

ENASE 2016

11th International Conference on Evaluation of
Novel Approaches to Software Engineering

PROCEEDINGS

Rome, Italy

27 - 28 April, 2016

► www.enase.org

SPONSORED BY:



LOGISTICS PARTNER:



PAPERS AVAILABLE AT:



ENASE 2016

Proceedings of the
11th International Conference on
Evaluation of Novel Software Approaches to
Software Engineering

Rome - Italy

April 27 - 28, 2016

Sponsored by

INSTICC - Institute for Systems and Technologies of Information, Control and Communication

In Cooperation with

AFEI - Association for Enterprise Information

ACM SIGAPP - ACM Special Interest Group on Applied Computing

ACM SIGSOFT - ACM Special Interest Group on Software Engineering

Copyright © 2016 by SCITEPRESS – Science and Technology Publications, Lda.
All rights reserved

Edited by Leszek Maciaszek and Joaquim Filipe

Printed in Portugal
ISBN: 978-989-758-189-2
Depósito Legal: 406877/16

<http://www.enase.org>
enase.secretariat@insticc.org

BRIEF CONTENTS

| | |
|--|------|
| INVITED SPEAKERS | IV |
| SPECIAL SESSIONS CHAIRS | IV |
| ORGANIZING COMMITTEES | V |
| PROGRAM COMMITTEE | VI |
| AUXILIARY REVIEWERS | VIII |
| SPECIAL SESSIONS PROGRAM COMMITTEE | VIII |
| SELECTED PAPERS BOOK | IX |
| FOREWORD | XI |
| CONTENTS | XIII |

INVITED SPEAKERS

Sergio Gusmeroli
TXT e-solutions SPA
Italy

Wil Van Der Aalst
Technische Universiteit Eindhoven
Netherlands

Ernesto Damiani
EBTIC-KUSTAR
United Arab Emirates

SPECIAL SESSIONS CHAIRS

SPECIAL SESSION ON MODEL-DRIVEN INNOVATIONS FOR SOFTWARE ENGINEERING

Gundars Alksnis, Riga Technical University, Latvia
Janis Osis, Riga Technical University, Latvia

SPECIAL SESSION ON COLLABORATIVE ASPECTS OF FORMAL METHODS

Anna Zamansky, University of Haifa, Israel
Maria Spichkova, RMIT University, Australia

ORGANIZING COMMITTEES

CONFERENCE CHAIR

Joaquim Filipe, Polytechnic Institute of Setúbal / INSTICC, Portugal

PROGRAM CHAIR

Leszek Maciaszek, Wroclaw University of Economics, Poland and Macquarie University, Sydney,
Australia

SECRETARIAT

Andreia Pereira, INSTICC, Portugal

Ana Ramalho, INSTICC, Portugal

GRAPHICS PRODUCTION AND WEBDESIGNER

André Lista, INSTICC, Portugal

Mara Silva, INSTICC, Portugal

WEBMASTER

Susana Rodrigues, INSTICC, Portugal

PROGRAM COMMITTEE

Marco Aiello, University of Groningen, Netherlands

Frederic Andres, Research Organization of Information and Systems, Japan

Guglielmo de Angelis, CNR - IASI, Italy

Oscar Avila, Universidad de los Andes, Colombia

Paul Bailes, The University of Queensland, Australia

Maria Bielikova, Slovak University of Technology in Bratislava, Slovak Republic

Jan Olaf Blech, RMIT University, Australia

Ivo Blohm, University of St. Gallen, Switzerland

Rem Collier, University College Dublin, Ireland

Rebeca Cortazar, University of Deusto, Spain

Massimo Cossentino, National Research Council, Italy

Bernard Coulette, Université Toulouse Jean Jaurès, France

Patrick Cousot, New York University, United States

Mariangiola Dezani, Università di Torino, Italy

Tadashi Dohi, Hiroshima University, Japan

Schahram Dustdar, Vienna University of Technology, Austria

Angelina Espinoza, Universidad Autónoma Metropolitana, Iztapalapa (UAM-I), Spain

Vladimir Estivill-Castro, Griffith University, Australia

Anna Rita Fasolino, Università degli Studi di Napoli Federico II, Italy

Maria João Ferreira, Universidade Portucalense, Portugal

Martin Gaedke, Distributed and Self-organizing Computer Systems Group, Chemnitz University of Technology, Germany

Stéphane Galland, Université de Technologie de Belfort Montbéliard, France

Frédéric Gervais, Université Paris-Est, LACL, France

Paolo Giorgini, University of Trento, Italy

Cesar Gonzalez-Perez, Institute of Heritage Sciences (Incipit), Spanish National Research Council (CSIC), Spain

Jose María Gutierrez, Universidad De Alcalá, Spain

Mahmoud EL Hamlaoui, University of Mohammed V Rabat/University of Toulouse Jean Jaurès, France

Brian Henderson-Sellers, University of Technology, Sydney, Australia

Rene Hexel, Griffith University, Australia

Benjamin Hirsch, EBTIC / Khalifa University, United Arab Emirates

Robert Hirschfeld, Hasso-Plattner-Institut, Germany

Zbigniew Huzar, Wroclaw University of Technology, Poland

Fuyuki Ishikawa, National Institute of Informatics, Japan

Mirjana Ivanovic, Faculty of Sciences, University of Novi Sad, Serbia

Stefan Jablonski, University of Bayreuth, Germany

Slinger Jansen, Utrecht University, Netherlands

Monika Kaczmarek, University of Duisburg-Essen, Germany

Georgia Kapitsaki, University of Cyprus, Cyprus

Siau-cheng Khoo, National University of Singapore, Singapore

Diana Kirk, Consultant, New Zealand

Paul Klint, Centrum Wiskunde & Informatica, Netherlands

Piotr Kosiuczenko, WAT, Poland

Nectarios Koziris, National Technical University of Athens, Greece

Rosa Lanzilotti, University of Bari, Italy

Robert S. Laramee, Swansea University, United Kingdom

Bixin Li, Southeast University, China

Huai Liu, RMIT University, Australia

André Ludwig, University of Leipzig, Germany

Ivan Lukovic, University of Novi Sad, Faculty of Technical Sciences, Serbia

Leszek Maciaszek, Wroclaw University of Economics, Poland and Macquarie University, Sydney, Australia

Lech Madeyski, Wroclaw University of Technology, Poland

Nazim H. Madhavji, University of Western Ontario, Canada

Michele Marchesi, University of Cagliari, Italy

Michael Mrissa, University of Lyon, France

Peter Axel Nielsen, Aalborg University, Denmark

Andrzej Niesler, Wroclaw University of Economics, Poland

Andreas Oberweis, Karlsruhe Institute of Technology (KIT), Germany

Janis Osis, Riga Technical University, Latvia

Mourad Oussalah, Laboratoire Lina Cnrs Fre 2729, University of Nantes, France

Justyna Petke, University College London, United Kingdom

Naveen Prakash, MRCE, India

Adam Przybylek, Gdansk University of Technology, Poland

Elke Pulvermueller, University of Osnabrueck, Germany

Lukasz Radlinski, West Pomeranian University of Technology, Poland

Camille Salinesi, University Paris 1 - Pantheon Sorbonne, France

Walt Scacchi, University of California Irvine, United States

Markus Schatten, University of Zagreb, Croatia

Josep Silva, Universidad Polit cnica de Valencia, Spain

Michal Smialek, Warsaw University of Technology, Poland

Ioana Sora, Politehnica University of Timisoara, Romania

Andreas Speck, Christian-Albrechts-University Kiel, Germany

Maria Spichkova, RMIT University, Australia

Witold Staniszkis, Rodan Development, Poland

Miroslaw Staron, University of Gothenburg, Sweden

Armando Stellato, University of Rome, Tor Vergata, Italy

Gunnar Stevens, University of Applied Science Bonn-Rhein-Sieg, Germany

Jakub Swacha, University of Szczecin, Poland

Rainer Unland, University of Duisburg-Essen, Germany

Olegas Vasilecas, Vilnius Gediminas Technical University, Lithuania

Stefan Wagner, Universit t Stuttgart, Germany

Krzysztof Wecel, Poznan University of Economics, Poland

Bernhard Westfechtel, University of Bayreuth, Germany

Jack C. Wileden, University of Massachusetts, United States

Martin Wirsing, Ludwig-Maximilians-Universit t M nchen, Germany

Igor Wojnicki, AGH University of Science and Technology, Poland

Michalis Xenos, Hellenic Open University, Greece

Kang Zhang, The University of Texas at Dallas, United States

Alfred Zimmermann, Reutlingen University, Germany

AUXILIARY REVIEWERS

Lorenzo Bettini, Università di Torino, Italy

Thomas Buchmann, University of Bayreuth, Germany

Mario Coppo, Università di Torino, Italy

Mohamad Gharib, University of Trento, Italy

Claudia Di Napoli, C.N.R., Italy

Jan-Peter Ostberg, Universität Stuttgart, Germany

Laure Petrucci, LIPN, CNRS UMR 7030, Université Paris 13, France

Elvinia Riccobene, University of Milan, Italy

Luca Sabatucci, National Research Council - Italy, Italy

Sven Verdoolaege, Polly Labs, Belgium

Fabian Wiedemann, Technische Universität Chemnitz, Germany

SPECIAL SESSIONS PROGRAM COMMITTEE

SPECIAL SESSION ON MODEL-DRIVEN INNOVATIONS FOR SOFTWARE ENGINEERING

Gundars Alksnis, Riga Technical University, Latvia

Vicente Garcia Diaz, Oviedo University, Spain

Liliana Dobrica, University Politehnica of Bucharest, Romania

Uldis Doninš, Riga Stradins University, Latvia

Petr Hnetynka, Charles University, Czech Republic

Zbigniew Huzar, Wroclaw University of Technology, Poland

Lajos Kollár, University of Debrecen, Hungary

Erika Nazaruka, Riga Technical University, Latvia

Janis Osis, Riga Technical University, Latvia

Raman Ramsin, Sharif University of Technology, Iran, Islamic Republic of

Janis Silins, Mapon SIA, Latvia

Fritz Solms, University of Pretoria, South Africa

Maria Spichkova, RMIT University, Australia

Artis Teilans, Rezekne University of Applied Sciences, Latvia

Edward Rolando Núñez Valdez, Charles III University of Madrid, Spain

SPECIAL SESSION ON COLLABORATIVE ASPECTS OF FORMAL METHODS

Gundars Alksnis, Riga Technical University, Latvia

Daniel Berry, University of Waterloo, Canada

Stefanie Betz, Karlsruhe Institute of Technology, Germany

Jan Olaf Blech, RMIT University, Australia

Ruzanna Chitchyan, University of Leicester, United Kingdom

Irit Hadar, University of Haifa, Israel

James Harland, RMIT University, Australia

Alan Hartman, University of Haifa, Israel

Peter Herrmann, NTNU, Norway

Ivan Jureta, University of Namur, Belgium

Janis Osis, Riga Technical University, Latvia

Daniel Ratiu, Siemens AG, Germany

Guillermo Rodriguez-Navas, Mälardalen University, Sweden

Bernhard Rumpe, RWTH Aachen University, Germany

Thomas Santen, Microsoft, Germany

Natalia Sidorova, Eindhoven University of Technology, Netherlands

Maria Spichkova, RMIT University, Australia

Rachel Tzoref-Brill, IBM Research, Israel

Colin Venters, University of Huddersfield, United Kingdom

Matthias Weidlich, Humboldt University of Berlin, Germany

Anna Zamansky, University of Haifa, Israel

Marc van Zee, University of Luxembourg, Luxembourg

SELECTED PAPERS BOOK

A number of selected papers presented at ENASE 2016 will be published by Springer in a CCIS Series book. This selection will be done by the Conference Chair and Program Chair, among the papers actually presented at the conference, based on a rigorous review by the ENASE 2016 Program Committee members.

FOREWORD

This book contains the proceedings of the 11th International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE 2016). This conference is sponsored by the Institute for Systems and Technologies of Information, Control and Communication (INSTICC) and held in cooperation with the ACM Special Interest Group on Applied Computing (SIGAPP), ACM Special Interest Group on Software Engineering (ACM SIGSOFT) and the Association for Enterprise Information (AFEI).

The mission of ENASE is to be a prime international forum to discuss and publish research findings and IT industry experiences with relation to the evaluation of novel approaches to software engineering. The conference acknowledges necessary changes in systems and software thinking due to contemporary shifts of computing paradigm to e-services, cloud computing, mobile connectivity, business processes, and societal participation. By comparing novel approaches with established traditional practices and by evaluating them against systems and software quality criteria, ENASE conferences advance knowledge and research in software engineering, including and emphasizing service-oriented, business-process driven, and ubiquitous mobile computing. ENASE aims at identifying most hopeful trends and proposing new directions for consideration by researchers and practitioners involved in large-scale systems and software development, integration, deployment, delivery, maintenance and evolution.

The meeting is complemented with the Special Session on Model-Driven Innovations for Software Engineering (MDI4SE) and the Special Session on Collaborative Aspects of Formal Methods (COLAFORM).

ENASE received 79 paper submissions, including the special sessions, from 28 countries, 13% were accepted as full papers which shows the intention of preserving a high quality forum for the next editions of this conference.

The conference program includes a panel and three invited talks delivered by internationally distinguished speakers, namely: Sergio Gusmeroli (TXT e-solutions SPA, Italy), Wil Van Der Aalst (Technische Universiteit Eindhoven, Netherlands) and Ernesto Damiani (EBTIC-KUSTAR, United Arab Emirates).

To recognize the best submissions and the best student contributions, several awards based on the combined marks of paper reviewing, as assessed by the Program Committee, and the quality of the presentation, as assessed by chairs at the conference venue, are conferred at the closing session of the conference.

We would like to express our thanks to all participants. First of all, to the authors, whose quality work is the essence of this conference. Next, we thank all the members of the program committee and the auxiliary reviewers for their diligence and expert reviewing. We must deeply thank the invited speakers for their excellent contribution in sharing their knowledge and vision. Finally, special thanks to all the members of the INSTICC team whose collaboration was fundamental for the success of this conference.

We wish you all an inspiring conference and an unforgettable stay at Rome, Italy. We look forward to having additional research results presented at the next edition of ENASE, details of which are available at <http://www.enase.org/>.

Leszek Maciaszek

Wroclaw University of Economics, Poland and Macquarie University, Sydney, Australia

Joaquim Filipe

Polytechnic Institute of Setúbal / INSTICC, Portugal

CONTENTS

INVITED SPEAKERS

KEYNOTE SPEAKERS

| | |
|--|---|
| The Sensing Enterprise - Enterprise Information Systems in the Internet of Things <i>Sergio Gusmeroli</i> | 5 |
| Green Data Science - Using Big Data in an “Environmentally Friendly” Manner <i>Wil Van Der Aalst</i> | 7 |
| Towards Model-Driven Big-Data-as-a-Service <i>Ernesto Damiani</i> | 9 |

SERVICE SCIENCE AND BUSINESS INFORMATION SYSTEMS

FULL PAPERS

| | |
|---|----|
| Are Suggestions of Coupled File Changes Interesting? <i>Jasmin Ramadani and Stefan Wagner</i> | 15 |
| Cloud Computing Adoption, Cost-benefit Relationship and Strategies for Selecting Providers: A Systematic Review <i>Antonio Carlos Marcelino de Paula and Glauco de Figueiredo Carneiro</i> | 27 |

SHORT PAPERS

| | |
|---|----|
| Semi-automatic Generation of OrBAC Security Rules for Cooperative Organizations using Model-Driven Engineering <i>Irvin Dongo and Vanea Chiprianov</i> | 43 |
| Extended Change Identification System <i>Parimala N. and Vinay Gautam</i> | 51 |

MOBILE SOFTWARE AND SYSTEMS

SHORT PAPER

| | |
|---|----|
| Preventing Hospital Acquired Infections through a Workflow-based Cyber-physical System <i>Maria Iuliana Bocicor, Arthur-Jozsef Molnar and Cristian Taslitchi</i> | 63 |
|---|----|

SOFTWARE ENGINEERING

FULL PAPERS

- Breaking the Boundaries of Meta Models and Preventing Information Loss in Model-Driven Software Product Lines 73
Thomas Buchmann and Felix Schwägerl
- Extending UML/MARTE-SAM for Integrating Adaptation Mechanisms in Scheduling View 84
Mohamed Naija and Samir Ben Ahmed
- A Methodology for Model-based Development and Safety Analysis of Transport Systems 91
Simon Hordvik, Kristoffer Øseth, Jan Olaf Blech and Peter Herrmann
- RA2DL-Pool: New Useful Solution to Handle Security of Reconfigurable Embedded Systems 102
Farid Adaili, Olfa Mosbahi, Mohamed Khargui and Samia Bouzefrane

SHORT PAPERS

- Towards Semantical DSMLs for Complex or Cyber-physical Systems 115
Blazo Nastov, Vincent Chapurlat, Christophe Dony and François Pfister
- Evolution Taxonomy for Software Architecture Evolution 124
Noureddine Gasmallah, Abdelkrim Amirat and Mourad Oussalah
- Systematic Mapping Study of Ensemble Effort Estimation 132
Ali Idri, Mohamed Hosni and Alain Abran
- A Novel R-UML-B Approach for Modeling and Code Generation of Reconfigurable Control Systems 140
Raja Oueslati, Olfa Mosbahi, Mohamed Khargui and Samir Ben Ahmed
- Constraints-based URDAD Model Verification 148
Fritz Solms, Priscilla Naa Dedei Hammond and Linda Marshall
- Evaluating A Novel Agile Requirements Engineering Method: A Case Study 156
Tanel Tenso, Alex Norta and Irina Vorontsova
- An Empirical Study of Two Software Product Line Tools 164
Kattiana Constantino, Juliana Alves Pereira, Juliana Padilha, Priscilla Vasconcelos and Eduardo Figueiredo
- Source and Test Code Size Prediction - A Comparison between Use Case Metrics and Objective Class Points 172
Mourad Badri, Linda Badri and William Flageol
- Self-Protection Mechanisms for Web Applications - A Case Study 181
Claudia Raibulet, Alberto Loporati and Andrea Metelli
- AWSM - Agile Web Migration for SMEs 189
Sebastian Heil and Martin Gaedke
- Validation of Loop Parallelization and Loop Vectorization Transformations 195
Sudakshina Dutta, Dipankar Sarkar, Arvind Rawat and Kulwant Singh
- Multi-variant Model Transformations — A Problem Statement 203
Felix Schwägerl, Thomas Buchmann and Bernhard Westfechtel

| | |
|---|-----|
| Automatic Refactoring of Component-based Software by Detecting and Eliminating Bad Smells - A Search-based Approach <i>Salim Kebir, Isabelle Borne and Djamel Meslati</i> | 210 |
| CURA: Complex-system Unified Reference Architecture - Position Paper: A Practitioner View <i>Ethan Hadar and Irit Hadar</i> | 216 |
| Evaluating the Evaluators - An Analysis of Cognitive Effectiveness Improvement Efforts for Visual Notations <i>Dirk van der Linden and Irit Hadar</i> | 222 |
| A Human-centred Framework for Combinatorial Test Design <i>Maria Spichkova and Anna Zamansky</i> | 228 |
| A Research Agenda on Visualizations in Information Systems Engineering <i>Jens Gulden, Dirk van der Linden and Banu Aysolmaz</i> | 234 |
| Towards an Engineering Process for Developing Accessible Software in Small Software Enterprises <i>Sandra Sanchez-Gordon, Mary-Luz Sánchez-Gordón and Sergio Luján-Mora</i> | 241 |
| Developing Green and Sustainable Software using Agile Methods in Global Software Development: Risk Factors for Vendors <i>Nasir Rashid and Siffat Ullah Khan</i> | 247 |
| An Enhanced Equivalence Checking Method to Handle Bugs in Programs with Recurrences <i>Sudakshina Dutta and Dipankar Sarkar</i> | 254 |
| Zoetic Data and their Generators <i>Paul Bailes and Colin Kemp</i> | 260 |
| Engineering Real-Time Communication Through Time-triggered Subsumption - Towards Flexibility with INCUS and LLFSMs <i>David Chen, René Hexel and Fawad Riasat Raja</i> | 272 |
| On Source Code Optimization for Interpreted Languages using State Models <i>Jorge López, Natalia Kushik and Nina Yevtushenko</i> | 282 |
| Managing Usability and Reliability Aspects in Cloud Computing <i>Maria Spichkova, Heinz W. Schmidt, Ian E. Thomas, Iman I. Yusuf, Steve Androulakis and Grischa R. Meyer</i> | 288 |
| End to End Specification based Test Generation of Web Applications <i>Khusbu Bubna</i> | 296 |
| An Appropriate Method Ranking Approach for Localizing Bugs using Minimized Search Space <i>Shanto Rahman and Kazi Sakib</i> | 303 |

SPECIAL SESSION ON MODEL-DRIVEN INNOVATIONS FOR SOFTWARE ENGINEERING

FULL PAPERS

| | |
|---|-----|
| Topological Functioning Model for Software Development within MDA (Survey) <i>Arturs Solomencevs</i> | 315 |
| The Validation Possibility of Topological Functioning Model using the Cameo Simulation Toolkit <i>Viktorija Ovchinnikova and Erika Nazaruka</i> | 327 |
| Comparison of Topological Functioning Model for Software Engineering with BPMN Approach in the Context of Model Driven Architecture <i>Janis Osis and Arturs Solomencevs</i> | 337 |
| Verification of BPMN Model Functional Completeness by using the Topological Functioning Model <i>Erika Nazaruka, Viktorija Ovchinnikova, Gundars Alksnis and Uldis Sukovskis</i> | 349 |

SPECIAL SESSION ON COLLABORATIVE ASPECTS OF FORMAL METHODS

SHORT PAPERS

| | |
|--|-----|
| Formal Behavioural Models to Facilitate Distributed Development and Commissioning in Industrial Automation <i>James Harland, Jan Olaf Blech, Ian Peake and Luke Trodd</i> | 363 |
| Teaching of Formal Methods for Software Engineering <i>Maria Spichkova and Anna Zamansky</i> | 370 |
| Reasoning about Inconsistency in RE - Separating the Wheat from the Chaff <i>Anna Zamansky, Irit Hadar and Daniel M. Berry</i> | 377 |
| Collaborative Model-based Development of a Remote Train Monitoring System <i>Peter Herrmann, Alexander Svae, Henrik Heggelund Svendsen and Jan Olaf Blech</i> | 383 |
| Let's Make it Fun: Gamifying and Formalizing Code Review <i>Naomi Unkelos-Shpigel and Irit Hadar</i> | 391 |
| Formal Methods in Collaborative Projects <i>Anna Zamansky, Guillermo Rodriguez-Navas, Mark Adams and Maria Spichkova</i> | 396 |
| AUTHOR INDEX | 403 |

INVITED SPEAKERS

KEYNOTE SPEAKERS

Evolution Taxonomy for Software Architecture Evolution

Nouredine Gasmallah^{1,2,3}, Abdelkrim Amirat² and Mourad Oussalah³

¹*Department of Computer Science, University of Annaba, Sidi-Amar, 23000, Annaba, Algeria*

²*Department of Maths and Computer Science, University of Souk-Ahras, Rte d'Annaba, 41000, Souk-Ahras, Algeria*

³*Department of Computer Science, University of Nantes, 2 Rue de la Houssinière BP-92208, 44322, Nantes, France
{gasmallahedi, abdelkrim.amirat}@yahoo.com, mourad.oussalah@univ-nantes.fr*

Keywords: Software Architecture, Software Evolution, Evolution Taxonomy, Quality Criteria.

Abstract: Nowadays, architects are facing the challenge of proliferation of stakeholder requirements for preserving and ensuring the effectiveness of the software, by using software evolution as a key solution. Hence, in terms of landscaping evolution space there is a great need to define the thinking on which efforts to deal with this issue have been based. In this paper, we propose a framework for software architecture evolution taxonomy based on four structural dimensions. This framework could both position existing evolution models in the field and highlight gray areas for the future. Mapping over framework dimensions, a set of quality factors and an investigation including 67 studies are performed to assess the proposals. The results contain a number of relevant findings, including the need to improve software architecture evolution by accommodating predictable changes as well as promoting the emergence of operating mechanisms.

1 INTRODUCTION

Nowadays, there is no doubt that software development is facing a cumbersome process of handling or modeling the inevitable evolution within open systems. Software architecture is a combination of a set of architectural elements and their interconnections which are unified to satisfy design requirements. However, the architecture should adhere to the changes required by the various stockholders in order to avoid system erosion (Perry and Wolf, 1992) and to meet their different goals. Thereby, software architecture must evolve as a systematic result of increased concern about the environment (Jazayeri, 2005). Consequently, evolution should be planned in the early phases of modeling the software. These requirements have in turn stimulated and challenged researchers to develop new approaches for dealing with the architecture evolution topic. Despite this widespread interest, few studies on architecture evolution taxonomy have been found in literature. Therefore, there is a clear need to bring structure into the software architecture domain to better landscape the wide array of research devoted to covering this field. This kind of study is vital for both categorization of the existing works and highlighting gaps that may provide new trends for the future. This paper refers to some of the most significant of these. First, (Buckley et al., 2005)

adopt a complementary way of thinking to position a taxonomy of software change using a non-exhaustive set of factors inherent to mechanisms used to evolve systems. These factors are categorized into two non-separate sets of: dimensions as characterizing and influencing factors. Meanwhile, in (Breivold et al., 2012), the authors identified five main categories of themes based substantially on research topics to conduct an investigation of 82 research papers. A set of specific characteristics is provided with a view to refine each category to subcategories reflecting a common specification on research focus, research concepts and context. The proposed overview does not mention an explicit framework for the proposed taxonomy but presents significant descriptions of many relevant studies for software evolution. However, a recent study (Ahmad et al., 2014) has identified six research themes of evolution reuse knowledge by investigating a set of existing methods, techniques and solutions either for systematic application or for empirical acquisition of architectural knowledge. The proposed thematic classification is focused on both time of evolution (design or runtime) and type of evolution (change execution or change mining) for reuse knowledge. The lack of existing studies on characterization of architectural evolution provide much scope for new thinking about classifying the existing works (Garlan et al., 2009)(Chaki et al., 2009). The aim of the

paper is threefold: (i) to provide a conceptual framework by addressing major concerns (what, why, who, how and when) about the evolution of software architecture, (ii) to identify the main taxonomy classes which could assist in both landscaping the field and highlighting gray areas, (iii) to identify a set of expected quality criteria which can elucidate the quality criteria focus for each evolution mechanism. Our motivations are driven by the need to promote synergy between the various existing mechanisms throughout the software evolution field. However, we attempt to find a simple and effective arrangement of approaches which may serve, subsequently, as a standard for evolution. To carry out this study, a set of 67 selected papers have been classified into broad categories and then utilized according to the experimental software engineering guidelines (Wohlin et al., 2012). The remainder of this paper is structured as follows: section two introduces the dimensions of evolution which draw the conceptual framework and the structure of the evolution taxonomy. The third section is devoted to presenting a brief definition of qualitative expectations for an evolution model. Section four conducts a proposal assessment to establish a more holistic understanding of our proposals. The last section recapitulates our major findings and discusses implications for further research.

2 TAXONOMIC FRAMEWORK

A taxonomic framework should be regarded rather as a tool which provides meaningful benchmarks for both coverage of the current state of the art and suppositions to guide new trends.

2.1 Framework for Software Evolution

By answering certain questions about: what, why, who, how and when does software architecture evolve? we can substantially identify the following four dimensions throughout the current studies: *levels* (who?), *object* (where?), *type* (what?) and the *operating mechanism* (how?) of evolution (OME in the following).

2.1.1 Levels of Evolution

This dimension emphasizes the importance of addressing architecture over one or several hierarchical levels (Amirat et al., 2011). This entails considering distinctively the modeling and abstraction levels during the evolution process as follows: (i) **Modeling level** (M_0 to M_3)- is an abstract representation of

the structure and behavior of a system with a view to provide a line of reasoning related to the system considered (Bézivin, 2003); changes can be performed in one or more of these levels (e.g. *each instance's manipulation is at a lower modeling level for the class concept*). (ii) **Abstraction level** (a_0 to a_n)- is used to address complex problems by defining appropriate levels of abstraction for relegation of irrelevant details to a lower level with a view to restrict the semantic information (Oussalah et al., 2014) (e.g. *refining classes to subclasses would be helpful and effective measures for designers*).

2.1.2 Object of Evolution

Means the subject on which the evolution is operated. Object can be: (i) **Artifact**- an abstraction of any element belonging to the architectural structure (architecture, component, service...), for example, when a change is performed on a component part of an architecture (user interface, shared variable, ...), the evolution is basically made on the artifact itself (component). (ii) **Process**- an abstraction of all isolated or combined operations or methods to be applied to a process, e.g. *installation of a new strategy in terms of rules and constraints*.

2.1.3 Operating Mechanism of Evolution (OME)

Outline the fashion in which an object of evolution is involved over the different hierarchical levels considered during the evolution of the architecture. Two main operating mechanisms can be identified according to (Brooks, 1989): (i) **Reduce**- Brooks defines the "reductionist" approach as a "classical" approach to problem solving whereby the overall resolution task is decomposed into subtasks. During this reductionist evolution, the operating mechanism goes through a pre-defined evolution path, until the solution is satisfied (evolved model) (e.g. *the evolution of classes impacts the instances*). Distinctions can be made in terms of three major cases: (a) the reduction operating mechanism can involve several modeling levels (*Modeling level reduce*) or only one modeling level either over (b) several abstraction levels (*Inter-abstraction level reduce*) or also only (c) one abstraction level (*Intra-abstraction level reduce*). (ii) **Emergence**- In contrast, during an "emergentist" evolution approach, the activity builds the path to a solution. Indeed, the emergence exposes a passage between the activity of "micro-level" and that of "macro-level" (e.g. *categorization of classes in super-classes*). *Modeling level emergence*, *inter-abstraction level emergence* and *intra-abstraction level emergence* are identified in the same way as by the reduce

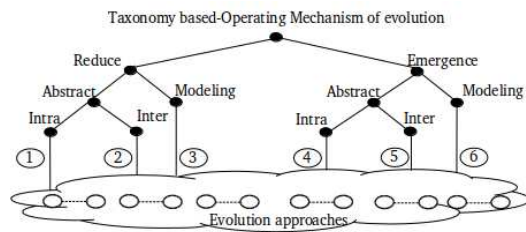


Figure 1: Operating mechanism across hierarchical levels.

OME (Fig. 1).

2.1.4 Type of Evolution

This dimension is commonly used throughout the software taxonomy (Williams and Carver, 2010)(Ahmad et al., 2014)(Buckley et al., 2005) but with different perceptions. The first embodies the behavioral aspect of maintainability according to type of maintenance. Meanwhile, the second type of evolution focuses on the development environment being used during the evolution (static, dynamic or load-time). In this light, the evolution type has to be considered from two relevant perspectives to achieve a reasonable assessment. The first concerns a technical view which applies the notion of maintainability (corrective, perfective, adaptive and preventive). For the sake of simplicity, this paper employs the following well known categories: corrective, perfective and adaptive (Swanson, 1976). The second perception adopts an architectural viewpoint which mainly considers architecture as an artifact of evolution (Chaki et al., 2009). It expresses the reason for supporting and conducting software changes whether after (curative) or before (predictable) software delivery, regardless of the technical type used.

(i) **Curative**- ensures that when new requirements arise unpredictably or were poorly defined or even unspecified during the life cycle, the environment will help to correct, perfect and adapt them by integrating the desired changes. Usually, the curative type depends on the nature of problem that is being addressed, mainly in relation to its context and the resource deployed, and often is applied in an ad-hoc manner as problems arise.

(ii) **Predictable**- ensures that evolution requirements are taken into account during the analysis phase and specified during the design. These requirements are specified at an earlier stage of the design process and define all anticipated changes allowed using any technical types for evolution. The latter can be divided into two main categories depending on the architects view: according to the predefinition of the final architecture (Chaki et al., 2009) and depending on the continuity of the evolved architecture (Oussalah et al.,

| Type of evolution | | | Technical Type of evolution | | |
|--------------------|-------------|----------|-----------------------------|------------|----------|
| | | | Corrective | Perfective | adaptive |
| Architectural Type | Predictable | Open | Break | | |
| | | Seamless | | | |
| | Close | Open | Break | | |
| | | Seamless | | | |
| | Curative | Open | Break | | |
| | | Seamless | | | |
| Close | Open | Break | | | |
| | Seamless | | | | |

Figure 2: Type of evolution sub-dimensions.

1999). The first category emphasizes two types of evolution: (a) *Open evolution*- means a deduction, from an initial architecture, of a new architecture reflecting a system solution in which a set of invariants and constraints are respected. (b) *Closed evolution*-the final architecture is considered as a premise. This evolution consists of performing a valid sequence of operations which, once they are applied to architecture, lead without any ambiguity to the final architecture. This involves oriented knowledge construction and requires a greater capacity of conceptual thinking. The second category provides two kinds of evolution: (c) *Evolution with break*- means that interventions are applied directly to the initial architecture without having the ability to go back on the trace of the made evolution (e.g. *evolution by extensibility (open white box)*, *changes are performed directly on codes*). (d) *Evolution with seamless*- denotes an evolution with trace where the architecture keeps a trace of its initial properties and operations performed before each evolving operation. Such continuity is ensured by the backup of the applied operation sequences (e.g. *in versioning technique, operations undertaken on versions are imperatively stored, which in turn preserves the evolution history*). In fact, open or closed types of evolution can be either with break or seamless (Fig. 2).

2.2 Evolution Taxonomy

The taxonomy allows presentation of the whole solution space of architecture evolution. The proposed taxonomy is based successively on the OME dimension and on the modeling and abstraction levels of the architecture affected during the evolution activity. During evolution process, each OME (reduce or emergent) can either impacts at least two modeling levels of the architecture (modeling levels) or preserving the same modeling level. In such case, the OME should be specified whether it concerns different abstraction levels (inter-Abstraction) or simply evolved within the same level of abstraction

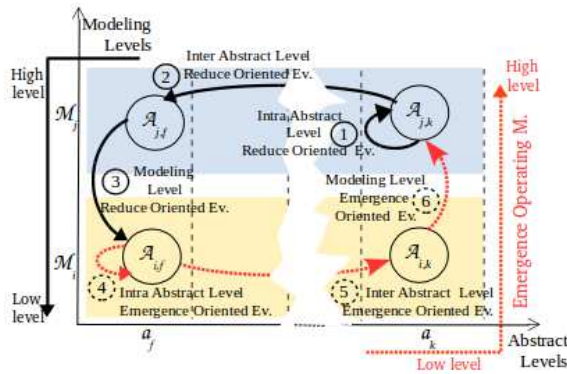


Figure 3: Architecture involvement through operating mechanisms.

(intra-abstraction). Thereby, the evolution taxonomy is structured around six classes for software architecture evolution, as follows:

Intra-abstraction Level Reduce Oriented Evolution (class 1)- The OME consists of evolving one or more architectural elements, of an architecture sited at a defined modeling level, within the same abstraction level (e.g. *modification of one or more component properties of an architecture*).

Inter-abstraction Level Reduce Oriented Evolution (class 2)- The OME consists of evolving one or more architectural elements sited at a defined modeling level, from their associated abstraction level to the lower abstraction levels (e.g. *evolution of class impacts several sub-classes*).

Modeling Level Reduce Oriented Evolution (class 3)- The OME consists of evolving one or more architectural elements on a downward modeling path i.e. from their initial modeling level to the lower modeling levels (e.g. *evolution of classes impacts instances*).

Intra-abstraction Level Emergence Oriented Evolution (class 4)- The OME consist of emerging one or more architectural elements, of an architecture sited at a defined modeling level, within the same abstraction level (e.g. *categorization of classes by creating a superclass*).

Inter-abstraction Level Emergence Oriented Evolution (class 5)- The OME consists of emerging one or more architectural elements, belonging to one defined modeling level, from their associated abstraction level up to the higher abstraction levels (e.g. *aggregation of classes in one class*).

Modeling Level Emergence Oriented Evolution (class 6)- The OME of evolution consists, on upward path, of emerging one or more architectural elements from their associated modeling level to the higher modeling levels (e.g. *creation of new classes to deal with differences on multiple instances*).

3 QUALITY EXPECTATION FOR AN EVOLUTION MODEL

Expectancy indicates that better effort will result in better performance (Vroom, 1964). Qualitative expectancy assumes that researchers have reasons for favoring one set of conscious criteria over others. Indeed, an architectural evolution model is operated to satisfy a set of subjective factors to achieve some valid goals. We are focused on devising a set of quality criteria for a new evolution approach. We have estimated that these are the minimum expected according to an architectural point of view. However, these criteria must not be interpreted as a restriction on quality factors but rather as the common specific criteria for our topic. Therefore, according to (Oussalah et al., 1999), an evolution approach must be: (i) enunciated to better provide a thorough behavior for the changes, (ii) expressed to better outline the development and refinement characteristics of the desired model, and finally, (iii) evaluated from the fact that the evolution model provides a set of quality appraisal criteria aimed to estimate the relevance of the model comparatively to a goal commitment. Thus, quality expectancy can be structured into three capacities: Q_n , Q_x and Q_v , respectively for the enunciation, expressiveness and evaluation capacities. Furthermore, a model of evolution can be evaluated to indicate its representativeness expectancy, which means the percentage to which the evolution quality has met the expected quality criteria. These percentages can be used as an indicator tool for assessing the evolution quality of a model.

3.1 Enunciation Capacity

Enunciation is formalized in terms of criteria to: formulate, manage the impacts of changes and keep track between the starting model (before changes) and the final model. This capacity encompasses criteria of: *Formulation of evolution (F)*- which reflects the level of evolution visibility in terms of applying operators to cause the initial architecture to evolve. *Impact management of the evolution (I)*- means the result due to the change of an architectural element of the model in terms of influence on the other elements. *Traceability (T)*- formalizes the model's ability to keep track of the sequence of evolution operations applied for changing an initial architecture. The enunciation capacity (Q_n) can be expressed through a parametric equation given by:

$$Q_n = \frac{a \times F(p) + b \times I(p) + c \times T(p)}{a + b + c} \quad (1)$$

Where parameter p is the degree of parameterization for criterion. Coefficients a , b and c are the associated weights by which we can establish a hierarchical order of preference between criteria.

3.2 Model Expressiveness Capacity

This capacity indicates what this model is actually capable of describing regarding the evolution of architecture. *Modeling level (M)*- appoints the different modeling levels affected during the OME. It can be flat trend in the case of an evolution on the same modeling level, otherwise it affects different modeling levels to achieve the result. *Abstraction level (A)*- defines the degree of refinement within an evolution in terms of the internal architecture details during the reuse time. These details are presented in a white, gray or black box. *Expressiveness mode (E)*- reflects the chosen representation to express the model evolution. This criterion focuses on accuracy and simplicity of expression and offers more semantics to enable reuse. *Operating mode (O)*- prescribes the mode of reasoning by which the evolution is managed. This can be done by deduction or by induction or by classification. *Domain (D)*- represents the scope covered by the solution of the evolution model. In fact, a generic domain means that multiple situations are likely to adopt this solution without giving details of their execution. In counterpart, the specific domain provides a further investment of multiple aspects of details to enable the architecture to evolve. The expressiveness capacity can be expressed by:

$$Q_x = (a \times M(p) + b \times A(p) + c \times E(p) + d \times O(p) + e \times D(p)) / (a + b + c + d + e) \quad (2)$$

3.3 Quality Evaluation Capacity of an Evolution Model

The third dimension reflects the ability to measure the capacities of the model to assess its quality, through: *Re-usability (R)*- represents the degree of re-use given by a model of evolution. *Adaptability (A)*- expresses the ability to control and enable an evolution model to be adapted dynamically. *Performance (P)*- defines the faculty of the model to make the desired changes by optimizing time, cost, space and speed ratios. *Support of evolution (S)*- describes if the evolution model has an predefined or implemented tools for the used evaluation mechanism. The evaluation capacity can be formulated as:

$$Q_v = \frac{a \times R(p) + b \times A(p) + c \times P(p) + d \times S(p)}{a + b + c + d} \quad (3)$$

Application Example: For the simplicity of understanding the example, some assumptions are made: (i) for handling criteria values, numerics 1, 0.5 and 0 are assigned respectively to explicit, implicit and not recognized criteria, (ii) the common associated weights used in each equation such as a , b , c , ..., d have been set to 1, (iii) parameters p - are considered equivalent for simplicity of calculation, thus equal values have been assigned to each one of them, and (iv) representativeness ratio is in the range of three intervals: $0 \leq weak < 1/3 \leq medium < 2/3 \leq high \leq 1$. Prospection of the paper by (Oussalah et al., 2006) led to the following criteria results: enunciation capacity ($F=1$, $I=1$, $T=0$), expressiveness capacity ($M=1$, $A=0.5$, $E=1$, $O=1$, $D=1$) and evaluation quality capacity ($R=1$, $A=1$, $P=1$, $S=0$). Then applying equations (1),(2), (3) results: $Q_n = (1 \times 1 + 1 \times 1 + 1 \times 0) / 3 = 0.67$; $Q_x = (1 \times 1 + 1 \times 0.5 + 1 \times 1 + 1 \times 1 + 1 \times 1) / 5 = 0.90$; $Q_v = (1 \times 1 + 1 \times 1 + 1 \times 1 + 1 \times 0) / 4 = 0.75$. The quality expectancy given by the capacities: $Q(Oussalah et al., 2006) = (0.66 + 0.90 + 0.75) / 3 = 0.77$ which means that the model presents a high representativeness for software architecture evolution.

4 PROPOSAL ASSESSMENT

Substantially, selection of papers focused on: (i) papers studying either a software architecture evolution or software engineering evolution thematic, or both of these, and (ii) according to two potential criteria: first, papers used within other classifications research, and second, well known papers using the indicator cited index paper. The preparation of the review was done through devising a prospective study sheet in accordance with (Wohlin et al., 2012). Once the appropriate information had been gathered, calculations were conducted as shown in the previous example. Afterward, representativeness expectancy was assessed and studies with a weak expectancy were removed. Finally, 67 selected papers were selected that to the best of our knowledge are the most representative studies in the field.

4.1 Evolution Mechanism Categories

In order to provide clarification and therefore wider understanding of the proposed taxonomy, the selected papers were divided into five thematic categories to reflect the broadest range of well known mechanisms for dealing with evolution. Each category includes mechanisms that share conceptually low close reflections regardless of the technique used for modeling (static or dynamic):

(i) *Evolution Change-based Approaches* (Gottlob et al., 1996)(Bengtsson et al., 2004)(Oreizy et al., 1999)- encompass work related to: (i) Maintainability- reflects changes into software after its implementation, (ii) Modifiability- describes the architectures ability to be changed in response to changes due to the environment or stockholder requirements or functional specifications, and (iii) Self-changes- comprise automation in computing which refers to execution of important computing operations. It includes: automatic computation, self-management, self-organization, self-adaptive.

(ii) *Evolution Algorithmic-based Approaches* (Engel and Browning, 2008)(Wermelinger and Fiadeiro, 2002)- include all reflections for improving software architecture by providing simple structures, dealing with similarity of objects. It contains mechanisms of categorization, reorganization, incremental approaches and refactoring.

(iii) *Evolution Trace-based Approaches* (Cicchetti et al., 2008)(Herrmannsdoerfer et al., 2009)- includes mechanisms where traceability is seen as a key insight for evolution to promote consistency and compliance. This category encompasses mechanisms related to: migration, co-evolution, versioning and view-point.

(iv) *Evolution Transformation based Approaches* (Zhao et al., 2007)(Engels and Heckel, 2000)- include mechanisms wherein changes are applied using one or more transformation rules for transforming or verifying or validating models (Mens and Van Gorp, 2006). Model driven architecture approaches and graph-transformation approaches are approaches dealing with such topics.

(v) *Evolution style based approaches* (Garlan et al., 2009)(Le Goar et al., 2010)- designate high-level modeling approaches, using architectural style for modeling the evolution. Evolution-styles help to specify the basic structure to evolve in software architecture.

4.2 Results and Discussion

Results are discussed according to:

(i) **Framework Dimensions:** Investigated papers were compared to the framework using the representativeness percentage, which represents the quotient of the number of explicit and implicit (weighted) expressions of criteria relatively to the total number of papers within a category. Afterwards, percentages were reported in the Table 1, in which: Explicitly, overall the studied evolution mechanisms focus on: (i) the artifact as an object of evolution with more than 70%, to the detriment of the process object, which remains a very promising direction (minus 30%), par-

Table 1: Representativeness of the framework dimensions.

| Framework dimensions | Metrics | % |
|----------------------|---------------|-------|
| Object of evolution | Artifact | 71.64 |
| | Process | 28.36 |
| Level of evolution | Modeling | 67.76 |
| | Abstraction | 32.24 |
| OME of evolution | Reduce | 89.56 |
| | Emergence | 10.44 |
| Type of evolution | Technical | 55.33 |
| | Architectural | 44.67 |

ticularly because the process is usually considered as a dynamic specification of the artifact, without neglecting that the artifact provides an advantageous alternative through UML modeling specifications, (i) abstraction level has attracted less interest (nearly 33%) from the evolution community for the reason that it overlaps with the modeling level concept, (iii) the "Reduce" OME is the most covered, scoring more than 89% as against 11% for the "Emergent", mainly due to the developer opting for rigorous mathematical reasoning based on a formal logic, which favors the deduction process, and (iv) technical type (55%) is relatively better represented than architectural type. The study has also found an overwhelming rate for the "Predictable" type (97%), absolutely justified by the rough dynamicity of the environment and the instability of stakeholder requirements at the different modeling levels, with an advantage for the "Open" type of more than 70%. In addition, the study indicates that almost 80% of the selection are devoted to corrective and perfective typology. It should furthermore be noted that adaptability produced a low representation percentage (less than 20%), mainly due to the difficulty of dynamic evolution at the design-time.

(ii) **Evolution Taxonomy:** Evolution mechanisms were ranked according to the suggested taxonomy with the aim of assessing the cover achieved by each category of mechanisms and to deduce the least considered taxonomy classes. By using the percentage of representativeness in the same way as described previously, Table 2 displays the different hedging of each mechanism and clarifies that existing studies have fostered the modeling-level reduce oriented evolution class (class 3), with trace-based approaches being the most significant in 68% of the work. The 14% of studies in intra-abstraction level reduce oriented evolution (class 1) is justified by the presence of techniques supporting quality, assessment, and analysis at the architectural level. Classification of the selection reveals a

Table 2: Class percentages of evolution taxonomy.

| C-1 | C-2 | C-3 | C-4 | C-5 | C-6 |
|-------|------|-------|------|------|------|
| 14.93 | 5.97 | 68.66 | 5.96 | 1.49 | 2.99 |

Table 3: Criterion and capacity percentages (%).

| $Q_{enunciation}=22.60$ | | | $Q_{expressiveness}=54.29$ | | | | | $Q_{evaluation}=23.11$ | | | |
|-------------------------|------|------|----------------------------|------|-------|-------|-------|------------------------|------|------|------|
| F | I | T | M | A | E | O | D | R | A | P | S |
| 11.80 | 6.16 | 4.64 | 12.51 | 6.56 | 12.21 | 10.90 | 12.11 | 6.66 | 5.95 | 6.26 | 4.24 |

significant shortage in research focusing on the emergence OME.

(iii) **Quality Expectancy:** This evaluation can rule on potential strengths and weaknesses of each category of the architectural evolution mechanisms. Table 3, which presents the percentage of representativeness of a quality criterion across all the studied papers, shows that for the studied models the appropriateness of the measurement formulation capacity for the enunciation dimension is the most respected aspect in terms of the fact that all the selected works were revised and published. Traceability capacity had the weakest representation, possibly because of the priority that approaches attach to finding a new solution. Regarding the expressiveness capacities, all modeling levels were covered, from the lowest level (M_0) to the meta-modeling level (M_2). However, a weak separation between modeling and abstraction levels was formulated (almost 32%). Expressiveness mode, operating mode and domain applicability are strongly expressed criteria in the studied models. In addition, evaluation criteria are considered in a less rigorous manner in the selection, by which we deduce that the field of architectural evolution has not yet reached a sufficient level of saturation and maturity to focus primarily on quality evaluation. Nevertheless, the research displays more consideration of re-usability criterion, and to a lesser extent of adaptability and performance criteria, while the support of quality assessment tools proposed by approaches remains a very promising track. Table 4 sets out the representativeness percentage of the expected qualitative capacities for each evolution mechanism. On this basis, the main findings are as follows: (i) the transformation based approaches present the higher enunciation ratio in comparison to the other categories, due essentially to its great ability for traceability and formalization, (ii) the algorithmic-based approaches focus largely on criteria related to expressiveness capac-

ity through the operating and expressiveness modes, domain of specification and modeling level identification, and (iii) throughout all categories, there is insufficient representation of quality evaluation capacity which denotes slightly higher representation of change based and style-based approaches.

5 CONCLUSION

The first goal of this study was to devise a framework for software architecture evolution. This framework attempts to provide a structural organization which, while far from considering qualitative specifications, is organized in four main dimensions: levels, object, type and operation mechanism of evolution. The evolution taxonomy, as a second goal, offers a landscape of what was being done and what remains to be done. It has unveiled a lack of emergence approaches. The third goal was to deal with the real challenge of achieving representativeness of a given evolution model in the field. The paper recapitulated a large number of quality criteria crucial to evaluation of effectiveness of the effort-making in terms of devising a model that meets the quality expectations. The latter were arranged into the following three important capacities: enunciation, expressiveness and quality evaluation. The results shows that criteria related to expressiveness are often given more weight as opposed to enunciation and evaluation quality. Likewise, our study limitation relates to the estimation of qualitative criteria either in terms of assigning a particular value for each criterion (explicit and implicit) or in terms of adopting an equal weight value for parameters over all criteria. Thus, we believe that the conceptual framework that we have presented could be applied in future studies to: (i) determine the appropriate evolution approach according to the criteria that seem the most relevant, (ii) highlight how the quality criteria of an evolution approach would influence its representativeness, and (iii) provide guidance in particular research fields, such as the study of emergent operating mechanisms. Moreover, future studies could use the proposed framework to explore many of the research papers investigated by previous taxonomic studies in the software engineering domain. In this light, it would be invaluable to provide a meta-classification which could be instantiated according to the field specialization.

Table 4: Capacity percentages by mechanism category.

| Evolution mech. (%) | Q_{Enun} | Q_{Expr} | Q_{Eval} |
|---------------------|------------|------------|------------|
| Change-based | 20.77 | 54.62 | 24.62 |
| Algorithmic-based | 20.99 | 57.41 | 21.60 |
| Trace-based | 21.47 | 55.21 | 23.31 |
| Transformat-based | 30.77 | 55.38 | 13.85 |
| Style-based | 23.75 | 51.91 | 24.34 |
| Average | 22.60 | 54.29 | 23.11 |

REFERENCES

- Ahmad, A., Jamshidi, P., and Pahl, C. (2014). Classification and comparison of architecture evolution reuse knowledgea systematic review. *Journal of Software: Evolution and Process*, 26(7):654–691.
- Amirat, A., Menasria, A., and Gasmallah, N. (2011). Evolution framework for software architecture using graph transformation approach.
- Bengtsson, P., Lassing, N., Bosch, J., and van Vliet, H. (2004). Architecture-level modifiability analysis (alma). *Journal Syst.and Software*, 69(1):129–147.
- Bézivin, J. (2003). La transformation de modèles. INRIA-ATLAS & Universit de Nantes, 2003. Ecole dEt dInformatique CEA EDF INRIA 2, cours #6.
- Breivold, H. P., Crnkovic, I., and Larsson, M. (2012). A systematic review of software architecture evolution research. *Information and Software Technology*, 54(1):16–40.
- Brooks, R. A. (1989). A robot that walks; emergent behaviors from a carefully evolved network. *Neural computation*, 1(2):253–262.
- Buckley, J., Mens, T., Zenger, M., Rashid, A., and Kniesel, G. (2005). Towards a taxonomy of software change. *Journal of Software Maintenance and Evolution: Research and Practice*, 17(5):309–332.
- Chaki, S., Diaz-Pace, A., Garlan, D., Gurfinkel, A., and Ozkaya, I. (2009). Towards engineered architecture evolution. In *Proceedings of the 2009 ICSE Workshop on Modeling in Software Engineering*, pages 1–6. IEEE Computer Society.
- Cicchetti, A., Di Ruscio, D., Eramo, R., and Pierantonio, A. (2008). Automating co-evolution in model-driven engineering. In *Enterprise Distributed Object Computing Conference, 2008. EDOC’08. 12th International IEEE*, pages 222–231. IEEE.
- Engel, A. and Browning, T. R. (2008). Designing systems for adaptability by means of architecture options. *Systems Engineering*, 11(2):125–146.
- Engels, G. and Heckel, R. (2000). Graph transformation as a conceptual and formal framework for system modeling and model evolution. In *Automata, Languages and Programming*, pages 127–150. Springer.
- Garlan, D., Barnes, J. M., Schmerl, B., and Celiku, O. (2009). Evolution styles: Foundations and tool support for software architecture evolution. In *Software Architecture, 2009 & European Conference on Software Architecture. WICSA/ECSA 2009. Joint Working IEEE/IFIP Conference on*, pages 131–140. IEEE.
- Gottlob, G., Schrefl, M., and Röck, B. (1996). Extending object-oriented systems with roles. *ACM Transactions on Information Systems (TOIS)*, 14(3):268–296.
- Herrmannsdoerfer, M., Benz, S., and Juergens, E. (2009). Cope-automating coupled evolution of metamodels and models. In *ECOOP 2009–Object-Oriented Programming*, pages 52–76. Springer.
- Jazayeri, M. (2005). Species evolve, individuals age. In *Principles of Software Evolution, Eighth International Workshop on*, pages 3–9. IEEE.
- Le Goaer, O., Tamzalit, D., and Oussalah, M. (2010). Evolution styles to capitalize evolution expertise within software architectures. In *SEKE 2010*, pages to-appear.
- Mens, T. and Van Gorp, P. (2006). A taxonomy of model transformation. *Electronic Notes in Theoretical Computer Science*, 152:125–142.
- Oreizy, P., Gorlick, M. M., Taylor, R. N., Heimbigner, D., Johnson, G., Medvidovic, N., Quilici, A., Rosenblum, D. S., and Wolf, A. L. (1999). An architecture-based approach to self-adaptive software. *IEEE Intelligent systems*, 14(3):54–62.
- Oussalah, M. et al. (1999). *Génie objet: analyse et conception de l’évolution*. Hermès Science publications.
- Oussalah, M. et al. (2014). *Architectures logicielles : Principes, techniques et outils*. Hermes Science Publications (12 fvrier 2014).
- Oussalah, M., Sadou, N., and Tamzalit, D. (2006). Saev: A model to face evolution problem in software architecture. In *Proceedings of the International ERCIM Workshop on Software Evolution*, pages 137–146.
- Perry, D. E. and Wolf, A. L. (1992). Foundations for the study of software architecture. *ACM SIGSOFT Software Engineering Notes*, 17(4):40–52.
- Swanson, E. B. (1976). The dimensions of maintenance. In *Proceedings of the 2nd international conference on Software engineering*, pages 492–497. IEEE Computer Society Press.
- Vroom, V. H. (1964). *Work and motivation*. new york: John willey & sons.
- Wermelinger, M. and Fiadeiro, J. L. (2002). A graph transformation approach to software architecture re-configuration. *Science of Computer Programming*, 44(2):133–155.
- Williams, B. J. and Carver, J. C. (2010). Characterizing software architecture changes: A systematic review. *Information and Software Technology*, 52(1):31–51.
- Wohlin, C., Runeson, P., Höst, M., Ohlsson, M. C., Regnell, B., and Wesslén, A. (2012). *Experimentation in software engineering*. Springer Science & Business Media.
- Zhao, C., Kong, J., Dong, J., and Zhang, K. (2007). Pattern-based design evolution using graph transformation. *Journal of Visual Languages & Computing*, 18(4):378–398.