Towards a Systematic Requirement-Based Test Generation Framework: Industrial Challenges and Needs

Shokoofeh Hesari
Certus V&V Centre, Simula Research Laboratory
University of Oslo
Oslo, Norway

Razieh Behjati, and Tao Yue
Certus V&V Centre, Simula Research Laboratory
Oslo, Norway

***Abstract*— Requirement-based test generation (RBTG) is a verification and validation technique, which ensures the conformance of a final product with its requirements. In collaboration with an industry partner, we studied and analyzed their current practice of applying RBTG in the context of developing a family of subsea oil and gas production systems, which are cyber-physical systems. The company aims at improving their current RBTG practice by enhancing the reuse of test artifacts across different products. Due to the complexity of developing such systems and being in the context of system product-line engineering, achieving this goal requires a systematic approach for RBTG. As the first step to this end, we conducted a domain analysis with the industry partner to characterize their current practice of applying RBTG and to identify their needs and challenges. In this paper, we report results of the domain analysis. Moreover, we discuss the limitations of employing existing RBTG approaches in an industrial setting and suggest directions for improvement.**

Index Terms- Requirement-based test generation; Product Line Engineering; Requirements Specification; Natural language; Test Derivation; Test Scenarios; Configuration.

I. INTRODUCTION

To increase quality and productivity and to reduce costs of product development, many industrial organizations resort to software product-line engineering approaches [1][2], which enable the generation of a multitude of product variants at reasonable cost. Since 1976, when the idea of software product families was described in [3], many traditional software development techniques have been adapted for product-lines, where reuse and customization of the development artifacts (e.g., requirements, design, and test) is a fundamental aspect. Among these techniques is the requirement-based test generation technique—a verification and validation technique that ensures the conformance of products to their specifications. Several approaches (e.g., [4][5][6]) to requirement-based test generation in the context of product families have been proposed in the literature. To the best of our knowledge, however, the application of requirement-based test generation in large-scale industrial product families and challenges being faced by industry are rarely reported in the literature.

In collaboration with an industry partner, FMC Technologies Inc., [7], we studied potentials and challenges of applying requirement-based test generation in an industrial setting. FMC is a global provider of subsea oil production systems. These systems are usually categorized as cyber-physical systems [8], which are large-scale, distributed software-intensive systems. The company has a number of product families and employs requirement-based test generation to validate and verify their products.

To reduce the cost and effort of test generation, FMC is looking for an approach to maximize the reuse of its test artifacts. In particular, the company seeks solutions [9] that can automatically derive product-specific test artifacts from product requirements configuration information.

We conducted a domain analysis to characterize the current practice of requirement-based test generation at the company. We further identified the company's expectations from the aforementioned automation support for reuse. These expectations are reported in this paper in the form of a set of needs, which are then used as a basis for evaluating the existing approaches, identifying the gap, and discussing possible directions for improvement.

Results of evaluating the literature show that existing requirement-based test generation solutions and commercial or open source requirements engineering tools should be tailored and extended to accommodate the needs of our industry partner and the characteristics of their development process. Furthermore, domain-specific and scalable solutions for requirement specifications, variability specification and management, and trace links establishment and management are all needed, as integrated parts of the overall framework, to provide the infrastructure required for supporting automation. These might also be the case for other organizations producing cyber-physical systems.

In the next section, we explain requirement-based test generation in the context of product-line engineering. Section III describes our industrial context and its related needs. In Section IV, we analyze the existing solutions and summarize the open questions. Finally, Section V concludes the paper.

II. REQUIREMENT-BASED TEST GENERATION

Requirement-based test generation [4][10][11] is an approach for generating test artifacts (e.g., test cases and test scenarios) from requirement specifications. An immediate benefit of this approach is the establishment of trace links between individual requirements and test artifacts. These trace links can be used for verifying a system against its requirements. Requirement-based test generation can be applied in the context of a single system (e.g., [11]) as well

as product families (e.g., [4][5][6]). In this paper, we limit the scope to requirement-based test generation in the context of product families. In the following, Section II.A briefly introduces product-line engineering, and Section II.B presents a generic framework, which is used in the rest of the paper for discussing and comparing the related work, identifying the gap, and proposing improvement directions that can address the industry's expectations and challenges.

A. Software product-line engineering

Software product-line engineering [1][2] is a paradigm for developing families of software applications instead of focusing on a single software system. A *product family* is a collection of similar software systems that share some common functionality, but vary in some aspects or features. To take advantage of the common functionality, reusable artifacts (e.g., requirements, design, tests) are developed, which can be customized and reused by different members of the family. Each reusable artifact introduces a number of *variation points* in the product family.

Commonly, a software product-line engineering framework distinguishes two phases: the *domain engineering* and the *application engineering* phases [12]. Domain engineering focuses on a product family as a whole. During this phase, commonalities and variabilities are defined and the reusable artifacts are developed. Application *engineering*, on the other hand, focuses on the production of a particular product from the reusable product-line artifacts. Production, in this context, involves selecting and customizing (e.g., via assigning values to variation points) the reusable artifacts according to specific needs of a particular product. We refer to this as the *Configuration* process.

B. Requirement-based test generation for product families

Figure 1 presents the generic framework for requirement-based test generation in the context of product families, which summarizes our understanding of the literature. This framework has incorporated the idea of reuse and provides an infrastructure for minimizing the overall effort required for supplying test artifacts during the application-engineering phase. The upper part of Fig. 1 depicts product-line artifacts that are involved in the test generation process, while the lower part contains product-specific artifacts that can be generated from the product-line artifacts. In the remainder of this section, we explain each of the involved artifacts, and then explain the test generation process and discuss how the idea of reuse is incorporated in this framework.

1) *Product-specific artifacts*

Product requirements describe functionalities, constraints and features specific to a product. *Product test* artifacts test a product against its requirements.

2) *Product-line artifacts*

A *product-line requirement* specifies a common (i.e., mandatory) or variable (i.e., optional) functionality or feature of the product family members. As a reusable artifact, a product-line requirement may introduce variation points, which should be configured in the application-engineering phase to obtain product requirements.

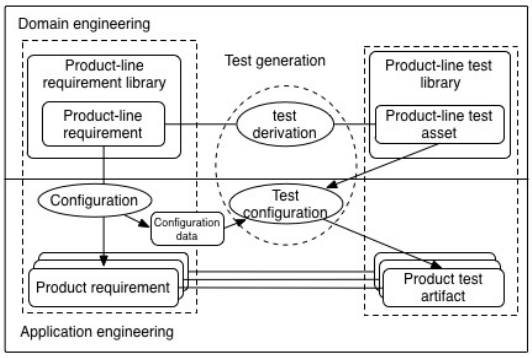

Fig. 1. Requirement-based test generation in product-line engineering

Product-line requirement library is an organized collection of product-line requirements to refer or reuse. Such a library distinguishes between common and variable requirements and specifies the relationships (e.g., imply, exclude) among the requirements.

Product-line test assets are reusable assets (e.g., test patterns [5], intermediate test models [4], or even test scenarios) that are used as a basis for generating product test artifacts. Product-line test assets and product-line requirements can either be combined (e.g., test assets can be attached to requirement specifications in the form of additional test information) or captured separately. A product-line test asset may introduce variation points. In the case of separated product-line requirements and test assets, these variation points should be traced back to the variation points introduced by the related requirements.

Product-line test library is an organized collection of product-line test assets to refer or reuse. The library may distinguish between common and variable tests, or rely on the commonalities and variabilities captured in the product-line requirement library and trace links between tests and requirements.

3) *Test generation*

Test generation in the context of product-line engineering consists of two activities: (1) *test derivation*, dealing with the derivation of product-line test assets from product-line requirements and establishing trace links between them, and (2) *test configuration*, which deals with the generation of product test artifacts from product-line test assets based on the configuration data provided during the application requirements engineering [1] process. Test derivation is typically done manually, while in most approaches (e.g., [4][5]) test configuration is performed automatically.

III. Our Industry Case Study and the Needs

FMC Technologies Inc. is a leading company in the maritime and energy industry. One of the company's key technologies is subsea production systems, which are cyber-physical systems intended for managing the exploitation of

[1] Application requirement engineering is the first step of application engineering during which product-line requirements are configured to produce product requirements [2].

oil and gas production fields. These systems consist of hundreds of mechanical, hydraulic, electrical and software components that are typically geographically distributed and connected through network.

Similar to other practices of developing cyber-physical systems in other domains (e.g., Medical systems developed by Siemens [4], Video Conferencing Systems produced by Cisco [15]), FMC has a product-line for developing its products and applies requirement-based test generation to verify and validate them. In such industrial settings, test generation often consumes large amount of resources, therefore leading to high cost (monetary wise). To reduce the overall cost and effort of test generation, automated solutions for improving the reuse of test assets should be devised and employed. In collaboration with our industrial partner, we strived to identify industrial expectations from such automated solutions.

In the rest of this section, we summarize the current practice of test generation in the context of cyber-physical systems, and report the identified industrial expectations from the prospective automated solutions for reuse. We have organized these expectations as a set of needs associated to the main components of the generic framework (Fig. 1).

A. Requirements specification

In the domain of cyber-physical systems, requirements (at both levels of domain and application engineering) are commonly specified using natural language. The use of natural language is particularly important in practice, because multiple stakeholders (e.g., customers, testers, and test engineers) are often involved in specifying and reviewing requirements [12]. Consequently, a significant need at our industrial partner is to provide requirement specifications based on natural language that can be equally expressive to all the stakeholders. Furthermore, to ensure the quality of derived test artifacts and to enable the verifiability of final products, all the involved stakeholders should have a consistent interpretation of the requirement specifications.

To be usable in the context of an automated reuse solution, these requirement specifications should further (1) be well-structured and unambiguous, and (2) systematically capture commonalities and variabilities. Taking into account all the aforementioned factors, the following needs can be derived:

Need#1: A method for specifying well-structured, unambiguous and expressive textual requirement

In the current practice of requirement engineering, document templates are often used to structure requirement specifications [13]. In addition, formal specifications (e.g., UML [14]) are occasionally used to compensate for the inherent ambiguity of textual specifications. A systematic domain-specific approach for guiding and ensuring consistent and systematic use of these techniques is however missing.

Need#2: A systematic approach for capturing commonalities and variabilities of requirements

To provide a systematic reuse infrastructure, commonalities and variabilities should be explicitly captured and documented in all types of product-line artifacts [12], including requirements. Therefore, an applicable approach for systematically capturing variabilities and commonalities in textual requirements specifications is expected.

B. Test assets specification

In the current practice of the company, test artifacts and test assets are both documented as test scenarios containing test steps specified in natural language, and expected results specified in a tabluar format. The test steps are executed manually as some steps require manually setting up a testing environment such as hardware simulators. To enable automated reuse of such test scenarios, we first need to explicitly document the set of reusable test assets and their variation points. Therefore, we elicit the following need:

Need#3: A systematic approach to capture commonalities and variabilities in test assets

Similar to Need#2, an applicable approach for systematically capturing commonalities and variabilities in textual test specifications is required.

C. Test generation approach

According to the generic framework discussed in Section II, the reuse of test assets involves the configuration of test assets and ensuring the conformance between the configured test assets and the configured requirements. These activities are currently performed manually at our industry partner, as they require excessive domain knowledge and expertise. The practice is similar to what one can often observe from other industrial settings in the domain of cyber-physical systems such as Cisco [15]. To perform the afformentioned activities, configuration experts should know requirements and their related test assets very well. Furthermore, they must know how to fill in the semantic gap between the test assets and the product-line requirements. To automate the reuse of test assets, the domain knowledge, which in this case is tacit, must be systematically captured and documented. This is addressed in the following two needs:

Need#4: Establishment of trace links during the test derivation process

Suitable trace links should be established between product-line requirements and their related test assets as well as their variation points. The industry is particularly interested in practical methods that enable the establishment of the trace links during the test derivation activity.

Need#5: Capturing the semantic gap between the configuration of requirements and tests

To systematically capture and document the semantic gap between the test assets and the product-line requirements, a solution including a structured, natural language-based specification method with application guidelines is needed to explicitly capture the required domain knowledge.

D. Integrated computer-based solution

To provide an efficient automated solution, which is applicable in practice, an integrated computer-based solution should be devised. Therefore, the following need is raised:

Need#6: An integrated computer-based solution for reusing test artifacts

This computer-based solution should be consistent with the company's practice of requirement-based test generation, specification methods for the requirements and the test artifacts, and its method for capturing the semantic gap.

IV. Analysis of existing approaches

The industrial needs and expectations from a solution for the automated reuse of test assets (Section III) can be categorized into four groups: (1) A systematic requirement specification method, (2) A systematic test specification method, (3) A solution for capturing the semantic gap between test assets and product-line requirements and establishing trace links, and (4) An integrated computer-based solution for reusing test artifacts.

In this section, for each of the above categories, we analyze the relevant solutions existing in the literature. We discuss their strengths and weaknesses in the context of the needs reported in Section III.

A. Systematic requirement specification method

As discussed in Section III, we identified two needs regarding the specification of requirements: a method for specifying well-structured, unambiguous and expressive textual requirements (Need#1), and a systematic approach for capturing commonalities and variabilities of requirements (Need#2) are needed.

A wide range of requirement specification languages exists in the literature. These include formal specification languages (e.g., Z [16]), graphical notations (e.g., UML and i* [18]), and textual specifications. The latter are the most widely adopted requirement specification methods in practice. Due to their high expressiveness and communicability, textual specifications are an appropriate choice in our industrial case, as multiple stakeholders from different organizations are involved in the requirements engineering process. In the following, we evaluate the existing textual requirement specification methods according to the two relevant needs: Need#1 and Need#2.

Natural language-based methods, although highly expressive and easy to use and train, in comparison with other types of specification methods, often introduce ambiguities. In particular, it is very likely for the involved stakeholders to have different interpretations of the same set of requirement specifications. Existing approaches reduce the inherent ambiguity of natural language by either limiting the vocabulary used in requirements specifications [19], or constraining the grammar, allowing only simple sentence structures to be used when requirements are documented (e.g., [20]). Another approach for reducing the ambiguity in textual requirements is the one provided by SysML [21]. It enables the specification of text-based requirements and modeling their relationship with other requirements, design elements, and test artifacts to support traceability.

Existing approaches for capturing the variabilities of textually specified requirements can be categorized into two groups. The first groups are the ones that use textual tags to directly document variation points in product-line requirement specifications (e.g., [5][6]). The second groups are the ones that use an intermediate model (e.g., activity diagrams as in [4]) to capture the commonalities and variabilities of product-line requirements. The first group of approaches can be employed with little overhead in terms of training: stakeholders only need to learn tag names. However, such tags can easily clutter requirement specifications and increase their complexity, hence impairing the readability of the whole document. This drawback can be addressed to a certain extent by better structuring the requirement specifications [22]. The second group of approaches ensures the simplicity of the requirement specifications by capturing variabilities separately in intermediate models. However, using these approaches imposes extra overhead of learning modeling notations. Maintaining intermediate models and their trace to the requirements is another issue, which can be defeated using approapriate tool support [4].

Concluding remarks

Considering all the factors discussed above, the most appropriate choice for specifying requirements at the company is to use a constrained and structured natural language. Such an approach should reduce the ambiguities inherent in natural languages, and ease the specification and management of variabilities.

Among the existing requirement-specification approaches, SysML seems to be the most suitable option, as it allows text-based requirement specifications. Furthermore, SysML offers built-in extension mechanisms (e.g., UML profile) that can provide the required foundation (1) for defining additional structures and constraints on the textual specifications, and (2) for modeling variabilities.

In the process of defining structures and constraints on textual specifications, one should note that constraining natural language hampers its expressiveness [22]. Therefore, the first step in this process is to find and characterize the right balance between expressiveness and ambiguity of the specification method.

Using SysML, and by extending it, one can define textual tags to model variabilities as an integrated part of the requirement specifications. To bound complexity and to avoid cluttering of the specifications, one can use inherent SysML model elements to structure complex requirements into smaller comprehensible chunks. In addition to textual tags, using SysML, one can define mechanisms for capturing variability on the model elements representing requirements and their relationships.

B. Systematic test specification method

As previously discussed in Section III, for enabling automated reuse of test scenarios, their commonalities and variabilities should be explicitly captured (i.e., Need#3) in the test assets. Similar to the case of requirement specifications, variabilities in test assets can be captured either jointly as part of their specifications or separately as intermediate models.

In [6], the authors proposed an approach that directly injects variation points into test specifications. In their approach, test asset specifications are configured manually according to the configuration of their corresponding requirements. To enable automated test configuration, trace

links are established between variation points in the requirements and the test assets. These trace links are established manually during the test derivation process.

Variability specification using intermediate models has been applied in [4], where UML activity diagrams are used as intermediate models to capture variabilities in test assets. These activity diagrams serve as blueprints that help with test configuration, i.e., test assets are configured so that they conform to the configuration of their corresponding requirements. A test engineer has to manually configure the activity diagrams according to the configured product requirements. From the configured activity diagrams, test artifacts can be reused automatically.

Concluding remarks

Considering the above discussion, in both cases, automated test configuration requires the establishment and maintenance of trace links between variabilities in both requirements and test assets. However, in the case of having an intermediate model, trace links should pass through the intermediate model, which adds to complexity and makes the maintenance of the trace links difficult. Hence, for our industrial case, capturing variation points directly in test specifications appears to be a better choice. In particular, similar to [6], variabilities can be captured using textual tags added to textual specifications of test scenarios. The remaining question is how scalable this approach is. Scalability is a major concern in the case of large-scale cyber physical systems, where large repositories of test scenarios often exist. In such cases, documenting all variation points together with test specifications may hamper scalability and so the practicality of the approach.

C. *Capturing the semantic gap and trace links between requirements and tests*

Two fundamental needs for automating the reuse and configuration of test assets are the establishment of trace links (Need#4) and capturing the semantic gap between tests assets and product-line requirements (Need#5). Trace links are needed to relate requirements to tests, as well as making the relation between the variabilities situated in the requirements and the ones situated in the tests. The semantic gap is about the information needed to derive the test configuration from requirement configuration, ensuring the conformance of product requirements and test artifacts.

There are various approaches for establishing trace links between requirements and downstream artifacts including test assets, such as trace matrix [23] and traceability metamodels [24]. However, none of these approaches considers the establishment of trace links among variabilities of product line requirements and test assets. Among the product-line requirement-based test generation approaches that we discussed in this paper (i.e., [4][5][6]), only the approach proposed in [4] establishes trace links between variabilities. However, this approach is not applicable in our context, because it uses intermediate test models, which, as discussed previously (in Section IV.A), are not an applicable option in our case. As discussed in Section III, a systematic methodology is needed to explicitly capture and document domain knowledge that is required in their current practice to bridge the semantic gap between the configuration of requirements and that of test assets. Such a methodology is rarely reported in the literature and we could not find any approach applicable to our industrial setting, i.e., cyber-physical systems.

Concluding remarks

Considering the above discussion, the main challenge in addressing Need#4 is in the establishment and maintenance of trace links between the variation points. This is particularly challenging because as mentioned in Sections IV.A and IV.B, in our industrial case, we need the variation points to be captured as an integrated part of the textual specifications, both for requirements and for test assets.

Systematically capturing the semantic gap between product-line requirements and test assets remains an open issue to be addressed in future research.

D. *An integrated computer-based solution for reusing test artifacts*

As mentioned in Section III, we aim at automating the reuse of test assets with the purpose of reducing cost, effort, and errors in the derivation of product test artifacts. To this end, we need to investigate the applicability of existing requirements engineering tools as an integrated solution (Need#6). Below, we discuss how existing tools address the needs of our industrial partner.

1) *Tool support for systematic requirements (and tests) specifications*

As discussed in Sections IV.A and IV.B, constrained and structured natural languages are appropriate options for specifying requirements and tests. Most of existing requirement engineering tools (e.g., PTC Integrity [25] and DOORS [26]) allow the requirements and their downstream artifacts to be specified in natural language. However, none of them assures the conformance of requirements and test specifications with respect to imposed constraints and structures. Furthermore, most of these tools don't support the variability management of requirements and tests as a built-in functionality. However, there are a few requirements engineering tools (e.g., PTC Integrity [25]) that distinguish between common and variable requirements. For example, PTC Integrity allows users to define textual tags for capturing requirements varaibilities. However, no solutions are provided for capturing variabilities in tests.

2) *Tool support for capturing the semantic gap between requirements and tests*

Automated generation of product test artifacts from product-line assets requires (1) trace links between product-line requirements and test assets to be captured (Need#4), and (2) transformation rules for deriving test configurations from requirement configurations to be fed as input to a requirements engineering tool (i.e., Need#5). All existing tools support the former to various extents, while the latter is missing from all of them. However, in the existing requirements engineering tools that can manage variabilities of a product family, such as PTC Integrity and pure::variants [27] (in the combination with DOORS), trace

links can be used to support automated selection (reuse without customization) of test artifacts.

Concluding remarks

In summary, three capabilities are missing from the existing tools: (1) Capturing textually specified requirement and tests variabilities, (2) Ensuring the conformance of requirements and test specifications with the defined constraints and structure, and (3) Ensuring the conformance of test configurations to requirements configurations.

V. CONCLUSION AND FUTURE WORK

According to a previously conducted domain analysis, we identified and reported the needs of an industry partner for improving its practice of requirement-based test generation. Subsequently, with analyzing the literature and tools, we revealed that there is a gap between the expectations of such industrial settings and existing approaches and tools. In relation to this gap, several open questions are formulated, which are (Question#1) capturing and documenting variation points in requirements and tests, (Question#2) assuring the conformance of requirement and test specifications with constraints and structures imposed on natural language, and (Question#3) affirming the conformance of test configurations with requirements configurations.

In the future, we will focus on answering these open questions, considering the characteristics specific to our domain, i.e., large-scale industrial cyber-physical systems, and different types of variation points (reported in [28]) with following assumptions and hypothesis for each open question:

- (Question#1) As mentioned in Sections IV.A and IV.B, our candidate solution for addressing this question is to capture and document variabilities in the specification of requirements and test assets. However, the large-scale nature of our industrial setting challenges the scalability and practicality of existing approaches (e.g., textual tags). In the future, we will extend SysML with a variability modeling mechanism to address this challenge. We expect this approach to be more scalable than approaches such as feature model in our case.
- (Question#2) As the first step in addressing the second open question, we need to define domain-specific rules to constrain and structure textual requirements and tests specifications. Automated tool support is then needed to ensure the conformance of requirement or test specifications to the defined rules. Such a tool support should be integrated with the requirement-engineering tool used at our industrial partner.
- (Question#3) The challenge in addressing the third open question is twofold. First, we need to devise a systematic approach to consisely capture the tacit knowledge related to the semantic gap between test assets and product-line requirements. Then, we have to develop tools that, using the captured tacit knowledge, can automatically check the conformance between requirements configurations and test configurations.

REFERENCES

[1] H. Gomaa, Designing Software Product Lines with UML: From Use Cases to Pattern- Based Software Architectures. Addison Wesley Longman Publishing Co., Inc., 2004.

[2] K. Pohl, G. Böckle, and F. J. Linden. Software Product Line Engineering: Foundations, Principles and Techniques, Springer-Verlag New York, Inc., 2005.

[3] D. L. Parnas, "On the Design and Development of Program Families", TSE, vol. 2(1), 1976, pp. 1-9.

[4] A. Reuys, E. Kamsties, K. Pohl, and S. Reis, "Model-based system testing of software product families", CAiSE, 2005, pp.19– 534.

[5] C. Nebut, Y. Le Traon, and J.M. Jezequel, "System Testing of Product Lines : From Requirements to Test Cases", Software Product Lines Springer Verlag (Ed.), 2006, pp. 447-478.

[6] A. Bertolino, S. Gnesi, "PLUTO: A Test Mehtodology for Product Families", Software Product-Family Engineering, 2004, pp. 181-197.

[7] FMC Technologies: http://www.fmctechnologies.com/

[8] Cyber-physical Systems: http://cyberphysicalsystems.org/

[9] T. Yue, S. Hesari, S. Ali, and B. Selic, "A Study on Industrial Requirements Engineering Practice of Highly Configurable Systems" Technical Report 2012-21, 2013.

[10] J. Gutierrez, C. Nebut, M.J. Escalona, M. Mejias, J. Torres, and A. H. Centeno, "A case study for generating test cases from use cases", RCIS, 2008, pp. 209-214.

[11] C. Nebut, F. Fleurey, Y. Le Traon, and J.M. Jezequel, "Automatic test generation: a use case driven approach", TSE, vol. 32(3), 2006, pp. 140-155.

[12] K. Pohl, Requirements Engineering: Fundamentals, Principles, and Techniques, Springer Publishing Company, Inc., 2010.

[13] T. Yue, L. Briand, and Y. Labiche, "A Systematic Review of Transformation Approaches between User Requirements and Analysis Models", Requirements Eng (Springer), vol. 16(2), 2011, pp.75-99.

[14] I. Jacobson, G. Booch, and J. Rumbaugh. *The Unified Modeling Language*, University Video Communications, 1996.

[15] S. Wang, S. Ali, and A. Gotlieb, "Minimizing Test Suites in Software Product Lines Using Weight-based Genetic Algorithms", GECCO, 2013.

[16] J. Michael Spivey, The Z notation, Prentice Hall, 1992.

[17] P. Stocks, and D. Carrington. "A framework for specification-based testing", TSE, vol. 22(11), 1996, pp. 777-793.

[18] E. Yu, "Towards Modelling and Reasoning Support for Early-Phase Requirements Engineering", Proc. 3rd IEEE Int. Symp. on Requirements Engineering, 1997, pp. 226–235.

[19] IEEE Computer Society, "IEEE recommended practice for software requirements specifications", IEEE, 1998.

[20] ISO 13628-6, "Petroleum and natural gas industries-Design and operation of subsea production systems-Part 6: Subsea production control systems", 2008.

[21] S. Friedenthal, A. Moore, and R. Steiner, A practical guide to SysML, the systems modeling language, Morgen Kufman, 2009.

[22] T. Yue, L. Briand, and Y. Labiche, "Facilitating the Transition from Use Case Models to Analysis Models: Approach and Experiments", TOSEM, vol. 22(1), 2013.

[23] B. Ramesh, "Factors influencing requirements traceability practice", Communications of the ACM, vol. 41(12) , 1998, pp. 37-44.

[24] R. Balasubramaniam, and M. Jarke, "Toward reference models for requirements traceability", TSE, vol. 27(1), 2001, pp. 58-93.

[25] http://www.mks.com/solutions/discipline/rm/requirements-management

[26] http://www-142.ibm.com/software/products/us/en/ratidoor/

[27] http://www.pure-systems.com/pure_variants.49.0.html

[28] R. Behjati, T. Yue, L. Briand, and B. Selic, "SimPL: A product-line modeling methodology for families of integrated control systems", IST, vol. 55(3), 2013, pp. 607–629.