Alignment of requirements specification and testing: A systematic mapping study

Zeinab Alizadeh Barmi
barmi@student.chalmers.se

Amir Hossein Ebrahimi
amirho@student.chalmers.se

Robert Feldt
robert.feldt@chalmers.se

Department of Computer Science and Engineering
Chalmers University
Gothenburg, Sweden

Abstract

Requirements should specify expectations on a software system and testing should ensure these expectations are met. Thus, to enable high product quality and efficient development it is crucial that requirements and testing activities and information are aligned. A lot of research has been done in the respective fields of Requirements Engineering and Testing but there is a lack of summaries of the current state of the art on how to link the two. This study presents a systematic mapping of the alignment of specification and testing of functional or non-functional requirements in order to identify useful approaches and needs for future research. In particular we focus on results relevant for non-functional requirements but since only a few studies was found on alignment in total we also cover the ones on functional requirements. We summarize the 35 relevant papers found and discuss them within six major sub categories with model-based testing and traceability being the ones with most prior results.

1. Introduction

With the ever-growing market demand for high quality software, the need and importance of testing has become more apparent in recent years. For more safety critical software systems - like medical diagnosis and space shuttle missions - software testing is a crucial aspect of their success since software errors can cause irreparable losses. On the other hand, Requirements Engineering (RE) represents a complementary view of a system and thus has a synergistic relationship with testing [1]. Therefore bringing RE and testing closer could benefit both disciplines. Making a strong link between them will improve the outcome of the software development process [2]. It also helps to discover possible errors early, which in turn can improve the product quality and lead to more satisfied customers [3]. From the project management perspective, linking requirements and testing would help to reach a more accurate testing plan, which in turn would improve project cost and schedule estimation. The likely result for the project is to be finished within the planned schedule and budget [1]. Although organizations are becoming more interested in linking requirements and testing, often this link is not provided and there is a gap between the areas. It is noticeable that in these efforts, the focus has been mainly on functional requirements (FRs) rather than on non-functional, quality requirements (NFRs). NFRs play a significant role in the success of software projects. Grunske [4] states that NFR's fulfillment is often more important than implementing FRs to have a satisfied customer. Matoussi and Laleau [5] point out that verification of NFRs are almost always done very late after finishing the implementation. Our aim in this research is to perform a systematic mapping on the alignment of functional or non-functional requirement specification and testing to get an overview of existing research in this area.

Among the work that has been done on alignment a lot of attention has been given to traceability. Traceability of requirements can help determine what requirement has been covered by which test and how the generated test cases cover these requirements [6]. Tracing from tests back to requirements is also helpful to find the root of a failed test. Another important reason for traceability is improving change management by helping to find out how change in the requirement is reflected in the test cases. Also in the alignment research area model based development has attracted a lot of attention. The idea behind MBT is the derivation of executable test code from test models by analogy to Model Driven Architecture (MDA) [6]. This technique is becoming of more interest in industry because it

provides automatic deriving of test cases from the behavioral model of the system called the test model.

As Petersen et al. [7] describe, a systematic mapping study consists of an overview of primary studies performed on a specific topic and the categorization of the results by providing a visual summary. As such, a systematic map offers an overview of a field, identifying potential gaps in research, whereas a systematic literature review [8] provides a more detailed study of the identified results. The systematic mapping process consists of five phases [7]: 1) Definition of the research questions, 2) Conducting the search for primary studies, 3) Screening of papers for inclusion or exclusion, 4) Keywording the abstracts, and 5) Data extraction and mapping of the studies.

This paper is organized as follows: Section 2 describes the phases of our systematic mapping process. Some of these phases are broken down into smaller steps. In section 3 the answer to our research questions is provided. Discussion and conclusions are provided in sections 4 and 5, respectively.

2. Research Method

2.1. Definition of research questions

To investigate existing research on the alignment of functional or non-functional requirement specification and testing, we formulated the following research questions:

RQ1. Which studies have been done on linking the specification and testing of requirements?

Aim: We want to find out which topics have been investigated and to what extent? Which of these studies are focused on NFRs? What are the different perspectives that address the alignment of requirements and testing, e.g., how common testing approaches try to address alignment or traceability? This would help identify the needs for complementary research.

RQ2. What types of solutions are represented in these researches? We want to find the solutions given in different topics that address alignment such as method, process, framework, tool, etc.

RQ3. In which fora is research on alignment of requirement and testing published? An overview of the range of fora in which the researches are published also could help with our research.

2.2. Conducting the search for primary studies

In order to conduct our search string we needed to obtain an overview of the requirements specification and testing area and the alignment of these two. So we gathered an initial set of previously known publications in the area, through an exploratory search [1, 3-5, 9-14]. We then tried to extend our initial set using forward/backward referencing, i.e. looking at which papers were referenced in or referred to papers in our initial set. The study of the resulting 24 papers helped us to explore and choose relevant keywords for the systematic search. From our research questions and based on our study of the initial set of papers we derived some categories for conducting the search string. These categories focused on NFRs (named C1), specification of NFRs (named C2), testing of NFRs (named C3), and linking the specification and testing of NFRs (named C4). We formulated a combination of these categories to reach our search string, which was "C1 AND C2 AND C3 AND C4". As we decided to cover all researches in the last 10 years, we limited the search on papers that were written in English and published between 2001 and 2010 (it should be noted that since the search was done in November of 2010 not all 2010 papers will be included in the results). We conducted the search on 4 databases (Scopus, Inspec Engineering Village, ISI Web of Knowledge, and IEEE Xplore). We then followed an incremental refinement of the search string in five steps. In each refinement step we checked the results to see if they contained papers from our initial set that were on the alignment area. In case the items of the initial set were missed or the results were not relevant we improved our search string further. One major refinement was removing the C2 (specification) category from the search string. The reason was that many papers in our initial set did not include the terms belonging to the C2 (specification) category in their title or abstract. As there was little research with focus on NFRs we decided to get some ideas from the alignment of requirements and testing in general. So we added another category named C5 with the "requirement" item and we changed our search string to "(C1 OR C5) AND C3 AND C4". The final version of categories is shown in Table 1. After the final iteration of the search string refinement we reached 591 hits.

Table 1. Search string categories

NFR (C1)
"nonfunctional requirement" OR "nonfunctional requirements" OR "non functional requirement" OR "non functional requirements" OR "non functional software requirement" OR "non functional software requirements" OR "nonbehavioral requirement" OR "nonbehavioral requirements" OR "nonbehavioural requirement" OR "nonbehavioural requirements" OR "non behavioral requirement" OR "non behavioral requirements" OR "non behavioural requirement" OR "non behavioural requirements" OR "nonfunctional property" OR "nonfunctional properties" OR "non functional property" OR "non functional properties" OR "quality attribute" OR "quality attributes" OR "quality requirement" OR "quality requirements" OR "quality attribute requirement" OR

"quality attribute requirements"
Testing (C3)
"test" OR "testing" OR "verify" OR "verifying" OR "verification" OR "validate" OR "validating" OR "validation"
Alignment (C4)
"align" OR "aligning" OR "alignment" OR "trace" OR "tracing" OR "traceable" OR "traceability" OR "link" OR "linking" OR "bridge"
Requirement (C5)
Requirement

2.3. Screening papers for inclusion or exclusion and keywording the abstracts

In this phase we defined inclusion and exclusion criteria in order to achieve a common understanding between the team members that would perform the screening. The paper screening process was performed in two steps.

In the first step the title and abstract (if needed) of all papers were considered. The main criteria for inclusion were papers that describe alignment of specification and testing of functional or non-functional requirements. Conference proceedings and papers that were not focused on software development, for example papers that focused on hardware or network development were excluded. In this step if a researcher was unsure about excluding a paper, this paper was included for the second step. 546 papers were excluded during this step. In the second step the full text of the papers were studied. Posters, opinion papers i.e. papers that express the personal opinion of author on what is good or bad [15], and short papers (with less than 6 pages) were excluded. Papers were only included if they had been subject to peer review. If researchers did not agree on the inclusion or exclusion of some papers these papers were discussed until a decision was made by consensus. 10 papers were excluded during this step. At the end the number of hits was reduced to 35papers.

During this phase we developed our data extraction form, partly based on the one used by Ivarsson and Gorschek[16]. To build this form some keywords and concepts, like context and domain were reached through the study of paper abstracts by each researcher. The keywords were evolved as papers were studied in detail. Using the keywords, we finally came up with the following key attributes in the form *research focus* (Model-centric approaches, code-centric approaches/ traceability/ formal approaches/ test cases/ problems and set of good practices), *contribution type* (Tool/ process/ model, framework/ guideline/ method/ metric and other), *quality requirements/attributes the paper focus on*, *research method*, *context* (academia/industry/ open-source software), *domain*, and *scale*. It should be noted that some research focus items contain some subcategories. Model centric approaches include 2 subcategories of Model based testing (MBT) and Goal-oriented development. Code centric approaches category is divided into 3 subcategories of Test driven development (TDD), Storytest driven development and Behavior driven development (BDD). Test cases category is also broken down to 2 subcategories of Test case generation (manual/automatic) and Test case coverage. During the study of the full text of included papers this extraction form was filled for each individual paper.

3. Results (data extraction and mapping)

As the full text of the papers was being studied the data extraction form for each paper was also filled. We answer our research questions by analyzing the extracted data from the papers.

3.1. RQ1. Which studies have been done on linking the specification and testing of requirements?

We have identified several research focuses on the alignment of requirements specification and testing. In the last part of this section we have mentioned which of these approaches support NFRs. The distribution of research focus is shown in Figure 1.The distribution of research focus over different years is shown in Table 2.

3.1.1. Model Based Testing (MBT)

In Model based testing (MBT), which has the largest number of hits, informal requirements of the system are the base for developing a test model which is a behavioral model of the system. This test model is used to automatically generate test cases. One problem in this area is that the generated tests from the model cannot be executed directly against an implementation under test (IUT) because they are at the same level of abstraction as the model. Arnold et al. address this problem [17] and propose a solution. They also claim that their scenario-driven approach supports the traceability between generated and executed test cases, and executions of an IUT. Their approach supports both FRs and NFRs.

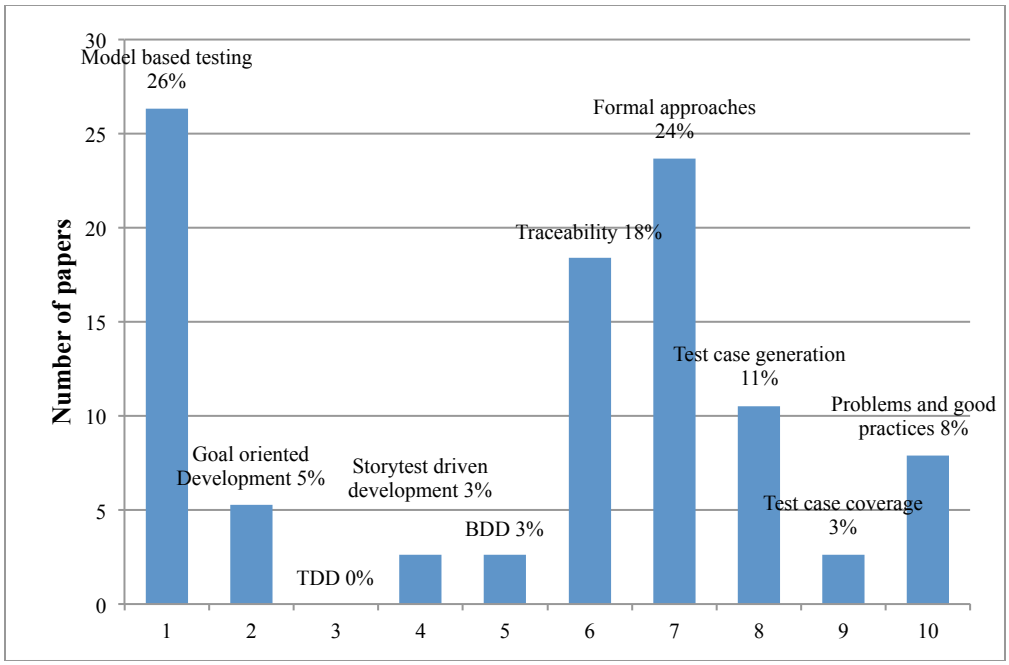

Figure 1. Distribution of research focus

There are some other prior research that are scenario based. Goel et al. [18] propose a model-driven approach in which strengths of both scenarios-based and state-based modeling styles are combined. Their Footprinter tool makes it possible to trace from requirements to testing and vice versa in a round-trip engineering approach. The Web Services Testing Framework (WSTF) is proposed by Tasi et al. for Scenario-based testing of Web Services [19]. In this approach test scripts and test cases are automatically generated based on a scenario specification by the test master component. Some prior research address development process in model based testing. Pfaller et al. propose [20] using different levels of abstraction in the development process to derive test cases and link them with the corresponding user requirements. Boulanger and Dao propose an approach [21] in which RE is done in different phases of the V-model in order to facilitate requirements validation and traceability. Lobo and Arther have worked on a project [22] which aims to reduce the gap between RE and the V&V process. This is done by applying V&V in a two-phase model: in the first phase, V&V is performed right after requirements elicitation for each distinct function of the system. In the second phase, the quality of linkages between requirement sets is verified and validated. Several studies are on model based testing of service oriented systems. Felderer et al. focus on model–driven testing of service-oriented systems in a test–driven manner [6]. They believe that Telling TestStories tool could support traceability between all kinds of modeling and system artifacts. Tasi et al. proposed framework - which was described as a scenario based approaches also aims for testing of web services [18]. There are also other MBT approaches. Marelly et al. extend sequence charts (LSCs) with symbolic instances and symbolic variables [23] in order to reach linking requirements and testing.

Zou and Pavlovski propose control cases to complement use cases by modeling NFRs which cannot be addressed by use cases [11]. They can be applied during the software development life cycle and can also be used to verify whether the implemented system meets the specified NFRs.

3.1.2. Goal oriented development

In the field of Model driven engineering (MDE), Goal-oriented modeling can easily realize stakeholder's concerns and their interdependencies using concepts which are less dependent on the underlying implementation technology and closer to the problem domain [24].

Goals, which can be at various levels of abstraction, define stakeholders' expectations from the system [24]. Hard goals are states that actors can attain and soft goals are goals that can never be fully satisfied. Both FRs and NFRs can be represented by goal oriented modeling. FRs are modeled by hard goals, and NFRs like efficiency and reliability are represented by soft goals which means they are expected to be satisfied within an acceptable limits rather than absolutely [24].

Table 2. Distribution of research focus over years

Research Focus	2002	2003	2004	2005	2006	2007	2008	2009	2010	Total	%
Model based testing				1	3	1	2	2	3	12	33
Goal- oriented development					1				2	3	8
Storytest driven development				1						1	3
BDD									1	1	3
Traceability				1	1		3	2	2	9	25
Formal Approaches		1							1	2	6
Test case generation				1	1		1	1		4	11
Test case coverage					1					1	3
Problems and good practices				2					1	3	8
Total	0	1	0	6	7	1	6	5	10	36	

This type of development improves productivity as well as model quality. Nan et al. propose a framework [24] for tracing aspects from requirement to implementation and testing. Language support is also provided to transform models into aspect oriented programs. Test cases can be derived from these models to help in the verification process [24].

3.1.2.1. Agent oriented Software Engineering (AOSE)

AOSE methodologies are based upon the Agent paradigm, and help to develop complex distributed systems [25]. AOSE partially addresses the link between requirements and testing. This is done by specification-based formal verification and object oriented testing techniques. Duy et al. introduce a testing framework for AOSE methodologies called TROPOS [25]. This framework provides a systematic way of deriving test cases from goal analysis. This approach is called goal oriented testing.

3.1.2.2. Aspect Oriented Software Development (AOSD)

Nan et al. present AOSD as a solution for transformation from the goal model to the implementation [24]. They describe how aspects are introduced from the goal models and introduce a framework with which aspects can maintain traceability from the requirement level down to implementation and test levels.

3.1.3. Traceability

Abbors et al. [26] present an approach for requirements traceability across a MBT process and the tools that are used for each phase. Some prior researches address requirement based testing to facilitate traceability between requirements and testing. Méndez et al. present a requirement-based testing process and define a guideline on how to keep the traceability among requirements down to the test cases in this process [27]. Quinn et al. propose the TraceFunction Method by which requirements of a software component can be specified in one easily used reference document in order to facilitate traceability between requirements and testing [28]. Some researches address traceability issues using scenarios. Naslavsky et al. believe that different kinds of scenarios are useful for tracing requirements to tests through the development life cycle and each can be used as test scenarios [29]. Test generating and traceability issues in three different scenario-based system modeling languages are studied by Goel and Roychoudhury [30]. Other researches construct meta-models to facilitate traceability. In order to establish the relationship between software components that include the requirements, design test cases and code, Ibrahim et al. construct a meta-model with top-down and bottom-up traceability support [31].Dubois et al. propose a meta-model called DARWIN4REQ which aims to keep the traceability link between three phases of requirement elicitation, design and V&V of requirements [32].

3.1.4. Formal approaches

Post et al. focus on translating requirements into scenario-based formal language which in turn could be linked to software verification [2]. Bouquet et al. use a

subset of UML 2.0 diagrams and Object Constraint Language (OCL) operators to formalize the expected system behavior [33]. The model is used for automatically generating executable test scripts. Kelleher and Simonss propose a new requirement modeling approach [34] in which use cases are replaced with use-case classes in UML 2.0. Use case classes are formal templates for describing rules on modeling requirements with instances. This replacement, together with utilizing explicit traceability links, facilitates bridging the gap between requirements and testing. Sabetta et al.discuss [35] that sometimes it might be needed to transform UML models into different analysis models which could each be used to verify (in a formal way) one kind of NFR. Some of these models are Petri nets, queuing networks, formal logic, etc. For this purpose, their abstraction-raising approach can transform UML models to different kind of analysis models in different formalisms. Hassan et al. focus on security requirements [36]. They propose the first goal-oriented software security engineering approach, Formal Analysis and Design for Engineering Security (FADES), aiming to produce software with high level of security in a systematic manner. FADES' support of automatic derivation of B formal method specifications and a suite of acceptance test cases from the requirements model ensures better alignment of security requirements and testing. Hussain and Eschbach present a model-based safety analysis approach [37] that automatically composes formal models of the system and produces a fault tree which can be used to generate test cases for the software system. Therefore test cases can be directly bound to the safety requirements and assure traceability between testing activity and safety requirements.

3.1.5. Code-centric approaches

Mugridge presents Storytest-Driven Development as a complementary form of TDD which can be applied to overall system development [38]. Storytests which are executable and business-oriented examples for each scheduled story are written by customers as an alternative to detailed requirements documents. As executable documents, they significantly reduce the need to derive independent tests because they help developers to continuously verify their consistency with the system. Baillon and Mongardé describe Behavior Driven Development (BDD) as a new development paradigm [39] in order to address traceability problems. They introduce the XReq tool which supports BDD in the Ada and other statically typed languages.

3.1.6. Problems and set of good practices in aligning requirements and testing

Uusitalo et al. present a set of good practices which can be applied to create a stronger link between requirements engineering and testing [3]. Some of these practices are involving testers during project planning and requirements reviews, which would lead to higher quality requirements and improved testability. A systematic approach is presented by Kukkanen et al. [1] for improving requirements and testing processes together with the aim of linking requirement and testing. They describe lessons learnt and best practices determined from applying new processes in an industrial case study. Sabaliauskaite et al. have carried out a survey in a large software company in Sweden, investigating the experienced obstacles in the alignment of requirements and testing [40].

3.1.7. Test cases

Nebut et al. concentrate on a guideline for automatic test case generation on embedded systems that are based on object oriented concepts [41]. The system requirements are described via use cases, contracts, and scenarios. If any other information for the requirements is needed, it is provided by different UML artifacts like sequence diagrams. Whalen et al. mention several problems of measuring the adequacy of black box testing using executable artifacts [42]. They also present coverage metrics based on formal high level software requirements. Conrad et al. presented a test case generation strategy which has been in use in an automotive company [43]. Siegl et al. are also interested in automotive industry proposed EXtended Automation Method (EXAM) for automatic generation of test cases, and the Timed Usage Model process for derivation of test cases from requirements [44]. Riebisch and Hubner concentrate on the first step of test case generation [45]. In this step their proposed method uses a description of the natural language and transforms it to an expression with formally defined syntax and semantics.

3.1.8. Approaches which support NFRs

Arnold et al. validation framework supports the modeling and automated validation of FRs and NFRs against several candidates IUTs [17]. In another research they have worked on a MBT approach, which is based on Requirements Notation (URN). URN is one of few approaches that address the modeling and validation of both FRs and NFRs [46]. The approach proposed by Felderer et al. is suitable for testing Service level agreements (SLA) which is considered as

non-functional properties [6]. In their case study performance and security are included in modeling requirements. Duy et al. proposed framework – TROPOS is goal oriented in which NFRs are specified by softgoals [25]. For discovering aspects from goal models in AOSD, goals should be elicited and categorized to hard (FRs) and soft (NFRs) goals, hence AOSD can support NFRs [24]. In the method proposed by Méndez et al. specifying test cases (TCs) based on use cases (UCs) enables traceability between tests and FRs and NFRs [27]. In this approach the TCS Table can be used to define TC procedures associated to NFRs. The meta-model presented by Dubois et al. allows a full traceability of both FRs and NFRs through software development process [32]. In their research, Sabetta et al. transform UML models to different kind of analysis models, such as Petri nets, queuing networks, formal logic, etc. Each of these models could be used in formal verification of different NFRs [35]. Hassan et al. focus on alignment of security requirements and testing through supporting of automatic derivation of B formal method specifications and a suite of acceptance test cases from the requirements model [36]. In another research, Mugridge states that both Storytest-driven development and TDD depend on advanced automated testing techniques, including tests for non-functional requirements [38].

3.2. RQ2. What types of solutions are represented in these studies?

Figure 2 shows a map of existing research focusing on the alignment of requirements specification and testing, distributed over type of contribution and research type. It should be noted that a publication might provide multiple contributions e.g. both a tool and method.

3.3. RQ3. In which fora is research on alignment of requirement and testing published?

Distribution of research shows that most research is published in conferences and workshops 29/35(82%). There is also one book chapter and five journal publications (a full table of the distribution of publication fora can be found on the url http://www.cse.chalmers.se/~feldt/publications/alizadeh_2011_revvert.html). Distribution of publication types over time is shown in Table 3.

Figure 2. Map of research focus

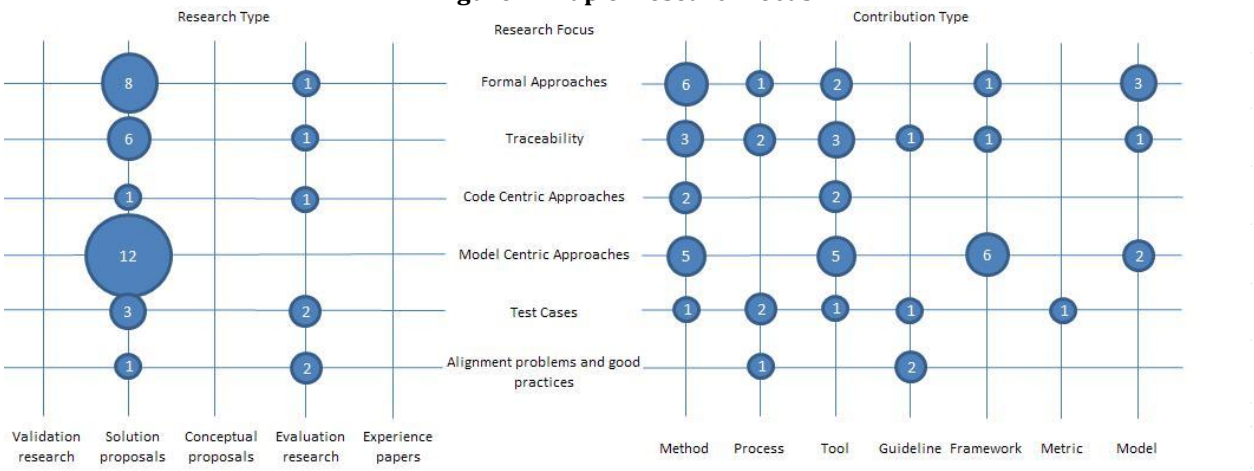

Table 3. Distribution of publication types over years

Publication Type	2002	2003	2004	2005	2006	2007	2008	2009	2010	Total
Conference	1	0	0	2	5	0	4	4	7	23
Workshop	0	0	0	3	1	1	1	0	0	6
Journal	0	1	0	0	1	0	1	0	2	5
Book	0	0	0	0	0	0	0	1	0	1
Total	1	1	0	5	7	1	6	5	9	35

4. Discussion

There are several challenges in using MBT approach for aligning requirements and testing. Researchers have tried to address these challenges. One challenge is to make test cases executable, as the tests are not at the same level of detail as the implementation code [17, 46]. Another challenge is to find interesting test cases. Test cases are said to be interesting if they cover requirements and can discover potential errors with a high probability [20]. Requirements traceability in MBT process is another important issue, which is the focus of several studies (such as [6, 20, 30]). In these the focus is mainly on FRs, and NFR traceability is still open for further research [26]. There are some studies on using MBT in service-oriented systems (like [6, 19]). The use of scenario notation for specification of system models has also attracted some attention (such as [17, 19, 30, 46]).

The focus on traceability issues is also conceivable since requirements traceability helps to determine the degree of test case coverage of requirements, and improves change management, which is crucially important to industry. As Abbors et al. mention [26] traceability reduces the time needed for debugging of the specification or the implementation of the system, by giving fast feedbacks.

Formal approaches which are another main research interest address translating informal requirements into formal models, generating tests from these formal models, and tracing between the informal requirements and tests [30]. Applying formal methods for aligning requirements and testing has some advantages and drawbacks. In this approach requirements are formulated in a precise, provable and correct representation. The representation is unambiguous and consistent [36]. This makes formal methods one of the best options for modeling and testing of safety critical systems. On the other hand using formal methods is difficult for practitioners [36]. Experienced people in this field are hard to come by and expensive to employ. The application of formal methods especially for large and complex systems is challenging because of their high cost and limited scalability. There is room for researches that combine the advantages of formal methods – formulating requirements in a precise, reliable and provable representation and the strength of informal methods – easy to learn and apply, to align requirement and testing.

Looking at figure 2 the dominant research type in all research focus areas is solution proposal. This means that challenges in each research focus area are well understood, but the proposed solutions are just proposals and very little researches focus on the actual use and evaluation of proposals. Table 4 shows that only half of the papers have evaluated their ideas in industrial case studies, and their validity discussion is mostly medium (that is the author has mentioned threats to validity without a detailed discussion). In addition most of the studies have not been published in journals. As mentioned in section 3.3, 29/35 (82%) of studies are published in conferences and workshops and only 5 out of 35 papers in a journal. All in all aligning the requirements and testing seems to be a relatively immature area and is in need of more empirical and practical work.

The contribution type is mostly method 17/55 (31%), tool 13/55 (24%), followed by framework. Presenting new methodologies or enhancing existing ones are needed to establish a strong link between requirements and testing. In order for them to be practical in industry, supportive tools and frameworks should be built.

Table 3 shows that interest in this field has grown in recent years, which could also serve as a motivation for more research.

Another important point is that most efforts in aligning requirements and testing have been on functional requirements. Table 5 shows that just 10 of 41 (24%) of papers present approaches that could also be used for NFRs. This is a low percentage considering how important NFRs are in today's software systems.

Table 4. Validity discussion and case study

	#	%
Weak	7	20
Medium	17	49
Strong	11	31
Total	35	
With case study	16	46

Table 5. Distribution over FRs and NFRs

	#	%
FR	31	76
NFR	10	24
Total	41	

5. Conclusions

This paper presents a systematic mapping on aligning the specification and testing of functional or non-functional requirements. We identified 35 papers published between 2001 and 2010 by searching in four major databases. After studying these papers we could

divide the prior work on aligning requirements and testing that they represent in these focus areas: Model-based approaches, Code-centric approaches, Traceability, Formal approaches, Test cases, Problems and set of good practices in aligning requirements and testing. The major focus is on model-based testing and traceability issues. The type of contribution of the papers is mostly methods (26%), tools (24%) followed by frameworks. Most of the prior research has been published in conferences and workshops (82%). There is also one book chapter and five journal publications. Although industry is becoming increasingly interested in establishing a strong link between requirements and testing, we conclude that there is still a significant gap between these areas. This shows high potential for future work in establishing methods and processes with supportive tools. In particular, the current approaches to alignment have paid little attention to non-functional, quality requirements even though they play a critical role in achieving successful software systems. As such, this area has high potential for further research.

References

[1] J. Kukkanen, K. Vakevainen, M. Kauppinen*, et al.*, "Applying a systematic approach to link requirements and testing: a case study," in *Asia-Pacific Software Engineering Conference (APSEC)*, Piscataway, NJ, USA, 2009, pp. 482-488.

[2] H. Post, C. Sinz, F. Merz*, et al.*, "Linking functional requirements and software verification," in *17thIEEE International Requirements Engineering Conference(RE)*, Piscataway, NJ, USA, 2009, pp. 295-302.

[3] E. J. Uusitalo, M. Komssi, M. Kauppinen*, et al.*, "Linking requirements and testing in practice," in *16th IEEE International Requirements Engineering Conference*, NJ, USA, 2008, pp. 265-70.

[4] Lars Grunske, "Specification Patterns for Probabilistic Quality Properties," in *30th International Conference on Software Engineering (ICSE 2008), Leipzig, Germany.*, 2008, pp. 31-40.

[5] Abderrahman Matoussi and Régine Laleau, "A Survey of Non-Functional Requirements in Software Development Process," *Technical report TR-LACL-2008-7, LACL (Laboratory of Algorithms, Complexity and Logic), University of Paris-Est (Paris 12),* 2008.

[6] M. Felderer, P. Zech, F. Fiedler*, et al.*, "A Tool-based Methodology for System Testing of Service-oriented Systems," in *Second International Conference on Advances in System Testing and Validation Lifecycle (VALID)*, Los Alamitos, CA, USA, 2010, pp. 108-13.

[7] K. Petersen, R. Feldt, S. Mujtaba*, et al.*, "Systematic mapping studies in software engineering," *12th International Conference on Evaluation and Assessment in Software Engineering (EASE) University of Bari, Italy, 26–27 June,* 2008.

[8] B.A. Kitchenham and S. Charters, "Guidelines for performing Systematic Literature Reviews in Software Engineering," *Technical Report EBSE-2007-01,* 2007.

[9] Lawrence Chung and Julio Cesar Prado Leite, "On Non-Functional Requirements in Software Engineering," in *Conceptual Modeling: Foundations and Applications*, T. B. Alexander, K. C. Vinay, G. Paolo*, et al.*, Eds., ed: Springer-Verlag, 2009, pp. 363-379.

[10] Marie-Agnes, Peraldi-Frati, and Arnaud Albinet, "Requirement traceability in safety critical systems," presented at the Proceedings of the 1st Workshop on Critical Automotive applications: Robustness and Safety, Valencia, Spain, 2010.

[11] J. Zou and C. J. Pavlovski, "Control cases during the software development life-cycle," in *IEEE Congress on Services Part 1 (SERVICES-1)*, Piscataway, NJ, USA, 2008, pp. 337-344.

[12] J. Metsa, M. Katara, and T. Mikkonen, "Testing Non-Functional Requirements with Aspects: An Industrial Case Study," in *Quality Software, 2007. QSIC '07. Seventh International Conference on*, 2007, pp. 5-14.

[13] Breno Lisi Romano, Glaucia Braga e Silva, Henrique Fernandes de Campos*, et al.*, "Software Testing for Web-Applications Non-Functional Requirements," presented at the Proceedings of the 2009 Sixth International Conference on Information Technology: New Generations, 2009.

[14] M. Glinz, "On Non-Functional Requirements," in *15th IEEE International Requirements Engineering Conference. RE '07.* , 2007, pp. 21-26.

[15] Roel Wieringa, Neil Maiden, Nancy Mead*, et al.*, "Requirements engineering paper classification and evaluation criteria: a proposal and a discussion," *Requir. Eng.,* vol. 11, pp. 102-107, 2005.

[16] M. Ivarsson and T. Gorschek, "Technology transfer decision support in requirements engineering research: a systematic review of REj," *Requirements Engineering, vol. 14, 2009, pp. 155-175.,* 2009.

[17] D. Arnold, J. P. Corriveau, and Shi Wei, "Modeling and validating requirements using executable contracts and scenarios," in *8th ACIS International Conference on Software Engineering Research, Management and Applications (SERA)*, CA, USA, 2010, pp. 311-20.

[18] A. Goel, B. Sengupta, and A. Roychoudhury, "Footprinter: Round-trip engineering via scenario and state based models," in *31st International Conference on Software Engineering - Companion Volume - ICSE-Companion*, Piscataway, NJ, USA, 2009, pp. 419-420.

[19] W. T. Tsai, R. Paul, L. Yu*, et al.*, "Scenario-Based Web Services Testing with Distributed Agents," *IEICE Transactions on Information and Systems,* vol. E86-D, pp. 2130-2144, 2003.

[20] C. Pfaller, A. Fleischmann, J. Hartmann*, et al.*, "On the integration of design and test: A model-based approach for embedded systems," in *Proceedings of the 2006 international workshop on Automation of software test(AST)* 2006, pp. 15-21.

[21] J. L. Boulanger and V. Q. Dao, "Requirements engineering in a model-based methodology for embedded automotive software," in *IEEE International Conference on Research, Innovation and Vision for the Future in Computing & Communication Technologies(RIVF)*, Ho Chi Minh City, Vietnam, 2008, pp. 263-268.

[22] L. O. Lobo and J. D. Arthur, "Local and global analysis: Complementary activities for increasing the effectiveness of requirements verification and validation," in *Proceedings of the 43rd Annual ACM Southeast Conference*, Kennesaw, GA, 2005, pp. 2256-2261.

[23] R. Marelly, D. Harel, and H. Kugler, "Multiple instances and symbolic variables in executable sequence charts," in *17th International Conference on Object-Oriented Programming, Systems, Languages and Applications (OOPSLA 2002)*, USA, 2002, pp. 83-100.

[24] Niu Nan, Yu Yijun, B. Gonzalez-Baixauli*, et al.*, "Aspects across software life cycle: a goal-driven approach," in *Transactions on Aspect-Oriented Software Development. VI. Special Issue on Aspects and Model-Driven Engineering*, ed Berlin, Germany: Springer Verlag, 2009, pp. 83-110.

[25] Nguyen Duy Cu, A. Perini, and P. Tonella, "A goal-oriented software testing methodology," in *Agent-Oriented Software Engineering VIII. 8th International Workshop, AOSE 2007*, Berlin, Germany, 2008, pp. 58-72.

[26] F. Abbors, D. Truscan, and J. Lilius, "Tracing requirements in a model-based testing approach," in *2009 First International Conference on Advances in System Testing and Validation Lifecycle (VALID)*, Piscataway, NJ, USA, 2009, pp. 123-8.

[27] E. Mendez, M. Perez, and L. E. Mendoza, "Improving software test strategy with a method to specify test cases (MSTC)," in *10th International Conference on Enterprise Information Systems. Databases and Information Systems Integration*, Setubal, Portugal, 2008, pp. 159-64.

[28] C. Quinn, S. Vilkomir, D. Parnas*, et al.*, "Specification of software component requirements using the trace function method," in *International Conference on Software Engineering Advances*, Tahiti 2006, pp. 50 - 50

[29] L. Naslavsky, T. A. Alspaugh, D. J. Richardson*, et al.*, "Using scenarios to support traceability," in *Proceedings of the 3rd international workshop on Traceability in emerging forms of software engineering(TEFSE)* California,USA, 2005, pp. 25-30.

[30] A. Goel and A. Roychoudhury, "Synthesis and traceability of scenario-based executable models," in *2006 2nd International Symposium on Leveraging Applications of Formal Methods, Verification and Validation (ISoLA 2006), 15-19 Nov. 2006*, Piscataway, NJ, USA, 2008, pp. 347-54.

[31] S. Ibrahim, M. Munro, A. Deraman*, et al.*, "A software traceability validation for change impact analysis of object oriented software," in *Proceedings of the International Conference on Software Engineering Research and Practice and Conference on Programming Languages and Compilers SERP'06*, USA, 2006, pp. 453-9.

[32] Hubert Dubois, Marie-Agnès Peraldi-Frati, and Fadoi Lakhal, "A model for requirements traceability in an heterogeneous model-based design process. Application to automotive embedded systems," in *15th IEEE International Conference on Engineering of Complex Computer Systems (ICECCS)*, Oxford, UK, 2010, pp. 233-242.

[33] F. Bouquet, C. Grandpierre, B. Legeard*, et al.*, "A subset of precise UML for model-based testing," in *Proceedings of the 3rd international workshop on Advances in modelbased testing (AMOST)*, 2007, pp. 95-104.

[34] J. Kelleher and M. Simonsson, "Utilizing use case classes for requirement and traceability modeling," in *Proceedings of the 17th IASTED International Conference on Modelling and Simulation*, Anaheim, CA, USA, 2006, pp. 617-25.

[35] A. Sabetta, D. C. Petriu, V. Grassi*, et al.*, "Abstraction-raising transformation for generating analysis models," in *Satellite Events at the MoDELS 2005 Conference. MoDELS 2005 International Workshops.* , Berlin, Germany, 2005, pp. 217-26.

[36] R. Hassan, M. Eltoweissy, S. Bohner*, et al.*, "Formal analysis and design for engineering security automated derivation of formal software security specifications from goal-oriented security requirements," *IET Software,* vol. 4, pp. 149-60, 2010.

[37] Tanvir Hussain and Robert Eschbach, "Automated Fault Tree Generation and Risk-Based Testing of Networked Automation Systems," in *Proceedings of 15th IEEE Conference on Emerging Technologies and Factory Automation (ETFA 10)* Bilbao, Spain, 2010.

[38] R. Mugridge, "Managing agile project requirements with storytest-driven development," *IEEE Software,* vol. 25, pp. 68-75, 2008.

[39] C. Bâillon and S. Bouchez-Mongardé, "Executable requirements in a safety-critical context with Ada," *Ada User Journal,* vol. 31, pp. 131-135, 2010.

[40] G. Sabaliauskaite, A. Loconsole, E. Engstrom*, et al.*, "Challenges in Aligning Requirements Engineering and Verification in a Large-Scale Industrial Context," in *Requirements Engineering: Foundation for Software Quality. 16th International Working Conference, REFSQ*, Berlin, Germany, 2010, pp. 128-42.

[41] C. Nebut, F. Fleurey, Y. Le Traon*, et al.*, "Automatic test generation: a use case driven approach," *IEEE Transactions on Software Engineering,* vol. 32, pp. 140-55, 2006.

[42] M. W. Whalen, M. P. E. Heimdahl, A. Rajan*, et al.*, "Coverage metrics for requirements-based testing," in *Proceedings of the international symposium on Software testing and analysis(ISSTA)* 2006, pp. 25-35.

[43] M. Conrad, I. Fey, and S. Sadeghipour, "Systematic Model-Based Testing of Embedded Automotive Software," *Proceedings of the Workshop on Model Based Testing(MBT), Electronic Notes in Theoretical Computer Science,* vol. 111, pp. 13-26, 2005.

[44] Sebastian Siegl, Kai-Steffen Hielscher, and Reinhard German, "Model Based Requirements Analysis and Testing of Automotive Systems with Timed Usage Models," in *18th IEEE International Requirements Engineering Conference*, Sydney, New South Wales Australia, 2010.

[45] M. Riebisch and M. Hubner, "Traceability-driven model refinement for test case generation," in *Proceedings. 12th IEEE International Conference and Workshops on the Engineering of Computer-Based Systems*, CA, USA, 2005, pp. 113-20.

[46] D. Arnold, J. P. Corriveau, and Shi Wei, "Scenario-Based Validation: Beyond the User Requirements Notation," in *Software Engineering Conference (ASWEC), 2010 21st Australian*, 2010, pp. 75-84.