Preface

In the early beginnings, the construction of automated programs for computers was seen as a special kind of scientific art. With the emergent use and the advent of higher-level programming languages, the term "Software Engineering" was coined in the late 1970s as a title of a NATO conference bringing together international experts in that field. At that conference, the experts wanted to tackle the pressing problems of software programming, resulting in poor stability and exceeding costs of the software. They aimed for a complete redefinition, where the development of software was seen as "engineering discipline". A systematic and measurable approach of building computer programs was the ultimate goal. Following the principles of established engineering disciplines, the quality of developing software should then also improve. As of today, Software Engineering (SE) distinguishes a number of areas, related to the software life cycle, most importantly requirements engineering, software design, quality management (including testing), construction, and maintenance; and furthermore, software development methodologies and programming languages.

With improved methods and practices, the quality of software clearly improved over the decades, despite the dramatically increased complexity. Also the development costs decreased by order of magnitudes when compared to the complexity of the software artifacts. However, as Fred Brooks prominently said, there will be no "silver bullet" for building software in a perfect and error-less manner.

The development of the first larger expert systems started around the 1970s as special purpose computer programs. In the early years, the development process was similar to the early software engineering days: The construction of expert systems followed no systematic approach, often specialized inference engines were programmed ad-hoc in hand with the knowledge acquisition phase. Custom tools were developed from project to project. In consequence, the same problems popped-up as well-known from the early software engineering days: Projects developed unpredictable costs, the quality of the systems was not measurable, and many systems were abandoned due to their non-maintainability. The term "Knowledge Engineering" (KE) was born to tackle these problems. With the same systematic approach, many established methods and practices from Software

Engineering were adapted to the needs of building intelligent systems. With the development of tailored knowledge-based methods, the Software Engineering community realized the potential of intelligent methods for Software Engineering and thus the intellectual loop was closed by influential methods of Knowledge Engineering in Software Engineering. Until today, this symbiotic relationship exists and produces fruitful results. Currently, a number of international journals and conferences are devoted to the interrelations of Software Engineering and Knowledge Engineering.

In recent years, this relation is even stronger, with the rise of "intelligent/smart" software penetrating many aspects of daily lives. Smart homes, intelligent assistants, chatbots, autonomous cars, and intelligent manufacturing; these all are software systems that require knowledge to develop and employ their intelligent behavior and operation to the end user.

This volume compiles a number of submissions originated in the KESE workshop series: The first workshop *KESE: Knowledge Engineering and Software Engineering* was held in 2005 at the 28th German Conference on Artificial Intelligence (KI-2005) in Koblenz, Germany. The idea of the organizers was the realignment of the discipline Knowledge Engineering and its strong relation to Software Engineering, as well as to the classical Artificial Intelligence (AI) research. The practical aspects of AI systems emphasize the need for combining KE and SE methods and techniques. Due to their initial success, the KESE workshops were annually organized at the German AI conference (KI). In the years 2006–2010 KESE was held together with the KI in Bremen, Osnabruck, Kaiserslautern, Paderborn, and Karlsruhe. In these years, the workshop gathered a motivated and active international community. Thanks to it, in 2011 the KESE7 workshop was co-located with the Spanish AI conference CAEPIA 2011, held in San Cristobal de la Laguna, Tenerife. Then, in 2012 KESE moved the European AI conference, ECAI, then held in Montpellier. In 2013, it shortly moved back to the KI, held in Koblenz again. The tenth and last edition of KESE occurred in 2014. It was again co-located with ECAI then held in Prague. In its ten years of continuous existence, KESE attracted a stable flow of high quality papers. In the years 2005–2014, the following numbers of papers were presented at KESE: 8, 5, 6, 9, 7, 5, 7, 10, 8, 10, total of 75 papers. Starting from 2007, the workshop had its own proceedings published via CEUR WS website (http://ceur-ws.org), as CEUR volumes: 282, 425, 486, 636, 805, 949, 1070, and 1289. Moreover, in 2011 a Special Issue on Knowledge and Software Engineering for Intelligent Systems of the International Journal of Knowledge Engineering and Data Mining Vol. 1 No. 3 was published. All the details of the history of KESE were made available on a dedicated website: http://kese.ia.agh.edu.pl

The KESE community not only appreciated the scientific quality of the workshop, but also its important social aspects. Every year the participants held additional vibrant discussions during the so-called balcony-sessions, commonly referred to as b-sessions.

The workshop series always encouraged submissions describing methodological research combining Knowledge Engineering and Software Engineering but also the presentation of successful applications demanding for both disciplines. In fact, besides regular papers, starting from 2009 KESE also solicited tool presentations. In the 10 years of annual workshops, the topics of interest varied from AI methods in software/knowledge engineering (knowledge and experience management, declarative, logic-based approaches, constraint programming, agent-oriented software engineering, issues of maintenance) and knowledge/software engineering methods in AI (collaborative engineering of the Semantic Web, database and knowledge base management in AI systems, tools for intelligent systems, evaluation of (intelligent) systems: verification, validation, assessment, process models) to a range of topics located on the intersection of several disciplines relevant for KESE. In the final edition these included: knowledge and software engineering for the Semantic Web, knowledge and software engineering for Linked Data, ontologies in practical knowledge and software engineering, business systems modeling, design and analysis using KE and SE, practical knowledge representation and discovery techniques in software engineering, context and explanation in intelligent systems, knowledge base management in KE systems, evaluation and verification of KBS, practical tools for KBS engineering, process models in KE applications, software requirements and design for KBS applications, software quality assessment through formal KE models, and declarative, logic-based, including constraint programming approaches in SE.

After 10 bright years of annual workshops, we decided it is the time summarize the decade of works of the KESE community. Thus, we decided to prepare and edit this "memorial" volume. It compiles thirteen intriguing contributions closely related to the KESE topics with both methodological and application background grouped in two separate parts.

The first methodological part is composed of seven chapters. Ralph Schäfermeier and Adrian Paschke introduce an ontology development process that is inspired by Aspect-Oriented Programming. By following the aspect-oriented development, the complexity of the construction process can be simplified by strict modularization. Ralph Bergmann and Gilbert Müller improve the (re-)use of workflows in process-oriented information systems by introducing methods for the retrieval and adaptation of (best practice) workflows applicable to new process problems. Isabel María Del Águila and José Del Sagrado describe an approach to building of intelligent systems using Bayesian networks. They use UML known from Software Engineering and introduce the meta-model BayNet that supports the development and maintenance process. How can software development be supported by Knowledge Engineering techniques? Paraskevi Smiari et al. represent anti-patterns in software development by Bayesian networks and include this into a knowledge-based framework. During the software development the data is acquired and problems are identified as anti-patterns. Bayesian networks are used to assess the anti-pattern in the concrete problem situation. The quality of intelligent systems is considered by the contribution of Rainer Knauf: He introduces formal methods for the validation and finally the refinement of intelligent systems. Automated methods for test case generation and support for the human inspection are presented. The quality of intelligent systems is also discussed in the next contribution:

Marius Brezovan and Costin Badica present a novel method for the verification of knowledge-based systems. They apply a mathematical language for modeling the static and dynamic properties of the systems. Finally, in her contribution, Kerstin Bach tackles the problem of building large (knowledge-intensive decision-support) systems. With the industry-inspired term "Knowledge Line" a methodology for the modularization of knowledge is presented and exemplified.

The second part is devoted to applications and is composed of six chapters. The authors Paolo Ciancarini et al. report on the development of a mission-critical and knowledge-intensive system. They define the process model iAgile which is based on the successful Scrum methodology widely applied in Software Engineering. Pascal Reuss et al. also report on construction of an intelligent system, here the diagnosis and maintenance of civil aircrafts. A multi-agent approach is used to organize and implement the problem domain. A practical application report from the medical domain is given by Paulo Novais et al. A model of computer interpretable guidelines is introduced by using the ontology language OWL. Together with the language, an implementation in a clinical system is given. The web traversals and actions of users in the web is an interesting asset for market research. Adrian Giurca introduces the ontology Metamarket that models the user preferences and interactions. Apache Maven is a popular and broadly used tool for supporting the development of general software. In their contribution Adrian Paschke and Ralph Schäfermeier extend and customize the Maven approach for the distributed development and management of ontologies. A different approach for supporting software development is presented by Andrea Janes: data is collected during the software development process in a ubiquitous manner. The measurements are used to identify knowledge in the process data and to analyze the software creation and usage process.

In the past few years, a strong rise of interest in AI can clearly be observed in research, IT industry, as well as the general public. Today, the development of complex intelligent systems clearly benefits from the synergy of knowledge engineering and software engineering. Let this book, be a timely and valuable contribution to this goal.

Kraków, Poland — Grzegorz J. Nalepa
Würzburg, Germany — Joachim Baumeister
2017

Springer

http://www.springer.com/978-3-319-64160-7

Synergies Between Knowledge Engineering and Software Engineering
Nalepa, G.J.; Baumeister, J. (Eds.)
2018, XIV, 294 p. 93 illus., 73 illus. in color., Softcover
ISBN: 978-3-319-64160-7