

Preface

Back in 1968, the first NATO Conference on Software Engineering has announced the “software crisis”. Today we can see this as a “chronic affliction” of missed deadlines, extended budgets (poor productivity) and unsatisfied clients (poor quality). The cause of this affliction is the inherent complexity of software systems. Today, we can identify several major areas of research and practice that aim at overcoming this complexity and thus achieving an increase in productivity and quality of software in various application domains. First of the areas consists in increasing levels of abstraction for programming constructs. Recent approaches in this field include domain-specific languages, various requirements languages and domain modelling. Second of the areas consists in assuring a better quality of produced software. It includes various approaches to predict and eliminate defects by analysing code or by creating new testing methods. Third of the areas consists in optimising software development and operation cycles. This direction comprises such elements as continuous product delivery, frequent product releases, high end-user involvement and feedback, as well as tight integration of teams. The above three areas of software engineering are reflected in three parts of this volume.

The first part—Requirements and Domain Modelling—treats the problems concerning initial phases of software development, which are domain or business modelling, and requirements specification. The basic questions asked in this part are: “what requirements specification languages should be used?” and “how to represent requirements specifications in these languages to guarantee that they are complete and are equally understandable by all stakeholders?” The paper “Business Rule Patterns Catalog for Structural Business Rules basing on the OMG recommendation Semantics of Business Vocabulary and Business Rules” proposes a catalogue of patterns for structural business rules. The catalogue is intended to contain a limited number of patterns, which cover the most typical cases that can be found by a business analyst during the business rule identification process. The paper “Sketching Use-case Scenarios Based on Use-Case Goals and Patterns” is in some sense similar to the previous one but focuses on use cases. It describes and evaluates the idea of a catalogue containing use-case patterns as a basis for a

semi-automatic approach to generate use-case scenarios, and a programming tool supporting this generation. The next two papers are complementary to one another. The first one, “Domain Modelling Based on Requirements Specification and Ontology”, presents an approach to UML class diagram construction on the basis of a given domain ontology expressed using the SUMO language and a given user requirements model. The second one, “Semantic Validation of UML Class Diagrams with the Use of Domain Ontologies Expressed in OWL 2”, considers how to check compliance of a given UML class diagram with a given domain ontology expressed in OWL. The last paper in this part, “RSL-DL: Representing Domain Knowledge for the Purpose of Code Generation”, proposes a declarative specification language to define domain logic, being a new extension to the existing RSL language. The new language facilitates construction of precise requirements specifications and their further generation into code structured using the MVP architectural pattern.

In the next part—Quality Assurance—the main issues under consideration are prediction of defects, mutation testing and practical problems of measuring the quality of software. The paper “Assessment of the Software Defect Prediction Cost Effectiveness in an Industrial Project” deals with a real problem found in the industry: how the proposed method for predicting defects, assisted by an open source software measurement framework, affects the cost of software development. This preliminary investigation is encouraging, indicating the possibility of significant financial savings. The paper “Dynamic Stylometry for Defect Prediction” hypothesises that the indirect measure of software defects can be estimated by the measures defined for the software development process. The proposed model of defects prediction is evaluated experimentally on a group of computer science students. A new idea for mutation testing of Web Services is presented in the paper “Mutant generation for WSDL”. A tool called Web Services Mutant Generator that supports mutation testing of Web Services and examples of its usage are presented. The paper “Improving Measurement Certainty by Using Calibration to Find Systematic Measurement” states a very practical question: how to reduce the uncertainty of measurement results obtained from instruments with an unknown measurement method? When answering this question, the authors postulate to replace standard statistical analysis by an appropriate calibration procedure. The last paper in this part also refers to practical issues. The aim of the paper “Performance comparison of CRM systems dedicated to reporting failures to IT department” is to compare efficiency of a client-side architecture system vs. a server-side architecture system, under the condition of their significant load. The obtained results confirm the importance of choosing the client-side architecture for small and medium-sized systems.

The third part of this volume—Software Process Improvement—discusses selected aspects and mechanisms conducive to improving quality of software development. The first paper, “Software Engineering Needs Agile Experimentation: a New Practice and Supporting Tool”, meets the needs of collecting high-quality data during lightweight experimentation in real-world software development. It proposes and assesses a new practice called “agile experimentation”, together with

a dedicated tool supporting the practice. The next paper, “An Industrial Survey on Business Analysis Problems and Solutions”, analyses a set of business analysis techniques used in selected industrial projects and their impact on the most often reported problems encountered by business analysts. The main outcome of the paper is a recommendation regarding ways in which these techniques should be applied. The paper “Beyond Software Architecture Knowledge Management Tools” is a reflection on architecture decision-making tools in isolation from a particular practical context. It postulates the need for an architecture knowledge management infrastructure integrated with popular tools that are used for software development.

The Rational Unified Process is a widely accepted formal software development methodology. It can be used as a pattern for assessment of process maturity in software development organisations. The paper “Model of RUP processes maturity assessment in IT organisations” presents the design of a system that analyses application of RUP-compliant processes in real organisations within actual projects. The last paper, “A collaborative approach to developing a part-time Ph.D. program in IT Architecture with industry cooperation”, concerns the important problem of professional education at the Ph.D. level. A motivation model for establishing Ph. D. studies is proposed. Its leading idea is to make an agreement between academy and industry based on the SWOT analysis.

The above 15 papers constitute a selection from 59 submissions, which gives the total acceptance rate of less than 26 %. The editors would like to express their cordial thanks to all the authors for their significant contributions. We are very grateful to all reviewers for their excellent reviews and comments supporting the selection process and leveraging the quality of the selected papers.

Wrocław
Warsaw
Wrocław
Wrocław
May 2016

Lech Madeyski
Michał Śmiełek
Bogumiła Hnadkowska
Zbigniew Huzar

Software Engineering: Challenges and Solutions
Results of the XVIII KKIO 2016 Software Engineering
Conference 2016 held at September 15-17 2016 in
Wroclaw, Poland

Madeyski, L.; Śmiałek, M.; Hnatkowska, B.; Huzar, Z.
(Eds.)

2017, XII, 215 p. 53 illus., 32 illus. in color., Softcover
ISBN: 978-3-319-43605-0