Requirements Development and Management of Embedded Real-Time Systems

Jiale Zhou
School of Innovation, Design and Engineering
Mälardalen University
Västerås, Sweden
zhou.jiale@mdh.se

***Abstract*—It is well recognized that most of the anomalies, discovered in the development of embedded real-time systems, belong to requirement and specification phases. To ease the situation, many efforts have been investigated into the area. For requirements development, especially requirements validation and verification, model-driven architecture techniques can be considered as a cost-efficient solution. In order to utilize such advantages, the design of the proposed system is often specified in terms of analyzable models at the certain level of abstraction. Further, different levels of requirements are translated into verifiable queries and fed into the models to be either validated or verified. For requirements management, requirements traceability provides critical support for performing change impact analysis, risk analysis, regression testing, etc. In this thesis, we cover several topics about requirements validation, requirements verification, and requirements traceability. In particular, the technical contributions are three-fold: 1) we propose an approach to requirements validation by using the extended Timed Abstract State Machine (TASM) language with newly defined TASM constructs and, 2) we present a simulation-based method which is powered up by statistical techniques to conduct requirements verification, working with industrial applications and, 3) we introduce an improved VSM-based requirements traceability recovery approach using a novel context analysis. Further, we have demonstrated the applicability of our contributions in real world usage through various case studies.**

I. Introduction

With the growing complexity of Embedded Real-Time Systems (ERTS), requirements development and management become more critical activities for developing such systems. Studies have revealed that most of the anomalies, discovered in the development of complex systems, belong to requirement and specification phases [1] [2] [3]. A *Requirement* is defined as "a statement that identifies a product or process operational, functional, or design characteristic or constraint, which is unambiguous, testable or measurable, and necessary for product or process acceptability (by consumers or internal quality assurance guidelines)" [4]. There is a common misconception that requirements are just specified at the outset of the System Development Life Cycle (SDLC). On the contrary, various levels of requirements should be specified for the corresponding phases of the SDLC, rather than a single phase. In this thesis, we follow a three-level requirements model, which consists of customer-level, system-level, and architecture-level (design-level in other words), as shown in Figure 1. In the V-model SDLC, requirements are firstly specified at corresponding levels of abstraction and then are used for different integration and verification purposes. More detailed description of Figure 1 can be found in Section II.

Requirements Development (RD) includes, but is not limited to, the tasks of:

- eliciting, analyzing, validating and communicating customer-level requirements,
- transforming customer-level requirements into subsequent levels of requirements,
- allocating requirements to hardware, software and test cases,
- verifying requirements,
- validating the set of requirements.

Many efforts have discussed the importance of a set of well-defined requirements [5] [6] and the implications caused by ill-defined requirements [2]. Nevertheless, it is still a challenge to define a set of complete, correct, and verifiable requirements. Especially, non-functional requirements of ERTS, in terms of timing and resource consumption (i.e., power consumption in this thesis), are of great importance to be validated and verified. RD using Model-Driven Architecture (MDA) techniques [7] can remedy the situation, which can be considered as a cost-efficient solution. Typically, model-based RD has the advantages of accelerating project development, requiring less human efforts, and making it possible to detect and fix the potential defects and errors on a fairly early stage of the SDLC. In order to utilize such advantages, the design of the proposed system is often specified in terms of analyzable models at the certain level of abstraction. Further, different levels of requirements are translated into verifiable queries and fed into the models to be either validated or verified.

Requirements Management (RM) plays an important role in keeping track of requirements and handling a large amount of software development artifacts (i.e., the artifacts hereafter), such as design specification, source codes, and test cases. The quality of RM is of great importance for systems development, mattering to e.g., customers satisfaction, requirements coverage, efficient utilization of money, time and other resources. The heart of RM is Requirements Traceability (REQT), which is defined as "the ability to describe and follow the life of a requirement, in both a forwards and backwards direction (i.e., from its origins, through its development and specification,

978-1-4799-3032-6/14/$31.00 © 2014 IEEE

RE 2014, Karlskrona, Sweden

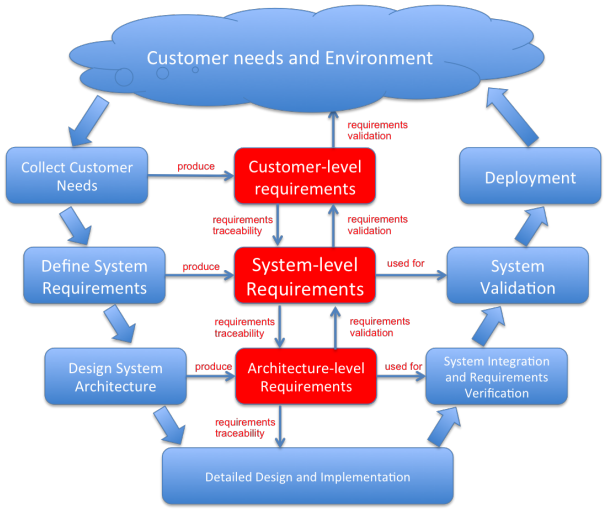

Fig. 1. The V-model SDLC and the three-level requirements model considered in this thesis.

to its subsequent deployment and use, and through periods of on-going refinement and iteration in any of these phases)" [8]. REQT not only provides critical support for system developers throughout the entire SDLC, but also is necessarily required by the certification process. Tracing requirements can help to, but is not limited to, perform change impact analysis, risk analysis, criticality analysis, regression testing, etc. However, there are numerous challenges that make it difficult to achieve successful traceability in practice. These challenges include technical issues related to physically creating, recovering, maintaining, and using thousands of interrelated and relatively vulnerable traceability links[1]. As a result, many organizations and companies are unwilling to implement and maintain traceability links, even though it is widely accepted as a critical element of the SDLC. In order to overcome the significant challenges in creating, recovering, maintaining, and using traceability, the research community has been actively addressing traceability challenges through the exploration of topics related to automating the traceability creation, facilitating the evolution and maintenance of trace links, visualizing traceability, verifying and validating systems development through traceability practices, and developing traceability practices that apply across a wide range of domains such as safety-critical applications, aspect-oriented and agile software development, as well as various regulated industries [9].

In this thesis, we cover several topics centering around RD and RM, which are mainly about requirements validation and verification, and REQT. In particular, the technical contributions are three-fold:

- We propose an approach to requirements modeling by using the Timed Abstract State Machine (TASM) language with some necessary extensions and the TASM query mechanism. To be specific, our approach describes both functional behaviors and non-functional properties, including timing and resource consumption, of the target system, and it provides a means to requirements validation and verification by using the developed TASM Toolset and a model checker.
- We extend TASM with the capabilities of modeling non-functional properties of ERTS w.r.t timing and power consumption, using probabilistic distributions. To this end, we propose a simulation-based method which is powered up by advanced statistical techniques to conduct requirements verification. Moreover, the extended TASM inherits the easy-to-use features from the Abstract State Machine language, which is a literate specification language understandable and usable without extensive mathematical training.
- We introduce an improved VSM-based requirements traceability recovery approach using a novel context analysis. As aforementioned, in the three-level requirements model featured with customer-level, system-level and architecture-level, our method can better utilize the context information extracted from the high-level requirements to discover the subsequent artifacts at the lower levels.

[1]Once an endpoint of a trace link is changed, the trace link is no more valid.

II. Background

This section firstly gives some background knowledge about requirements development centering around requirements model, requirements validation and verification in Sections II-A1 and II-A2 respectively. Next a brief overview of IR-based traceability recovery as in requirements management is presented in Section II-B.

A. Requirements Development

1) Requirements Model: Although there are various models for developing systems for different sizes and areas of projects [10], the development of requirements are usually divided into layers or levels which have different nuances but are similar. In this thesis, we follow the three-level requirements model, as shown in Figure 1. At the beginning of the SDLC, it is an essential task to elicit a sound set of requirements for the proposed system based on the opinions of various stakeholders, i.e., customer-level requirements. In doing so, a basis for discussion with the stakeholders and a means of clarifying their needs together with some potential solutions will be given. One example is "the customer wants a Brake-by-Wire system to substitute the traditional mechanical brake system". Rather than proceeding straight to the design phase, it is necessary to first determine what functionalities and properties the proposed system must have, regardless of more detailed design. Therefore, the initial customer-level requirements will be further transformed, in terms of being decomposed or refined, into a set of system-level requirements, to help to establish a common understanding of the proposed solution to the domain-specific problem. One typical example is "the system shall provide a base brake functionality where the driver presses the brake pedal so that the braking system starts decelerating the vehicle". Based off of the system-level

requirements, it is now possible to consider alternative design architectures. The design architecture defines what components the proposed system consists of and how such components interact with each other. Subsequently, the architecture-level requirements stipulate the requirements for each system component w.r.t their functionalities, interaction constrains, and any other required properties featured with performance quality, reliability, safety, etc. One example is "the brake torque calculator shall compute the driver requested torque and send the value to the vehicle brake controller, when a brake pedal displacement is detected". Note that, in some cases, the components at the architecture-level are still too complex to be implemented directly, and therefore will be further refined into more specific requirements corresponding to software components, hardware components which are associated with system implementation.

2) Requirements Validation and Verification: Requirements validation and verification are two independent activities that are performed together to ensure that the final system would meet requirements and specifications, which are imposed at the beginning of the development process. Different from system validation which confirms that the built system does what it is supposed to do, requirements validation is the process of confirming the *completeness* and *consistency* of requirements. Further, once completeness and consistency of requirements are validated, then the *correctness* of such requirements can be achieved. To be specific:

1) Completeness refers to situations where a specification entails all the requirements that are regarded as necessary to satisfy the proposed system, and there is no undefined object or entity.
2) Consistency refers to situations where a specification contains no internal contradictions.

Each of the three-level requirements has their own validation purpose as follows:

- The customer-level requirements must be guaranteed that they collect all of the stated business objectives, clarify the needs of various stakeholders, and are easy-to-use for communication between stakeholders and developers.
- The system-level requirements provide a high-level solution to implementation of the designed system, which must be validated to be sure that they address the stated business objectives, meet the needs of stakeholders, and contain no internal contradictions.
- The validation procedure for the architecture-level requirements not only ensures that they are consistent and complete with the upper-level requirements (i.e., system-level requirements), but also makes sure the design solution is feasible, unambiguous and verifiable.

Requirements verification is the process of confirming that the designed and implemented system fully addresses documented requirements. Contrary to the purpose of requirements validation focusing on "Do the right thing", requirements verification ensures the outcome at each phase of the SDLC with the satisfaction of the pertaining requirements, in the sense of "Do the thing right". In the model-based development, verification procedure involves modeling of the target system by using domain-specific models, which will then be analyzed by using either simulation-based methods or exhaustive analysis in terms of model checking. Among various techniques for model-based requirements verification, one interesting piece of work to mention is Statistical Model Checking (SMC) [11], which plays a more and more important role nowadays. Specifically, SMC refers to a series of techniques that observe several runs of the (system) model with respect to certain properties, and then analyze the results from the perspective of statistics, which are used to check the design correctness. There are several advantages of this application: 1) It is quite simple to implement, understand and use in practice and, 2) It requires very little or no extra modeling efforts, but simply an operational model of the target system that can be simulated w.r.t the property and, 3) The use of statistics allows to analyze non-deterministic problems. SMC gives us a strong motivation to power up our specification language, namely TASM, with advanced statistical methods.

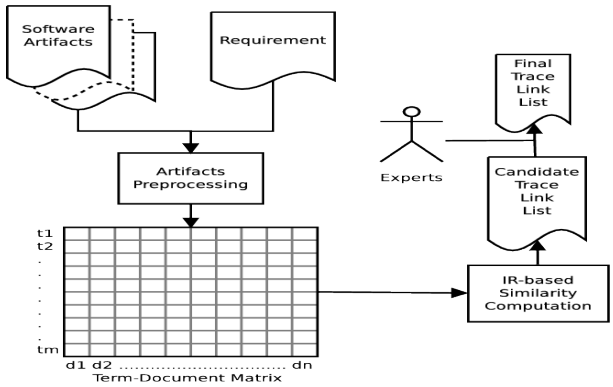

Fig. 2. An IR-based traceability recovery process.

B. Requirements Traceability

The heart of RM is building requirements traceability. The IR-based traceability recovery aims at utilizing information retrieval (IR) techniques to compare a set of source artifacts as queries (e.g., requirements), against another set of target artifacts (e.g., source code files), and to calculate the textual similarities of all possible pairs of artifacts. The textual similarity between two artifacts is based on the occurrences of terms (words) within the artifacts contained in the repository. Pairs with a similarity score lower than a certain threshold (usually defined based on engineers' experience) are filtered out, and the reserved pairs form the candidate trace link list. The ranked list of candidate trace links are then analyzed by software engineers to decide if such links are true positive or not. Typically, an IR-based traceability recovery process follows the steps depicted in Figure 2. The artifacts have to be preprocessed before they are used to compute similarity scores. The preprocessing of the artifacts includes a text normalization by removing most non-textual tokens (e.g., operators, punctuations) and splitting compound identifiers into separate words by using the underscore or Camel Case

splitting heuristic. Furthermore, common terms, referred to as "stop words" (e.g., articles, prepositions, common usage verbs, and programming language keywords), which contribute less to the understanding about artifacts, are also discarded by using a stop word filter. Words with the length less than a defined threshold are also pruned out. In addition, stemmer is commonly used to perform a morphological analysis, which reduces the inflected words to their root, e.g., returning verb conjugations and removing plural nouns. After preprocessing, an artifact (e.g., a UC requirement, a source code file) can be represented as a plain document containing a list of terms (in this paper, we use *documents* and *artifacts* interchangeably). The extracted terms are generally stored in a $m \times N$ matrix (called term-by-document matrix), where m is the number of all the terms that occur in all the documents, and N is the number of documents in the corpus. A generic entry $w_{i,j}$ of the matrix denotes a measure of the relevance of the i_{th} term in the j_{th} document. Based on the term-by-document matrix representation, different IR methods can be used to calculate textual similarities between paired artifacts.

III. Research Questions

In this thesis, we are aiming at contributing to a general research question as follow:

- **General research question:** How to improve requirements development and management of (complex dependable) embedded real-time systems in a structured and practical way, in accordance with industrial demand?

In particular, the general research question can be further refined into three specific research questions which guide our work:

- **Research question one (RQ1):** How can we validate non-functional requirements in terms of timing and resource consumption requirements, in an early stage of model-based development of embedded real-time systems?
  - **The challenge associated with RQ1 (Ch1):** In order to increase the confidence in the correctness of the requirements, model-based *formal methods* techniques have been to a large extend investigated into the field of requirements validation [12] [13]. In these techniques, the system structure and behaviors derived from low-level requirements is often specified in terms of analyzable models at a certain level of abstraction. Further, high-level requirements are formalized into verifiable queries or formulas and then fed into the models to perform model checking and/or theorem proving. In this way, the low-level requirements are reasoned about to resolve contradictions, and it is also verified that they are neither so strict to forbid desired behaviors, nor so weak to allow undesired behaviors. However, such formal methods techniques also suffer from some limitations, such as how to ease the demand of heavy mathematics background knowledge to perform theorem proving, and how to model the target without having the state explosion problem of model checking occurred.
  - **Our contribution to RQ1 (OC1):** We extend TASM with the *Observer* construct, and propose an approach to requirements modeling by using the TASM language for checking the completeness and consistency of functional and non-functional requirements at both the system-level and architecture-level.
- **Research question two (RQ2):** How can we verify system design in accordance with non-functional requirements in embedded real-time systems, featured with large-size and complicated behaviors of timing and resource consumption?
  - **The challenge associated with RQ2 (Ch2):** For any embedded real-time system for instance of consumer electronics and automobiles, it should be ensured that the system design indeed meets all the specified requirements in terms of functional behaviors and non-functional properties, in a particular context. Early-stage abstract models of the system's structure and behaviors can become the artifacts on which exhaustive analysis is usually carried out, in an attempt to increase trust in the system's correct operation. However, the most common limitation of exhaustive analysis is the state-explosion problem which can cost unacceptable time. In particular, when the system behaves in a non-deterministic way, it is almost impossible to perform exhaustive analysis.
  - **Our contribution to RQ2 (OC2):** We extend TASM with the capabilities of modeling the behaviors of systems confirming to probabilistic distributions, and then apply some statistical techniques to analyze models w.r.t functional behaviors and non-functional properties in terms of timing and resource-usage, for verification purpose.
- **Research question three (RQ3):** How can we improve the accuracy of candidate trace links between requirements and the subsequent artifacts of embedded real-time systems development?
  - **The challenge associated with RQ3 (Ch3):** In traditional industrial practices, especially for legacy systems [14], trace links are manually established and maintained. Such activities tend to be costly to implement and are therefore perceived as financially non-viable by many companies [8], [15]. To address this problem, many efforts [16], [17], [18], [19], [20], [21], [22] have been devoted to semi-automatic or fully-automatic trace link creation. However, the precision and recall of generated trace links is still at a low level of accuracy, such that the trace link creation throughout the entire systems development process, remains a challenging issue [23].
  - **Our contribution to RQ3 (OC3):** We improve the VSM-based requirements traceability recovery approach by using a novel context analysis. Specifically,

the analysis method can better utilize context information extracted from requirements (e.g., use cases) to discover relevant subsequent artifacts (e.g., source code files).

IV. Research Methodology

The overarching goal of our research is to improve the development process of software-intensive embedded real-time systems. Our approach focuses on extending the theoretical knowledge in the areas of requirements specification (i.e., requirements modeling in this thesis), requirements verification and validation, and requirements traceability. In order to adequately address the questions listed above, it is important to adopt an appropriate research methodology, suitable for such a given setting. The methodology used in our research is based on the research steps proposed by Shaw [24], which is summarized in the following four steps:

1) Conducting literature reviews and surveys to form new research goals in specific.
2) Analysis of the current state-of-the-art progress in requirements development and management field based on the defined research questions.
3) Answering the research questions by presenting the proposed solutions and achieved research results.
4) Validating whether the research results can be applied in the real-world applications.

V. Papers

In this section we provide the included papers in this thesis. Table I presents the relations between the research questions and papers.

TABLE I
Relations between the papers and research questions.

Paper	Research Question	Current State
Paper A	RQ1	Published at Ada-Europe'14
Paper B	RQ1	Accepted by RE'14
Paper C	RQ1&2	Plan to submit to RET'14
Paper D	RQ2	To be decided
Paper E	RQ3	Published at SEAA'13

- **Paper A:** ***A TASM-based Requirements Validation Approach for Safety-critical Embedded Systems*** *Published at the 19th International Conference on Reliable Software Technologies - Ada-Europe (Ada-Europe'14), 2014.*
  Requirements validation is an essential activity to carry out in the software development life cycle, and it confirms the completeness and consistency of requirements through various levels. Model-based formal methods can provide a cost-effective solution to requirements validation in a wide range of domains such as safety-critical applications. In this paper, we extend a formal language Timed Abstract State Machine (TASM) with two newly defined constructs *Event* and *Observer*, and propose a novel requirements validation approach based on the extended TASM. Specifically, our approach can: 1) model both functional and non-functional (e.g. timing and resource consumption) requirements of the system at different levels and, 2) perform requirements validation by utilizing our developed toolset and a model checker. Finally, we demonstrate the applicability of our approach in real world usage through an industrial case study of a Brake-by-Wire system.
- **Paper B:** ***Towards Feature-oriented Requirements Validation for Automotive Systems*** *Accepted by the 22nd IEEE International Requirements Engineering Conference (RE'14), 2014.*
  In the modern automotive industry, feature models have been widely used as a domain-specific requirements model, which can capture commonality and variability of a software product line through a set of features. Product variants can thus be configured by selecting different sets of features from the feature model. For feature-oriented requirements validation, the variability of feature sets often makes the hidden flaws such as behavioral inconsistencies of features, hardly to avoid. In this paper, we present an approach to feature-oriented requirements validation for automotive systems w.r.t both functional behaviors and non-functional properties. Our approach first starts with the behavioral specification of features and the associated requirements by following a restricted use case modeling approach, and then formalizes such specifications by using a formal yet literate language for analysis. We demonstrate the applicability of our approach through an industrial application of a vehicle locking-unlocking system.
- **Paper C:** ***The Observer-based Technique for Requirements Validation in Embedded Real-time Systems*** *Plan to submit to the 1st International Workshop on Requirements Engineering and Testing.*
  Model-based requirements validation is an increasingly attractive approach to discovering hidden flaws in the requirements in the early phases of systems development life cycle (SDLC). The application of using traditional methods such as model checking for the validation purpose is limited by the growing complexity of embedded real-time systems (ERTS). The observer-based technique is a lightweight validation technique, which has shown its potential as a means to validate the correctness of model behaviors. In this paper, the novelty of our contribution is three-fold: 1) we formally define the observer constructs and the corresponding operational semantics and, 2) we propose the Events Monitoring Logic (EvML) to facilitate the observer specification and, 3) we show how to use observers to validate the requirements describing the system functional behaviors and non-functional properties (such as timing) of ERTS. We also illustrate the applicability of the extended TASM through an industrial application of a Vehicle Locking-Unlocking system.

- **Paper D:** ***Title To Be Decided*** *Conference to be decided.* To model and analyze complicated behaviors of non-functional properties of ERTS w.r.t timing and power consumption, is a challenging yet critical task for requirements verification. We propose a solution to remedy the situation by extending TASM with the capabilities of modeling such complex system behaviors in an elegant and powerful manner using probabilistic distributions. For the purposes of requirements verification, we power up simulation-based method by using some advanced statistical techniques, with the focus on analysis of both average cases and extreme cases of the target system. More interestingly, our approach inherits the easy-to-use and easy-to-learn features from ASM, which is a literate specification language understandable and usable without extensive mathematical training.

- **Paper E:** ***A Context-based Information Retrieval Technique for Recovering Use-Case-to-Source-Code Trace Links in Embedded Software Systems*** *Published at the 39th Euromicro Conference on Software Engineering and Advanced Applications (SEAA'13), 2013.*
  Requirements traceability is the ability to relate requirements (e.g., use cases) forward to corresponding design documents, source code and test cases by establishing trace links. This ability is becoming ever more crucial within embedded systems development, as a critical activity of testing, verification, validation and certification. However, semi-automatic or fully-automatic generating accurate trace links remains an open research challenge, especially for legacy systems. The contribution of this paper is an improved VSM-based requirements traceability recovery approach using a novel context analysis. Specifically, the analysis method can better utilize context information extracted from use cases to discover relevant source code files. Our approach is evaluated by using three different embedded applications in the domains of industrial automation, automotive and mobile platform.

VI. Time Plan

The following activities, as shown in Table II, are planned to be completed before the licentiate defence. The proposed time plan starts from March, 2014:

TABLE II
Time Plan

Activity	Time	Date
Writing paper C	5 weeks	May
Writing paper D	4 weeks	June
First draft of the thesis	5 weeks	July
Final version of the thesis	2 weeks	August
Defence		**Oct**

The licentiate thesis defence can be performed in October 2014.

References

[1] N. G. Leveson, *Safeware: System Safety and Computers*. NY, USA: ACM, 1995.

[2] A. T. Bahill and S. J. Henderson, "Requirements development, verification and validation exhibited in famous failures," *Syst. Eng*, 2005.

[3] A. Ellis, "Achieving safety in complex control systems," in *Proceedings of SCSC'95*. Springer London, 1995, pp. 1–14.

[4] M. E. C. Hull, K. Jackson, and J. Dick, Eds., *Requirements Engineering, Third Edition*. Springer, 2011.

[5] B. Berenbach, F. Schneider, and H. Naughton, "The use of a requirements modeling language for industrial applications," in *Proceedings of RE'12*, 2012, pp. 285–290.

[6] E. Bjarnason, P. Runeson, M. Borg, M. Unterkalmsteiner, E. Engström, B. Regnell, G. Sabaliauskaite, A. Loconsole, T. Gorschek, and R. Feldt, "Challenges and practices in aligning requirements with verification and validation: a case study of six companies," *Empirical Software Engineering*, pp. 1–47, 2013.

[7] Model-Driven Architecture (MDA), http://www.omg.org/mda/, November Accessed: November 2011.

[8] O. C. Z. Gotel and A. C. W. Finkelstein, "An Analysis of the Requirements Traceability Problem," in *Proceedings of the 2nd IEEE International Requirements Engineering Conference (RE' 94)*, 1994, pp. 94–101.

[9] C.-H. Jane, G. Orlena, and Z. Andrea, *Software and Systems Traceability*. Springer, 2012.

[10] N. M. A. Munassar and A. Govardhan, "A comparison between five models of software engineering," *IJCSI*, vol. 7, no. 5, pp. 94–101, 2010.

[11] P. E. Bulychev, A. David, K. G. Larsen, M. Mikucionis, D. B. Poulsen, A. Legay, and Z. Wang, "Uppaal-smc: Statistical model checking for priced timed automata," in *QAPL*, 2012, pp. 1–16.

[12] A. Cimatti, M. Roveri, A. Susi, and S. Tonetta, "From informal requirements to property-driven formal validation," in *Proceedings of FMICS'09*. Berlin, Heidelberg: Springer-Verlag, 2009, pp. 166–181.

[13] A. Iliasov, "Augmenting formal development with use case reasoning," in *Proceedings of Ada-Europe'12*. Springer Berlin Heidelberg, 2012, pp. 133–146.

[14] J. Kraft, Y. Lu, C. Norström, and A. Wall, "A Metaheuristic Approach for Best Effort Timing Analysis targeting Complex Legacy Real-Time Systems," in *Proceedings of the 14th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS' 08)*, 2008.

[15] B. Ramesh and M. Jarke, "Toward reference models for requirements traceability," *IEEE Trans. Softw. Eng.*, vol. 27, no. 1, pp. 58–93, Jan. 2001.

[16] S. Deerwester, S. Dumais, T. Landauer, G. Furnas, and L. Beck, "Improving information retrieval with latent semantic indexing," in *Annual Meeting of the American Society for Info. Science 25*, 1988.

[17] D. Blei, A. Ng, and M. Jordan, "Latent Dirichlet Allocation," *Journal of Machine Learning Research*, no. 3, pp. 993–1022, 2003.

[18] J. Cleland-Huang, R. Settimi, C. Duan, and X. Zou, "Utilizing supporting evidence to improve dynamic requirements traceability," in *Proceedings of the 13th IEEE International Conference on Requirements Engineering*, ser. RE '05. Washington, DC, USA: IEEE Computer Society, 2005, pp. 135–144.

[19] N. Ali, Y.-G. Gueheneuc, and G. Antoniol, "Trust-Based Requirements Traceability," in *Proceedings of the 19th International Conference on Program Comprehension (ICPC' 11)*, 2011, pp. 111–120.

[20] W.-K. Kong and J. H. Hayes, "Proximity-based Traceability: An Empirical Validation using Ranked Retrieval and Set-based Measures," in *Proceedings of the 1st International Workshop on Empirical Requirements Engineering (EmpiRE' 11)*, 2011, pp. 45–52.

[21] A. D. Lucia, M. D. Penta, R. Oliveto, A. Panichella, and S. Panichella, "Improving ir-based traceability recovery using smoothing filters," *International Conference on Program Comprehension*, vol. 0, pp. 21–30, 2011.

[22] A. Mahmoud, N. Niu, and S. Xu, "A semantic relatedness approach for traceability link recovery." in *Proceedings of the 20th IEEE International Conference on Program Comprehension (ICPC' 12)*, 2012, pp. 183–192.

[23] O. Gotel, J. Cleland-Huang, J. H. Hayes, A. Zisman, A. Egyed, P. Grünbacher, A. Dekhtyar, G. Antoniol, and J. Maletic, *The Grand Challenge of Traceability (v1.0)*. Springer-Verlag London Limited, 2012, pp. 343–412.

[24] M. Shaw, "The coming-of-age of software architecture research," in *Proceedings of ICSE'01*. Washington, DC, USA: IEEE Computer Society, 2001, pp. 656–664a.