applied sciences

MDPI

Editorial

Special Issue on Requirements Engineering, Practice and Research

Alberto Rodrigues da Silva [1,*] and Luis Olsina [2]

[1] INESC-ID, Instituto Superior Técnico, Universidade de Lisboa, 1000-029 Lisboa, Portugal
[2] Facultad de Ingeniería, Universidad Nacional de La Pampa, General Pico L6360, Argentina
* Correspondence: alberto.silva@tecnico.ulisboa.pt

1. Introduction

With recent developments in cloud and mobile computing technologies, the growing need for secure, trustworthy, and cost-efficient software and the shortage of highly skilled professional software developers have given rise to a new generation of problems that require improved forms of specification and representation of such systems at multiple abstraction levels, with various concerns and stakeholder perspectives.

Requirements engineering (RE) is a multidisciplinary, human-centered process integrated with both systems engineering and software engineering, which provides a shared vision and understanding of complex systems throughout their life cycles.

RE involves disparate activities such as eliciting, analyzing, defining, specifying, validating, and managing requirements. The literature extensively reports and discusses the negative consequences of ignoring these early RE activities.

RE needs to be sensitive to how people perceive and understand the world around them, how they interact with each other and their systems, and how the sociology and culture of the organizations affect their actions. Thus, RE draws on the cognitive and social sciences to provide theoretical grounding and practical techniques for eliciting and specifying requirements, including a diversity of areas such as computer science, software engineering, cognitive psychology, anthropology, sociology, and linguistics.

This Special Issue aimed to unite researchers and practitioners from the RE and related communities; explore the technologies that power contemporary RE tools and platforms; discuss the emerging problems and open challenges identified by the RE community; share case studies and empirical studies of applying RE tools and best practices; discuss textual- and visual-domain-specific languages; and discuss aspects of RE for emerging specific domains.

Topics of interest considered relevant were the following: RE control natural languages (CNLs); RE visual and modeling languages; RE for specific domains, such as cyber-physical systems, big data, or AI applications; automatic analysis of requirement specifications; RE and natural language processing (NLP) techniques; software tools and platforms for RE; RE quality; RE and software testing; RE and document automation techniques; and RE for privacy, security, and safety aspects.

This Special Issue received 17 submissions and accepted 10 high-quality papers (a 58.8% acceptance rate). Reviewing the chosen papers, various topics have been addressed in relation to three main groups: (i) CNL-based approaches and NLP techniques; (ii) RE and model-driven approaches; and (iii) other topics.

2. CNL-Based Approaches and NLP Techniques

Natural languages are the most common form of requirements representation; however, they exhibit characteristics that often introduce quality problems, such as inconsistency, incompleteness, and ambiguity. Three papers address these concerns by proposing

check for updates

Citation: Rodrigues da Silva, A.; Olsina, L. Special Issue on Requirements Engineering, Practice and Research. *Appl. Sci.* **2022**, *12*, 12197. https://doi.org/10.3390/app122312197

Received: 9 November 2022
Accepted: 24 November 2022
Published: 29 November 2022

and discussing approaches based on controlled natural languages (CNLs) and natural language processing (NLP) techniques to mitigate these quality problems.

A. R. da Silva and D. Savić (*Linguistic Patterns and Linguistic Styles for Requirements Specification: Focus on Data Entities*) [1] adopt the notions of linguistic pattern and linguistic style and discuss the relevance to produce better technical documentation. Their paper focuses on the textual specification of data entities, which are elements commonly referred to throughout different types of requirements, such as use cases, user stories, or functional requirements. This paper discusses how to textually represent the following elements: data entity, attribute, data type, data entity constraint, attribute constraint, and even cluster of data entities. This paper shows concrete examples and supports the discussion with three linguistic styles, represented by a rigorous requirements specification language and two informal controlled natural languages, one with a more compact and another with a more verbose, expressive, and complete representation. They analyzed how other languages coped with the representation of these data entity elements and complemented that analysis and comparison based on the PENS classification scheme. They conducted a pilot evaluation session and received encouraging feedback.

C. Liu, Z. Zhao, L. Zhang, and Z. Li (*Automated Conditional Statements Checking for Complete Natural Language Requirements Specification*) [2] argue that defects such as the duality and the incompleteness in natural language software requirements specification significantly affect the success of software projects. Thus, their paper focuses on the requirements incompleteness implied by the conditional statements and proposes a sentence-embedding- and antonym-based approach for detecting the requirements' incompleteness. The basic idea is that when one condition is stated, its opposite condition should also be present. Otherwise, the requirements specification is incomplete. Based on the state of the art of machine learning and natural language processing techniques, the proposed approach extracts conditional sentences from the requirements specification. It elicits conditional statements which contain one or more conditional expressions. Then, the conditional statements are clustered using the sentence-embedding technique. The conditional statements in each cluster are analyzed to detect potential incompleteness by using negative particles and antonyms. A benchmark dataset from an aerospace requirements specification was used to validate the proposed approach.

A. Gärtner, T. Fay, and D. Göhlich (*Fundamental Research on Detecting Contradictions in Requirements: Taxonomy and Semi-Automated Approach*) [3] argue that requirements documents can include several thousand individual requirements, which shall be error free to avoid unnecessary complications and costs in the later product development stages. An important part of their proposal is to identify contradictions between pairs of requirements. The first step is to define what contradictions are and in what form they can occur in requirement documents. Based on the Aristotelian Logic theory, the authors define specific subtypes of contradictions to match them to the requirements engineering field. The identification of these subtypes is conducted via a formalization of the requirement sentences and a subsequent analysis by means of simple questions. To validate the method, industrial requirement documents were searched for contradictions. For each detected type of contradiction, they present an example of the detection process. Thereby, they show that the method is easy to apply and may also be used by non-specialists. Thus, their approach provides a taxonomy as a basis for further research on automated contradiction detection and automated quality analysis of requirements documents.

3. RE and Model-Driven Approaches

Requirements engineering and model-driven approaches have been combined to produce better requirement specifications and quickly produce executable system mockups or prototypes from these specifications. Three papers discuss promising approaches that can leverage how we specify requirements and create new software systems.

M. Filipović, Ž. Vuković, L. Dejanović, and G. Milosavljević (*Rapid Requirements Elicitation of Enterprise Applications Based on Executable Mockups*) [4] discuss the use of executable UI

mockups in the initial phase of functional requirements elicitation during the development of business applications. Despite significant research progress in recent years, model-driven techniques still require improvement to enable customer participation from the initial phase of requirement specifications based on working prototypes. These prototypes can be reused directly throughout the development process. To meet this goal, the authors have developed an open-source solution called Kroki that enables rapid collaborative development. They conducted 10 joint user sessions with domain experts from different domains and backgrounds, resulting in prototype specifications ranging from 7 to 20 screen mockups accompanied with domain models, developed in two-hour time frames.

A. Carvalho (*REX: General-Purpose CNL with Code Generation Support*) [5] argues that controlled natural languages (CNLs) have been proposed to address some of the issues of natural language when it is used to express requirements. However, CNLs are based on formal grammar, which can be complex, hard to read, and especially difficult to write. Implementing support tools can also demand significant effort. Moreover, unanticipated constructions cannot be handled or must be dealt with in unexpected and cumbersome ways. This paper presents REX, a CNL with simple grammar that is easy to understand and support, but it is still a general-purpose language. To accomplish this, users specify their own language and how natural it is. Another benefit of CNLs is the possibility of automating the transformation of a text or specification into something useful, thereby reducing manual labor and transformation errors. The paper also presents the support tools used to transform a REX text into code of a complete application.

P. Galhardo and A. Silva (*Combining Rigorous Requirements Specifications with Low-Code Platforms to Rapid Development Software Business Applications*) [6] discuss a model-driven approach that semi-automatically produces software business applications by combining rigorous requirement specifications (defined with the ITLingo ASL language) with a concrete low-code platform (Quidgest Genio). First, they analyze the common concepts in both ITLingo ASL and Genio languages. Then, they discuss model-to-model transformations that allow ASL specifications to be converted into Genio low-code projects. Finally, the code generation capabilities of the Genio low-code platform are employed to generate the source code of the target software applications. To evaluate the consistency of the proposed approach, they use and discuss a simple and representative case study based on a fictitious system, the Invoice Management System (IMS), whose requirements are like those found in many business applications, for instance, as previously discussed in [1].

4. Other Topics

Finally, this Special Issue includes four papers that address complementary concerns about software customization and its impact on the quality of SasS, product derivation methods, proactive management of change requirements, and a literature review on RE for IoT systems.

A. Qasem Ali, A. Md Sultan, A. Abd Ghani, and H. Zulzalil (An Empirical Investigation of Software Customization and Its Impact on the Quality of Software as a Service: Perspectives from Software Professionals) [7] argue that, although customization plays a significant role in the provision of software as a service (SaaS), delivering a customizable SaaS application that reflects the tenant's specific requirements with an acceptable level of quality is a challenge. Drawing on a pre-developed software customization model for SaaS quality, two fundamental objectives of this study were to determine whether different software customization approaches directly affect SaaS quality and to assess the reliability and validity of the model. A questionnaire-based survey was used to collect data from 244 software professionals with experience in SaaS development. Structural equation modeling was employed to test the reliability, validity, and research hypotheses. The measurement model assessment suggested that the six-construct model with 39 items exhibited good reliability and validity. The findings of the structural model assessment show that all customization approaches other than the integration approach significantly influence the quality of SaaS applications. The results also indicate that both configuration and composi-

tion approaches positively impact SaaS quality, while the impacts of the other approaches are negative. This model's empirical assessment and evaluation, which features a rich set of information, provides considerable benefits to both researchers and practitioners.

L. Gräßler, C. Oleff, and D. Preuß (*Proactive Management of Requirement Changes in the Development of Complex Technical Systems*) [8] observe that requirement changes and cascading effects of change propagation are significant sources of inefficiencies in product development and increase the risk of project failure. Thus, proactive change management of requirement changes yields the potential to handle such changes efficiently. The paper proposes a systematic approach to proactive change management to assess and reduce the risk of a requirement change with reasonable effort in industrial application, named Proactive Management of Requirement Changes (ProMaRC). This approach was developed in close collaboration with industry experts and evaluated based on workshops, pilot users' feedback, three industrial case studies from the automotive industry and five development projects from research. To limit the application effort, the authors also developed an automated technique for dependency analysis based on the machine learning technique BERT and semi-automated assessment of change likelihood and impact using a modified PageRank algorithm. Applying that technique, the risks of requirement changes are assessed systematically and reduced by means of proactive change measures. The evaluation shows a high performance of dependency analysis and confirms the applicability and usefulness of the technique.

X. Wang, W. Wang, and H. Liu (*Product Model Derivation from Feature Model and Formal Specification*) [9] consider that existing product derivation methods (i.e., the process of building a specific product from a software product line) can only obtain abstract feature models, lacking detailed specifications of individual features. The authors argue that these models focus on deriving code assets or class diagram templates without precise model descriptions for specific products. Thus, they propose a product derivation approach to obtain a formal specification of a particular product based on the feature model and formal specification. They use the integration ordering and behavior-preserving integration techniques to integrate the formal specification for each feature pair. That method is divided into two steps. First, it determines the feature formal specification integration ordering based on the feature model. Second, the behavior-preserving integration is conducted for pairs, including declaration integration, functional scenario path generation, and function integration based on path matching. Behavior-preserving integration guarantees consistent behavior to ensure the quality of the formal specification after integration. The authors also developed a support tool with which they conducted a case study. That tool first guides the user to perform feature functional scenario path matching, then performs functional integration based on the matching results and repeats the above steps to generate a product model.

J. Aguilar-Calderón, C. Tripp-Barba, A. Zaldívar-Colado, and P. Aguilar-Calderón, (*Requirements Engineering for Internet of Things (loT) Software Systems Development: A Systematic Mapping Study*) [10] argue that the development of IoT systems is complicated compared to that of traditional software systems, especially concerning RE. The RE of IoT systems is not implemented frequently due to their broad aspects, such as their variety of user needs, making these systems challenging to construct. Thus, the research community has not explored the use of loT-based systems to provide well-planned proposals to improve their quality. In this paper, the authors present a comprehensive and inclusive review of the RE techniques for IoT-based systems, based on the systematic mapping study (SMS) process used to sort and organize research publications to gain knowledge on progress and identify research gaps. Thus, this paper classifies existing research publications concerning the RE techniques used for IoT systems and discusses the implications for future research.

Funding: This research received no external funding.

Acknowledgments: This Special Issue was made possible by the many talented authors, hardworking reviewers, and the dedicated editorial team of *Applied Sciences*. Thanks to all authors; regardless of the final decisions of the submitted manuscripts, the reviewers and editors' feedback, comments, and suggestions helped the authors improve their work. Thanks to all the anonymous reviewers for their commitment, constructive criticism, and suggestions that helped improve the submitted papers. Finally, our thanks to the editorial team of *Applied Sciences*, who were very helpful and professional throughout this process.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Silva, A.R.; Savić, D. Linguistic Patterns and Linguistic Styles for Requirements Specification: Focus on Data Entities. *Appl. Sci.* **2021**, *11*, 4119. [CrossRef]
2. Liu, C.; Zhao, Z.; Zhang, L.; Li, Z. Automated Conditional Statements Checking for Complete Natural Language Requirements Specification. *Appl. Sci.* **2021**, *11*, 7892. [CrossRef]
3. Gärtner, A.; Fay, T.; Göhlich, D. Fundamental Research on Detecting Contradictions in Requirements: Taxonomy and Semi-Automated Approach. *Appl. Sci.* **2022**, *12*, 7628. [CrossRef]
4. Filipović, M.; Vuković, Ž.; Dejanović, I.; Milosavljević, G. Rapid Requirements Elicitation of Enterprise Applications Based on Executable Mockups. *Appl. Sci.* **2021**, *11*, 7684. [CrossRef]
5. Carvalho, A. REX: General-Purpose CNL with Code Generation Support. *Appl. Sci.* **2022**, *12*, 7700. [CrossRef]
6. Galhardo, P.; Silva, A.R. Combining Rigorous Requirements Specifications with Low-Code Platforms to Rapid Development Soft-ware Business Applications. *Appl. Sci.* **2022**, *12*, 9556. [CrossRef]
7. Qasem Ali, A.; Md Sultan, A.; Abd Ghani, A.; Zulzalil, H. An Empirical Investigation of Software Customization and Its Impact on the Quality of Software as a Service: Perspectives from Software Professionals. *Appl. Sci.* **2021**, *11*, 1677. [CrossRef]
8. Gräßler, I.; Oleff, C.; Preuß, D. Proactive Management of Requirement Changes in the Development of Complex Technical Systems. *Appl. Sci.* **2022**, *12*, 1874. [CrossRef]
9. Wang, X.; Wang, W.; Liu, H. Product Model Derivation from Feature Model and Formal Specification. *Appl. Sci.* **2022**, *12*, 6241. [CrossRef]
10. Aguilar-Calderón, J.; Tripp-Barba, C.; Zaldívar-Colado, A.; Aguilar-Calderón, P. Requirements Engineering for Internet of Things (loT) Software Systems Development: A Systematic Mapping Study. *Appl. Sci.* **2022**, *12*, 7582. [CrossRef]