alternating simulation relation, i.e., a relation  $\rho$  from  $P$  to  $Q$  for which  $(s, t) \in \rho$  implies that all input moves from  $t$  can be simulated by  $s$  and all output moves from  $s$  can be simulated by  $t$ . A refinement between interface automata is defined as the existence of an alternating simulation between the initial states, i.e., an interface automaton  $P$  refines an interface automaton  $Q$ ,  $P \preceq Q$ , if (1) the set of input actions of  $Q$  is a subset of the input actions of  $P$ , (2) the set of output actions of  $P$  is a subset of the output actions of  $Q$ , and (3) there is an alternating simulation  $\rho$  from  $P$  to  $Q$  such that  $(s_0, t_0) \in \rho$ , where  $s_0$  and  $t_0$  are initial states, respectively, of  $P$  and  $Q$ . The most abstract solution under alternating simulation of  $P \parallel R \preceq Q$  is given by  $R = (P \parallel Q^\perp)^\perp$ , where  $P^\perp$  is the same as  $P$ , except that the input actions in  $P$  become the output actions in  $P^\perp$  and similarly the output actions in  $P$  are the input actions in  $P^\perp$ . This  $P^\perp$  operator reminds of the inverse FSM introduced in the solution of the model matching problem discussed in [26].

In [21] Avnit and Sowmya discussed a variant of the converter synthesis problem, in which, instead of requiring that the composition satisfies the specification, the objective is to insure what is called a correct conversion: a correct converter is compatible with both protocols, and never causes an overflow or an underflow of its buffers. The problem is solved by deleting redundant transitions; the authors do not discuss the fact that when dealing with sequential systems one should rather delete redundant sequences than redundant transitions, otherwise, some solutions may be lost.

Given two automata representing two protocols, in [19] Watanabe *et al.* introduced a synthesizer that judges whether each state of the product automaton is legal or not in terms of data dependency. The output transducer is an FSM which is the subset of the product automaton that consists of legal states. A transducer is required to satisfy the property that illegal tuples are removed according to the following rules: (a) a data word which has not been received must not be sent; (b) if a state from the master automaton is a final state, all the states from the slave automaton must be final states; (c) tuples which inevitably



transit to an illegal tuple are also illegal. The first rule does not care about which automaton has received or sent a data word. The resulting solution is a sub-machine of the product automaton, i.e., all machines have the same set of actions. There is no explicit mention of the overall specification, similar to a case-study in [17], where the transducer is a given. Notice that each component protocol is represented as a composition of smaller automata, so that there is no need of a monolithic component automaton when deriving a transducer. In summary, they construct the whole transducer from a set of partial transducers, and the synthesized transducer consists of several FSMs to handle parallel transactions.

Cao and Nymeyer presented in [27] a theoretical model of a converter including buffers and allowing the specification by the designer of CTL conditions, to be verified by means of a model checker.

Sinha et al. developed in [1, 23, 28–30] a compositional approach for the integration of multiple components with a wide range of protocol mismatches into a single System-on-Chip (SoC). They construct the SoC either in a single step or incrementally, and discuss the pros and cons of the two approaches; the latter has advantages with respect to scalability, reuse, wiring congestion and latency errors. They can handle mismatches such as multiple clocks, multi-directional IP communication, control and data-width mismatches, but not yet complex mismatches such as interface inconsistencies. They also ensure the satisfaction of multiple constraints (specifications) on the behaviour of the SoC, which are specified as properties in temporal logic CTL; these properties are classified as control constraints over the state labels in the protocols, data constraints over the data counters used to track down the data communication between IPs, and control-data constraints using state labels and counter values in the same CTL formula. Protocols are described as Synchronous Kripke Structures (SKS) which are Finite State Machines (FSMs), augmented with a set of propositions (denoting control labels and data labels) that label each state to describe its control and data input/output status. In the proposed methodology, a clock automaton is introduced to handle multiple clocks, i.e., each protocol is oversampled to describe its behaviour with respect to the fastest clock in the SoC. Then the parallel composition of all the SKSs is computed, and finally inputs and outputs are partitioned as follows: uncontrollable signals passed to/from the protocols as soon as available, shared signals emitted and read by protocols, missing control signals to be generated by the converter. The synthesis of converters is based on a recursive algorithm originally presented in [31], which is extended to construct a graph corresponding to *all* the valid behaviours of the protocols such that the given constraints are satisfied (the original algorithm did not enumerate all possible ways in which an assertion may be satisfied). Then a deterministic converter is obtained by extracting a sub-graph from the previous graph representing the maximal non-deterministic converter. The converter behaves acts as follows: it forwards all uncontrollable signals to the environment/protocols in the same clock tick; it buffers shared signals from the protocols and forwards them after one or more clock ticks; it maintains a 1-place buffer for each shared signal; it may hide a buffered signal from the protocols in a tick to disable an

undesirable transition, and it may generate some control signals emitted neither by the environment nor by the protocols. The converter executes based on the fastest clock of the SoC, and at each step it follows a precise sequence of micro-steps with the environment and the protocols: it reads the uncontrollable signals from the environment as an input formula, it forwards the uncontrollable signals and converter-controlled signals (buffered or generated) to the protocols, it reads the signals generated by the IPs emitting any uncontrollable outputs to the environment and buffering relevant control signals. All together these actions form a macro-step; the result is a converter SKS (CSKS), whose parallel composition with the protocols defines the matched system. Results are reported with SKS abstractions of IP protocols extracted from Advanced Microcontroller Bus Architecture (AMBA) white papers and Hardware Description Language (HDL) implementations.

In [32] Autili et al. survey the automatic synthesis of connectors at the level of networked systems to achieve protocol interoperability, i.e., the ability to communicate and coordinate correctly. The challenge is for heterogeneous protocols in an ubiquitous computing environment to cooperate to reach some common goals, even though meeting dynamically and without a priori knowledge of each other. Connectors may be either coordinators (the networked systems are already able to communicate, but they need to be coordinated) or mediators (the requirement is both to enable communication and achieve coordination). Even though automated and run-time interoperability is still an open challenge for networked systems, the paper reports the state-of-art on formal methods for the automatic synthesis of coordinators and mediators, concluding with necessary and sufficient interoperability conditions to guarantee the existence of correct mediators. Automatic synthesis of a secure orchestrator for a set of BPMN (Business Process Model and Notation) processes is presented in [33], based on synthesis by partial model checking.

In the following sections, we will present a theoretical approach and a software tool to synthesize all protocol converters for any topology of parallel and synchronous composition, when the components to interface are modeled by finite automata and finite state machines; we will compare in Sect. 5 with the most relevant previous work.

### 3 Equations over Languages and Automata

Consider a composition topology, shown in Fig. 1, where the composition of two interconnected components (the context or plant  $A$ , and the unknown component  $X$ ) defines a behaviour contained or equal to the one defined by the specification  $C$ ; the context and the unknown interact through internal signals, and communicate with the environment with external input and output signals. According to what connections are present, one may define simplified topologies, e.g., the rectification topology when the unknown component has no external input and output signals. Our goal is to find the most general unknown component that contains all the behaviours such that the composition is included in the specification. In the next subsections, we will introduce two types of composition:

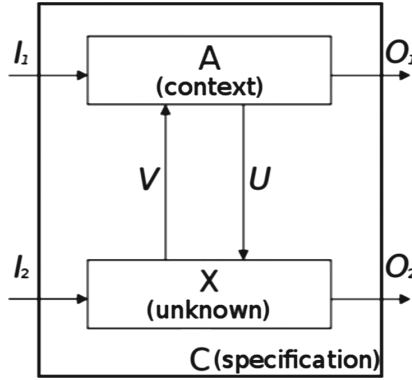


Fig. 1. General topology.

**parallel composition** (a.k.a. as **asynchronous composition**) and **synchronous composition**.

### 3.1 Equations with Parallel Composition

In order to introduce parallel composition we must define two new operators: expansion and restriction. The first one, denoted by the symbol  $\uparrow$ , is used to extend the alphabet of the automaton, while the second one, denoted by the symbol  $\downarrow$ , is used to restrict the alphabet of the automaton.

**Definition 1.** Given a language  $L$  over alphabet  $X$  and an alphabet  $V$ , consider the mapping  $e : X \rightarrow 2^{(X \cup V)^*}$  defined as  $e(x) = \{\alpha x \beta \mid \alpha, \beta \in (V \setminus X)^*\}$ , then the language  $L_{\uparrow V} = \{e(\alpha) \mid \alpha \in L\}$  over alphabet  $X \cup V$  is the **expansion** of language  $L$  to alphabet  $V$ , or  $V$ -expansion of  $L$ , i.e., words in  $L_{\uparrow V}$  are obtained from those in  $L$  by inserting anywhere in them words from  $(V \setminus X)^*$ . Notice that  $e(\epsilon) = \{\alpha \mid \alpha \in (V \setminus X)^*\}$ .

**Definition 2.** Given a language  $L$  over alphabet  $X \cup V$ , consider the homomorphism  $r : X \cup V \rightarrow V^*$  defined as  $r(y) = y$  if  $y \in V, r(y) = \epsilon$  if  $y \in X \setminus V$ , then the language  $L_{\downarrow V} = \{r(\alpha) \mid \alpha \in L\}$  over alphabet  $V$  is the **restriction** of language  $L$  to alphabet  $V$ , or  $V$ -restriction of  $L$ , i.e., words in  $L_{\downarrow V}$  are obtained from those in  $L$  by deleting all the symbols in  $X$  that are not in  $V$ . Notice that  $r(\epsilon) = \epsilon$ .

**Definition 3.** Given the pairwise disjoint alphabets  $I, U, O$ , language  $L_1$  over  $I \cup U$  and language  $L_2$  over  $U \cup O$ , the **parallel composition** of languages  $L_1$  and  $L_2$  is the language  $[(L_1)_{\uparrow O} \cap (L_2)_{\uparrow I}]_{\downarrow I \cup O}$ , denoted by  $L_1 \diamond_{I \cup O} L_2$ , defined over  $I \cup O$ .

With these operators it is possible to define an equation over languages and write a closed form for its solution.

**Definition 4.** Given the pairwise disjoint alphabets  $I, U, O$ , a language  $A$  over alphabet  $I \cup U$  and a language  $C$  over alphabet  $I \cup O$ , language  $B$  over alphabet  $U \cup O$  is called a **solution** of the equation  $A \diamond X \subseteq C$  iff  $A \diamond B \subseteq C$ .

**Definition 5.** The **largest solution** is a solution that contains any other solution.

**Theorem 1.** (proof in [9], Theorem 2.16, p. 24) The largest solution of the equation  $A \diamond X \subseteq C$  is the language  $S = \overline{A \diamond C}$ .

Language equations can be solved effectively when their languages have finitary representations, usually by means of automata generating them, e.g., finite automata for regular languages. In that case, we can interpret a given language equation as denoting an equation over the automata generating the given languages. So, we talk interchangeably of language equation or automaton equation (where the automata generate the languages).

**Definition 6.** A **finite automaton (FA)** is a 5-tuple  $F = \langle S, \Sigma, \Delta, r, Q \rangle$ .  $S$  represents the finite state space,  $\Sigma$  represents the finite alphabet of actions, and  $\Delta \subseteq \Sigma \times S \times S$  is the next state relation, such that  $(\sigma, p, n) \in \Delta$  iff  $n \in S$  is a next state of present state  $p \in S$  on action  $i \in \Sigma$ . The initial or reset state is  $r \in S$  and  $Q \subseteq S$  is the set of final or accepting states. The automaton  $F$  is deterministic, if for each state  $s \in S$  and any action  $\sigma \in \Sigma$  there exists at most one state  $s'$ , such that  $(\sigma, s, s') \in \Delta$ , otherwise it is nondeterministic. A variant of FA allows the introduction of  $\epsilon$ -moves, meaning that  $\Delta \subseteq (\Sigma \cup \{\epsilon\}) \times S \times S$ ; by the closure procedure one obtains an equivalent deterministic FA without  $\epsilon$ -moves [34].

Finally, we introduce equations over FSMs, by interpreting them as equations over languages produced by FSMs; these languages are regular and so are generated by finite automata.

**Definition 7.** A **finite state machine (FSM)** is a 5-tuple  $M = \langle S, I, O, T, r \rangle$  where  $S$  represents the finite state space,  $I$  represents the finite input space,  $O$  represents the finite output space and  $T \subseteq I \times S \times S \times O$  is the transition relation. On input  $i$ , the FSM at present state  $p$  may transit to next state  $n$  and produce output  $o$  iff  $(i, p, n, o) \in T$ . State  $r \in S$  represents the initial or reset state.

For ease of discussion, we assume that FSMs are complete, i.e., at least one transition is specified for each present state and input pair.

There are different ways of associating a language to an FSM, by considering the automaton underlying the FSM, according to the semantics of choice. Here we introduce an interleaving semantics, naturally associated with parallel composition, where the language of an FSM is defined over the alphabet  $I \cup O$ . as follows.

**Definition 8.** Given an FSM  $M = \langle S, I, O, T, r \rangle$ , consider the finite automaton  $F(M) = \langle S \cup (S \times I), I \cup O, \Delta, r, S \rangle$ , where  $(i, s, (s, i)) \in \Delta \wedge (o, (s, i), s') \in \Delta$  iff  $(i, s, s', o) \in T$ . The language accepted by  $F(M)$  is denoted  $L_r^\cup(M)$ , and by

definition is the  $\cup$ -language of  $M$  at state  $r$ . Similarly  $L_s^\cup(M)$  denotes the language accepted by  $F(M)$  when started at state  $s$ , and by definition is the  $\cup$ -language of  $M$  at state  $s$ . By construction,  $L_s^\cup(M) \subseteq (IO)^*$ , where  $IO$  denotes the set  $\{io \mid i \in I, o \in O\}$ .

In short, the automaton is obtained from the original FSM, by replacing each edge  $(i, s, s', o)$  by the pair of edges  $(i, s, (s, i))$  and  $(o, (s, i), s')$  where  $(s, i)$  is a new node (non-accepting state). All original states are made accepting. This automaton generates the language that we associate to the FSM, and it can be interpreted as an Input/Output Automaton [35].

Similarly, we can define the composition of FSMs and equations over FSMs.

**Definition 9.** The parallel composition of FSMs  $M_A$  and  $M_B$  yields a reduced observable FSM  $M_A \diamond M_B$  with language

$$L(M_A \diamond M_B) = L(M_A) \diamond L(M_B) \cap (IO)^*.$$

The previous definition relies on the fact that  $L(M_A \diamond M_B) = L(M_A) \diamond L(M_B) \cap (IO)^*$  is a unique FSM language to which corresponds any FSM whose associated automaton accepts such language (there are many such behaviorally equivalent FSMs). The resulting FSM may be made unique by standard operations like subset construction and state minimization, which produce a reduced observable FSM. By definition, an FSM is observable if for each triple  $(i, p.o) \in I \times S \times O$  there is at most one next state  $n$  such that  $(i, p, n, o) \in T$ , i.e., for each input, present state and output there is a unique next state, equivalent to the fact that the underlying finite automaton is deterministic; an FSM is reduced if there are no two equivalent states, i.e., states that cannot be distinguished by their external behaviour.

**Definition 10.** The parallel FSM equation

$$M_A \diamond M_X \preceq M_C,$$

corresponds to the related language equation

$$L(M_A) \diamond L(M_X) \subseteq L(M_C) \cup \overline{(IO)^*},$$

where  $L(M_A)$  and  $L(M_C)$  are the FSM languages associated with FSMs  $M_A$  and  $M_C$ .

In the following, we summarize the steps to solve parallel FSM equations. For ease of notation, given an FSM  $M_X$  we denote by  $X$  its associated automaton and generated language.

Given the parallel FSM equation

$$M_A \diamond_{I_1 \cup I_2 \cup O_1 \cup O_2} M_{X_{I_2 \cup U \cup V \cup O_2}} \subseteq M_C, \tag{1}$$

one derives the corresponding automata  $A$  and  $C$  and solves the automaton equation

$$A \diamond_{I_1 \cup I_2 \cup O_1 \cup O_2} X_{I_2 \cup U \cup V \cup O_2} \subseteq C, \tag{2}$$

equivalent to

$$((A_{I_1 \cup V \cup U \cup O_1})_{\uparrow I_2 \cup O_2} \cap (X_{I_2 \cup U \cup V \cup O_2})_{\uparrow I_1 \cup O_1})_{\downarrow I_1 \cup I_2 \cup O_1 \cup O_2} \subseteq C_{I_1 \cup I_2 \cup O_1 \cup O_2}.$$

The largest automaton solution is given by

$$X = \overline{A \diamond_{I_2 \cup U \cup V \cup O_2} \overline{C}} \quad (3)$$

equivalent to

$$X_{I_2 \cup U \cup V \cup O_2} = \overline{((A_{I_1 \cup V \cup U \cup O_1})_{\uparrow I_2 \cup O_2} \cap (\overline{C}_{I_1 \cup I_2 \cup O_1 \cup O_2})_{\uparrow U \cup V})_{\downarrow I_2 \cup U \cup V \cup O_2}}. \quad (4)$$

The Eq. (3) is intuitively derived as follows: we take the unwanted behaviours (the complement of the specification) and we compose them with the context in order to obtain only the behaviours that we want to delete from our system. Then we complement this result and so we obtain all the acceptable behaviours; by construction this is the largest automaton solution of the Eq. (2). In a similar way it is possible to obtain the largest FSM solution of the Eq. (1), as follows.

The largest FSM solution requires enforcing the hypothesis that an input must be followed by an output before another input can be produced. This particular constraint is necessary because we do not assume any extension of the FSM model to store symbols (i.e., FSMs with buffers).

Setting  $I = I_1 \cup I_2$  and  $O = O_1 \cup O_2$ , the largest FSM solution is given by

$$X = \overline{A \diamond_{I_2 \cup U \cup V \cup O_2} (\overline{C} \cap (IO)^*)} \cap (UV)^* \quad (5)$$

that is equivalent to

$$X_{I_2 \cup U \cup V \cup O_2} = \overline{((A_{I_1 \cup V \cup U \cup O_1})_{\uparrow I_2 \cup O_2} \cap (\overline{C}_{I_1 \cup I_2 \cup O_1 \cup O_2} \cap (IO)^*)_{\uparrow U \cup V})_{\downarrow I_2 \cup U \cup V \cup O_2} \cap ((UV)^*)_{\uparrow I_2 \cup O_2}}. \quad (6)$$

Once the unknown  $X$  has been computed, as a sanity check one may verify whether the computed  $X$  satisfies the inequality (2)

$$A \diamond_{I_1 \cup I_2 \cup O_1 \cup O_2} X_{I_2 \cup U \cup V \cup O_2} \subseteq C,$$

equivalent to

$$((A_{I_1 \cup U \cup V \cup O_1})_{\uparrow I_2 \cup O_2} \cap (X_{I_2 \cup U \cup V \cup O_2})_{\uparrow I_1 \cup O_1})_{\downarrow I_1 \cup I_2 \cup O_1 \cup O_2} \subseteq C_{I_1 \cup I_2 \cup O_1 \cup O_2}. \quad (7)$$

If the composition is equivalent to the specification then the equation is solvable, otherwise a solution of the inequality yields a strict refinement of the specification. Similar formulas can be derived for the simplified topology or other topologies, introducing the appropriate supports of variables.

### 3.2 Equations with Synchronous Composition

We can repeat the previous steps to define equations with respect to synchronous composition and obtain dual results in terms of characterization of the solutions. To save space, we will not go through again the all derivation, but only point out the main differences, and refer to [9] for the details.

To define synchronous composition, instead of the operators of expansion and restriction, we need lifting and projection.

**Definition 11.** *Given a language  $L$  over alphabet  $X \times V$ , consider the homomorphism  $p : X \times V \rightarrow V^*$  defined as  $p((x,v)) = v$ , then the language  $L_{\downarrow V} = \{p(\alpha) \mid \alpha \in L\}$  over alphabet  $V$  is the **projection** of language  $L$  to alphabet  $V$ , or  $V$ -projection of  $L$ . By definition of substitution  $p(\epsilon) = \epsilon$ .*

**Definition 12.** *Given a language  $L$  over alphabet  $X$  and an alphabet  $V$ , consider the substitution  $l : X \rightarrow 2^{(X \times V)^*}$  defined as  $l(x) = \{(x,v) \mid v \in V\}$ , then the language  $L_{\uparrow V} = \{l(\alpha) \mid \alpha \in L\}$  over alphabet  $X \times V$  is the **lifting** of language  $L$  to alphabet  $V$ , or  $V$ -lifting of  $L$ . By definition of substitution  $l(\epsilon) = \{\epsilon\}$ .*

The given substitution operators change a language and its alphabet of definition; in particular the operators  $\uparrow$  and  $\downarrow$  vary the components that are present in the Cartesian product defining the language alphabet.

**Definition 13.** *Given the alphabets  $I, U, O$ , language  $L_1$  over  $I \times U$  and language  $L_2$  over  $U \times O$ , the **synchronous composition** of languages  $L_1$  and  $L_2$  is the language  $[(L_1)_{\uparrow O} \cap (L_2)_{\uparrow I}]_{\downarrow I \times O}$ , denoted by  $L_1 \bullet_{I \times O} L_2$ , defined over  $I \times O$ .*

Finally, the language associated to an FSM under the semantics of synchronous composition is the one generated by the automaton coinciding with the original FSM where all states are made accepting and the edges carry a label of the type  $(i, o)$ .

**Definition 14.** *Given an FSM  $M = \langle S, I, O, T, r \rangle$ , consider the finite automaton  $F(M) = \langle S, I \times O, \Delta, r, S \rangle$ , where  $((i, o), s, s') \in \Delta$  iff  $(i, s, s', o) \in T$ . The language accepted by  $F(M)$  is denoted  $L_r^\times(M)$ , and by definition is the  $\times$ -**language** of  $M$  at state  $r$ . Similarly  $L_s^\times(M)$  denotes the language accepted by  $F(M)$  when started at state  $s$ , and by definition is the  $\times$ -language of  $M$  at state  $s$ .*

The rest follows by mimicking the steps described in Sect. 3.1.

### 3.3 BALM-II

All these operations are available in the new software package BALM-II [11], which is an extension of an existent software called BALM [12].

BALM, a branch of the MVSIS [36] project, implements many operations to solve synchronous inequalities and equations over automata and FSMs, like lifting and projection, which can be used in the synthesis/resynthesis of sequential

circuits. However, the original version of BALM did not handle inequalities and equations with respect to parallel composition; therefore BALM was upgraded to BALM-II, a version extended with new procedures and commands to solve parallel equations (see the User’s manual in [10]).

BALM-II inherits all the representation formats and commands from BALM, and extends it with features to represent and manipulate automata encoding FSMs interpreted with the semantics of parallel composition, (i.e., expansion, restriction, etc.). Notice that, under the semantics of synchronous composition, FSMs are translated into automata simply by merging the input and output variables of the FSM into the “inputs” of the automaton, since the automata are defined over the cartesian product  $A_1 \times A_2 \times \dots \times A_n$ , where  $A_1, \dots, A_n$  are the FSM’s alphabets; instead, under the semantics of parallel composition, the translation is less straightforward, and it requires splitting the transitions of the FSM and introducing new states, because the resulting automata are defined over the union of alphabets  $A_1 \cup A_2 \cup \dots \cup A_n$  (see [9, 10] for a description of the transformation procedure). BALM-II makes available the commands to translate FSMs into the corresponding automata, then to operate on the automata to solve the equations, and finally to extract from the automata the resulting FSMs (of course, if the original problem is specified directly by means of automata, there is no need of this round trip from and to FSMs). This can be done for both the synchronous and parallel semantics. We refer to [9] for a theoretical exposition, and to [10] for implementation details and examples of usage.

In the next section we will apply this computational framework to the synthesis of the largest protocol converter on case-studies from the literature. For simplicity, we will describe examples modeled directly by automata, and refer the reader to [10] for examples with FSMs showing also the conversion procedures from FSMs to automata and viceversa.

## 4 An Example of Asynchronous Protocol

We define and solve an equation over finite automata to solve a problem of converter synthesis, i.e., the design of an automaton translating between two different protocols. A communication system has a sending part and a receiving part that exchange data through a specific protocol. A mismatch occurs when two systems with different protocols try to communicate. The mismatch problem is solved by designing a converter that translates between the receiver and the sender, while respecting the overall service specification of the behavior of the composed communication system with respect to the environment. It is very unlikely that both parts send and receive a message at the same time instant (unless there is a special reason for forcing it), so the behaviour of the overall system is described by the parallel composition of the receiving and sending modules, together with the converter to be synthesized. Therefore, we formulate the problem as a parallel language equation: given the service specification  $C$  of a communication system, a component sender and a component receiver, the goal is to find a converter  $X$  whose composition with the sender and receiver  $A$  meets the system specification after hiding the internal signals:  $A \diamond X \subseteq C$ .



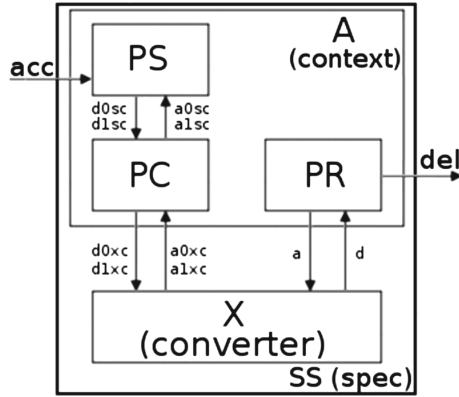


Fig. 2. Communication system described in Sect. 4.

As an example we consider the problem of designing a protocol converter to interface: an *alternating-bit* (AB) sender and a *non-sequenced* (NS) receiver. This problem was proposed originally in [22,37], and this exposition is adapted from [9]. A communication system based on an alternating bit protocol is composed of two processes, a sender and a receiver, which communicate over a half duplex channel that can transfer data in either directions, but not simultaneously. Each process uses a control bit called the alternating bit, whose value is updated by each message sent over the channel in either direction. The acknowledgment is also based on the alternating bit: each message received by either process in the system corresponds to an acknowledgment message that depends on the bit value. If the acknowledgment received by a process does not correspond to the message sent originally, the message is resent until the correct acknowledgment is received. On the other hand, a communication system is non-sequenced when no distinction is made among the consecutive messages received, or their corresponding acknowledgments. This means that neither messages nor their acknowledgments are distinguished by any flags as it happens with the alternating bit.

Figure 2 shows the block diagram of the composed system. Each component is represented by a rectangle with incoming and outgoing labeled arrows to indicate the inputs and outputs, respectively. The sender consists of an AB protocol sender (*PS*) and of an AB protocol channel (*PC*). Meanwhile, the receiving part includes an NS protocol receiver (*PR*). The converter *X* must interface the two mismatched protocols and guarantee that its composition with *PS*, *PC* and *PR* refines the service specification (*SS*) of the composed system. The events *Acc* (*Accept*) and *Del* (*Deliver*) represent the interfaces of the communication system with the environment (the users). The converter *X* translates the messages delivered by the sender *PS* (using the alternating bit protocol) into a format that the receiver *PR* understands (using the non-sequenced protocol). For example, acknowledgment messages *a* delivered to the converter by the receiver

are transformed into acknowledgments of the alternating bit protocol ( $a0xc$  to acknowledge a 0 bit and  $a1xc$  to acknowledge a 1 bit) and passed to the sender by the channel ( $a0cs$  to acknowledge a 0 bit and  $a1cs$  to acknowledge a 1 bit); data messages are passed from the sender to the channel ( $d0sc$  for a message controlled by a 0 bit and  $d1sc$  for a message controlled by a 1 bit) and then from the channel to the converter ( $d0cx$  for a message controlled by a 0 bit and  $d1cx$  for a message controlled by a 1 bit) to be transformed by the converter into a data message  $d$  for the receiver.

We model the components as finite automata which recognize prefix-closed regular languages, and solve the language equation  $PS \diamond PC \diamond PR \diamond X \subseteq SS$ , where  $PS \diamond PC \diamond PR$  is the context,  $X$  is the unknown protocol converter and  $SS$  is the specification.

The largest automaton solution of the previous equation is obtained by the computation  $S = \overline{PS \diamond PC \diamond PR \diamond SS}$ , that is the instantiation of the equation (3). The automaton solution can be obtained by running a few simple commands on BALM-II (like `complement`, `product` (i.e., intersection), `expansion` and `restriction`).

The automata of the components of the communication system and of the largest prefix-closed solution of the converter problem can be found in [10], with a comparison with respect to the other solutions reported in the literature.

The protocol conversion problem was addressed in [22], as an instance of supervisory control of discrete event systems, where the converter language is restricted to be a sublanguage of the context  $A$ , and in [37] with the formalism of input-output automata. In [22] the problem is modeled by the equation  $A \diamond X = C$  over regular languages with the rectification topology. The solution is given as a sublanguage of  $A$  of the form  $A \diamond C \setminus A \diamond \overline{C}$  (not the largest solution). An algorithm to obtain the largest compositionally progressive solution is provided that first splits the states of the automaton of the unrestricted solution (refining procedure, exponential step due to the restriction operator), and then deletes the states that violate the desired requirement of progressive composition (linear step). This algorithm does not generalize, as it is, to topologies where the unknown component depends also on signals that do not appear in the component  $A$ .

## 5 Conclusions

We transformed the protocol converter synthesis problem into solving inequations and equations over regular languages represented by finite automata and finite state machines. The solutions are computed with BALM-II, a software package that finds the largest solution of parallel and synchronous inequations and equations over automata and FSMs. For illustration we showed how to solve a classical problem to design a protocol converter for interfacing an alternating-bit sender and a non-sequenced receiver; we reported an example with parallel composition and one with synchronous composition. The ability to derive all protocol converters that solve the problem is an advantage over computational techniques that deliver only one or a few solutions (which might yield inferior

implementations, e.g., as sequential circuits). For simplicity in this paper we dealt only with inequations, but we could filter out the solutions that do not satisfy the strict equality, thus solving strict equations and avoiding trivial empty solutions. Notice that our approach works as it is also for non-deterministic specifications, and in general delivers a non-deterministic result representing a collection of deterministic solutions. Our current algorithm deals only with FA and FSMs, where all data are represented explicitly. In order to deal with real protocols having data words (with 8, 16, 32, 64 bits) there must be some abstraction mechanism, which is out of scope for this paper, which is focussed on proposing an automatic converter synthesis procedure for the control part. We will investigate as future work how to handle data paths on top of our proposed approach for control logic, starting from the techniques proposed in the literature. Another related practical issue is scalability. The worst-case of the proposed algorithm is exponential in the input state space, however this worst-case in practice is achieved only in pathological examples. To assess the practicality of the proposed methods and compare them, the availability of an established collection of shared benchmarks would help the research community.

## References

1. Sinha, R., Roop, P.S., Salcic, Z., Basu, S.: Correct-by-construction multi-component SoC design. In: Proceedings of the Conference on Design, Automation and Test in Europe, DATE 2012, San Jose, CA, USA. EDA Consortium, pp. 647–652 (2012)
2. Martin, G., Bailey, B., Piziali, A.: ESL Design and Verification: A Prescription for Electronic System Level Methodology. Morgan Kaufmann, San Mateo (2007)
3. Passerone, R., Rowson, J.A., Sangiovanni-Vincentelli, A.: Automatic synthesis of interfaces between incompatible protocols. In: Proceedings of the 35th Annual Design Automation Conference, DAC 1998. ACM, New York (1998). <http://doi.acm.org/10.1145/277044.277047>
4. Petrenko, A., Yevtushenko, N.: Solving asynchronous equations. In: Budkowski, S., Cavalli, A., Najm, E. (eds.) FORTE XI/PSTV XVIII 1998, pp. 231–247. Kluwer Academic Publishers, Dordrecht (1998)
5. Yevtushenko, N., Villa, T., Brayton, R., Petrenko, A., Sangiovanni-Vincentelli, A.: Solution of parallel language equations for logic synthesis. In: The Proceedings of the International Conference on Computer-Aided Design, pp. 103–110, November 2001
6. Yevtushenko, N., Villa, T., Brayton, R., Petrenko, A., Sangiovanni-Vincentelli, A.: Solution of synchronous language equations for logic synthesis. In: The Biannual 4th Russian Conference with Foreign Participation on Computer-Aided Technologies in Applied Mathematics, September 2002
7. Yevtushenko, N., Villa, T., Brayton, R., Petrenko, A., Sangiovanni-Vincentelli, A.: Sequential synthesis by language equation solving, Tech. Report No. UCB/ERL M03/9, Berkeley, CA, April 2003

8. Yevtushenko, N., Villa, T., Zharikova, S.: Solving language equations over synchronous and parallel composition operators. In: Kunc, M., Okhotin, A. (eds.) Proceedings of the 1st International Workshop on Theory and Applications of Language Equations, TALE 2007, Turku, Finland, 2 July 2007, pp. 14–32. Turku Centre for Computer Science (2007)
9. Villa, T., Yevtushenko, N., Brayton, R., Mishchenko, A., Petrenko, A., Sangiovanni-Vincentelli, A.: The Unknown Component Problem: Theory and Applications. Springer, New York (2012)
10. Castagnetti, G., Piccolo, M., Villa, T., Yevtushenko, N., Mishchenko, A., Brayton, R.K.: Solving parallel equations with BALM-II, EECS Department, University of California, Berkeley, Tech. Report UCB/EECS-2012-181, July 2012. <http://www.eecs.berkeley.edu/Pubs/TechRpts/2012/EECS-2012-181.html>
11. Castagnetti, G., Piccolo, M., Villa, T.: BALM-II. <http://esd.scienze.univr.it/index.php/it/balm-ii.html>
12. B. R. Group.: BALM, website and User's Manual. <http://embedded.eecs.berkeley.edu/Respep/Research/mvsi/balm.html>
13. Androutsopoulos, V., Brookes, D., Clarke, T.: Protocol converter synthesis. Comput. Digital Tech. IEE Proc. **151**(6), 391–401 (2004)
14. Jiang, Y., Jin, Y.: Protocol converter synthesis: an application of control synthesis, EE219C Class Project Report, December 1999
15. Passerone, R., Rowson, J.A., Sangiovanni-Vincentelli, A.L.: Automatic synthesis of interfaces between incompatible protocols. In: DAC, pp. 8–13 (1998)
16. Passerone, R., de Alfaro, L., Henzinger, T.A., Sangiovanni-Vincentelli, A.L.: Convertibility verification and converter synthesis: two faces of the same coin. In: ICCAD, pp. 132–139 (2002)
17. Passerone, R.: Semantic foundations for heterogeneous systems, Ph.D. dissertation, EECS Department, University of California, Berkeley, Tech. Report No. UCB/ERL M98/30 (2004). <http://www.eecs.berkeley.edu/Pubs/TechRpts/1998/3445.html>
18. Passerone, R.: Interface specification and converter synthesis. In: Zurawski, R. (ed.) Embedded Systems Handbook. CRC Press, Taylor and Francis Group, Boca Raton (2005)
19. Watanabe, S., Seto, K., Ishikawa, Y., Komatsu, S., Fujita, M.: Protocol transducer synthesis using divide and conquer approach. In: Design Automation Conference, 2007, ASP-DAC 2007, Asia, South Pacific, pp. 280–285, January 2007
20. Bhaduri, P., Ramesh, S.: Interface synthesis and protocol conversion. Formal Aspects Comput. **20**, 205–224 (2008)
21. Avnit, K., Sowmya, A.: A formal approach to design space exploration of protocol converters. In: The Proceedings of the Design, Automation and Test in Europe Conference, pp. 129–134, April 2009
22. Kumar, R., Nelvagal, S., Marcus, S.: A discrete event systems approach for protocol conversion. Discrete Event Dyn. Syst. Theory Appl. **7**(3), 295–315 (1997)
23. Sinha, R., Girault, A., Goessler, G., Roop, P.S.: A formal approach to incremental converter synthesis for system-on-chip design. ACM Trans. Des. Autom. Electron. Syst. **20**(1), 13:1–13:30 (2014). <http://doi.acm.org/10.1145/2663344>
24. Buffalov, S., El-Fakih, K., Yevtushenko, N., Bochmann, G.: Progressive solutions to a parallel automata equation. In: König, H., Heiner, M., Wolisz, A. (eds.) FORTE 2003. LNCS, vol. 2767. Springer, Heidelberg (2003)
25. El-Fakih, K., Buffalov, S., Yevtushenko, N., Bochmann, G.: Progressive solutions to a parallel automata equation. Theoret. Comput. Sci. **362**, 17–32 (2006)
26. DiBenedetto, M.D., Sangiovanni-Vincentelli, A., Villa, T.: Model matching for finite state machines. IEEE Trans. Autom. Control **46**(11), 1726–1743 (2001)

27. Cao, J., Nymeyer, A.: Formal model of a protocol converter. In: Downey, R., Manyem, P. (eds.) *Fifteenth Computing: The Australasian Theory Symposium (CATS 2009)*, CRPIT, vol. 94. ACS, Wellington, pp. 107–117 (2009)
28. Sinha, R., Roop, P., Basu, S.: A model checking approach to protocol conversion. In: *Workshop on Model-driven High-level Programming of Embedded Systems (2007)*
29. Sinha, R., Roop, P.S., Basu, S.: SoC design approach using convertibility verification. *EURASIP J. Emb. Sys.* **2008** (2008)
30. Sinha, R., Roop, P.S., Basu, S., Salcic, Z.: Multi-clock SoC design using protocol conversion. In: *Proceedings of the Conference on Design, Automation and Test in Europe, DATE 2009*. European Design and Automation Association, pp. 123–128 (2009)
31. Sinha, R.: Automated techniques for formal verification of SoCs, Ph.D. dissertation, University of Auckland, New Zealand (2009)
32. Autili, M., Inverardi, P., Mignosi, F., Spalazzese, R., Tivoli, M.: Automated synthesis of application-layer connectors from automata-based specifications. In: Dediu, A.-H., Formenti, E., Martín-Vide, C., Truthe, B. (eds.) *LATA 2015*. LNCS, vol. 8977, pp. 3–24. Springer, Heidelberg (2015)
33. Ciancia, V., Martin, J., Martinelli, F., Matteucci, I., Petrocchi, M., Pimentel, E.: Automated synthesis and ranking of secure BPMN orchestrators. *Int. J. Secur. Softw. Eng.* **5**(2), 44–64 (2014). <http://dx.doi.org/10.4018/ijssse.2014040103>
34. Hopcroft, J., Motwani, R., Ullman, J.: *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley Publishing Company, Reading (2001)
35. Lynch, N., Tuttle, M.: An introduction to input/output automata. *CWI-Q.* **2**(3), 219–246 (1989)
36. M. R. Group: MVSIS, software package. <http://embedded.eecs.berkeley.edu/Respep/Research/mvsis/software.html>
37. Hallal, H., Negulescu, R., Petrenko, A.: Design of divergence-free protocol converters using supervisory control techniques. In: *7th IEEE International Conference on Electronics, Circuits and Systems, ICECS 2000*, vol. 2, pp. 705–708, December 2000
38. Avnit, K., D’Silva, V., Sowmya, A., Ramesh, S., Parameswaran, S.: A formal approach to the protocol converter problem. In: *DATE*, pp. 294–299 (2008)
39. D’Silva, V., Ramesh, S., Sowmya, A.: Bridge over troubled wrappers: Automated interface synthesis. In: *Proceedings of the 17th International Conference on VLSI Design, VLSID 2004*. IEEE Computer Society, Washington, D.C., p. 189 (2004). <http://dl.acm.org/citation.cfm?id=962758.963411>
40. Tivoli, M., Fradet, P., Girault, A., Göbller, G.: Adaptor synthesis for real-time components. In: Grumberg, O., Huth, M. (eds.) *TACAS 2007*. LNCS, vol. 4424, pp. 185–200. Springer, Heidelberg (2007)

# An Experimental Spatio-Temporal Model Checker

Vincenzo Ciancia<sup>1</sup>(✉), Gianluca Grilletti<sup>2</sup>, Diego Latella<sup>1</sup>,  
Michele Loreti<sup>3,4</sup>, and Mieke Massink<sup>1</sup>

<sup>1</sup> Istituto di Scienza e Tecnologie dell'Informazione 'A. Faedo', CNR, Pisa, Italy  
vincenzoml@gmail.com

<sup>2</sup> Scuola Normale Superiore, Pisa, Italy

<sup>3</sup> Università di Firenze, Florence, Italy

<sup>4</sup> IMT Alti Studi, Lucca, Italy

**Abstract.** In this work we present a spatial extension of the global model checking algorithm of the temporal logic CTL. This classical verification framework is augmented with ideas coming from the tradition of topological spatial logics. More precisely, we add to CTL the operators of the *Spatial Logic of Closure Spaces*, including the *surrounded* operator, with its intended meaning of a point being surrounded by entities satisfying a specific property. The interplay of space and time permits one to define complex spatio-temporal properties. The model checking algorithm that we propose features no particular efficiency optimisations, as it is meant to be a reference specification of a family of more efficient algorithms that are planned for future work. Its complexity depends on the product of temporal states and points of the space. Nevertheless, a prototype model checker has been implemented, made available, and used for experimentation of the application of spatio-temporal verification in the field of *collective adaptive systems*.

## 1 Introduction

A *collective system* consists of a large set of interacting individuals. The temporal evolution of the system is not only determined by the decisions taken by the individuals at the *local* level, but also by their interactions, that are observable at the *global* level. By their own nature, such systems feature a “spatial” distribution of the individuals (e.g., locations in physical space, or nodes of some digital or social network), affecting interaction possibilities and patterns. Verification of collective systems and of their adaptation mechanisms requires one to take such spatial constraints into account.

In this work, we provide a preliminary study on the feasibility of model checking as a fully automated analysis of spatio-temporal models. Our work is grounded on the so-called *snapshot models* (see [12] for an introduction). Spatial

---

Research partially funded by EU project QUANTICOL (nr. 600708) and IT MIUR project CINA.

information is encoded by some topological structure, in the tradition of *topological spatial logics* [1], whereas temporal information is described by a *Kripke frame*. The valuation of atomic propositions is a function of temporal states, and spatial locations. We employ *Čech closure spaces* for the spatial part of the modelling, following the research line initiated in [6] with the definition of the *Spatial Logic of Closure Spaces* (SLCS). Čech closure spaces are a generalisation of topological spaces also encompassing directed graphs.

Starting from a spatial and a temporal formalism, *spatio-temporal* logics may be defined, by introducing some mutually recursive nesting of spatial and temporal operators. Several combinations can be obtained, depending on the chosen spatial and temporal fragments, and the permitted forms of nesting of the two. A great deal of possibilities are explored in [12], for spatial logics based on topological spaces. We investigate one such structure, in the setting of closure spaces, namely the combination of the temporal logic *Computation Tree Logic* (CTL) and of SLCS, resulting in the *Spatio-Temporal Logic of Closure Spaces* (STLCS). STLCS permits arbitrary mutual nesting of its spatial and temporal fragments. As a proof of concept, we define a simple model checking algorithm, which is a variant of the classical CTL labelling algorithm [2, 8], augmented with the algorithm in [6] for the spatial fragment. The algorithm, which operates on finite spaces, has been implemented as a prototype [10], that we discuss in the current paper. The same algorithm is also currently implemented in the tool *topochecker* [4], which is meant to be further developed and maintained, and was used in [7] in order to check spatio-temporal properties of bike sharing systems.

*Related Work.* The literature on topological spatial logics is rich (see [1]). However, model checking is typically not taken into account; this is discussed in detail in [6]. Relevant exceptions are the recent works [11], where statistical model checking is used on a (linear) spatio-temporal logics of signals, and [13], also developing a model checker for a linear spatio-temporal logic of signals, augmented with some metric information, and inheriting the approach of closure spaces from [6]. Our work diverts from these research lines in that the underlying temporal model is branching, thus permitting thorough evaluation of system properties in the presence of information about nondeterministic choice in the model. In computer science, the term *spatial logics* has also been used for logics that predicate about the internal structure of processes in process calculi. A model checker for such kind of logics was developed in [3]. Indeed, the theory and tool we present are linked to topological spatial logics rather than the area of process calculi, thus the developed algorithms are very different in nature.

## 2 Motivating Example: Adaptive Smart Transport Network

This work is part of a larger research effort aimed at formal verification of spatio-temporal requirements of *collective adaptive systems*, in the scope of the EU FP7

QUANTICOL project<sup>1</sup>. In order to motivate the proposed tool in the theory of verification of adaptive systems, we briefly report on a recent case study, detailed in [5], where the STLCS model checker has been used in the context of adaptive systems, and in particular of *smart transport networks*. The context is the bus network of a city. The model checker is primarily used to identify occurrences of *clumping* of buses, that is, buses of the same line that are “too close in space-time” to each other, resulting in several buses of the same line passing by the same stops within a short amount of time, and longer intervals without any buses at certain stops. More precisely, a bus is part of a *clump* if *it is close to a point where another bus of the same line will be very soon*. This statement is inherently spatio-temporal, and classical temporal logics do not have the ability to directly express it. It turns out that there is some ambiguity in the formalisation of this sentence, resulting in different possible STLCS formulas characterising it. Once established these formulas, the bus coordination system is equipped with an adaptation layer, enabling buses to wait for some time at a stop, in order to avoid the emergence of clumps at the expenses of some additional delay on the line. The underlying hypothesis is that clumping happens when some buses are forced to delay (e.g. because of traffic conditions) but the system evolves immediately afterwards, in such a way that subsequent buses of the same line do not delay. The STLCS model checker is used to define an analysis methodology that estimates the impact of adaptation, before deployment, starting from existing traces (logs) of the system. Each trace, in the form of a series of GPS coordinates for each bus, is considered as a deterministic system. For traces featuring clumping (checked using the model checker), the expected non-deterministic behaviour of the system under the effect of the adaptation layer is then computed as a spatio-temporal model, by augmenting the existing trace with the possible “wait” steps of each bus. The counterexample-generation capabilities of the model checker are finally used on such Kripke frame to analyse the impact of the adaptation, by identifying new traces containing wait instructions that correct the problem. By doing this, one is able to check if, and under what conditions, the adaptation strategy succeeds in mitigating or eliminating the clumping problem, and confirm or disprove (depending on the actual situation) the hypothesis underlying the choice of the adaptation strategy. For more details on the specific case study, we refer the reader to [5]; in the remainder of the paper, we shall focus on the formal definition of the STLCS logic, and its model checking algorithm, as both were not presented in [5].

### 3 Closure Spaces

In this work, we use *closure spaces* to define basic concepts of *space*. Below, we recall several definitions, most of which are explained in [9]. See also [6] for a thorough description of SLCS, the spatial logic of closure spaces, and its model-checking algorithm. A closure space is a set equipped with a *closure operator*

<sup>1</sup> See the web site <http://www.quanticol.eu>.



obeying to certain laws. In the finite case, closure spaces are graphs, but also (infinite) topological spaces are an instance of the more general constructions.

**Definition 1.** A closure space is a pair  $(X, \mathcal{C})$  where  $X$  is a set, and the closure operator  $\mathcal{C} : 2^X \rightarrow 2^X$  assigns to each subset of  $X$  its closure, obeying to the following laws, for all  $A, B \subseteq X$ :

1.  $\mathcal{C}(\emptyset) = \emptyset$ ;
2.  $A \subseteq \mathcal{C}(A)$ ;
3.  $\mathcal{C}(A \cup B) = \mathcal{C}(A) \cup \mathcal{C}(B)$ .

The notion of *interior*, dual to closure, is defined as  $\mathcal{I}(A) = X \setminus \mathcal{C}(X \setminus A)$ . Closure spaces are a generalisation of *topological spaces*. The axioms defining a closure space are also part of the definition of a *Kuratowski closure space*, which is one of the possible alternative definitions of a topological space. More precisely, a topological space is a closure space where the axiom  $\mathcal{C}(\mathcal{C}(A)) = \mathcal{C}(A)$  (idempotency) holds. We refer the reader to, e.g., [9] for more information.

Various notions of *boundary* can be defined. The *closure boundary* (often called *frontier*) is used for the *surrounded operator* in STLCS.

**Definition 2.** In a closure space  $(X, \mathcal{C})$ , the boundary of  $A \subseteq X$  is defined as  $\mathcal{B}(A) = \mathcal{C}(A) \setminus \mathcal{I}(A)$ . The interior boundary is  $\mathcal{B}^-(A) = A \setminus \mathcal{I}(A)$ , and the closure boundary is  $\mathcal{B}^+(A) = \mathcal{C}(A) \setminus A$ .

A closure space may be derived starting from a *binary relation*, that is, a *graph*. In particular all finite spaces are in this form. This is easily seen by the equivalent characterization of *quasi-discrete* closure spaces.

**Definition 3.** Consider a set  $X$  and a relation  $R \subseteq X \times X$ . A closure operator is obtained from  $R$  as  $\mathcal{C}_R(A) = A \cup \{x \in X \mid \exists a \in A. (a, x) \in R\}$ .

Closure spaces derived from a relation can be characterised as *quasi-discrete* spaces (see also Lemma 9 of [9] and the subsequent statements).

**Definition 4.** A closure space is quasi-discrete if and only if one of the following equivalent conditions holds: (i) each  $x \in X$  has a minimal neighbourhood<sup>2</sup>  $N_x$ ; (ii) for each  $A \subseteq X$ ,  $\mathcal{C}(A) = \bigcup_{a \in A} \mathcal{C}(\{a\})$ .

**Proposition 1.** A closure space  $(X, \mathcal{C})$  is quasi-discrete if and only if there is a relation  $R \subseteq X \times X$  such that  $\mathcal{C} = \mathcal{C}_R$ .

Summing up, a closure space enjoys minimal neighbourhoods, and the closure of  $A$  is determined by the closure of the singletons composing  $A$ , if and only if the space is derived from a relation using Definition 3.

---

<sup>2</sup> A *minimal neighbourhood* of  $x$  is a set  $A$  that is a *neighbourhood* of  $x$ , namely,  $x \in \mathcal{I}(A)$ , and is included in all other neighbourhoods of  $x$ .

## 4 The Spatio-Temporal Logic of Closure Spaces

We define a logic interpreted on a variant of Kripke models, where valuations are interpreted at points of a closure space. Fix a set  $P$  of proposition letters.

**Definition 5.** *STLCS formulas are defined by the following grammar, where  $p$  ranges over  $P$ :*

$$\begin{array}{l|l} \Phi ::= \top & [\text{TRUE}] \\ | p & [\text{ATOMIC PREDICATE}] \\ | \neg \Phi & [\text{NOT}] \\ | \Phi \vee \Phi & [\text{OR}] \\ | \mathcal{N} \Phi & [\text{CLOSE}] \\ | \Phi \mathcal{S} \Phi & [\text{SURROUNDED}] \\ | \mathbf{A} \varphi & [\text{ALL FUTURES}] \\ | \mathbf{E} \varphi & [\text{SOME FUTURE}] \end{array}$$

$$\begin{array}{l|l} \varphi ::= \mathcal{X} \Phi & [\text{NEXT}] \\ | \Phi \mathcal{U} \Phi & [\text{UNTIL}] \end{array}$$

The logic STLCS features the CTL path quantifiers  $\mathbf{A}$  (“for all paths”), and  $\mathbf{E}$  (“there exists a path”). As in CTL, such quantifiers must necessarily be followed by one of the path-specific temporal operators, such as<sup>3</sup>  $\mathcal{X}\Phi$  (“next”),  $\mathbf{F}\Phi$  (“eventually”),  $\mathbf{G}\Phi$  (“globally”),  $\Phi_1 \mathcal{U} \Phi_2$  (“until”), but unlike CTL, in this case  $\Phi$ ,  $\Phi_1$  and  $\Phi_2$  are STLCS formulas that may make use of spatial operators. Further operators of the logic are the boolean connectives, and the spatial operators  $\mathcal{N}\Phi$ , denoting closeness to points satisfying  $\Phi$ , and  $\Phi_1 \mathcal{S} \Phi_2$ , denoting that a specific point satisfying  $\Phi_1$  is surrounded, via points satisfying  $\Phi_1$ , by points satisfying  $\Phi_2$ . The mutual nesting of such operators permits one to express spatial properties in which the involved points are constrained to certain temporal behaviours. Let us proceed with a few examples. Consider the STLCS formula  $\mathbf{EG}$  (**green S blue**). This formula is satisfied in a point  $x$  in the graph, associated to the initial state  $s_0$ , if there exists a (possible) evolution of the system, starting from  $s_0$ , in which point  $x$ , in every state in the path, satisfies *green* and is surrounded by blue. A further, nested, example is the STLCS formula  $\mathbf{EF}$  (**green S (AX blue)**). This formula is satisfied by a point  $x$  in the graph, in the initial state  $s_0$ , if there is a (possible) evolution of the system, starting from  $s_0$ , in which point  $x$  is eventually green and surrounded by points  $y$  that, for every possible evolution of the system from then on, will be blue in the next time step.

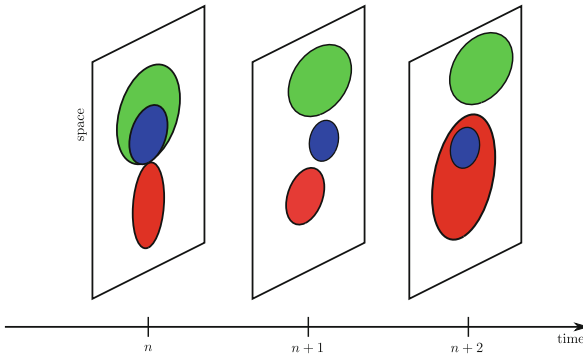
A model  $\mathcal{M}$  is composed of a Kripke structure  $(S, \mathcal{T})$ , where  $S$  is a non-empty set of *states*, and  $\mathcal{T}$  is a non-empty *accessibility relation* on states, and a closure space  $(X, \mathcal{C})$ , where  $X$  is a set of points and  $\mathcal{C}$  the closure operator. Every state  $s$  has an associated valuation  $\mathcal{V}_s$ , making  $((X, \mathcal{C}), \mathcal{V}_s)$  a *closure model* according to

<sup>3</sup> Some operators may be derived from others; for this reason, e.g., in Definition 5, and Sect. 5, we use a minimal set of connectives. As usual in logics, there are several different choices for such a set.

Definition 6 of [6]. Equivalently, valuations have type  $S \times X \rightarrow 2^P$ , where  $P$  is the set of atomic propositions, thus, the valuation of atomic propositions depends both on states and points of the space. Intuitively, there is a set of possible worlds, i.e. the states in  $S$ , and a spatial structure represented by a closure space. In each possible world there is a different valuation of atomic propositions, inducing a different “snapshot” of the spatial situation which “evolves” over time. In this paper we assume that the spatial structure  $(X, \mathcal{C})$  does not change over time. Other options are indeed possible. For instance, when space depends on  $S$ , one may consider an  $S$ -indexed family  $(X_s, \mathcal{C}_s)_{s \in S}$  of closure spaces.

**Definition 6.** A model is a structure  $\mathcal{M} = ((X, \mathcal{C}), (S, \mathcal{T}), \mathcal{V}_{s \in S})$  where  $(X, \mathcal{C})$  is a closure space,  $(S, \mathcal{T})$  is a Kripke frame, and  $\mathcal{V}$  is a family of valuations, indexed by states. For each  $s \in S$ , we have  $\mathcal{V}_s : P \rightarrow \mathcal{P}(X)$ .

A path in the Kripke structure is a sequence of *spatial models* (in the sense of [6]) indexed by instants of time; see Fig. 1, where space is a two-dimensional structure, and valuations at each state are depicted by different colours.



**Fig. 1.** In spatio-temporal logics, a temporal path represents a sequence of *snapshots* induced by the time-dependent valuations of the atomic propositions (Color figure online).

**Definition 7.** Given Kripke frame  $\mathcal{K} = (S, \mathcal{T})$ , a path  $\sigma$  is a function from  $\mathbb{N}$  to  $S$  such that for all  $n \in \mathbb{N}$  we have  $(\sigma(i), \sigma(i + 1)) \in \mathcal{T}$ . Call  $\mathcal{P}_s$  the set of infinite paths in  $\mathcal{K}$  rooted at  $s$ , that is, the set of paths  $\sigma$  with  $\sigma(0) = s$ .

The evaluation contexts are of the form  $\mathcal{M}, x, s \models \Phi$ , where  $\Phi$  is a STLCS formula,  $s$  is a state of a Kripke structure, and  $x$  is a point in space  $X$ .

**Definition 8.** Satisfaction is defined in a model  $\mathcal{M} = ((X, \mathcal{C}), (S, \mathcal{T}), \mathcal{V}_{s \in S})$  at point  $x \in X$  and state  $s \in S$  as follows:

$$\begin{aligned}
 \mathcal{M}, x, s &\models \top \\
 \mathcal{M}, x, s &\models p && \iff x \in \mathcal{V}_s(p) \\
 \mathcal{M}, x, s &\models \neg\Phi && \iff \mathcal{M}, x, s \not\models \Phi \\
 \mathcal{M}, x, s &\models \Phi \vee \Psi && \iff \mathcal{M}, x, s \models \Phi \text{ or } \mathcal{M}, x, s \models \Psi \\
 \mathcal{M}, x, s &\models \mathcal{N}\Phi && \iff x \in \mathcal{C}(\{y \in X \mid \mathcal{M}, y, s \models \Phi\}) \\
 \mathcal{M}, x, s &\models \Phi \mathcal{S} \Psi && \iff \exists A \subseteq X. x \in A \wedge \forall y \in A. \mathcal{M}, y, s \models \Phi \wedge \\
 &&& \quad \wedge \forall z \in \mathcal{B}^+(A). \mathcal{M}, z, s \models \Psi \\
 \mathcal{M}, x, s &\models \mathbf{A} \varphi && \iff \forall \sigma \in \mathcal{P}_s. \mathcal{M}, x, \sigma \models \varphi \\
 \mathcal{M}, x, s &\models \mathbf{E} \varphi && \iff \exists \sigma \in \mathcal{P}_s. \mathcal{M}, x, \sigma \models \varphi \\
 \\ 
 \mathcal{M}, x, \sigma &\models \mathcal{X}\Phi && \iff \mathcal{M}, x, \sigma(1) \models \Phi \\
 \mathcal{M}, x, \sigma &\models \Phi \mathcal{U} \Psi && \iff \exists n. \mathcal{M}, x, \sigma(n) \models \Psi \text{ and } \forall n' \in [0, n). \mathcal{M}, x, \sigma(n') \models \Phi
 \end{aligned}$$

The syntax we provide is rather essential. Further operators can be derived from the basic ones; e.g., one can define conjunction and implication using negation and disjunction; spatial *interior* is defined as the dual of  $\mathcal{N}$ ; several derived path operators are well-known for the temporal fragment, by the theory of CTL. We do not attempt to make an exhaustive list; for the classical temporal connectives, see e.g., [2]; for spatial operators, [6] provides some interesting examples. In Sect. 6 we show some simple spatial and spatio-temporal formulas. More complex formulas can be found in [5].

## 5 Model Checking

In this section we describe the model checking algorithm, which is a variant of the well-known CTL labelling algorithm. For more information on CTL and its model checking techniques, see e.g., [2] or [8]. This algorithm operates in the case of *finite, quasi-discrete closure spaces*, represented as finite graphs. Assume the type `Set` implementing a finite set-like data structure<sup>4</sup>, with elements of type `El` and operations `union`, `inter`, `diff`, `times`, `emptyset`, with the obvious types.

We represent a *finite directed graph* as the triple

$$(\mathbf{G} : \text{Set}, \text{Pred\_G} : \text{El} \rightarrow \text{Set}, \text{Cl\_G} : \text{Set} \rightarrow \text{Set})$$

where the argument and result of the operators implementing closure `Cl_G`, and predecessor `Pred_G`, are constrained to belong to `G`. We describe a model by a pair of graphs  $\mathcal{M} = (\mathcal{X}, \mathcal{T})$  where the spatial component is  $\mathcal{X} = (\mathbf{X}, \text{Pred\_X}, \text{Cl\_X})$ , and the temporal component (which can be thought of as a Kripke frame) is  $\mathcal{T} = (\mathbf{T}, \text{Pred\_T}, \text{Cl\_T})$ .

Consider the finite set  $\mathbf{S} = \mathbf{X} \text{ times } \mathbf{T}$  of *points in space-time*; given a subset  $\mathbf{A} \subseteq \mathbf{S}$ , and a state  $\mathbf{t} \in \mathbf{T}$ , we let `space_sec(A, t)` be the subset of `X` containing the points  $x$  such that  $(x, \mathbf{t}) \in \mathbf{A}$ ; we define `time_sec` in a similar way. With

<sup>4</sup> We remark that the complexity of operations on such type affect the complexity of the algorithm; however, since the algorithm is global, the `Set` type may be implemented using an explicit lookup table, that is, an array of boolean values indexed by states, as usual in model checking, obtaining the complexity that we discussed.

**choose** we indicate the operation of choosing an element from a non-empty set (without making explicit how to pick it). For  $\Phi$  an STLCS formula, and  $\mathcal{M}$  a model, we let  $\llbracket \Phi \rrbracket^{\mathcal{M}} = \{(x, t) \subseteq S \mid \mathcal{M}, x, t \models \Phi\}$ .

Given a formula  $\Phi$  and a model  $\mathcal{M}$ , the algorithm proceeds by induction on the structure of  $\Phi$ ; the output of the algorithm is the set  $\llbracket \Phi \rrbracket^{\mathcal{M}}$ . In the following, we present the relevant code portions addressing each case of the syntax; we omit the cases for the boolean connectives, and use a minimal set of connectives, apt to efficient verification, for the temporal part, namely  $\text{EX}$ ,  $\text{AF}$ ,  $\text{EU}$ . The cases for  $\Phi = \text{EX}\Phi'$  and  $\text{E}(\Phi_1 \mathcal{U}\Phi_2)$  make use of the auxiliary function `pred_time`:

```
function pred_time(A)
  F := emptyset;
  foreach ((x,t) in A)
    U := Pred_T(t);
    F := F union ({x} times U);
  return F;
```

*Case  $\Phi = \mathcal{N}\Phi'$ :* The result is computed as the set  $\bigcup_{(x,t) \in \llbracket \Phi' \rrbracket^{\mathcal{M}}} \{(y, t) \mid y \in \mathcal{C}_X(x)\}$ , which is correct in a quasi-discrete closure space  $(X, \mathcal{C})$ , as, for all sets  $A$ , we have  $\mathcal{C}(A) = \bigcup_{x \in A} \mathcal{C}(\{x\})$ .

```
let A =  $\llbracket \Phi' \rrbracket^{\mathcal{M}}$ ;
P := emptyset;
foreach ((x,t) in A)
  P := P union (Cl_X({x}) times {t});
return P;
```

*Case  $\Phi = \Phi_1 \mathcal{S}\Phi_2$ :* For every state  $t$ , we compute the spatial components of  $\llbracket \Phi_1 \rrbracket^{\mathcal{M}}$  and  $\llbracket \Phi_2 \rrbracket^{\mathcal{M}}$  at state  $t$  (called  $R$  and  $Bs$  in the pseudo-code). Then we apply the algorithm described in [6].

```
let A =  $\llbracket \Phi_1 \rrbracket^{\mathcal{M}}$ ;
let B =  $\llbracket \Phi_2 \rrbracket^{\mathcal{M}}$ ;
F := emptyset;
foreach (t in T)
  R := space_sec(A, t);
  Bs := space_sec(B, t);
  U := R union Bs;
  D := Cl_X(U) diff U;
  while (D != emptyset)
    s := choose(D);
    N := ( Cl_X({s}) inter R ) diff Bs;
    R := R diff N;
    D := ( D union N ) diff {s};
  F := F union (R times {t})
return F;
```

*Case  $\Phi = \text{EX}\Phi'$ :* The set of predecessors (in time) of the points in space-time belonging to the semantics of  $\Phi'$  are computed and returned.

```
let A =  $\llbracket \Phi' \rrbracket^{\mathcal{M}}$ ;
return pred_time(A);
```

*Case  $\Phi = \text{AF}\Phi'$ :* The case for AF is essentially the efficient algorithm for EG presented in [2], except that it is presented in “dual” form, using the fact that  $\llbracket \text{EG}\Phi' \rrbracket^{\mathcal{M}} = \llbracket \neg \text{AF}(\neg\Phi') \rrbracket^{\mathcal{M}}$ . The algorithm is iterated for each point of the space. More precisely, for each  $x \in X$ , vector `count`, whose indices are states in T, is used to maintain the following invariant property along the `while` loop: *whenever `count[t]` is 0, we have  $\mathcal{M}, x, t \models \text{AF}\Phi'$* . In order to establish such invariant property, before the `while` loop, `count[t]` is initialised to 0 for each point in F, which is the set of points t such that there is some x, with  $\mathcal{M}, x, t \models \Phi'$  (therefore, also  $\mathcal{M}, x, t \models \text{AF}\Phi'$  by definition). For each remaining state t, the value of `count[t]` is set to the number of its successors. Along the `while` loop, the set U is the set of states t that, at the previous iteration (or at initialisation), have been shown to satisfy  $\mathcal{M}, x, t \models \text{AF}\Phi'$ . At each iteration, for each t in U, function `sem_af_aux` is used to inspect each predecessor y of t and decrease the value of `count[y]`. When `count[y]` becomes 0, y is added to U, as it is proved that all the successors of y satisfy  $\text{AF}\Phi'$ ; no state is added twice to U (which is guaranteed by the check `if count[y] > 0` in function `sem_af_aux`).

```

let A =  $\llbracket \Phi' \rrbracket^{\mathcal{M}}$ ;
M := emptyset;
foreach (x in X)
  F := time_sec(A, x);
  U := F;
  foreach (t in (T minus F))
    count[t] := cardinality (Cl_T({t}));
  foreach (t in F)
    count[t] := 0;
  while (U != emptyset)
    U' := U;
    U := emptyset;
    foreach (t in U')
      sem_af_aux(F, U, count, t);
  M := M union ({x} times F);
return M;

```

```

function sem_af_aux(F, U, count, t)
  foreach (y in Pred_T(t))
    if count[y] > 0 then
      count[y] := count[y] - 1;
      if (count[y] = 0)
        then
          U := U union {y};
          F := F union {y};

```

*Case  $E(\Phi_1 \mathcal{U}\Phi_2)$ :* In this case, the algorithm computes the set of points that either satisfy  $\Phi_2$ , or satisfy  $\Phi_1$  and can reach points satisfying  $\Phi_2$  in a finite number of temporal steps. This is accomplished by maintaining, along the `while` loop, the set F of points that have already been shown to be in this situation (initialised to the points satisfying  $\Phi_2$ ), and the set L of points that satisfy  $\Phi_1$ , are not in F, and can reach F in one (temporal) step. At each iteration, F is augmented by the

points in  $L$ , and  $L$  is recomputed. When  $L$  is empty,  $F$  contains all the required points. The set  $P$ , initialised to the points satisfying  $\Phi_1$ , is used to guarantee termination, or more precisely, that no node is added twice to  $L$ .

```

let A =  $\llbracket \Phi_1 \rrbracket^M$ ;
let B =  $\llbracket \Phi_2 \rrbracket^M$ ;
F := B;
P := A diff B;
L := pred_time(F) inter P;
while(L <> emptyset)
  F := F union L;
  P := P diff L;
  L := pred_time( L ) inter P;
return F

```

In the implementation, available at [10], the definition of the Kripke structure is given by a file containing a graph, in the plain text graph description language<sup>5</sup> `dot`. Quasi-discrete closure models are provided either in the form of a graph, or in the form of a set of images, one for each state in the Kripke structure, having the same size. The colours of the pixels in the image are the valuation function, and atomic propositions actually are colour ranges for the red, green, and blue components of the colour of each pixel. In this case, the model checker verifies a special kind of closure spaces, namely finite regular grids.

The model checker interactively displays the image corresponding to a “current” state. The most important command of the tool is `sem colour formula`, that changes the colour of points satisfying the given formula, to the specified colour, in the current state. The tool has the ability to define parametrised names for formulas (no recursion is allowed). Formulas are automatically saved and restored from a text file. The implementation is that of a so-called “global” model checker, that is, all points in space-time satisfying the given formulas are coloured/returned at once. More information on the tool, as well as the complete source code, is available at [10].

The complexity of the currently implemented algorithm is linear in the product of number of states plus arcs of the temporal model, subformulas, and points plus arcs of the space, which is a consequence of the algorithm described in [6] being linear in the number of points, and the classical algorithm for CTL being linear in the number of states (in both cases, for each specific formula). Such efficiency is sufficient for experimenting with the logic (see [5]), but if both the space and the Kripke structures are large, model checking may become impractical.

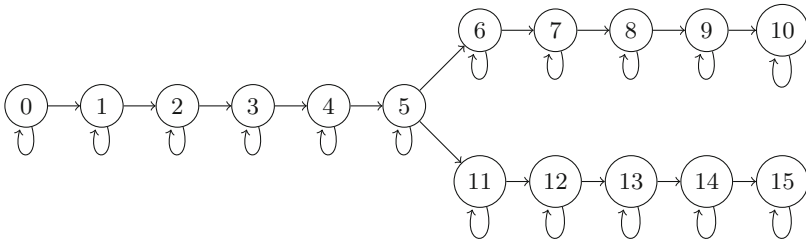
*Remark 1.* Even though we consider a thorough performance analysis of the basic algorithm beyond the scope of this preliminary investigation, and possibly redundant, we can provide some hints about the feasibility and the efficiency problems of spatio-temporal model checking. Our prototype has been implemented in OCaml, trying to make use of the declarative features of the language. For example, we use the `Set` module of OCaml, implementing a purely

<sup>5</sup> Further information on the `dot` notation can, for example, be found at <http://www.graphviz.org/Documentation.php>.

functional data type for sets, in order to make use of Definition 8 directly, rather than attempting to use bit arrays to improve performance, as it is typical in global model checking. In the example of Sect. 2, we considered rather small Kripke frames, in the order of one hundred states. However, the images associated to each state contain around one million points. Therefore, even though the state space seems rather small, the number of examined points in space-time is in the order of 50–100 millions of states. The model checker is able to perform the required analyses in a time that roughly varies between some seconds and 30 min, depending on the formula, on a quite standard laptop computer. On the one hand, this proves that non-trivial examples may be analysed using the simple algorithm we proposed, but on the other hand, the same data strongly suggests that effective optimisations need to be found to make large-scale spatio-temporal model checking feasible (more on this in the conclusions).

## 6 Examples

Finally, we show some simple examples to illustrate operation of the tool. Consider the Kripke frame in Fig. 2. To each state, an image is associated, that the model checker considers an undirected graph whose nodes are pixels, and whose arcs go from each pixel to the neighbouring ones, in the four main directions *north*, *south*, *east*, *west*. The image associated to each state are shown in Fig. 3.



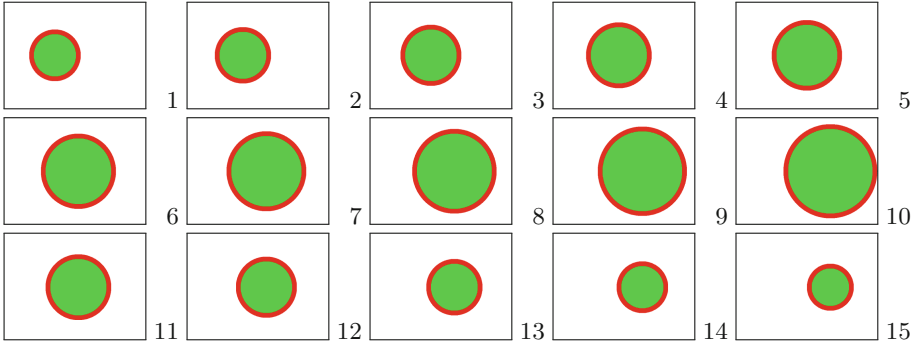
**Fig. 2.** The Kripke frame of our example.

Let us consider the green circle with red boundary, in the first image of Fig. 3. The centre of the circle in the figures moves along time towards the right. Its radius grows at constant speed in turn. Then, in state 5, there is a non-deterministic choice point. In the first possible future (states 6–10), the radius keeps growing, whereas in the second future (states 11–15) the radius shrinks. In the following, we shall use atomic propositions  $g$ ,  $r$ , evaluating to the green and red points (boundary of the green area) in the figures.

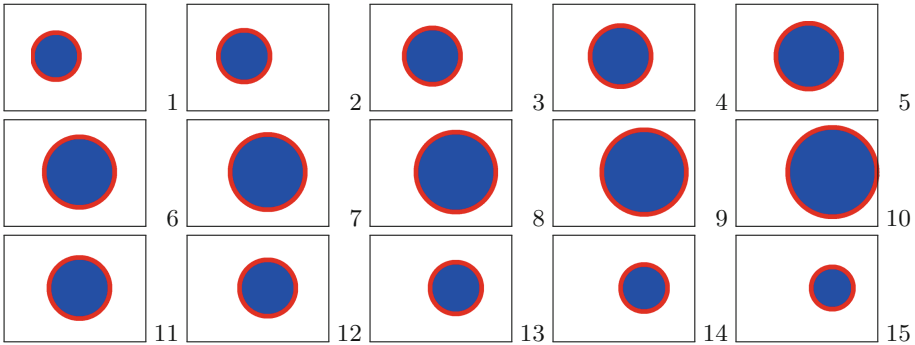
Let us first consider the spatial formula  $g\mathcal{S}r$  (green points surrounded by a red boundary). Such formula is evaluated, colouring in blue the points satisfying it, by executing the command below. Its output is displayed in Fig. 4, for each point in space and state of the Kripke structure:

```
sem blue S[<g>, <r>]
```





**Fig. 3.** The images providing valuations for the atomic propositions. Each valuation depicts a green, filled circle with a red border (Color figure online).



**Fig. 4.** The points satisfying  $gSr$  are displayed in blue. These are the filled circles; their borders remain red (Color figure online).

A second example is the spatio-temporal formula  $EF(gSr)$ , computed by:

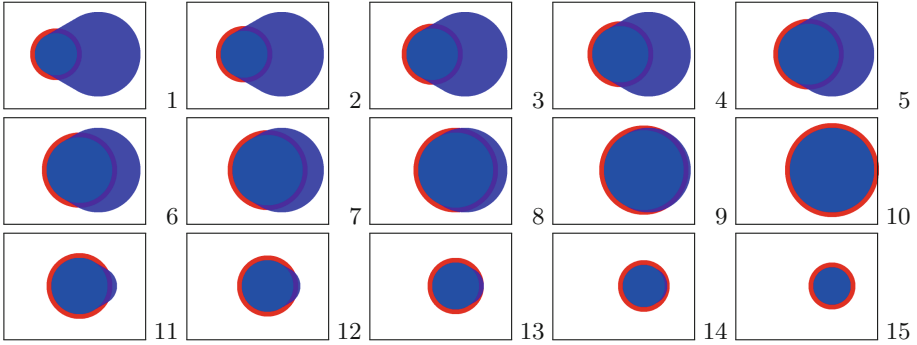
```
sem blue EF (S [<g>, <r>])
```

See output in Fig. 5. For each point in space and temporal state, the points that will eventually satisfy green and be surrounded by red, are coloured in blue.

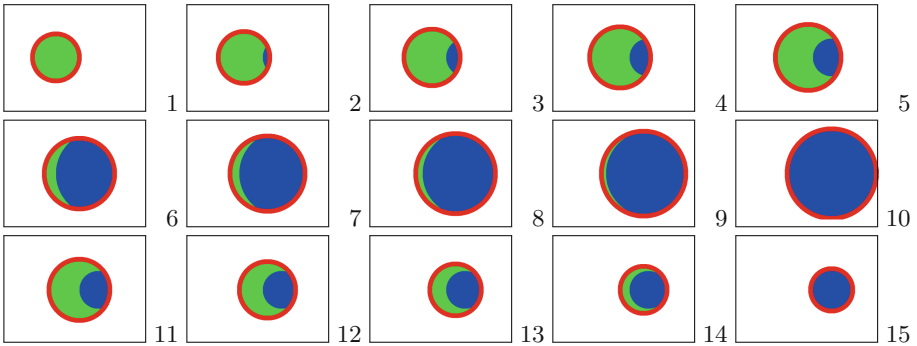
Finally, we show the semantics of the spatio-temporal formula  $AGg$ , characterising points that will be green forever in all futures. In Fig. 6 we show the output of

```
sem blue AG <g>
```

In states 1–5 of Fig. 6, the valuation of the formula in each state is the intersection of two circular areas, namely the intersection of the green area in the chosen state and the green area in state 15. By this, in particular, the valuation of the formula is the empty set in state 1. In states 11–15, the situation is similar, since state 15 is a possible future. On the other hand, in states 6–10, state 15 is not reachable, thus the area which will be forever green is larger.



**Fig. 5.** In blue, the semantics of  $EF(gSr)$  (Color figure online).



**Fig. 6.** In blue, the semantics of  $AGg$  (Color figure online).

## 7 Conclusions and Future Work

In this paper we studied an extension of the Spatial Logic of Closure Spaces of [6] with classical CTL temporal logic operators. A simple, proof-of-concept, temporal extension of the spatial model checker for SLCS has also been presented together with a simple example. The spatio-temporal model checker has been used for a urban transportation case study, as described in detail in [5], and for analysing properties of bike sharing systems [7].

The use of a spatial model checker provides us with a sophisticated tool for checking properties of systems where location plays an important role, as it does in many collective adaptive systems. By enhancing this standpoint with a temporal perspective, the interplay of space and time allows one to define complex spatio-temporal formulas, predicating over the relation between points of a spatial model that varies over branching time.

Current work is focused on defining *collective* variants of spatial and spatio-temporal properties; that is, the satisfaction value of a formula is defined on a set of points, rather than on a single point, so that the satisfaction value of a

formula with respect to a set of points (a collective property) is not necessarily determined by the satisfaction values over the points composing the set (an individual property). Such interpretation of spatio-temporal logics is particularly motivated by the setting of collective adaptive systems and *emergent properties*.

High priority in future work will be given to the investigation of various kinds of optimisations for spatio-temporal model checking, including partition refinement of models, symbolic methods, and on-line algorithms taking advantage of differential descriptions of the change between system states. An orthogonal, but nevertheless interesting, aspect of spatio-temporal computation is the introduction of probability and stochastic aspects, as well as the introduction of metrics, yielding bounded versions of the introduced spatio-temporal connectives. Such features will be studied in the context of STLCS. Investigating efficient model checking algorithms in this setting is important for practical applications, which are very often quantitative rather than boolean.

Another ongoing work is the development of qualitative and quantitative spatio-temporal analysis of the behaviour of complex systems, which was started in [13], and features an extension of *Signal Temporal Logic* to accommodate spatial information. In that case, models are deterministic (thus non-branching) and *monitoring* plays a central role. Single, infinite traces (intended to be the outcome of some approximation of a complex system, described by a system of differential equations) are analysed to check whether specific spatio-temporal properties are satisfied, such as, the formation of specific patterns.

## References

1. Aiello, M., Pratt-Hartmann, I., van Benthem, J. (eds.): Handbook of Spatial Logics. Springer, Netherlands (2007)
2. Baier, C., Katoen, J.P.: Principles of Model Checking. MIT Press, Cambridge (2008)
3. Caires, L., Vieira, H.T.: SLMC: a tool for model checking concurrent systems against dynamical spatial logic specifications. In: Flanagan, C., König, B. (eds.) TACAS 2012. LNCS, vol. 7214, pp. 485–491. Springer, Heidelberg (2012)
4. Ciancia, V.: topochecker - a topological model checker (2015). <http://fmt.isti.cnr.it/topochecker> - <https://github.com/vincenzoml/topochecker>
5. Ciancia, V., Gilmore, S., Grilletti, G., Latella, D., Loretì, M., Massink, M.: Spatio-temporal model-checking of vehicular movement in transport systems. submitted for journal publication, available from the authors
6. Ciancia, V., Latella, D., Loretì, M., Massink, M.: Specifying and verifying properties of space. In: Diaz, J., Lanese, I., Sangiorgi, D. (eds.) TCS 2014. LNCS, vol. 8705, pp. 222–235. Springer, Heidelberg (2014)
7. Ciancia, V., Latella, D., Massink, M., Paskauskas, R.: Exploring spatio-temporal properties of bike-sharing systems. In: 9th International Conference on Self-Adaptive and Self-Organizing Systems Workshops. IEEE (volume to appear, 2015)
8. Clarke, E.M., Grumberg, O., Peled, D.: Model Checking. MIT Press, Cambridge (2001)
9. Galton, A.: A generalized topological view of motion in discrete space. Theor. Comput. Sci. **305**(1–3), 111–134 (2003)

10. Grilletti, G., Ciancia, V.: STLCS model checker (2014). <https://github.com/cherosene/ctl.logic>
11. Haghighi, I., Jones, A., Kong, Z., Bartocci, E., Grosu, R., Belta, C.: Spatel: a novel spatial-temporal logic and its applications to networked systems. In: Proceedings of the 18th International Conference on Hybrid Systems: Computation and Control, HSCC 2015, Seattle, WA, USA, 14–16 April 2015, pp. 189–198. ACM (2015)
12. Kontchakov, R., Kurucz, A., Wolter, F., Zakharyashev, M.: Spatial logic + temporal logic = ? In: Aiello, M., Pratt-Hartmann, I., van Benthem, J. (eds.) Handbook of Spatial Logics, pp. 497–564. Springer, Netherlands (2007)
13. Nenzi, L., Bortolussi, L., Ciancia, V., Loretì, M., Massink, M.: Qualitative and quantitative monitoring of spatio-temporal properties. In: Bartocci, E., Majumdar, R., et al. (eds.) RV 2015. LNCS, vol. 9333, pp. 21–37. Springer, Heidelberg (2015)

# Dependable Composition of Software and Services in the Internet of Things: A Biological Approach

Amleto Di Salle, Francesco Gallo<sup>(✉)</sup>, and Alexander Perucci

University of L'Aquila, Via Vetoio 1, 67100 L'Aquila, Italy  
{amleto.disalle, francesco.gallo}@univaq.it,  
alexander.perucci@graduate.univaq.it  
<http://www.univaq.it>

**Abstract.** If we pause a moment to reflect on the innovations of the last twenty years in the field of information technology, we realize immediately as consumer electronics, computers and telecommunications have changed their balance of power. Today, Internet is so ingrained in the culture of the people who seems to be always there, and you can hardly imagine to live without it. Today mobile devices are computers, and their inter-connection and connection with several kinds of technological objects is ever more increasing. This has led to the emergence of new concepts, such as the Internet of Things (IoT), Machine to Machine (M2M), and People to Machine (P2M) and the consequent need to provide frameworks that allows communication and interoperability between heterogeneous objects. Furthermore, the increasing availability of data and especially of computational power allows *things* to serve not only as data producers but also as consumers.

Thus we can think about *internet of things as a cloud of services* software and services composition.

In such a highly mobile environment, both the user and “things” may be subject to frequent movement, demanding a frequent recomposition. In this paper, we propose a preliminary biological-inspired approach for adaptive software composition at run time. The approach leverages the concept of immune system to ensure dependability e.g. availability and reliability, of a composition of software and services in the Internet of Things.

**Keywords:** Adaptive composition · Run-time composition · Service composition

## 1 Introduction

Information and communication systems are invisibly embedded in the environment around us. This results in the generation of enormous amounts of data which have to be stored, processed and presented in a seamless, efficient and easily interpretable form. This model will consist of services that are commodities and delivered in a manner similar to traditional commodities.

The computing criterion will need to go beyond traditional mobile computing scenarios that use smart phones and portables, and evolve into connecting everyday existing objects and embedding intelligence into our environment.

The traditional Internet concept is radically evolving into a Network of interconnected objects that not only harvest information from the environment (sensing) and interacts with the physical world (actuation/command/control), but also uses existing internet standards to provide services for information transfer, analytics, applications and communications. So, we have seen the birth of new concepts such as *Internet of Things*, that [1,2], define as: (1) *‘Things’ are active participants in business, information and social process where they are enabled to interact and communicate among themselves and with the environment by exchanging data and information sensed about the environment, while reacting autonomously to real/physical world events and influencing it by running processes that trigger actions and create services with or without direct human intervention*, and (2) *Interconnection of sensing and actuating devices providing the ability to share information across platforms through a unified framework, developing a common operating picture for enabling innovative applications. This is achieved by seamless large scale sensing, data analytics and information representation using cutting edge ubiquitous sensing and cloud computing.*

These definitions promote the development of novel tools for the agile, transparent, and automated composition of objects in heterogeneous environments, dominated by a high dynamism. Dynamism involves access to resources and services that can not be guaranteed over time, causing the user to perceive the “Net” as not dependable. Being inspired by biological systems, an immune system is a form of highly dynamic system by its nature. In particular, an immune system is composed of loosely-coupled elements, which can interact with each other by exchanging asynchronous messages, providing a high capability to composition. Moreover, it can react to unforeseen events, e.g., intrusions, and it has the ability to adapt to context changes and provide the correct behaviour, making the system dependable and responsive.

In this paper, we propose a biological-inspired preliminary approach that leverages the peculiarities of immune systems so to allow the composition of software and services in a dynamic and reliable way, in order to provide a tool able to manage the **combination** and **selection** of software and services in the context of IoT. This composition is produced in order to satisfy both functional and non functional requirements, such as dependability. Moreover, given the need to write data centric applications (e.g. “Big Data”), of which the Internet of things is large producer, and the need to have applications with the ability to scale horizontally and remaining dependable, however, it has suggested to adopt the use of paradigm functional programming based because, in our view, offers effective technique for the challenges of our context, see Sect. 4.

The paper is organized as follows: in the Sect. 2, we provide an overview about immune systems by also discussing their main elements. In Sect. 3, we introduce basic concepts about software and services composition, by also discussing the new challenges to be faced in the Internet of Things context. In Sect. 4, we present

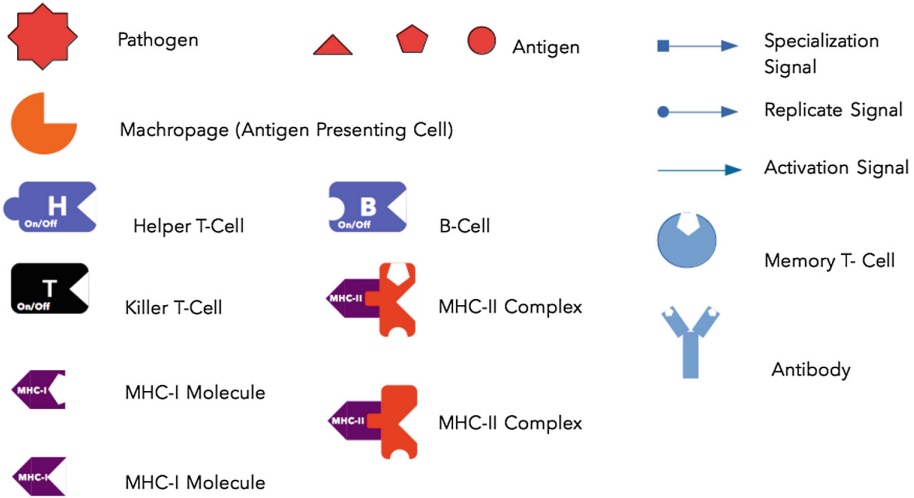


Fig. 1. Immune system components

our biological-inspired approach for the dynamic and dependable composition of software and services in the Internet of Things. We provide a preliminary implementation of the proposed approach that makes use of functional programming. Section 5 concludes and discusses future research directions we should undertake towards the realization of our preliminary proposal.

## 2 Immune System Aspects

An immune system [3] is a particular kind of biological system that is self-protecting against diseases. It is made of biological structures and processes within an organism. The minimum biological structure within an immune system is the cell, which in turn is made of molecules.

A key feature of an immune system is the ability to distinguish between (i) non-infectious structures, which must be preserved since they do not represent a disease, and (ii) infectious structures, i.e., *pathogens*, which must to be removed since they result in injuries to the organism the immune system belongs to.

But above all, its action immunizing, is performed through the selection and composition of components better suited to deal with the threat.

By referring to [4], the main elements of an immune system can be summarized as follows:

- **Lymphocytes.** They are the cells of an immune system and, for the purposes of this paper, it is enough to distinguish between *T Cells* and *B Cells*:
  - **T Cells.** They can be of two kinds, *T Helper* and *T Killer*. The former are cells responsible for preventing infections by managing and strengthening the immune responses enabled by the recognition of antigens. The

latter are cells able to destroy certain tumor cells, viral-infected cells, and parasites. Furthermore, they are responsible for down-regulating immune responses, when needed.

- **B Cells.** They are responsible for producing *antibodies* in response to foreign proteins of bacteria, viruses, and tumor cells. B cells are continuously produced in the bone marrow. When the B cell receptor, on the surface of the cell, matches the detected antigens present in the body, the B cell proliferates and secretes a free form of those receptors (antibodies) with identical binding sites as the ones on the original cell surface. The interesting aspect is that the different variants are generated in a seemingly random fashion. Versions that are not biologically significant or useless are automatically deleted.

Both B Cells and T Cells carry receptor molecules that make them able to recognize specific pathogens. In particular, T Killers recognize pathogens only after antigens have been processed and presented in combination with a *Major Histocompatibility Complex* (MHC) molecule. In contrast, B Cells recognize pathogens without any need for antigen processing.

- **Macrophages.** They are important in the regulation of immune responses. They are often referred to as *Antigen-Presenting Cells* (APC) because they pick up and ingest foreign materials and present these antigens to other cells of the immune system such as T Cells and B Cells. This is one of the important first steps in the initiation of an immune response.
- **Memory Cells.** When B Cell and T Cell are activated and begin to replicate, some cells belonging to their progeny become long-lived *Memory Cells*. The role of Memory Cells is to build an immunological memory that makes the immune system stronger in being self-protecting to future infections/attacks. In particular, a Memory Cell remembers already recognized antigens and lead to a stronger immune response when these antigens are recognized again.
- **Immune response.** An immune response to foreign antigens requires the presence of APC in combination with B Cells or T Cells. When an APC presents an antigen to a B Cell, the B Cell produces antibodies that specifically bind to that antigen in order to kill/destroy it. If the APC presents an antigen to a T Cell, the T Cell becomes active. Active T Cells essentially proliferate and kill target cells that specifically express the antigen presented by the APC. The production of antibodies and the activity of T Killers are highly regulated by T Helpers. They send *signals* to T Killers in order to regulate their activation, proliferation (replication) and efficiency (specialization).

Antibodies are protein complexes with a modular structure which share a basic structure which, in turn, show considerable variability in specific regions capable of binding to structurally complementary particles as antigens. Figure 2 represents the antibodies structure. The variable part, i.e., the V-shaped one, forms the binding site for antigens.

The peculiarity of antibodies, as crucial entities of immune systems, is the specificity with which they are able to recognize antigens. They are able, in fact, to distinguish between protein fragments which differ even for a single amino



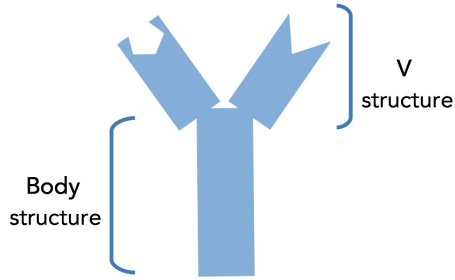


Fig. 2. Antibody example

acid, but also to bind more different antigens. The specificity, however, comes from the ability to create always different antibodies which can bind different antigens capable of building a repertoire of antibodies extremely broad.

Following the description above, Fig. 1 shows the main elements of an immune system as constructs of a simple graphical notation that we use in Sect. 2.1 for immune system modeling purposes. The figure shows also the messages (signals) that the system elements can exchange.

## 2.1 Immune System Scenarios

By leveraging the graphical modeling notation introduced above, in this section, we briefly describe two scenarios in the immune systems domain, which are representative for adaptive systems in general. The scenarios provide the reader with a high-level description of the interactions that happen among the elements of an immune system during an immune response.

**Scenario 0.** Figure 3(a) shows a scenario where the elements of an immune system are in an *inactive* state, marked with the *off* label. In particular, two APC engulf a virus or bacteria: each APC decomposes the pathogen (virus or bacteria) and exposes on its surface a piece of the pathogen, i.e., an antigen (see the triangle and pentagon in the figure). Metaphorically, we can think of this as a setup phase of a computer system, where each system’s component is in an idle state and the antigen is a “perturbation” that comes from the outside or even by the system itself. In our context, we can see this perturbation as new system behavior or goal to achieve.

**Scenario 1.** In this scenario, a B Cell becomes active after that it caught a known antigen and the T Helper close to it became active. In particular, we note that B-cell produces antibodies that exhibit an area of contact (V), complementary to the antigen, which is then captured and subsequently engulfed.

These simple, yet representative scenarios, lead us to observe that, as a particular kind of adaptive and dependable system in a specific domain:

- an immune system is composed of loosely-coupled elements;

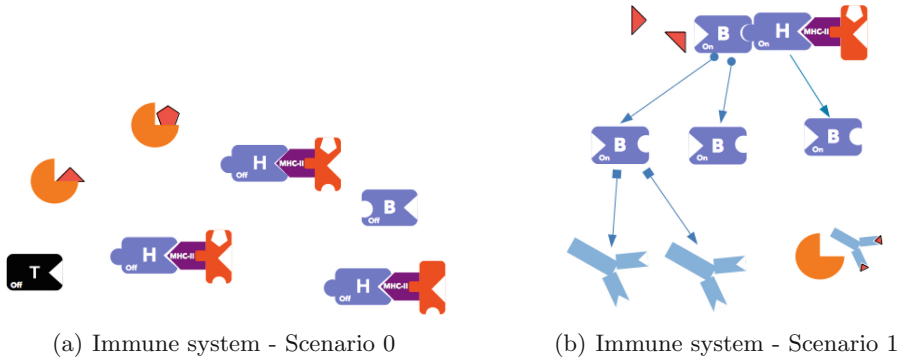


Fig. 3. Scenario example

- the elements of an immune system can interact with each other by exchanging asynchronous messages, providing a high capability to composition;
- an immune system can react to unforeseen events, e.g., intrusions;
- the elements of an immune system have the ability to adapt to context changes and provide the correct behaviour, making the system dependable and responsive.

### 3 Software and Services Composition

In the previous section it was emphasized that the immune response is based on the ability of its components to respond efficiently and effectively to external attacks or malfunctioning, through their composition.

In the traditional context of services composition, an implicit assumption is that services are running on an *enterprise service bus* (ESB) [5]. This proprietary product hide within itself the complexity of the coordination of the different components. Moreover service provider hosts a network accessible software module (an implementation of a given service). The service provider defines a service description of the service and publishes it to a client (or service discovery agency) through which a service description is published and made discoverable. The service client (requestor) discovers a service (endpoint) and retrieves the service description directly from the service (through meta-data exchange) or from a registry or repository. Service provider and service client roles are logical constructs and a service may exhibit characteristics of both.

Actually, the emergence of new concepts, such as the Internet of Things (IoT), Machine to Machine (M2M), and People to Machine (P2M) caused by the growth of smart devices, allows us to think about the possibility of using all these *things* not only as data producers or consumers, but as services provider. Thus we can think about *internet of things as a cloud of services* software and services composition.

This led to the creation of *mobile providers* which will probably be very different from traditional ESBs; for example they may be context-aware services

or location based services, or, given the presence of several sensors which are now equipped mobile devices, providing information from the real world. Moreover, in the context of P2M, they may also act as intermediaries between the people, in order to turn them into *human provided services* [6].

Given the variability of the context, a big challenge is to *automate and efficiently generate a composition of services or software applications that meet exactly the expectations of the requester, which is dependable in time and distributed.*

In the next section we propose a preliminary approach based on immune system, in particular on the antibody component, for supporting dependable and dynamic composition of software and services in the context of the Internet of Things.

## 4 Dependable Composition of Software and Services in the IoT: A Programmatic Solution

With the popularity and continued growth of smart mobile devices, service providers contained in a composition may be mobile devices. They constantly move and may cause the crash of the composition when, for example, are located outside of the communication range, or one of the device goes offline. The selection of the services, therefore, must make the composition as much as possible *dependable*, so as to avoid frequent recompositions.

In order to achieve a dependable composition it is necessary to provide tools that can handle natively and independently one possible reconfiguration, but above all the ability to adapt to changes in context, while still keeping the goal of the composition. As said in Sect. 3, we can summarize the composition process in:

**Service description:** is key to service management. Service description enables both the customer and the service provider to know what to expect and not expect from a service. Clearly defined services enable customers to understand service offerings, including what each service does and does not include, eligibility, service limitations, cost, how to request services, and how to get help.

**Service classification:** once performed discovery of services, the service classification can support the service combination and service selection. For example, it allows to aggregate services discriminating against non-functional requirements, so that one can select the most appropriate service among those identified. It also helps to improve the performance and efficiency of the composition by decreasing the number of services.

**Service combination:** it is possible to identify two possible approaches: (1) *bottom-up*, which is synthesized in a composition of known services, defining an executable workflow, or (2) *top-down*, where a composer creates abstract non-executable workflow template, and then forwarding the template to the next phase of service selection.

Service Composition Concepts	Immune System Concepts
Service Selection	Memory Cell/Antibody
Service Combination	Immune Response
Service Classification	Macrophage
Service Description	Antigene

Fig. 4. Service and immune system mapping

Immune System Signal	Service Message
Specialization Signal	specializationMessage
Replication Signal	replicationMessage
Activation Signal	activationMessage

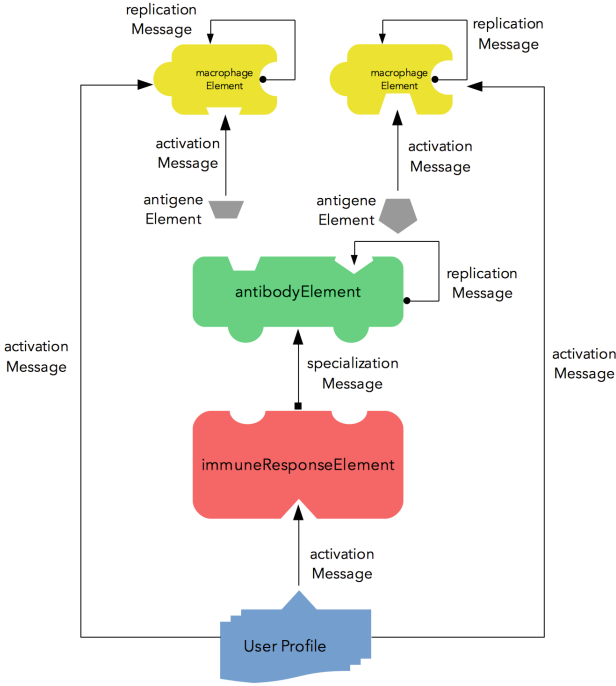
Fig. 5. Service message and immune system mapping

**Service selection:** since a workflow consisting of abstract activity, service selection selecting an appropriate service efficiently for each task. In fact, through the service classification, each activity may have several candidate services with the same functions, that however have, for example, different non-functional requirements.

Since the classification of Fig. 4, we propose a preliminary study in order to provide a tool able to manage efficiently and reliably the **combination** and **selection** of software and services in the context of IoT. To address this challenge, we propose an approach based on the immune system and Functional Programming (FP):

- the elements of the immune system are able to be composed between them in order to build complex structures that once activated have the ability to respond to external stimuli, for example antigens, and exhibit a countermove. This makes it an extremely adaptable, resilient and reliable.
- on the other hand, functional programming encourage the: (1) *compose function*, that favors building programs from the bottom up by composing functions together; (2) *immutability*, since functional programming favors pure functions, which can't mutate data, it also makes use of immutable data. Instead of modifying an existing data, a new one is efficiently created; (3) *transform, rather than mutate, data*, FP uses functions to transform immutable data. One data is put into the function, and a new immutable data structure comes out.

In our view, the services and the software components, are atomic and immutable data that are combined through mechanisms of composition, i.e. functions, in order to ensure the desired goal.



**Fig. 6.** Antibody architecture

In Fig. 6, we defined the elements constituting the software architecture of the process of Fig. 3(b). Given the highly heterogeneous environment that characterizes the Internet of Things, every actor is equipped with a specific *user profile*, which can be defined as: *a record of user-specific data that define the user’s working environment*.

This definition is deliberately general, to ensure the management of profiles that apply to both human and “things”, but at the same time *guide* the *classification* and *selection* of services. A further motivation to support the use of a user profile is given by the fact that together with the description of any goal to be achieved, also non-functional requirements can be specified, such as dependability, that for a mobile user is perceived as a fundamental requirement.

The architecture components were represented without *smooth surfaces*, with the intent to represent the ability of each of them to create links in a totally adaptive way, that allows the composition with services or software applications as much automated as possible. The invariant is given by the User Profile, which defines the semantic boundaries, within which to make the classification and the choice of components to be classified.

As introduced in the previous section, we can make some considerations about the formalism and computer science concepts that could support a theoretical implementation of the proposed architecture (Fig. 6). In particular, we consider

the concepts of *Partially Applied Functions* [7], where in a function with an parameters list, you can define a new function where you omit one or more parameter of the list.

Let a program  $p$ , and its input  $i$ , and we say that we want to split  $i$  in a known static part  $s$  and a dynamic (unknown)  $d$ . Then, given a function of specialization  $S$ , we specialize  $p$  with respect to  $s$ :

$$S(p, < s, - >) = p_s$$

In our context, the function  $S$  is the service to be provided to the user  $p$ , that uses as input parameters: the User Profile ( $s$ ) and available services at the time ( $d$  or  $-$ ).

This function is computable, and it can be implemented [8]. It is strongly linked to the concept of functional programming. This approach promotes modularity, and thus a natural ability to compose, promoting reuse, parallelization, generality, and automated reasoning. There are numerous programming languages that allow one to adopt the functional programming paradigm; one of these is the Scala language<sup>1</sup>.

For simplicity, we map some of the elements of the architecture of Fig. 6 to simple mathematical functions, please see Fig. 7(a) and (b). In particular, we have:

- $map^2$  is *immuneResponseElement*
- $s$  is *UserProfile*
- $fsf$  is *antibodyElement*
- $f1$  and  $f2$  is *antigenElement*

```
def fs[X](f:x =>x)(s: Seq[X]) = s map f
def f1(x: int) = x * 2
def f2(x: int) = x * x
def fsf[X](f:x =>x) = fs(f) _
val fsf1 = fsf(f1)
val fsf2 = fsf(f2)
```

(a) Partial Function Mapping

```
Output:
fsf1(List(0,1,2,3)) -> List(0,2,4,6)
fsf2(List(0,1,2,3)) -> List(0,1,4,9)
```

(b) Output

Fig. 7. Example

With this simple example, we have shown how the architecture in Fig. 6 can be mapped in a natural way in a programming language that implements natively functional programming, highlighting the compositional capabilities it offers.

<sup>1</sup> <http://www.scala-lang.org>.

<sup>2</sup> Scala Language Command - Map works by applying a function to each element in the list.

## 5 Summary and Conclusions

In this paper we presented a preliminary work with the aim to investigate the opportunity to derive from biological processes software programming concepts that can provide software and services composition solutions in the IoT context. Mimic the behavior of nature is not new in the field of computer science; classical example are neural network [9,10], the genetic algorithm [11], the same immune system [12,13], and more generally the biologically inspired computation [14], they have always evoked great interest. Often, they have characteristics as distributivity, adaptability, heterogeneity, dynamism, dependability, resilience: all new challenges that the advent of mobile devices presents every day.

The interesting aspect of the IT and biology world, it is the fact that some computing paradigms allow you to define in an elegant and formal way some biological processes, such as functional programming, which itself is well supported natively in some modern programming languages like the Scala language. The next steps involve a careful formalization of the proposed architecture (Fig. 6), using the Scala language and the definition of a concrete case study to evaluate the efficiency and fairness of the approach for the construction of systems through dynamic and dependable composition of software and services in the IoT.

**Acknowledgment.** This research work has been supported by the Ministry of Education, Universities and Research, prot. 2012E47TM2 (project IDEAS - Integrated Design and Evolution of Adaptive Systems), by the European Union's H2020 Programme under grant agreement number 644178 (project CHOReVOLUTION - Automated Synthesis of Dynamic and Secured Choreographies for the Future Internet), and by the Ministry of Economy and Finance, Cipe resolution n. 135/2012 (project INCIPICT - INnovating CIty Planning through Information and Communication Technologies).

## References

1. Sundmaeker, H., Guillemin, P., Friess, P., Woelfflé, S. (eds.): Vision and Challenges for Realising the Internet of Things. Publications Office of the European Union, Luxembourg (2010)
2. Gubbi, J., Buyya, R., Marusic, S., Palaniswami, M.: Internet of things (iot): a vision, architectural elements, and future directions. *Future Gener. Comput. Syst.* **29**(7), 1645–1660 (2013). Including Special sections: Cyber-enabled Distributed Computing for Ubiquitous Cloud and Network Services Cloud Computing and Scientific Applications – Big Data, Scalable Analytics, and Beyond
3. Hofmeyr, S.A.: An interpretative introduction to the immune system. In: *Design Principles for the Immune System and Other Distributed Autonomous Systems*, pp. 3–26. Oxford University Press (2000)
4. Janeway Jr., C.A., Travers, P., Walport, M., et al.: *Immunobiology: The Immune System in Health and Disease*. 5th edn. Garland Science, USA (2013)
5. Chappell, D.A.: *Enterprise Service Bus - Theory in Practice*. O'Reilly, Sebastopol (2004)

6. Li, Q., Liu, A., Liu, H., Lin, B., Huang, L., Gu, N.: Web services provision: solutions, challenges and opportunities (invited paper). In: Proceedings of the 3rd International Conference on Ubiquitous Information Management and Communication, ICUIMC 2009, pp. 80–87. ACM, New York (2009)
7. Jones, N.D., Gomard, C.K., Sestoft, P.: Partial Evaluation and Automatic Program Generation. Prentice-Hall Inc, Upper Saddle River (1993)
8. Kleene, S.C.: Introduction to Metamathematics. Ishi Press International, New York (2009)
9. Tahmasebi, P., Hezarkhani, A.: A hybrid neural networks-fuzzy logic-genetic algorithm for grade estimation. *Comput. Geosci.* **42**, 18–27 (2012). doi:[10.1016/j.cageo.2012.02.004](https://doi.org/10.1016/j.cageo.2012.02.004)
10. Haykin, S.: Neural Networks: A Comprehensive Foundation, 2nd edn. Prentice Hall PTR, Upper Saddle River (1998)
11. Banzhaf, W., Francone, F.D., Keller, R.E., Nordin, P.: Genetic Programming: An Introduction: on the Automatic Evolution of Computer Programs and Its Applications. Morgan Kaufmann Publishers Inc., San Francisco (1998)
12. Kephart, J.O.: A biologically inspired immune system for computers. In: Artificial Life IV: Proceedings of the Fourth International Workshop on the Synthesis and Simulation of Living Systems, pp. 130–139. MIT Press (1994)
13. de Castro, L.N.: Artificial Immune Systems: A New Computational Intelligence Approach. Springer, London (2002)
14. Parsons, S.: Imitation of life: how biology is inspiring computing by Nancy Forbes, MIT Press, 176 pp., \$8.95, ISBN 0-262-06241-0. *Knowl. Eng. Rev.* **21**, 95–95 (2006)



## Author Index

- Adams, Mark 53
- Barbuti, Roberto 176
- Baresi, Luciano 215
- Blech, Jan Olaf 43
- Bompadre, Alessandro 176
- Bove, Pasquale 176
- Brayton, Robert 281
- Castagnetti, Giovanni 281
- Cerone, Antonio 139, 145, 198
- Ciancia, Vincenzo 297
- Cini, Chiara 192
- Cintia, Paolo 108
- De Nicola, Rocco 145
- Di Noia, Tommaso 228
- Di Salle, Amleto 240, 312
- Eckert, Jürgen 22
- Erbatur, Serdar 75
- Farchi, Eitan 35, 68
- Gallo, Francesco 240, 312
- Galpin, Vashti 161
- German, Reinhard 22
- Gnesi, Stefania 266
- Grilletti, Gianluca 297
- Grossi, Valerio 93
- Guidotti, Riccardo 108
- Guinea, Sam 215
- Herrmann, Peter 43
- Hofmann, Martin 75
- Homm, Daniel 22
- Janssen, Marijn 124
- Kherrazi, R. 11
- Klievink, Bram 124
- Latella, Diego 266, 297
- Loreti, Michele 297
- Massink, Mieke 266, 297
- Meinke, Karl 3
- Merelli, Emanuela 192
- Milazzo, Paolo 176, 198
- Mishchenko, Alan 281
- Mongiello, Marina 228
- Monreale, Anna 93
- Nakagawa, Hiroyuki 253
- Nanni, Mirco 93
- Nieto Coria, Cesar A. 192
- Noroozi, N. 11
- Nycander, Peter 3
- Ogawa, Kento 253
- Pardini, Giovanni 176
- Pedreschi, Dino 93
- Perucci, Alexander 240, 312
- Piccolo, Matteo 281
- Quattrocchi, Giovanni 215
- Scarcella, Giuseppe 192
- Setiawan, Suryana 198
- Shams, Farshad 145
- Straccia, Umberto 228
- ter Beek, Maurice H. 266
- Tesei, Luca 192
- Tsuchiya, Tatsuhiro 253
- Turini, Franco 93
- van der Meer, A.P. 11
- van Engelenburg, Sélinde 124
- Villa, Tiziano 281
- Wierda, A. 11
- Yevtushenko, Nina 281
- Zamansky, Anna 35, 68