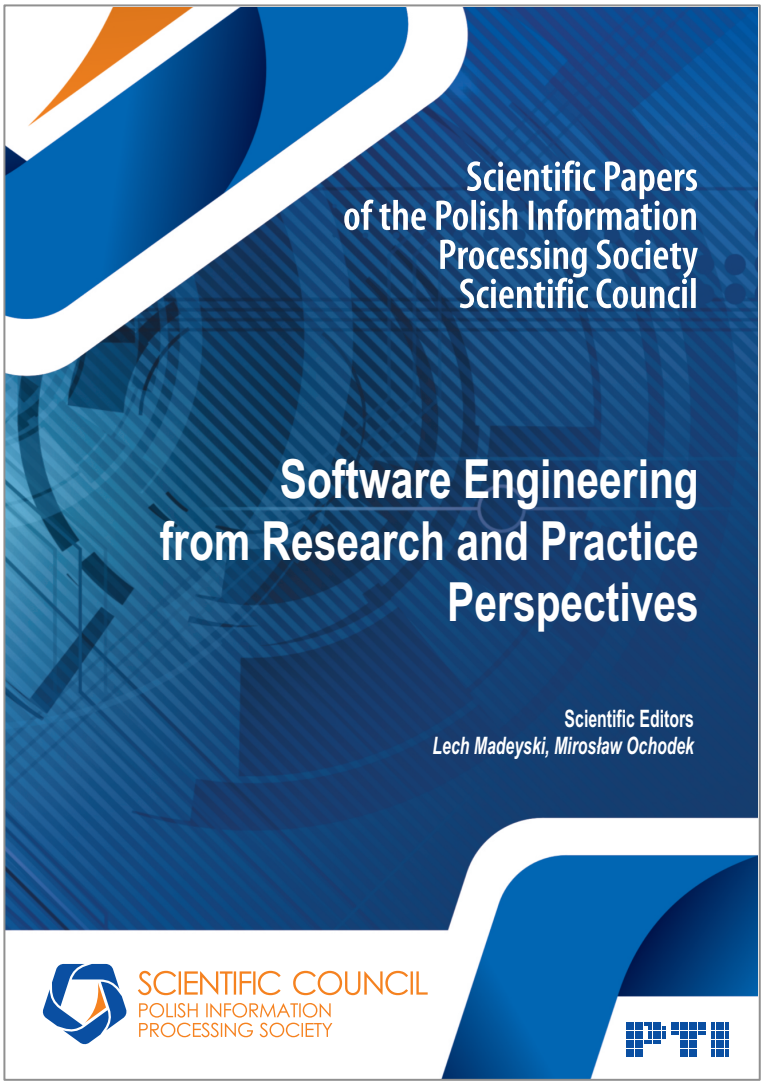
CHAPTER 1

Recent Polish achievements in Software Engineering

By **Jakub Jurkiewicz, Piotr Kosiuczenko, Lech Madeyski, Mirosław Ochodek, Cezary Orłowski, Łukasz Radliński**



How to cite:

Jakub Jurkiewicz, Piotr Kosiuczenko, Lech Madeyski, Mirosław Ochodek, Cezary Orłowski, Łukasz Radliński. "Recent Polish achievements in Software Engineering." In Software Engineering from Research and Practice Perspective (Eds. Lech Madeyski, M. Ochodek), pp. 15-38, Scientific Papers of the Polish Information Processing Society Scientific Council, ISBN 978-83-63919-16-0, 2014.

Table of Contents

Authors and affiliations

Preface
Lech Madeyski
Wroclaw University of Technology, Faculty of Computer Science and Management, Institute of Informatics
lech.madeyski@pwr.edu.pl
Mirosław Ochodek
Poznan University of Technology, Faculty of Computing
miroslaw.ochodek@cs.put.poznan.pl

CHAPTER 1
Jakub Jurkiewicz
Poznan University of Technology, Faculty of Computing
jakub.jurkiewicz@cs.put.poznan.pl
Piotr Kosiuczenko
Military University of Technology in Warsaw, Department of Cybernetics
pkosoiczenko@wat.edu.pl
Lech Madeyski
Wroclaw University of Technology, Faculty of Computer Science and Management, Institute of Informatics
lech.madeyski@pwr.edu.pl
Mirosław Ochodek
Poznan University of Technology, Faculty of Computing
miroslaw.ochodek@cs.put.poznan.pl
Cezary Orłowski
IBM Center for Advanced Studies on Campus, Gdańsk,
cor@zie.pg.gda.pl
Łukasz Radliński
West Pomeranian University of Technology,
Faculty of Computer Science and Information Technology
lukasz.radlinski@zut.edu.pl

CHAPTER 2
Marek Majchrzak
Capgemini Poland
Wroclaw University of Technology, Faculty of Computer Science and Management, Institute of Informatics
marek.majchrzak@capgemini.com

Łukasz Stilger
Capgemini Poland
lukasz.stilger@capgemini.com
Marek Matczak
Capgemini Poland
marek.matczak@capgemini.com

CHAPTER 3
Tomasz Sitek
Gdańsk University of Technology, Faculty of Management and Economics
tsitek@zie.pg.gda.pl
Artur Ziółkowski
Gdańsk University of Technology, Faculty of Management and Economics
aziolko@zie.pg.gda.pl

CHAPTER 4
Miklós Biró
Software Competence Center Hagenberg,
miklos.biro@scch.at

CHAPTER 5
Stanisław Jerzy Niepostyn
Warsaw University of Technology, Institute of Computer Science
S.Niepostyn@ii.pw.edu.pl
Andrzej Tyrowicz
Agencja Europejska, Andrzej Tyrowicz
a@tyrowicz.eu

CHAPTER 6
Łukasz Radliński
West Pomeranian University of Technology,
Faculty of Computer Science and Information Technology
lukasz.radlinski@zut.edu.pl

CHAPTER 7
Bogumiła Hnatkowska
Wroclaw University of Technology, Faculty of Computer Science and Management, Institute of Informatics
Bogumila.Hnatkowska@pwr.edu.pl
Łukasz Wrona
Wroclaw University of Technology, Faculty of Computer Science and Management, Institute of Informatics
lukwro83@gmail.com

CHAPTER 8
Andrzej Ratkowski
Warsaw University of Technology,
Institute of Control and Computation Engineering
Krzysztof Gawryś
Warsaw University of Technology,
Institute of Control and Computation Engineering
Eliza Świątek
Warsaw University of Technology,
Institute of Control and Computation Engineering

CHAPTER 9
Ilona Bluemke
Warsaw University of Technology, Faculty of Electronics and Information Technology, Institute of Computer Science
I.Bluemke@ii.pw.edu.pl
Anna Stepień
Warsaw University of Technology, Faculty of Electronics and Information Technology, Institute of Computer Science
A.Stepien.1@stud.elka.pw.edu.pl

CHAPTER 10
Michał Ćmil
Poznan University of Technology, Faculty of Computing
cmilmichal@gmail.com
Bartosz Walter
Poznan University of Technology, Faculty of Computing
bartosz.walter@cs.put.poznan.pl

CHAPTER 11

Wojciech Frącz
AGH University of Science and Technology
fracz@iisg.agh.edu.pl
Jacek Dajda
AGH University of Science and Technology
dajda@agh.edu.pl

Chapter 1

Recent Polish achievements in Software Engineering

Publications in top research journals (indexed by ISI) as well as citations are crucial in any research field to position the work and to build on the work of others. The objective of this chapter is twofold: to give an overview of the achievements of Polish research centers in the field of software engineering since 2010, and to present few recent contributions by researchers with Polish affiliations in ISI journals in the field of software engineering or closely related fields.

1.1. Introduction

Glass was the first who two decades ago published an assessment of systems and software engineering scholars and institutions [Gla94]. The set of journals selected by Glass included *IEEE Transactions on Software Engineering* (*TSE*), *ACM Transactions on Software Engineering and Methodologies* (*TOSEM*), *IEEE Software* (*SW*), *Information and Software Technology* (*IST*), *Journal of Systems and Software* (*JSS*), and *Software: Practice and Experience* (*SPE*). In 2009 Wong et al. [WTGBC09] have analyzed publications in the period of 2002–2006 using this set of journals extended by the *Empirical Software Engineering* (*EMSE*) journal to emphasize the importance of strong empirical component. The most recent report by Wong et al. was published in 2011 [WTGBC11].

A complementary series of analyses of the most cited articles in the software engineering journals has been published by Wohlin. The most recent of analyses was published in 2009 on a basis of 18 software engineering journals [Woh09].

However, to the best of our knowledge, neither similar analyses but related to Polish research institutions or researchers involved in the software engineering field, nor an overview of selected of contributions by researchers with Polish affiliations in ISI software engineering (or closely related) journals have been published so far. Hence, the aim of this chapter is to fill this gap.

The aim of the first section is to give an overview of the contribution of researchers with Polish affiliations in comparison to other European countries. The aim of the subsequent sections is to go into details and present short overviews of selected contributions of Polish authors published by ISI indexed journals within the software engineering field or computer science in general. The presented contributions include identification of events in use cases, solving the invariability problem in OCL, predicting the flow of defect correction effort to optimize the

amount of quality assurance (QA) activities to minimize the total project effort, and model of a maturity capsule in software project management.

1.1.1. Selection decisions

An important decision when looking at Polish contributions to the field of software engineering is which ISI software engineering journals to include. We decided to include in the analyzed set of journals all of the software engineering journals analyzed by Wong et al. [WTGBC11] as well as Wohlin [Woh09], even if they changed their names (e.g., *Journal of Software: Evolution and Process* continues, since 2012, the tradition of the *Journal of Software Maintenance and Evolution: Research and Practice* and *Software Process: Improvements and Practice*, while *IET Software* continues, since 2007, the tradition of *IEE Proceedings - Software*). Then from the created superset of journals we excluded journals which are discontinued (*Annals of Software Engineering*, *Software Architecture*, *Software – Concepts and Tools*) or journals without impact factor in 2013. In spite of the fact that the created our set of journals based on the inclusion decisions of the renowned authors of previous analyses – Wong et al. [WTGBC11] and Wohlin [Woh09] – and minor constraints related to impact factor in year 2013, the created set of journals is by no means complete and can be extended even further. Our arbitrary decision is to extend the set of journals by adding the *Software and Systems Modeling* journal, which is strictly software engineering journal with assigned impact factor. There is also a wide range of computer science journals (e.g., *Computing and Informatics*, *Cybernetics and Systems: An International Journal*) which could be included on a paper by paper basis as some of them may be related to software engineering. However, it would need an extreme effort to check every published paper. As a result, the set of the analyzed journals is presented in Table 1.1.

All of these journals are indexed by Scopus, which provides an excellent search interface including ability to construct advanced search strings. The search string we used to constrain our search to the papers by authors with Polish affiliation published since 2010 in the aforementioned set of journals is presented below:

```
 ( ISSN(1049331X) OR ISSN(09288910) OR ISSN(13823256) OR ISSN(17518806) OR
ISSN(07407459) OR ISSN(00985589) OR ISSN(09505849) OR ISSN(02181940) OR
ISSN(1532060X) OR ISSN(09473602) OR ISSN(20477481) OR ISSN(09639314) OR
ISSN(09600833) OR ISSN(00380644) OR ISSN(01641212) OR ISSN(16191366)) AND
AFFIL(poland) AND (PUBYEAR > 2009)
```

Table 1.1. Set of analyzed software engineering journals.

ACM Transactions on Software Engineering and Methodology (TOSEM)
Automated Software Engineering (ASE)
Empirical Software Engineering (EMSE)
IET Software (IETSW)
IEEE Software (SW)
IEEE Transactions on Software Engineering (TSE)
Information and Software Technology (IST)
International Journal of Software Engineering and Knowledge Engineering (IJSEKE)
Journal of Software: Evolution and Process (JSEP)
Journal of Software Maintenance and Evolution: Research and Practice (JSME)
Requirements Engineering Journal (REJ)
Software and Systems Modeling (SoSyM)
Software Quality Journal (SQJ)
Software Testing, Verification and Reliability (STVR)
Software: Practice and Experience (SPE)
Journal of Systems and Software (JSS)

1.1.2. Search results

The search performed on 14 September 2014 returned 28 document results (sorted by number of citations):

1. L. Madeyski. "The impact of test-first programming on branch coverage and mutation score indicator of unit tests: An experiment". In: *Information and Software Technology* 52.2 (2010), pp. 169–184. DOI: *10.1016/j.infsof.2009.08.007*. URL: *http://dx.doi.org/10.1016/j.infsof.2009.08.007* – 19 citations
2. M. Ochodek, J. Nawrocki, and K. Kwarciak. "Simplifying Effort Estimation Based on Use Case Points". In: *Information and Software Technology* 53.3 (Mar. 2011), pp. 200–213. ISSN: 0950-5849. DOI: *10.1016/j.infsof.2010.10.005*. URL: *http://dx.doi.org/10.1016/j.infsof.2010.10.005* – 16 citations
3. L. Madeyski and N. Radyk. "Judy – A Mutation Testing Tool for Java". In: *IET Software Journal (formerly IEE Proceedings Software)* 4.1 (2010). Draft: *http://madeyski.e-informatyka.pl/download/Madeyski10b.pdf*, pp. 32–42. DOI: *10.1049/iet-sen.2008.0038*. URL: *http://dx.doi.org/10.1049/iet-sen.2008.0038* – 16 citations
4. A. Janik and K. Zielinski. "AAOP-based Dynamically Reconfigurable Monitoring System". In: *Information and Software Technology* 52.4 (Apr. 2010), pp. 380–396. ISSN: 0950-5849. DOI: *10.1016/j.infsof.2009.10.006*. URL: *http://dx.doi.org/10.1016/j.infsof.2009.10.006* – 15 citations
5. G. J. Nalepa and K. Kluza. "UML REPRESENTATION FOR RULE-BASED APPLICATION MODELS WITH XTT2-BASED BUSINESS RULES". in: *International Journal of Software Engineering and Knowledge Engineering* 22.04 (2012), pp. 485–524. DOI: *10.1142/S021819401250012X*. URL: *http://www.worldscientific.com/doi/abs/10.1142/S021819401250012X* – 10 citations
6. M. Miłkowski. "Developing an Open-source, Rule-based Proofreading Tool". In: *Software:*

Practice and Experience 40.7 (June 2010), pp. 543–566. ISSN: 0038-0644. DOI: *10.1002/spe.v40:7*. URL: *http://dx.doi.org/10.1002/spe.v40:7* – 10 citations

7. A. Janik and K. Zielinski. "Adaptability Mechanisms for Autonomic System Implementation with AAOP". in: *Software: Practice and Experience* 40.3 (Mar. 2010), pp. 209–223. ISSN: 0038-0644. DOI: *10.1002/spe.v40:3*. URL: *http://dx.doi.org/10.1002/spe.v40:3* – 4 citations
8. P. Bachara, K. Blachnicki, and K. Zielinski. "Framework for Application Management with Dynamic Aspects J-EARS Case Study". In: *Information and Software Technology* 52.1 (Jan. 2010), pp. 67–78. ISSN: 0950-5849. DOI: *10.1016/j.infsof.2009.06.003*. URL: *http://dx.doi.org/10.1016/j.infsof.2009.06.003* – 4 citations
9. J. Floch et al. "A Comprehensive Engineering Framework for Guaranteeing Component Compatibility". In: *Journal of Systems and Software* 83.10 (Oct. 2010), pp. 1759–1779. ISSN: 0164-1212. DOI: *10.1016/j.jss.2010.04.075*. URL: *http://dx.doi.org/10.1016/j.jss.2010.04.075* – 3 citations
10. S. Deorowicz. "Solving Longest Common Subsequence and Related Problems on Graphical Processing Units". In: *Software: Practice and Experience* 40.8 (2010), pp. 673–700. ISSN: 0038-0644. DOI: *10.1002/spe.v40:8*. URL: *http://dx.doi.org/10.1002/spe.v40:8* – 3 citations
11. A. Zalewski and S. Kijas. "Beyond ATAM: Early Architecture Evaluation Method for Large-scale Distributed Systems". In: *Journal of Systems and Software* 86.3 (Mar. 2013), pp. 683–697. ISSN: 0164-1212. DOI: *10.1016/j.jss.2012.10.923*. URL: *http://dx.doi.org/10.1016/j.jss.2012.10.923* – 2 citations
12. K. Łukasiewicz and J. Miler. "Improving agility and discipline of software development with the Scrum and CMMI". in: *Software, IET* 6.5 (2012), pp. 416–422. ISSN: 1751-8806. DOI: *10.1049/iet-sen.2011.0193* – 2 citations
13. M. Janicki, M. Katara, and T. Pääkkönen. "Obstacles and Opportunities in Deploying Model-based GUI Testing of Mobile Software: A Survey". In: *Software Testing, Verification and Reliability* 22.5 (Aug. 2012), pp. 313–341. ISSN: 0960-0833. DOI: *10.1002/stvr.460*. URL: *http://dx.doi.org/10.1002/stvr.460* – 2 citations
14. M. Ochodek, B. Alchimowicz, J. Jurkiewicz, and J. Nawrocki. "Improving the Reliability of Transaction Identification in Use Cases". In: *Information and Software Technology* 53.8 (2011), pp. 885–897. ISSN: 0950-5849. DOI: *10.1016/j.infsof.2011.02.004*. URL: *http://dx.doi.org/10.1016/j.infsof.2011.02.004* – 2 citations
15. R. Hofman. "Behavioral Economics in Software Quality Engineering". In: *Empirical Software Engineering* 16.2 (Apr. 2011), pp. 278–293. ISSN: 1382-3256. DOI: *10.1007/s10664-010-9140-x*. URL: *http://dx.doi.org/10.1007/s10664-010-9140-x* – 2 citations
16. J. Jurkiewicz, J. Nawrocki, M. Ochodek, and T. Głowacki. "HAZOP based identification of events in use cases". English. In: *Empirical Software Engineering* (2013), pp. 1–28. ISSN: 1382-3256. DOI: *10.1007/s10664-013-9277-5*. URL: *http://dx.doi.org/10.1007/s10664-013-9277-5* – 1 citation
17. A. Riel, A. Draghici, G. Draghici, D. Grajewski, and R. Messnarz. "Process and product innovation needs integrated engineering collaboration skills". In: *Journal of Software: Evolution and Process* 24.5 (2012), pp. 551–560. ISSN: 2047-7481. DOI: *10.1002/smr.497*. URL: *http://dx.doi.org/10.1002/smr.497* – 1 citation
18. P. Janczarek and J. Sosnowski. "Investigating software testing and maintenance reports: Case study". In: *Information and Software Technology* 0 (2014), pp. –. ISSN: 0950-5849. DOI:

http://dx.doi.org/10.1016/j.infsof.2014.06.015. URL: *http://www.sciencedirect.com/science/article/pii/S0950584914001542* – 0 citations

19. L. Madeyski and M. Jureczko. "Which Process Metrics Can Significantly Improve Defect Prediction Models? An Empirical Study". In: *Software Quality Journal* (accepted) (2014). DOI: *10.1007/s11219-014-9241-7*. URL: *http://dx.doi.org/10.1007/s11219-014-9241-7* – 0 citations
20. L. Madeyski, W. Orzeszyna, R. Torkar, and M. Józala. "Overcoming the Equivalent Mutant Problem: A Systematic Literature Review and a Comparative Experiment of Second Order Mutation". In: *IEEE Transactions on Software Engineering* 40.1 (2014), pp. 23–42. ISSN: 0098-5589. DOI: *10.1109/TSE.2013.44*. URL: *http://dx.doi.org/10.1109/TSE.2013.44* – 0 citations
21. J. Sobecki. "Comparison of Selected Swarm Intelligence Algorithms in Student Courses Recommendation Application". In: *International Journal of Software Engineering and Knowledge Engineering* 24.01 (2014), pp. 91–109 – 0 citations
22. B. Czarnacka-Chrobot. "RATIONALIZATION OF BUSINESS SOFTWARE SYSTEMS DEVELOPMENT AND ENHANCEMENT PROJECTS INVESTMENT DECISIONS ON THE BASIS OF FUNCTIONAL SIZE MEASUREMENT". in: *International Journal of Software Engineering and Knowledge Engineering* 23.06 (2013), pp. 839–868. URL: *http://www.worldscientific.com/doi/abs/10.1142/S0218194013500228* – 0 citations
23. T. Schulz, Ł. Radliński, T. Gorges, and W. Rosenstiel. "Predicting the Flow of Defect Correction Effort using a Bayesian Network Model". English. In: *Empirical Software Engineering* 18.3 (2013), pp. 435–477. ISSN: 1382-3256. DOI: *10.1007/s10664-011-9175-7*. URL: *http://dx.doi.org/10.1007/s10664-011-9175-7* – 0 citations
24. P. Kosiuczenko. "Specification of Invariability in OCL". in: *Software and Systems Modeling* 12.2 (May 2013), pp. 415–434. ISSN: 1619-1366. DOI: *10.1007/s10270-011-0215-y*. URL: *http://dx.doi.org/10.1007/s10270-011-0215-y* – 0 citations
25. W. Pedrycz. "KNOWLEDGE MANAGEMENT AND SEMANTIC MODELING: A ROLE OF INFORMATION GRANULARITY". in: *International Journal of Software Engineering and Knowledge Engineering* 23.01 (2013), pp. 5–11. URL: *http://www.worldscientific.com/doi/abs/10.1142/S0218194013400019* – 0 citations
26. J. J. Jung, R. P. Katarzyniak, and N. T. Nguyen. "GUEST EDITORS; INTRODUCTION". in: *International Journal of Software Engineering and Knowledge Engineering* 23.01 (2013), pp. 1–3. DOI: *10.1142/S0218194013020014* – 0 citations
27. R. P. Katarzyniak and G. Popek. "INTEGRATION OF MODAL AND FUZZY METHODS OF KNOWLEDGE REPRESENTATION IN ARTIFICIAL AGENTS". in: *International Journal of Software Engineering and Knowledge Engineering* 23.01 (2013), pp. 13–29. DOI: *10.1142/S0218194013400020*. URL: *http://www.worldscientific.com/doi/abs/10.1142/S0218194013400020* – 0 citations
28. M. Psiuk, D. Żmuda, and K. Zielinski. "Distributed OSGi Built over Message-oriented Middleware". In: *Software: Practice and Experience* 43.1 (Jan. 2013), pp. 1–31. ISSN: 0038-0644. DOI: *10.1002/spe.1148*. URL: *http://dx.doi.org/10.1002/spe.1148* – 0 citations

We repeated the search process for different countries (changing the part of the search string responsible for affiliation, e.g. from `AFFIL(poland)` into `AFFIL(germany)`) as well as the world (removing the part of the search string responsible for affiliation). Figure 1.1 presents a map of Europe including bubbles with bubble size proportional to the contribution of each country.

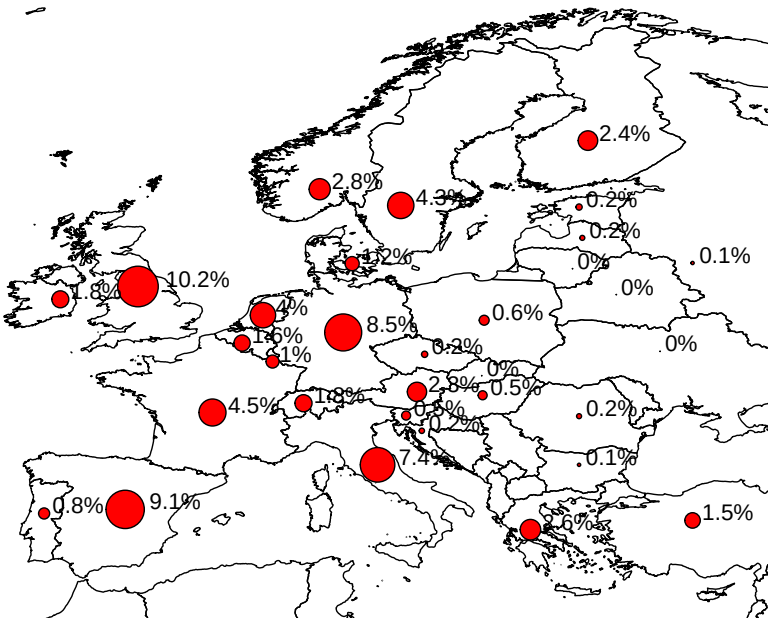


Figure 1.1. How European countries contribute to leading software engineering journals.

United Kingdom, Spain and Germany contribute the most in terms of the number of papers published in the analyzed set of leading software engineering journals and time frame, i.e. 10.2%, 9.1% and 8.5% respectively. It is worth mentioning that contribution of United States is about 21%.

Polish contribution (0.6%) is slightly less than the contribution of Portugal (0.8%), which cannot be considered an achievement taking into account that Poland is bigger than Portugal. Polish contribution is spread among 13 research institutions (e.g., Wroclaw University of Technology, AGH University of Technology, Poznan University of Technology, Warsaw University of Technology) and one software development company (Nokia Siemens Networks).

It would be interesting to check whether there are any interesting trends with regard to the contribution of Polish researchers in last years. A subsequent analysis presented in Figure 1.2 shows how selected European countries contributed to the analyzed set of journals in successive years. We cannot see any specific trend with regard to Polish contribution in the last five years.

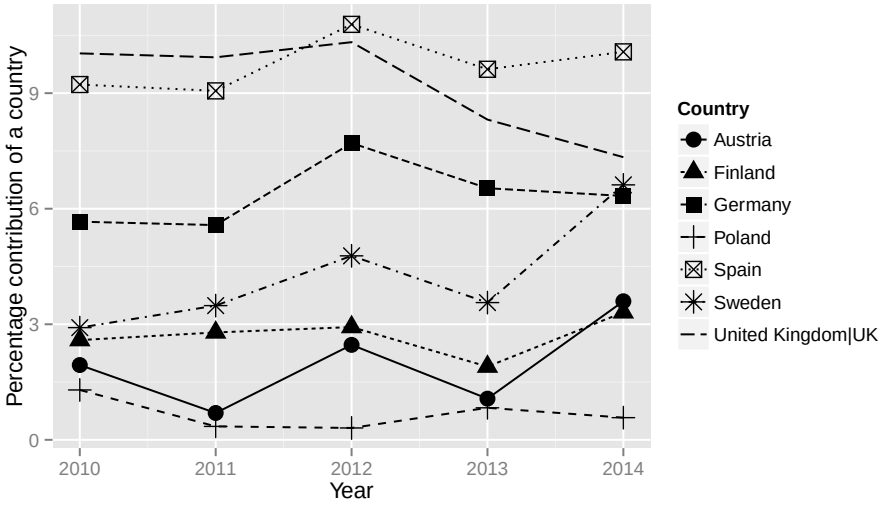


Figure 1.2. Percentage contribution of European countries to leading software engineering journals.

1.2. Identification of events in use cases

1.2.1. Research context

There are many quality attributes of requirements specifications, one of them is completeness. If one considers use cases for description of functional requirements, it is important to include complete list of events which may interrupt

main scenarios. Missing events can lead to higher project costs and overrunning schedule. Therefore, a question arises: *what is the effective and efficient method to identify events in use cases?* No specific method, aimed at identification of events in use cases, had been found, hence, as the first step, method based on HAZOP approach has been proposed and evaluated in comparison to the ad hoc approach [JNOG13b]. As the second step, automatic method of events identification has been proposed and evaluated.

1.2.2. Research objectives

The goal of this study was to propose methods aimed at identification of events in use cases. Moreover, these methods have been evaluated from the stand point of accuracy and speed.

1.2.3. Research methods

HAZOP method has been used as a fundament for the proposed H4U method, which is aimed at identification of events in use cases. H4U uses the notion of primary and secondary keywords in the process of analysis of use cases. In order to evaluate the proposed approach, two controlled experiments have been designed and conducted. In both experiments the H4U method has been compared to the ad hoc approach. Participants of the first experiment included 18 students and in the second experiment 64 IT professionals were involved. In both experiments, the accuracy and speed of the two approaches have been measured and evaluated. Moreover, an automatic method of events identification has been proposed. In order to elaborate this method, 160 use cases from software projects have been analyzed. This analysis let to naming 14 abstract event types and two inference rules. The automatic method has been evaluated from the point of view of speed and accuracy. Moreover, linguistic quality of the automatically identified events has been assessed in an experiment based on the assumptions of Turing-test. Benchmark use-case-based requirements specification was used in the evaluations of ad hoc approach, H4U method and automatic method.

1.2.4. Research results

In the first place, H4U method has been evaluated with comparison to the ad hoc approach. The first experiment (with students) showed that H4U method allows to achieve more accurate results. However, the participants who used the H4U method were slower in the analysis of use cases than the participants who used ad hoc approach. The second experiment (with IT professionals) con-

firmed these results. The results from both experiments showed that the accuracy of events identification ranged from 0.15 to 0.26. Experiment concerning the proposed automatic method of events identification showed that this method can achieve accuracy at the level of 0.8, which is better than manual approaches. Moreover, automatic method is faster than manual methods, i.e., it is able to analyze 10.8 steps per minute, while participant of the experiments were able to analyze on average 2.5 steps per minute with ad hoc approach and 0.5 steps per minute with H4U method. In terms of linguistic quality of the automatically identified events, it can be concluded that the understandability of event descriptions generated by computer was not worse than understandability of event descriptions written by humans.

1.2.5. Conclusions

The proposed H4U method, aimed at identification of use-case events, provides effective alternative to the ad hoc approach in terms of accuracy of event identification. The accuracy and speed of identification of events can be further improved by using the proposed automatic method.

1.3. Solving the invariability problem in OCL

1.3.1. Research context

There exist various methods and languages for the specification and modeling of object-oriented systems. Contracts are the prevailing way of specifying systems from the caller point of view (see [Mey88]). The Unified Modeling Language (UML) [OMG11] is often used in combination with the Object Constraint Language (OCL) [OMG12], a high-level language for a contractual specification of object-oriented systems. In OCL, one can express invariants and operations' pre- and post-conditions.

The specification of invariable system parts is a well known problem. Usually when a change of a large system happens, only its small part is modified and the rest remains unchanged. In case of complex systems one needs a means for avoiding extensive specification of those invariable parts. This is the so called *frame problem*. In case of object-oriented systems, one has to specify what happens with all objects' attributes and associations. However, without a proper means the resulting specification can be, and often is, very excessive. For a number of years this problem remained unsolved for OCL.

The frame problem

In general there exist four approaches to the frame problem: minimal-change approach, implicit specification and frame formulas. The minimal-change approach requires that the set of changed system parts is minimal, i.e., the change is in accordance with the specification in the usual sense and moreover the number of changed parts cannot be smaller. A serious disadvantage of this approach is that it is very hard to figure out such minimal sets, and the minimality proof can be complex and non-standard. Thus it is not useful in practice, and specially when tool support is needed

An implicit approach to invariability was used in case of OCL, however it dates back to Hoare logic. In this logic all variables which are not mentioned in the so called Hoare triple are assumed to be unchanged. The idea of the approach is similar: all system parts not mentioned in a specification must not change. This approach allows to write simple specifications and does not require any special means. However, it does not work well for contractual specifications because an operation execution can have very complex side-effects. This approach also heavily depends on the actual form of the specification and for specifications equivalent in the classical logical sense it may result in different variable parts.

The frame formulas are used in artificial intelligence (cf. [Sch90]). The idea is to specify modification of attributes using axiom schemata. This approach requires however proper means to make the specifications compact. In case of Java Modeling Language (JML, see [DM05]) and also Spec#, explicit invariability clauses are used for a compact specification of invariable properties. Invariability constraints can be checked at the compile-time. Thus, it is not possible to specify invariability requirements which cannot be checked statically. Moreover, these languages are much simpler than OCL.

1.3.2. Research objectives

The goal of research [Kos13c; Kos13a] was to provide specification primitives which address the shortcomings of previously existing approaches. In particular, these primitives should:

— allow one to specify the invariable part of object-oriented systems
— be language-based, not semantics-based, preferably OCL-based
— allow validity monitoring with standard OCL-tools
— ensure logical equivalence of OCL-specifications in the standard sense
— allow for the application of standard proof techniques

1.3.3. Research results

A state of an object-oriented system can be understood as a graph with labeled nodes and labeled edges. The nodes correspond to objects, their labels to their classes. The edges correspond to links between objects; the corresponding labels to class attributes and associations among them. A state change of an oo-system corresponds to a state change of such a labeled graph. Such a change concerns object creation and deletion, and also link modification. We let the object creation and deletion be governed by the OCL-specification. However we add new primitives to identify sets of those links which can be removed or replaced by new ones. Thus, a system change, object removal and creation as well as link modification can happen as long as the basic OCL-specification is satisfied and the links are modified only when they are specified by the primitives.

An OCL-specification of an operation has basically three parts. The first part of the specification declares the context, i.e., the signature of specified method and the class it belongs to. The second part specifies its pre-condition, i.e., the condition which has to be satisfied before the operation is executed. The third part is the post-condition, i.e., a condition which must be valid after the operation's execution. We add to operations' post-conditions invariability clauses of the form:

`in p modifies` t_1`::`a_1`, ...,` t_n`::`a_n

Clause `p` defines a set of classes; in general it can be a metamodel-based view definition [Kos13c]. Term t_i is an OCL-term defining a set of objects of a class C_i and a_i is an attribute or an association-end of this class. For objects defined by term t_i attribute/association a_i can be changed. Attribute a_i has to belong to classes defined by `p`. We do not specify in this clause, what happens with attributes not included in `p`.

As an example, consider a bank account with attribute `balance` storing information on the actual balance of a bank account and method `credit`. The way this method operates can be specified in the extended OCL in the following way:

```
context BankAccount::credit(amount :  Real)
post :  self.balance = self.balance@pre + amount
in BankAccount modifes :  self::balance
```

The primitive `@pre` can occur only in post-conditions. It delivers the value of attribute `balance` in the pre-state, i.e., in the state before the method execution. When it does not occur as a postfix of an attribute, then the value of attribute is computed in the post-state. Operation `credit` increases attribute `balance` by adding `amount`. The invariability clause says that `credit` modifies only the attribute `balance` of objects from class `BankAccount`.

1.3.4. Conclusions

Specification of invariability was a real problem in case of OCL-specifications. The problem with designing invariability primitives was the descriptive power of OCL and the plenitude of constructs facilitating specification writing. As a result of the presented research, a solution of this problem was proposed, which is simple in form and has natural semantics. It allows one for compact specifications of invariable system parts in a compact and precise way. The semantics is defined in terms of standard OCL; this allows for the application of standard OCL-models, techniques and tools. However, there are still issues to be addressed. For example, we need to define primitives for associations with multiple ends.

1.4. Predicting the Flow of Defect Correction Effort

1.4.1. Research context

Extensive literature on defect prediction usually deals with predicting number of defects or defect proneness of a software component. While such information is useful in many contexts it does not answer the question that is more important from the resource management perspective, i.e., *how much effort will be required to correct these defects?* The described study investigated possibility of predicting correction effort instead of raw defect count. [SRGR13b]

The environment for this study was the automotive company where software is developed according to the industrial standard V-model [Ind92] with four successive phases: requirements (RE), design (DE), implementation (IM), integration and testing (I&T). An earlier study [SRGR11] had confirmed that defect correction effort (DCE) depends on the phases where a defect was inserted and detected. Specifically, defects inserted in early phase, but detected in later, need more effort for their correction than if they are inserted and detected in the same phase. This flow of defect correction effort between phases makes the main rationale for the proposed predictive model.

1.4.2. Research objectives

The main goal of this study was to develop a model that could predict the defect correction effort at various development phases. This model, called a Defect Cost Flow Model (DCFM) reflects a V-model of a software development lifecycle – a real engineering process for developing embedded applications in the automotive industry. With this model it was possible to **optimize the amount of quality assurance** (QA) activities in different phases to minimize the total project effort.

1.4.3. Research methods

Technically, the DCFM is a Bayesian Network (BN). Among various reasons for choosing a Bayesian Network as a formal representation of DCFM the most important were:

— Model structure reflects cause-effect relationships for better understanding and fit to reality.
— BNs may incorporate expert knowledge combined with empirical data.
— They enable performing various types of analyses using rigorous probability calculus focused on decision support.

The research process involved the following main phases:

1. Problem definition using the Goal-Question-Metric approach.
2. Data gathering and analysis – using the internal *change and defect management system* as main source. The second data source was expert knowledge from researchers, developers and managers supporting this study. Exiting literature in the field served as the third data source.
3. Model creation and enhancement covered building initial version of the model as well as its multiple enhancements. Each version contained new elements (i.e. variables) and the whole model was calibrated using the data obtained in the previous phase. The model was created in an iterative process for easier validation and access to the working (partial) version at each time.
4. Model validation covered general model behavior, practical usefulness, detailed model behavior in numerous scenarios with different input data, sensitivity analysis, and in the possibility of adjusting and calibrating the core of DCFM.

1.4.4. Research results

The main result was the Defect Cost Flow model. Its structure is too large for display and discuss in a single figure. Thus, Figure 1.3 illustrates the core structure of the model while Figure 1.4 presents some details for the design phase. The defect correction effort flows from the phase where defects are inserted until they are detected and fixed.

Specifically, some defects inserted in the requirements phase are also fixed there. But since the review process is imperfect some defects are left and thus flow to the next phase (design). Correcting these defects in this phase requires more effort (4-5 times) as reflected by the *effort multiplier* (Figure 1.4 left). With higher *level of QA activity* more defects could be detected in the design phase. But still, some would be left and detected in later phases.

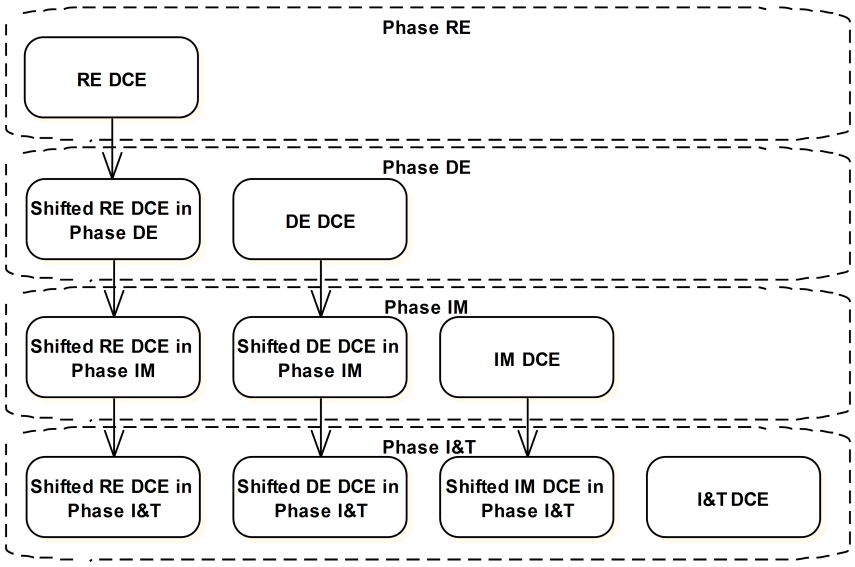


Figure 1.3. Schematic of DCFM [SRGR13b].

In the design phase also new defects are inserted. They would need to be detected and fixed – partially in the QA activity, and partially in the next phases as the defect correction effort (Figure 1.4 right).

The model incorporates various empirical data, e.g.:

— The probabilities for inserting defects slowly decreases in the first three phases and drops down rapidly in the I&T phase.
— Different levels for sufficiency of QA effort are defined as a percentage of the core development effort.
— The QA activities are the most efficient in the RE and DE phases. In IM and I&T phases they are significantly less efficient in detecting defects originating from earlier phases.

1.4.5. Conclusions

Model validation confirms that the model provides sensible predictions consistent with gathered empirical data and known literature in software engineering field. What is especially important is that this model has been applied in a real in-

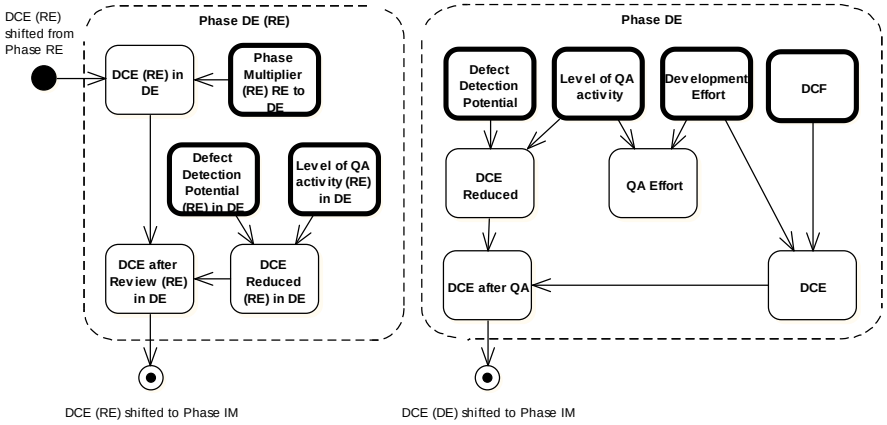


Figure 1.4. Model structure for design phase (DE) [SRGR13b].

dustrial process. It demonstrates high potential in finding the appropriate amount of review effort for specific development phases to minimize the overall costs. Thus, the model may be used in the industry for decision support. By extending and calibrating it can be tailored to meet the needs of specific development environment.

1.5. Model of a maturity capsule in software project management

1.6. Research context

Research conducted at the Center for Advanced Studies on Campus (CAS) focuses on issues of software project management and on finding solutions to improve management and development processes. The development methods used in project management and in the development environment prove inadequate to the problems of contemporary IT projects. It is therefore proposed that in project management the development and management processes should be monitored with the use of the innovative maturity capsule developed at CAS.

1.6.1. Research objectives

The main objective of the study was to define and apply in practice the maturity capsule in IT projects. It was assumed that the concept of the maturity capsule is to be understood as a set of maturity ratings of the supplier, client and project

(estimated through the scalar negentropy of the project) [KO14a]. To define the maturity capsule, it is necessary to establish the measurements of the maturity of the client and the supplier and the project negentropy. The knowledge resulting from the COBIT (Control Objectives for Information and Related Technology) and ITIL (Information Technology Infrastructure Library) standards is important for the initial processing of the project data, which aims at evaluating the maturity of the supplier and client organizations in question. The TOGAF standard is used mainly to evaluate a specific indicator, measuring the degree of global maturity of a project, called negentropy. The applicability of the model is verified in a number of environments, mainly in IT projects and in the organizations carrying out such projects [KO14b].

1.6.2. Research methods

The description of the project management processes and information technologies was based on a formal, discrete - time-linear dynamic description expanded with the essential nonlinear mechanisms in the form of a fuzzy - rule-based system (with the use of a linguistic estimation developed on the basis of answers given by experts to sets of questions in interview questionnaires). These descriptions were used to develop a useful model based on a sequence of the following three steps [SO14]:

1. the fuzzy modeling philosophy, based on the formation of the membership function, is an appropriate foundation of the universality of the maturity capsule.
2. the number of times of use is a relevant criterion for assessing the quality of the developed model.
3. the tuning of the model becomes possible through the identification of its parameters and variables (it is based on linguistic evaluations resulting from competency questions).

1.6.3. Research results

The conducted studies demonstrated how the maturity capsule can be used by those managing projects, by development teams and by customer teams in order to support the processes inside the project in terms of monitoring and predicting its development. Four levels of verification of the maturity capsule were proposed [Orł14]. On the first one, the usefulness of the capsule for the managers of IT projects was evaluated. On the second level, the support for processes which ensure corporate governance was referred to, as well as the use of project negentropy

in supporting the management processes of a company. The third level focused on the linguistic evaluation of supplier organization maturity in predicting its evolution. While, on the fourth level of 'control of the level of the client organization and the processes of its change, a linguistic description was used to support this evaluation.

1.6.4. Conclusions

In previous studies on project management processes, neither the analysis of the state nor the maturity of the project was as comprehensive as this one. Both elements have been included in the maturity capsule to predict and optimize information technologies in managing information technology projects. The presented analysis of the maturity capsule, the possibility to progress in terms of maturity, and the monitoring of the level of management all allow for predicting technologies to support the desired changes in the maturity capsule. In this sense, the developed solution provides an innovative perspective on the management processes of technologies and IT projects, involving the aggregation of knowledge about the maturity of the entities in the capsule (client, project and supplier) and the decomposition of information technologies into services and IT functionalities. The solution described in this work, regarding a comprehensive evaluation of a project and involving the use of the maturity capsule, requires indicating how frequent this evaluation is and analyzing its applicability for the organization/teams of the client and the supplier, which change dynamically during the project. In such cases, the standard use of evaluation questionnaires may be inadequate. A better solution would be to develop and apply a system in which specialized agents make the assessment. The purpose of such a system would be to evaluate the environment in which the IT project is carried out.

1.7. Conclusions

The contribution of Polish researchers to the software engineering research field is limited. The percentage of research papers in the analyzed set of leading software engineering journals and the time period (2010 – 2014 Sep 14) was about 0.6%. Fortunately, there are some valuable achievements of Polish researchers which we tried to present briefly in this chapter.

It is also worth mentioning that we did not analyzed which research institutions in Poland contribute the most as sometimes changing an affiliation of one or two researchers would influence the results to a large extent.

References

[Gla94] R. L. Glass. "An Assessment of Systems and Software Engineering Scholars and Institutions". In: *Journal of Systems and Software* 27.1 (Oct. 1994), pp. 63–67. ISSN: 0164-1212. DOI: *10.1016/0164-1212(94)90115-5*. URL: *http://dx.doi.org/10.1016/0164-1212(94)90115-5*.

[WTGBC09] W. E. Wong, T. H. Tse, R. L. Glass, V. R. Basili, and T. Y. Chen. "Controversy Corner: An Assessment of Systems and Software Engineering Scholars and Institutions (2002-2006)". In: *Journal of Systems and Software* 82.8 (Aug. 2009), pp. 1370–1373. ISSN: 0164-1212. DOI: *10.1016/j.jss.2009.06.018*. URL: *http://dx.doi.org/10.1016/j.jss.2009.06.018*.

[WTGBC11] W. E. Wong, T. Tse, R. L. Glass, V. R. Basili, and T. Chen. "An assessment of systems and software engineering scholars and institutions (2003–2007 and 2004–2008)". In: *Journal of Systems and Software* 84.1 (2011). Information Networking and Software Services, pp. 162 –168. ISSN: 0164-1212. DOI: *http://dx.doi.org/10.1016/j.jss.2010.09.036*. URL: *http://www.sciencedirect.com/science/article/pii/S0164121210002682*.

[Woh09] C. Wohlin. "An Analysis of the Most Cited Articles in Software Engineering Journals - 2002". In: *Information and Software Technology* 51.1 (Jan. 2009), pp. 2–6. ISSN: 0950-5849. DOI: *10.1016/j.infsof.2008.09.012*. URL: *http://dx.doi.org/10.1016/j.infsof.2008.09.012*.

[Mad10] L. Madeyski. "The impact of test-first programming on branch coverage and mutation score indicator of unit tests: An experiment". In: *Information and Software Technology* 52.2 (2010), pp. 169–184. DOI: *10.1016/j.infsof.2009.08.007*. URL: *http://dx.doi.org/10.1016/j.infsof.2009.08.007*.

[ONK11] M. Ochodek, J. Nawrocki, and K. Kwarciak. "Simplifying Effort Estimation Based on Use Case Points". In: *Information and Software Technology* 53.3 (Mar. 2011), pp. 200–213. ISSN: 0950-5849. DOI: *10.1016/j.infsof.2010.10.005*. URL: *http://dx.doi.org/10.1016/j.infsof.2010.10.005*.

[MR10] L. Madeyski and N. Radyk. "Judy – A Mutation Testing Tool for Java". In: *IET Software Journal (formerly IEE Proceedings*

Software) 4.1 (2010). Draft: *http://madeyski.e-informatyka.pl/download/Madeyski10b.pdf*, pp. 32–42. DOI: *10.1049/iet-sen.2008.0038*. URL: *http://dx.doi.org/10.1049/iet-sen.2008.0038*.

[JZ10a] A. Janik and K. Zielinski. "AAOP-based Dynamically Reconfigurable Monitoring System". In: *Information and Software Technology* 52.4 (Apr. 2010), pp. 380–396. ISSN: 0950-5849. DOI: *10.1016/j.infsof.2009.10.006*. URL: *http://dx.doi.org/10.1016/j.infsof.2009.10.006*.

[NK12] G. J. Nalepa and K. Kluza. "UML REPRESENTATION FOR RULE-BASED APPLICATION MODELS WITH XTT2-BASED BUSINESS RULES". In: *International Journal of Software Engineering and Knowledge Engineering* 22.04 (2012), pp. 485–524. DOI: *10.1142/S021819401250012X*. URL: *http://www.worldscientific.com/doi/abs/10.1142/S021819401250012X*.

[Mił10] M. Miłkowski. "Developing an Open-source, Rule-based Proofreading Tool". In: *Software: Practice and Experience* 40.7 (June 2010), pp. 543–566. ISSN: 0038-0644. DOI: *10.1002/spe.v40:7*. URL: *http://dx.doi.org/10.1002/spe.v40:7*.

[JZ10b] A. Janik and K. Zielinski. "Adaptability Mechanisms for Autonomic System Implementation with AAOP". In: *Software: Practice and Experience* 40.3 (Mar. 2010), pp. 209–223. ISSN: 0038-0644. DOI: *10.1002/spe.v40:3*. URL: *http://dx.doi.org/10.1002/spe.v40:3*.

[BBZ10] P. Bachara, K. Blachnicki, and K. Zielinski. "Framework for Application Management with Dynamic Aspects J-EARS Case Study". In: *Information and Software Technology* 52.1 (Jan. 2010), pp. 67–78. ISSN: 0950-5849. DOI: *10.1016/j.infsof.2009.06.003*. URL: *http://dx.doi.org/10.1016/j.infsof.2009.06.003*.

[Flo+10] J. Floch et al. "A Comprehensive Engineering Framework for Guaranteeing Component Compatibility". In: *Journal of Systems and Software* 83.10 (Oct. 2010), pp. 1759–1779. ISSN: 0164-1212. DOI: *10.1016/j.jss.2010.04.075*. URL: *http://dx.doi.org/10.1016/j.jss.2010.04.075*.

[Deo10] S. Deorowicz. "Solving Longest Common Subsequence and Related Problems on Graphical Processing Units". In: *Software: Practice and Experience* 40.8 (2010), pp. 673–700. ISSN: 0038-0644.

DOI: *10.1002/spe.v40:8*. URL: *http://dx.doi.org/10.1002/spe.v40:8*.

[ZK13] A. Zalewski and S. Kijas. "Beyond ATAM: Early Architecture Evaluation Method for
Large-scale Distributed Systems". In: *Journal of Systems and Software* 86.3 (Mar. 2013), pp. 683–697. ISSN: 0164-1212. DOI: *10.1016/j.jss.2012.10.923*. URL: *http://dx.doi.org/10.1016/j.jss.2012.10.923*.

[ŁM12] K. Łukasiewicz and J. Miler. "Improving agility and discipline of software development with the Scrum and CMMI". In: *Software, IET* 6.5 (2012), pp. 416–422. ISSN: 1751-8806. DOI: *10.1049/iet-sen.2011.0193*.

[JKP12] M. Janicki, M. Katara, and T. Pääkkönen. "Obstacles and Opportunities in Deploying
Model-based GUI Testing of Mobile Software: A Survey". In: *Software Testing, Verification and Reliability* 22.5 (Aug. 2012), pp. 313–341. ISSN: 0960-0833. DOI: *10.1002/stvr.460*. URL: *http://dx.doi.org/10.1002/stvr.460*.

[OAJN11] M. Ochodek, B. Alchimowicz, J. Jurkiewicz, and J. Nawrocki. "Improving the Reliability of Transaction Identification in Use Cases". In: *Information and Software Technology* 53.8 (2011), pp. 885–897. ISSN: 0950-5849. DOI: *10.1016/j.infsof.2011.02.004*. URL: *http://dx.doi.org/10.1016/j.infsof.2011.02.004*.

[Hof11] R. Hofman. "Behavioral Economics in Software Quality Engineering". In: *Empirical Software Engineering* 16.2 (Apr. 2011), pp. 278–293. ISSN: 1382-3256. DOI: *10.1007/s10664-010-9140-x*. URL: *http://dx.doi.org/10.1007/s10664-010-9140-x*.

[JNOG13a] J. Jurkiewicz, J. Nawrocki, M. Ochodek, and T. Głowacki. "HAZOP based identification of events in use cases". English. In: *Empirical Software Engineering* (2013), pp. 1–28. ISSN: 1382-3256. DOI: *10.1007/s10664-013-9277-5*. URL: *http://dx.doi.org/10.1007/s10664-013-9277-5*.

[RDDGM12] A. Riel, A. Draghici, G. Draghici, D. Grajewski, and R. Messnarz. "Process and product innovation needs integrated engineering collaboration skills". In: *Journal of Software: Evolution and Process* 24.5 (2012), pp. 551–560. ISSN: 2047-7481. DOI: *10.1002/smr.497*. URL: *http://dx.doi.org/10.1002/smr.497*.

[JS14] P. Janczarek and J. Sosnowski. "Investigating software testing and maintenance reports: Case study". In: *Information and Software Technology* 0 (2014), pp. –. ISSN: 0950-5849. DOI: *http://dx.doi.org/10.1016/j.infsof.2014.06.015*. URL: *http://www.sciencedirect.com/science/article/pii/S0950584914001542*.

[MJ14] L. Madeyski and M. Jureczko. "Which Process Metrics Can Significantly Improve Defect Prediction Models? An Empirical Study". In: *Software Quality Journal* (accepted) (2014). DOI: *10.1007/s11219-014-9241-7*. URL: *http://dx.doi.org/10.1007/s11219-014-9241-7*.

[MOTJ14] L. Madeyski, W. Orzeszyna, R. Torkar, and M. Józala. "Overcoming the Equivalent Mutant Problem: A Systematic Literature Review and a Comparative Experiment of Second Order Mutation". In: *IEEE Transactions on Software Engineering* 40.1 (2014), pp. 23–42. ISSN: 0098-5589. DOI: *10.1109/TSE.2013.44*. URL: *http://dx.doi.org/10.1109/TSE.2013.44*.

[Sob14] J. Sobecki. "Comparison of Selected Swarm Intelligence Algorithms in Student Courses Recommendation Application". In: *International Journal of Software Engineering and Knowledge Engineering* 24.01 (2014), pp. 91–109.

[CC13] B. Czarnacka-Chrobot. "RATIONALIZATION OF BUSINESS SOFTWARE SYSTEMS DEVELOPMENT AND ENHANCEMENT PROJECTS INVESTMENT DECISIONS ON THE BASIS OF FUNCTIONAL SIZE MEASUREMENT". In: *International Journal of Software Engineering and Knowledge Engineering* 23.06 (2013), pp. 839–868. URL: *http://www.worldscientific.com/doi/abs/10.1142/S0218194013500228*.

[SRGR13a] T. Schulz, Ł. Radliński, T. Gorges, and W. Rosenstiel. "Predicting the Flow of Defect Correction Effort using a Bayesian Network Model". English. In: *Empirical Software Engineering* 18.3 (2013), pp. 435–477. ISSN: 1382-3256. DOI: *10.1007/s10664-011-9175-7*. URL: *http://dx.doi.org/10.1007/s10664-011-9175-7*.

[Kos13b] P. Kosiuczenko. "Specification of Invariability in OCL". In: *Software and Systems Modeling* 12.2 (May 2013), pp. 415–434. ISSN: 1619-1366. DOI: *10.1007/s10270-011-0215-y*. URL: *http://dx.doi.org/10.1007/s10270-011-0215-y*.

[Ped13] W. Pedrycz. “KNOWLEDGE MANAGEMENT AND SEMANTIC MODELING: A ROLE OF INFORMATION GRANULARITY”. In: *International Journal of Software Engineering and Knowledge Engineering* 23.01 (2013), pp. 5–11. URL: *http://www.worldscientific.com/doi/abs/10.1142/S0218194013400019*.

[JKN13] J. J. Jung, R. P. Katarzyniak, and N. T. Nguyen. “GUEST EDITORS; INTRODUCTION”. In: *International Journal of Software Engineering and Knowledge Engineering* 23.01 (2013), pp. 1–3. DOI: *10.1142/S0218194013020014*.

[KP13] R. P. Katarzyniak and G. Popek. “INTEGRATION OF MODAL AND FUZZY METHODS OF KNOWLEDGE REPRESENTATION IN ARTIFICIAL AGENTS”. In: *International Journal of Software Engineering and Knowledge Engineering* 23.01 (2013), pp. 13–29. DOI: *10.1142/S0218194013400020*. URL: *http://www.worldscientific.com/doi/abs/10.1142/S0218194013400020*.

[PŻZ13] M. Psiuk, D. Żmuda, and K. Zielinski. “Distributed OSGi Built over Message-oriented Middleware”. In: *Software: Practice and Experience* 43.1 (Jan. 2013), pp. 1–31. ISSN: 0038-0644. DOI: *10.1002/spe.1148*. URL: *http://dx.doi.org/10.1002/spe.1148*.

[JNOG13b] J. Jurkiewicz, J. Nawrocki, M. Ochodek, and T. Głowacki. “HAZOP based identification of events in use cases”. In: *Empirical Software Engineering* (2013), pp. 1–28.

[Mey88] B. Meyer. *Object-oriented software construction*. Prentice Hall New York, 1988.

[OMG11] OMG. *Unified Modeling Language, Spec. ver. 2.4.1*. 2011.

[OMG12] OMG. *Object Constraint Language, Spec. ver. 2.3.1*. 2012.

[Sch90] L. Schubert. “Monotonic solution of the frame problem in the situation calculus”. In: *Knowledge representation and defeasible reasoning*. Springer, 1990, pp. 23–67.

[DM05] A. Darvas and P. Müller. “Reasoning about method calls in JML specifications”. In: *Proceedings of FTfJP’05*. 2005.

[Kos13c] P. Kosiuczenko. “Specification of invariability in OCL, Specifying invariable system parts and views”. In: *Software & Systems Modeling* 12.2 (2013), pp. 415–434.

[Kos13a] P. Kosiuczenko. “On the Validation of Invariants at Runtime”. In: *Fundamenta Informaticae* 125.2 (2013), pp. 183–222.

[SRGR13b] T. Schulz, Ł. Radliński, T. Gorges, and W. Rosenstiel. "Predicting the Flow of Defect Correction Effort using a Bayesian Network Model". English. In: *Empirical Software Engineering* 18.3 (2013), pp. 435–477. ISSN: 1382-3256. DOI: *10.1007/s10664-011-9175-7*. URL: *http://dx.doi.org/10.1007/s10664-011-9175-7*.

[Ind92] Industrieanlagen-Betriebsgesellschaft (IABG). *The V-Model — General Directive 250. Software Development Standard for the German Federal Armed Forces*. Ottobrunn, Germany, 1992.

[SRGR11] T. Schulz, Ł. Radliński, T. Gorges, and W. Rosenstiel. "Software Process Model using Dynamic Bayesian Networks". In: *Knowledge Engineering for Software Development Life Cycles: Support Technologies and Applications*. Ed. by M. Ramachandran. Hershey: Information Science Reference, 2011, pp. 289–310. ISBN: 978-1-60960-509-4. DOI: *10.4018/978-1-60960-509-4.ch016*.

[KO14a] Z. Kowalczuk and C. Orłowski. *Advanced Modeling of Management Processes in Information Technology*. Springer, 2014, pp. 1–203. ISBN: 978-3-642-40876-2.

[KO14b] Z. Kowalczuk and C. Orłowski. "Model of a Maturity Capsule in Managing IT Projects". In: *Cybernetics and Systems* 45.2 (2014), pp. 123–135.

[SO14] E. Szczerbicki and C. Orłowski. "Guest Editorial: Designing and Developing Smart Cognitive Systems: Implementation Lessons from the Real World". In: *Cybernetics and Systems* 45.2 (2014), pp. 89–91.

[Orł14] C. Orłowski. "Rule-based model for selecting integration technologies for Smart Cities systems". In: *Cybernetics and Systems* 45.2 (2014), pp. 136–145.