

## Axel Van Lamsweerde Software Requirements Engineering

The book presents both the current state of the art in requirements engineering and a systematic method for engineering high-quality requirements, broken down into four parts. The first part introduces fundamental concepts and principles including the aim and scope of requirements engineering, the products and processes involved, requirements qualities to aim at and flaws to avoid, and the critical role of requirements engineering in system and software engineering. The second part of the book is devoted to system modeling in the specific context of engineering requirements. It presents a multi-view modeling framework that integrates complementary techniques for modeling the system-as-is and the system-to-be. The third part of the book reviews goal-based reasoning techniques to support the various steps of the KAOS method. The fourth part of the book goes beyond requirements engineering to discuss the mapping from goal-oriented requirements to software specifications and to software architecture. Online software will accompany the book and will add value to both classroom and self-study by enabling students to build models and specifications involved in the book's exercises and case studies, helping them to discover the latest RE technology solutions. Instructor resources such as slides, figures and handouts are available from an accompanying website.

After Ole-Johan's retirement at the beginning of the new millennium, some of us had thought and talked about making a "Festschrift" in his honor. When Donald Knuth took the initiative by sending us the first contribution, the process began to roll! In early 2002 an editing group was formed, including Kristen Nygaard, who had known Ole-Johan since their student days, and with whom he had developed the Simula language. Then we invited a number of prominent researchers familiar with Ole-Johan to submit contributions for a book honoring Ole-Johan on the occasion of his 70th birthday. Invitees included several members of the IFIP 2.3 working group, a forum that Ole-Johan treasured and enjoyed participating in throughout his career. In spite of the short deadline, the response to the invitations was overwhelmingly positive. The original idea was to complete the book rather quickly to make it a gift he could read and enjoy, because by then he had had cancer for three years, and his health was gradually deteriorating. Kristen had been regularly visiting Ole-Johan, who was in the hospital at that time, and they were working on their Turing award speech. Ole-Johan was gratified to hear about the contributions to this book, but modestly expressed the feeling that there was no special need to undertake a book project on his behalf. Peacefully accepting his destiny, Ole-Johan died on June 29, 2002.

This book provides a coherent methodology for Model-Driven Requirements Engineering which stresses the systematic treatment of requirements within the realm of modelling and model transformations. The underlying basic assumption is that detailed requirements models are used as first-class artefacts playing a direct role in constructing software. To this end, the book presents the Requirements Specification Language (RSL) that allows precision and formality, which eventually permits automation of the process of turning requirements into a working system by applying model transformations and code generation to RSL. The book is structured in eight chapters. The first two chapters present the main concepts and give an introduction to

requirements modelling in RSL. The next two chapters concentrate on presenting RSL in a formal way, suitable for automated processing. Subsequently, chapters 5 and 6 concentrate on model transformations with the emphasis on those involving RSL and UML. Finally, chapters 7 and 8 provide a summary in the form of a systematic methodology with a comprehensive case study. Presenting technical details of requirements modelling and model transformations for requirements, this book is of interest to researchers, graduate students and advanced practitioners from industry. While researchers will benefit from the latest results and possible research directions in MDRE, students and practitioners can exploit the presented information and practical techniques in several areas, including requirements engineering, architectural design, software language construction and model transformation. Together with a tool suite available online, the book supplies the reader with what it promises: the means to get from requirements to code “in a snap”.

Requirements engineering is the process by which the requirements for software systems are gathered, analyzed, documented, and managed throughout their complete lifecycle. Traditionally it has been concerned with technical goals for, functions of, and constraints on software systems. Aurum and Wohlin, however, argue that it is no longer appropriate for software systems professionals to focus only on functional and non-functional aspects of the intended system and to somehow assume that organizational context and needs are outside their remit. Instead, they call for a broader perspective in order to gain a better understanding of the interdependencies between enterprise stakeholders, processes, and software systems, which would in turn give rise to more appropriate techniques and higher-quality systems. Following an introductory chapter that provides an exploration of key issues in requirements engineering, the book is organized in three parts. Part 1 presents surveys of state-of-the-art requirements engineering process research along with critical assessments of existing models, frameworks and techniques. Part 2 addresses key areas in requirements engineering, such as market-driven requirements engineering, goal modeling, requirements ambiguity, and others. Part 3 concludes the book with articles that present empirical evidence and experiences from practices in industrial projects. Its broader perspective gives this book its distinct appeal and makes it of interest to both researchers and practitioners, not only in software engineering but also in other disciplines such as business process engineering and management science.

In the past ten years or so, software architecture has emerged as a central notion in the development of complex software systems. Software architecture is now accepted in the software engineering research and development community as a manageable and meaningful abstraction of the system under development and is applied throughout the software development life cycle, from requirements analysis and validation, to design and down to code and execution level. This book presents the tutorial lectures given by leading authorities at the Third International School on Formal Methods for the Design of Computer, Communication and Software Systems, SFM 2003, held in Bertinoro, Italy, in September 2003. The book is ideally suited for advanced courses on software architecture as well as for ongoing education of software engineers using formal methods in their day-to-day professional work.

For more and more systems, software has moved from a peripheral to a central role, replacing mechanical parts and hardware and giving the product a competitive edge.

Consequences of this trend are an increase in: the size of software systems, the variability in software artifacts, and the importance of software in achieving the system-level properties. Software architecture provides the necessary abstractions for managing the resulting complexity. We here introduce the Third Working IEEE/IFIP Conference on Software Architecture, WICSA3. That it is already the third such conference is in itself a clear indication that software architecture continues to be an important topic in industrial software development and in software engineering research. However, becoming an established field does not mean that software architecture provides less opportunity for innovation and new directions. On the contrary, one can identify a number of interesting trends within software architecture research. The first trend is that the role of the software architecture in all phases of software development is more explicitly recognized. Whereas initially software architecture was primarily associated with the architecture design phase, we now see that the software architecture is treated explicitly during development, product derivation in software product lines, at run-time, and during system evolution. Software architecture as an artifact has been decoupled from a particular lifecycle phase. Modelling of information is necessary in developing information systems. Information is acquired from many sources, by using various methods and tools. It must be recognized, conceptualized, and conceptually organized efficiently so that users can easily understand and use it. Modelling is needed to understand, explain, organize, predict, and reason on information. It also helps to master the role and functions of components of information systems. Modelling can be performed with many different purposes in mind, at different levels, and by using different notions and different background theories. It can be made by emphasizing users' conceptual understanding of information on a domain level, on an algorithmic level, or on representation levels. On each level, the objects and structures used on them are different, and different rules govern the behavior on them. Therefore the notions, rules, theories, languages, and methods for modelling on different levels are also different. It will be useful if we can develop theories and methodologies for modelling, to be used in different situations, because databases, knowledge bases, and repositories in knowledge management systems, developed on the basis of models and used to technically store information, are growing day by day. In this publication, the interest is focused on modelling of information, and one of the central topics is modelling of time. Scientific and technical papers of high quality are brought together in this book.

This Festschrift volume, published in honor of John Mylopoulos on the occasion of his retirement from the University of Toronto, contains 25 high-quality papers, written by leading scientists in the field of conceptual modeling. The volume has been divided into six sections. The first section focuses on the foundations of conceptual modeling and contains material on ontologies and knowledge representation. The four sections on software and requirements engineering, information systems, information integration, and web and services, represent the chief current application domains of conceptual modeling. Finally, the section on implementations concentrates on projects that build tools to support conceptual modeling. With its in-depth coverage of diverse topics, this book could be a useful companion to a course on conceptual modeling.

Everyone expects the products and services they use to be secure, but 'building security in' at the earliest stages of a system's design also means designing for use as well. Software that is

unusable to end-users and unwieldy to developers and administrators may be insecure as errors and violations may expose exploitable vulnerabilities. This book shows how practitioners and researchers can build both security and usability into the design of systems. It introduces the IRIS framework and the open source CAIRIS platform that can guide the specification of secure and usable software. It also illustrates how IRIS and CAIRIS can complement techniques from User Experience, Security Engineering and Innovation & Entrepreneurship in ways that allow security to be addressed at different stages of the software lifecycle without disruption. Real-world examples are provided of the techniques and processes illustrated in this book, making this text a resource for practitioners, researchers, educators, and students. Non-Functional Requirements in Software Engineering presents a systematic and pragmatic approach to 'building quality into' software systems. Systems must exhibit software quality attributes, such as accuracy, performance, security and modifiability. However, such non-functional requirements (NFRs) are difficult to address in many projects, even though there are many techniques to meet functional requirements in order to provide desired functionality. This is particularly true since the NFRs for each system typically interact with each other, have a broad impact on the system and may be subjective. To enable developers to systematically deal with a system's diverse NFRs, this book presents the NFR Framework. Structured graphical facilities are offered for stating NFRs and managing them by refining and inter-relating NFRs, justifying decisions, and determining their impact. Since NFRs might not be absolutely achieved, they may simply be satisfied sufficiently ('satisficed'). To reflect this, NFRs are represented as 'softgoals', whose interdependencies, such as tradeoffs and synergy, are captured in graphs. The impact of decisions is qualitatively propagated through the graph to determine how well a chosen target system satisfies its NFRs. Throughout development, developers direct the process, using their expertise while being aided by catalogues of knowledge about NFRs, development techniques and tradeoffs, which can all be explored, reused and customized. Non-Functional Requirements in Software Engineering demonstrates the applicability of the NFR Framework to a variety of NFRs, domains, system characteristics and application areas. This will help readers apply the Framework to NFRs and domains of particular interest to them. Detailed treatments of particular NFRs - accuracy, security and performance requirements - along with treatments of NFRs for information systems are presented as specializations of the NFR Framework. Case studies of NFRs for a variety of information systems include credit card and administrative systems. The use of the Framework for particular application areas is illustrated for software architecture as well as enterprise modelling. Feedback from domain experts in industry and government provides an initial evaluation of the Framework and some case studies. Drawing on research results from several theses and refereed papers, this book's presentation, terminology and graphical notation have been integrated and illustrated with many figures. Non-Functional Requirements in Software Engineering is an excellent resource for software engineering practitioners, researchers and students.

This three-volume collection, titled Enterprise Information Systems: Concepts, Methodologies, Tools and Applications, provides a complete assessment of the latest developments in enterprise information systems research, including development, design, and emerging methodologies. Experts in the field cover all aspects of enterprise resource planning (ERP), e-commerce, and organizational, social and technological implications of enterprise information systems.

This book constitutes the refereed proceedings of the Third International Conference on Generative and Component-Based Software Engineering, GCSE 2001, held in Erfurt, Germany, in September 2001. The 14 revised full papers presented together with one invited paper were carefully reviewed and selected from 43 submissions. The papers are organized in topical sections on software product lines, aspects, generic and generative approaches, and

components and architectures.

Dynamic Coalitions denote a temporary collaboration between different entities to achieve a common goal. A key feature that distinguishes Dynamic Coalitions from static coalitions is Dynamic Membership, where new members can join and others can leave after a coalition is set. This thesis studies workflows in Dynamic Coalitions, by analyzing their features, highlighting their unique characteristics and similarities to other workflows, and investigating their relation with Dynamic Membership. For this purpose, we use the formal model of Event Structures and extend it to faithfully model scenarios taken as use cases from healthcare. Event Structures allow for workflows modeling in general, and for modeling Dynamic Membership in Dynamic Coalitions as well through capturing the join and leave events of members. To this end, we first extend Event Structures with Dynamic Causality to address the dynamic nature of DCs. Dynamic Causality allows some events to change the causal dependencies of other events in a structure. Then, we study the expressive power of the resulting Event Structures and show that they contribute only to a specific kind of changes in workflows, namely the pre-planned changes. Second, we present Evolving Structures in order to support ad-hoc and unforeseen changes in workflows, as required by the use cases. Evolving Structures connect different Event Structures with an evolution relation which allows for changing an Event Structure during a system run. We consider different approaches to model evolution and study their equivalences. Furthermore, we show that the history of a workflow should be preserved in our case of evolution in Dynamic Coalitions, and we allow for extracting changes from an evolution to support Process Learning. Third, to capture the goals of DCs, we equip Evolving Structures with constraints concerning the reachability of a set of events that represents a goal. The former extensions allow for examining the changes and evolutions caused by members, and examining members' contributions to goal satisfaction, through their join and leave events. Finally, we highlight many modeling features posed as requirements by the domain of our Dynamic-Coalition use cases, namely the healthcare, which are independent from the nature of Dynamic Coalitions, e.g. timing. We examine the literature of Event Structures for supporting such features, and we identify that the notion of Priority is missing in Event Structures. To this end, we add Priority to various kinds of Event Structures from the literature. Furthermore, we study the relation between priority on one side, and conjunctive causality, disjunctive causality, causal ambiguity and various kinds of conflict on the other side. Comparing to Adaptive Workflows, which are concerned with evolutions of workflows that occur as a response to changes, e.g. changes in the business environment or exceptions, this thesis shows that Dynamic-Coalition workflows are not only Adaptive but also Goal-Oriented. Besides, it adds one extra trigger for evolution in workflows—unique to Dynamic Coalitions—namely the join of new members who contribute to goal satisfaction in a Dynamic Coalition. Finally the thesis contributes to bridging the gap in modeling between theory and domain experts by supporting step-by-step modeling applied regularly in healthcare and other domains. Dynamische Koalitionen (DKen) bezeichnen eine temporäre Kollaboration zwischen verschiedenen Entitäten zum Erreichen eines gemeinsamen Ziels. Ein Schlüsselaspekt, welcher dynamische Koalitionen von statischen Koalitionen unterscheidet ist die dynamische Mitgliedschaft, durch die neue Mitglieder hinzu- kommen und andere die Koalitionen verlassen können, nachdem sie entstanden ist. Diese Arbeit studiert Workflows in dynamische Koalitionen durch eine Analyse ihrer Eigenschaften, das Herausstellen ihrer einzigartigen Charakteristika und Ähnlichkeiten zu anderen Workflows und durch eine Untersuchung ihrer Beziehung zu dynamischer Mitgliedschaft. In diesem Sinne nutzen wir das formale Model der Ereignisstrukturen (ESen) und erweitern es, um Fallstudien aus der Medizin angemessen zu modellieren. ESen erlauben sowohl eine generelle Workflow Modellierung als auch eine Darstellung von Eintritt- und Austrittereignissen von Mitgliedern. Zu diesem Zweck erweitern wir ESen zuerst um Dynamische Kausalität, um die dynamische Natur von DKs abzubilden.

Dynamische Kausalität erlaubt bestimmten Ereignissen die kausalen Abhängigkeiten anderer Ereignissen in einer Struktur zu verändern. Dann untersuchen wir die Ausdrucksstärke der resultierenden ESEN und zeigen, dass sie nur eine spezifische Art der Veränderung abbilden, die sogenannten vorgeplanten Veränderungen. Als Zweites präsentieren wir Evolving in ESEN um ad-hoc- und unvorhergesehene Veränderungen zu unterstützen, wie es durch unsere Fallstudien benötigt wird. Evolving in ESEN verbinden verschiedene ESEN mit einer Relation, welche eine Veränderung einer ES während eines Ablaufes erlaubt. Wir ziehen verschiedene Ansätze der Modellevolution in Betracht und untersuchen ihre Äquivalenzen. Des Weiteren zeigen wir, dass in unserem Fall der Evolution in DKEN die Geschichte eines Workflows erhalten bleiben muss und wir ermöglichen das Extrahieren von Veränderungen einer Evolution, um Process Learning zu unterstützen. Drittens: Um die Ziele von DKEN abzubilden, fügen wir den Evolving in ESEN mit Einschränkungen bezüglich der Erreichbarkeit einer Menge von Ereignissen hinzu, welche das Ziel repräsentieren. Die genannten Erweiterungen erlauben es sowohl die Änderungen und Evolutionen, die vom Mitgliedern verursacht werden als auch die Beiträge der Mitglieder zur Zielerreichung durch deren Eintritt- und Austrittereignissen zu untersuchen. Schlussendlich, stellen wir viele Modellierungseigenschaften dar, welche von den DK-Fallstudien aus der Medizin benötigt werden und unabhängig von der Natur der DKEN sind, wie z.B. Timing. Wir untersuchen die Literatur zu ESEN bezüglich Unterstützung für solche Eigenschaften und stellen fest, dass der Begriff Priorität in ESEN fehlt. Daher fügen wir Priorität zu verschiedenen ESEN aus der Literatur hinzu. Des Weiteren untersuchen wir die Beziehungen von Priorität auf zu Konjunktiver Kausalität, disjunktiver Kausalität, kausal Uneindeutigkeit und verschiedenen Formen von Konflikt. Im Vergleich zu Adaptive Workflows, welche sich mit der Evolution von Workflows beschäftigt, die als Reaktion auf Veränderungen entsteht, wie z.B. Veränderungen im Business Environment oder Exceptions, zeigt diese Arbeit das DKEN nicht nur adaptiv sondern auch zielorientiert sind. Außerdem fügt sie einen zusätzlichen Auslöser für Evolution in Workflows hinzu, welcher ausschließlich DKEN eigen ist: das Hinzukommen neuer Mitglieder welche zur Zielerreichung der DK beitragen. Zuletzt trägt diese Arbeit bei, die Lücke der Modellierung zwischen der Theorie und den Domänenexperten zu überbrücken, in dem sie eine Schritt-für-Schritt Modellierung unterstützt, welche regelmäßig in der Medizin und anderen Bereichen angewand wird.

Modern software systems increasingly use commercial-off-the-shelf (COTS) software products as building blocks. In some cases, major software systems are assembled with virtually no custom code in the system. The use of COTS software products as components offers the promise of rapid delivery to end users, shared development costs with other customers, and an opportunity for expanding mission or business capabilities and performance as improvements are made in the commercial marketplace. Few organizations today can afford the resources and time to replicate market-tested capabilities. Yet, the promise of COTS products is too often not realized in practice. There have been more failures than successes in using COTS software products. The research and software practitioner communities have been working with COTS-based software systems for a number of years. There is now sufficient documented experience in the community to collect, analyze, and disseminate success stories, common failings, lessons-learned, and research advances. The mounting experience shows that the effective use of COTS software products in major software systems demands new skills, knowledge, and abilities, changed roles and responsibilities, and different techniques and processes. The International Conference on COTS-Based Software Systems (ICCBSS) focuses on the challenges of building and maintaining systems that incorporate COTS software products. The conference sponsors, the National Research Council Canada, the Software Engineering Institute, and the University of Southern California Center for Software Engineering, aim to bring together managers, developers, maintainers, and researchers to share their expertise and experience.

This book constitutes the strictly refereed post-workshop proceedings of the International Workshop on Requirements Targeting Software and Systems Engineering, RTSE '97, held in Bernried, Germany in October 1997. The 15 revised full papers presented in the book were carefully revised and reviewed for inclusion in the book. Among the authors are internationally leading researchers. The book is divided in sections on foundations of software engineering, methodology, evaluation and case studies, and tool support and prototyping.

This volume contains the papers from the workshop "Radical Innovations of Software and Systems Engineering in the Future." This workshop was the ninth in the series of Monterey Software Engineering workshops for formulating and advancing software engineering models and techniques, with the fundamental theme of increasing the practical impact of formal methods. During the last decade object orientation was the driving factor for new system solutions in many areas ranging from e-commerce to embedded systems. New modeling languages such as UML and new programming languages such as Java and CASE tools have considerably influenced the system development techniques of today and will remain key techniques for the near future. However, actual practice shows many deficiencies of these new approaches: – there is no proof and no evidence that software productivity has increased with the new methods; – UML has no clean scientific foundations, which inhibits the construction of powerful analysis and development tools; – support for mobile distributed system development is missing; – for many applications, object-oriented design is not suited to producing clean well-structured code, as many applications show.

This book constitutes the refereed proceedings of the 8th International Conference on Software Reuse, ICSR-8, held in Madrid, Spain in July 2004. The 28 revised full papers presented were carefully reviewed and selected from numerous submissions. The papers are organized in topical sections on software variability: requirements; testing reusable software; feature modeling; aspect-oriented software development; component and service development; code level reuse; libraries, classification, and retrieval; model-based approaches; transformation and generation; and requirements. The Sixth International Conference on Reliable Software Technologies, Ada-Europe 2001, took place in Leuven, Belgium, May 14-18, 2001. It was sponsored by Ada-Europe, the European federation of national Ada societies, in cooperation with ACM SIGAda, and it was organized by members of the K.U. Leuven and Ada-Belgium. This was the 21st consecutive year of Ada-Europe conferences and the sixth year of the conference focusing on the area of reliable software technologies. The use of software components in embedded systems is almost ubiquitous: planes fly by wire, train signalling systems are now computer based, mobile phones are digital devices, and biological, chemical, and manufacturing plants are controlled by software, to name only a few examples. Also other, non-embedded, mission-critical systems depend more and more upon software. For these products and processes, reliability is a key success factor, and often a safety-critical hard requirement. It is well known and has often been experienced that quality cannot be added to software as a mere afterthought. This also holds for reliability. Moreover, the reliability of a system is not due to and cannot be built upon a single technology. A wide range of approaches is needed, the most difficult issue being their purposeful integration. Goals of reliability must be precisely defined and included in the requirements, the development process must be controlled to achieve these goals, and sound development methods must be used to fulfill these non-functional requirements.

This book constitutes the refereed proceedings of the 15th International Working Conference on Requirements Engineering: Foundation for Software Quality, REFSQ

2009, held in Amsterdam, The Netherlands, in June 2009. The 14 revised full papers were carefully reviewed and selected from 49 submissions. The papers are organized in thematic sections on value and risk, change and evolution, interactions and inconsistencies, organization and structuring, experience, elicitation, research methods, behavior modeling, empirical studies, and open-source RE.

This book constitutes the thoroughly refereed post-proceedings of the International Software Process Workshop, SPW 2005, held in Beijing, China in May 2005. The 30 papers presented here, together with 11 keynote addresses are organized in topical sections on process content, process tools and metrics, process management, process representation and analysis, as well as experience reports.

"Information security covers the protection of information against unauthorized disclosure, transfer, modification, and destruction, whether accidentally or intentionally. Quality of life in general and of individual citizens, and the effectiveness of the economy critically depends on our ability to build software in a transparent and efficient way. Furthermore, we must be able to enhance the software development process systematically in order to ensure software's safety and security. This, in turn, requires very high software reliability, i.e., an extremely high confidence in the ability of the software to perform flawlessly. Foundations of software technology provide models that enable us to capture application domains and their requirements, but also to understand the structure and working of software systems and software architectures. Based on these foundations tools allow to prove and ensure the correctness of software's functioning. New developments must pay due diligence to the importance of security-related aspects, and align current methods and techniques to information security, integrity, and system reliability. The articles in this book describe the state-of-the-art ideas on how to meet these challenges in software engineering."

Recommender systems use information filtering to predict user preferences. They are becoming a vital part of e-business and are used in a wide variety of industries, ranging from entertainment and social networking to information technology, tourism, education, agriculture, healthcare, manufacturing, and retail. Recommender Systems: Algorithms and Applications dives into the theoretical underpinnings of these systems and looks at how this theory is applied and implemented in actual systems. The book examines several classes of recommendation algorithms, including Machine learning algorithms Community detection algorithms Filtering algorithms Various efficient and robust product recommender systems using machine learning algorithms are helpful in filtering and exploring unseen data by users for better prediction and extrapolation of decisions. These are providing a wider range of solutions to such challenges as imbalanced data set problems, cold-start problems, and long tail problems. This book also looks at fundamental ontological positions that form the foundations of recommender systems and explain why certain recommendations are predicted over others. Techniques and approaches for developing recommender systems are also investigated. These can help with implementing algorithms as systems and include A latent-factor technique for model-based filtering systems Collaborative filtering approaches Content-based approaches Finally, this book examines actual systems for social networking, recommending consumer products, and predicting risk in software engineering projects. We can now say that it is really a big pleasure for us to welcome all of you to the proceedings of CAiSE 2005 which was held in Porto.

Trustworthiness is a key success factor in the acceptance and adoption of cyber-physical systems. The author first discusses various existing definitions of trust and trustworthiness and extends them to cyber-physical systems. A comprehensive framework is proposed, including methods that cover all phases of development: requirements engineering, system design, trustworthiness evaluation, run-time maintenance, and evidence-based assurance. To support a smooth integration of the methods into development projects, these methods are provided in the form of so-called capability patterns. A running example from the ambient assisted living domain is used to demonstrate the application of the methods. About the Author: Nazila Gol Mohammadi is currently working as an associate researcher at paluno – The Ruhr Institute for Software Technology in Essen, Germany. Her research interests include software engineering, requirements engineering, digitalization, cloud computing, cyber-physical systems, and trustworthiness of software systems.

This book constitutes the refereed proceedings of the 9th International Conference on Reliable Software Technologies, Ada-Europe 2004, held in Palma de Mallorca, Spain in June 2004. The 23 revised full papers presented together with 3 invited papers were carefully reviewed and selected from numerous submissions. The papers are organized in topical sections on static analysis, distributed systems, real-time systems, reflection and XML, testing, critical systems modeling, scheduling, and application programming interfaces.

A revolutionary concept-based approach to thinking about, designing, and interacting with software As our dependence on technology increases, the design of software matters more than ever before. Why then is so much software flawed? Why hasn't there been a systematic and scalable way to create software that is easy to use, robust, and secure? Examining these issues in depth, *The Essence of Software* introduces a theory of software design that gives new answers to old questions. Daniel Jackson explains that a software system should be viewed as a collection of interacting concepts, breaking the functionality into manageable parts and providing a new framework for thinking about design. Through this radical and original perspective, Jackson lays out a practical and coherent path, accessible to anyone—from strategist and marketer to UX designer, architect, or programmer—for making software that is empowering, dependable, and a delight to use. Jackson explores every aspect of concepts—what they are and aren't, how to identify them, how to define them, and more—and offers prescriptive principles and practical tips that can be applied cost-effectively in a wide range of domains. He applies these ideas to contemporary software designs, drawing examples from leading software manufacturers such as Adobe, Apple, Dropbox, Facebook, Google, Microsoft, Twitter, and others. Jackson shows how concepts let designers preserve and reuse design knowledge, rather than starting from scratch in every project. An argument against the status quo and a guide to improvement for both working designers and novices to the field, *The Essence of Software* brings a fresh approach to software and its creation.

ETAPS 2002 was the 7th instance of the European Joint Conferences on Theory and Practice of Software. ETAPS is an annual federated conference that was established in 1998 by combining a number of existing and new conferences. This year it comprised 5 conferences (FOSSACS, FASE, ESOP, CC, TACAS), 13 satellite workshops (ACL2, AGT, CMCS, COCV, DCC, INT, LDTA, SC, SFEDL, SLAP, SPIN, TPTS, and VISS), 8 invited lectures (not including those specific to the satellite events), and several tutorials. The events that comprise ETAPS address various aspects of the system development process, including specification, design, implementation, analysis, and improvement. The languages, methodologies, and tools which support these activities are all well within its scope. Different blends of theory and practice are represented, with an inclination towards theory with a practical motivation on one hand and soundly-based practice on the other. Many of the issues involved in software design apply to

systems in general, including hardware systems, and the emphasis on software is not intended to be exclusive.

Inhaltsangabe:Abstract: The first step during the software development, requirements engineering, is very critical because of the high effort (in time and costs) that has to be made to correct mistakes detected later that have been made in this early phase of software life-cycle. In order to support the aim of high-quality software, the goals of requirements engineering are developing a complete as possible specification, providing integrated representation formalisms and accomplishing a common agreement on the specification. The very first activity that has to be passed through is requirements elicitation. There are existing three main problems: the problem of defining the scope, the problem of understanding the users needs and the problem of requirements volatility over time. You can follow several heuristics and guidelines to find solutions to these problems. In addition, several techniques and methodologies have been suggested to support the process of requirements elicitation. They differ in several ways: the kind of problem they intend to solve, the methods used for achieving this aim, the kind of people involved, the level of abstraction and precision the requirements have to be formulated in. In this report, a selection of these techniques and methodologies is chosen and they are classified into a classification scheme worked out. The techniques and methodologies can coarsely be divided into four classes: interview-oriented approaches, objective and goal analysis-oriented approaches, viewpoint analysis-oriented approaches, and scenario analysis-oriented approaches. There are others that do not fit into this division, but provide nevertheless help for requirements elicitation. The developed classification scheme highlights the differences between the existing techniques. It should serve as an overview of existing techniques and methods as a guideline for analysts and developers for finding an appropriate method for problems at hand. Inhaltsverzeichnis:Table of Contents: 1.Introduction1 2.Conceptions and Guidelines5 2.1Requirements Elicitation Process Model5 2.2Guidelines for Requirements Elicitation9 3.Framework for Understanding Elicitation Approaches13 3.1Problem14 3.2Methods15 3.3People16 3.4Type17 3.5Solution18 4.Categorization of Requirements Elicitation Approaches19 4.1Interview-Oriented Approaches20 4.2Objective and Goal-Oriented Approaches25 4.3Viewpoint Analysis-Oriented Approaches32 4.4Scenario [...]

The Third International Conference on Product Focused Software Process Improvement (PROFES 2001) continued the success of the PROFES'99 and PROFES 2000 conferences. PROFES 2001 was organized in Kaiserslautern, Germany, September 10-13, 2001. The PROFES conference has its roots in the PROFES Esprit project (<http://www.ele.vtt.fi/profes/>), but it quickly evolved into a full fledged general purpose conference in 1999 and since then it has gained wide spread international popularity. As in previous years, the main theme of PROFES 2001 was professional software process improvement (SPI) motivated by product and service quality needs. SPI is facilitated by software process assessment, software measurement, process modeling, and technology transfer and has become a practical tool for quality software engineering and management. The conference addresses both the solutions found in practice as well as relevant research results from academia. The purpose of the conference is to bring to light the most recent findings and results in the area and to stimulate discussion between the researchers, experienced professionals, and technology providers for SPI.

Requirements EngineeringFrom System Goals to UML Models to Software SpecificationsJohn Wiley & Sons Incorporated

NLDB 2005, the 10th International Conference on Applications of Natural Language to Information Systems, was held on June 15–17, 2005 at the University of Alicante, Spain. Since the first NLDB conference in 1995 the main goal has been to provide a forum to discuss and disseminate research on the integration of natural language resources in information system engineering. The development and convergence of computing, telecommunications and

information systems has already led to a revolution in the way that we work, communicate with each other, buy goods and use services, and even in the way that we entertain and educate ourselves. The revolution continues, and one of its results is that large volumes of information will increasingly be held in a form which is more natural for users than the data presentation formats typical of computer systems of the past. Natural language processing (NLP) is crucial in solving these problems, and language technologies will make an indispensable contribution to the success of information systems. We hope that NLDB 2005 was a modest contribution to this goal. NLDB 2005 contributed to advancing the goals and the high international standing of these conferences, largely due to its Program Committee, composed of renowned researchers in the field of natural language processing and information system engineering. Papers were reviewed by three reviewers from the Program Committee. This clearly contributed to the significant number of papers submitted (95). Twenty-nine were accepted as regular papers, while 18 were accepted as short papers.

Discusses the technological developments, main issues, challenges, opportunities, and trends impacting every part of small to medium sized enterprises.

Because almost all technical systems are more or less interfaced with software these days, attacks against computer systems can cause considerable economic and physical damage. For this reason, understanding the dependability of such systems, as well as the improvement of cyber security and its development process, are amongst the most challenging and crucial issues in current computer science research. This book contains the lectures from the NATO Advanced Study Institute (ASI) Summer School entitled Engineering Dependable Software Systems, held in Marktoberdorf, Germany, in July and August 2012. This two week course for young computer scientists and mathematicians working in the field of formal software and systems was designed to give an in-depth presentation of state-of-the-art topics in the field, as well as promoting international contacts and collaboration and the teaming up of leading researchers and young scientists. The 12 lectures delivered at the school and presented here cover subjects including: model-based testing, formal modeling and verification, deductively verified software, model checking, performance analysis, integrating risk analysis, embedded systems and model checking, among others. The book will be of interest to all those whose work involves the development of large-scale, reliable and secure software systems.

This text combines a practical, hands-on approach to programming with the introduction of sound theoretical support focused on teaching the construction of high-quality software. A major feature of the book is the use of Design by Contract.

The more our society relies on electronic forms of communication, the more the security of these communication networks is essential for its well-functioning. Topics in this volume include developments in: security protocols; secure software engineering; and mobile agent security.

This book constitutes the refereed proceedings of the 6th International Conference on the Unified Modelling Language, UML 2003, held in San Francisco, CA, USA in October 2003. The 25 revised full papers, 4 tool papers, and 1 experience paper presented together with the abstracts of 3 invited talks and summaries on the UML 2003 workshop and tutorials were carefully reviewed and selected from initially 168 submissions. The papers are organized in topical sections on practical model management, time and quality of service, tools, composition and architecture, transformation, Web related issues, testing and validation, improving UML/OCL,

consistency, and methodology.

This book constitutes the refereed proceedings of the First joint International Software Process Workshop and the International Workshop on Software Process Simulation and Modeling, SPW/ProSim 2006. The 34 revised full papers presented together with 4 keynote addresses are organized in topical sections on process tailoring and decision-support, process tools and metrics, process management, process representation, analysis and modeling, process simulation modeling, process simulation applications, and experience report.

This volume focuses on the education of researchers, teachers, students and practitioners. As usual in engineering, a study and application of the relevant branches of mathematics is crucial both in education and practice.

[Copyright: 352ad9f5ec24ccf4938aeee01f1f9e3b](#)