Patterns for Development of Safety-Critical Systems with Agile: Trace Safety Requirements and Perform Automated Testing

Hafiza Maria Maqsood
University of Florence
Florence, Italy
hafizamaria.maqsood@unifi.it

Eduardo Martins Guerra
Free University of Bozen,Bolzano
Bolzano, Italy
guerraem@gmail.com

Xiaofeng Wang
Free University of Bozen
Bolzano, Italy
xiaofeng.Wang@unibz.it

Andrea Bondavalli
University of Florence
Florence, Italy
andrea.bondavalli@unifi.it

ABSTRACT

In safety-critical systems keeping complete trace of requirements and detailed testing is an extremely relevant part of software development life cycle. Safety standards like ISO 26262, DO178C and many others prescribe that critical requirements must be completely traceable. These standards also demand detailed and regression testing of system. Here we present some patterns that deal with these concerns in an agile way. First set of patterns describe the key mechanism to list the sources of safety requirements and a mechanism for traceability of those requirements. It uses an approach that satisfies safety standards and adapts agile behavior where possible. The second pattern is about test automation for safety-critical systems, which complements our first set of patterns. It decreases the amount of documentation required for traceability and testing of features but without any compromise on essential testing. These patterns will facilitate the team to perform requirement's traceability and regular, rigorous testing in a timely and cost efficient manner.

CCS CONCEPTS

• **Software and its engineering → Agile software development**; *Agile software development*; *Requirements analysis.*

KEYWORDS

Agile, Traceability, Documentation, Requirements, Scrum, Safety, Safety-Critical Systems, Testing, Test Automation, Testing in Agile

ACM Reference Format:
Hafiza Maria Maqsood, Eduardo Martins Guerra, Xiaofeng Wang, and Andrea Bondavalli. 2020. Patterns for Development of Safety-Critical Systems with Agile: Trace Safety Requirements and Perform Automated Testing. In *European Conference on Pattern Languages of Programs 2020 (EuroPLoP '20), July 1–4, 2020, Virtual Event, Germany*. ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3424771.3424800

EuroPLop '20,
,

ACM ISBN 978-1-4503-7769-0/20/07…$15.00
https://doi.org/10.1145/3424771.3424800

1 INTRODUCTION

Software traceability and testing have always been considered a very important property for well-engineered software. The definition is given by Center of Excellence for Software and Systems Traceability [5] (CoEST). They state that "ability to interrelate any uniquely identifiable software engineering artifact to any other, maintain required links over time, and use the resulting network to answer questions about both the software product and its development process is traceability" [5]. Traceability is an integral part of the certification and approval process of most of the safety-critical systems. Though traceability is a very important factor in safety-critical systems [6] [14], yet it is a very elusive quality of the software development process. The effort, cost, and formalism required to maintain links for traceability are considered extremely high and do not go hand in hand with rapid development like agile [2] [15].

Considering the importance of formal procedure for traceability of requirements, the U.S Department of Defense has identified the procedure as one of the seven most critical and needed research areas. According to them, it must be targeted to ensure the safe and accurate operations of present and future software-intensive systems [8]. Agile methodologies are based on 12 principles and four basic values [3]. A project is said to be truly agile if it follows all of these principles and values. However, in safety-critical systems, it is mandatory to be compliant with safety regulations that advocate detailed documentation of each step. This makes use of any single agile technique very difficult [22]. For safety-critical projects, the lack of safety-related documentation can impact communication and can harm the overall process of communication, especially in terms of safety requirements [29].

The agile manifesto states that it gives priority to working software over detailed documentation [3]. It seems there is a conflict between heavy documentation and agile principles [26]. Cohen [7] argues that there are issues that must be looked upon in written communication and we should not abandon documentation, instead we should use the documentation at and for appropriate points, especially for the development of safety-critical systems [29].

Test automation is one of the key concepts when we try to develop safety-critical systems with agile methodologies. There are certain challenges that a team needs to overcome when they want to develop safety-critical systems by using Scrum or XP (the two

most widely used agile approaches). These challenges can be test automation, short iterations, continuous integration and regression testing [27]. As the project moves, it becomes more complex to make sure that previously developed modules are working fine. In this scenario, automated tests can prove to be very useful.

The documentation of these patterns is part of an effort to identify practices that can be used to apply agile in safety-critical projects.We have performed a systematic literature review to find out the key issues and problems in adapting agile for safety-critical systems. It was concluded that agile cannot be adapted completely for development of safety-critical systems. However, there are some phases of software development life cycle where we can incorporate agile principles. In future , we will be working on three more approaches to support the process i.e. automate the requirement traceability and testing process (the current patterns are the first steps towards automation of traceability and testing), merge development and safety teams, and anticipate the safety verification.

'Everyone Responsible for Safety' is an idea about making safety as part of team responsibility. In Safety Verification we will work on an approach similar to Test Driven Development that argues to define test cases with requirements.

In current practices, development of safety-critical systems is carried out using traditional approaches like waterfall, V model etc. These patterns are developed for the teams who want to entirely replace traditional models with agile process models or want to use agile models in collaboration with traditional models. All agile iterative process models e.g. scrum, xp etc. are suitable for the proposed approach. This paper presents first set of patterns for tracing safety requirements and second pattern for test automation of safety-critical systems.

2 IDENTIFICATION AND TRACEABILITY OF SAFETY REQUIREMENTS

2.1 Context

In agile process models requirements are communicated in a very informal way, mostly by on-site meetings with customers. On the other hand, the traceability matrix requires a formal requirement specification document to proceed with the process [13]. The principles of agile processes make it difficult to follow a static document-centric model. To establish a link between requirement changes and design constructs, new/changed requirements must be mapped onto previous specifications. Agile in its nature does not provide room for traceability, hence to incorporate it in agile, the process has to be integrated from the very first step. The initial informal meetings about customer needs and their refinement to requirements must be addressed formally. Developing software is an exploratory process and thus there must be some mechanism to handle the change or more formally a mechanism of change management must be in place. To create safety-critical systems with agile, quality of process must be maintained with best practices for planning, traceability and continuous gathering and validation of requirements [13].

Solutions proposed for agile processes must be supportive of these human-centric practices and must be adapted in the form of user stories, plans or tasks to accommodate changes.

2.2 Motivation

For example, the DO-178C standard [25], which the USA Federal Aviation Administration (FAA) has established as the means of certifying that software aspects of airborne systems comply with airworthiness requirements, specifies a very detailed set of traceability requirements including the need to provide traceability between source code and low-level requirements "to enable verification of the absence of undocumented source code and verification of the complete implementation of the low-level requirements." Similarly, the USA Food and Drug Administration (FDA) states that traceability analysis must be used to verify that the software design implements the specified software requirements, all aspects of the design are traceable to software requirements and that all code is linked to established specifications and test procedures [10].

Christopher Lee et al. [21] claim that many informal techniques are used in agile methodologies especially for elicitation of requirements and implementing them. They further argue that the document-centric approach does not fit well with agile principles. To deal with the issue of traceability, it is important that we first deal with the technique for gathering requirements and refining them into formal specifications.

Scrum is one of the most widely used process models. It has user stories to deal with requirements which are a very informal way of handling requirements. It is based on meetings and views of different stakeholders, taken in terms of user stories written in no particular format. Jacobsson [18] argues that teams who follow Scrum consider it to be the magical solution. They think that, by following Scrum from start to end at each step, all problems will be solved, which is not the case in reality. Especially the informal way of handling requirements is not sufficient for traceability. Scrum suggests that for each sprint there is a meeting of all stakeholders and they come up with user stories as requirements written in an informal style. However, the challenge is, if we start making detailed documentation in every sprint, the process will loose its ability to deliver modules in short time sprints.

There is a need for some mechanism that can suggest a way of taking and tracing requirements [28] while staying agile in Scrum. By staying agile we mean doing it in a short time for each sprint of Scrum. This [24] becomes more critical when we talk about safety-critical systems where traceability of requirements is not an option instead it is a must-have property of the process when dealing with the development of safety-critical requirements [12] [4]. This difference in the basic nature of both processes makes it difficult to perform traceability with agile process models [30].

2.3 Forces

- **Traceability**: Traceability is an issue in safety-critical systems in its very own nature also, as for safety certifications it is required to have a complete track of traceability of requirements. To perform this efficiently is a challenge in itself.
- **Up-front Design**: Traditional approaches advocate up-front design; hence all requirements are agreed upon in the beginning and are locked at the end of the requirement phase. On the other hand, agile advocates evolutionary design instead of planned upfront design and hence accommodates changes throughout the development phase.

- **Safety Analysis Techniques**: The safety analysis techniques are designed for traditional approaches; they do not fit well with agile methodologies since they need a pre-defined stable architecture and set of requirements.
- **Requirement Elicitation**: Requirements elicitation takes place in the form of conversations, where insights into the problem space, clarification of assumptions, and deeper understanding takes place. While these conversations are rarely captured because of their ad-hoc nature, tools should provide a mechanism to record unstructured requirements elicitation and transcribe them to a more structured model when appropriate [21].

2.4 Tracing Safety Requirements

2.4.1 Problem. It is evident that to find a middle ground between agile principles and requirement traceability matrices we need to keep documentation composite. Hence all requirements do not need to be traced, there must be an approach to identify critical safety requirements which must be documented and updated.More specifically, in traceability pattern we address the problem of

"What factors determine that any given requirement is safety related requirement?"

2.4.2 Solution. In this pattern, we propose a method to identify safety requirements. As advocated before we propose to use an amalgamation of the traditional modeling approach and agile principles. It will shorten the documentation and time the agile team needs to spend on documenting requirements. Instead of taking a formal approach for all requirements, they will do it only for selected ones.

We propose a feature based criteria to select requirements for traceability. At the beginning of the project, the requirements can come from:

- Safety standards,
- project Stakeholders,
- Safety analysis.

At this point, in the beginning, we take these requirements and relate them to each feature of the product. One requirement may be related to more than one feature. For example user authentication is a requirement which will be fulfilled by developing a feature of login. But "login" can be required to access many other features of the system as well. Hence, when those features are developed it must be checked if login is still working properly OR it is not effected adversely. We design iterations around these features. In each iteration, we track safety requirements relevant to the feature being developed figure **??**.

At the start of every iteration, the whole team will have a meeting to decide safety requirements or if there is any change/update in these requirements. During the development, the change can come through two sources:

- Changes to existing safety requirements,
- Changes that will influence code that directly or indirectly belongs to a safety requirement.

The rest of the process for enlisting and tracking requirements remain the same during the whole development process.

To mark any requirement as "DONE", it must be verified that a relevant feature has been completely built and delivered AND all dependent features of that requirement are also built and delivered. The requirement will not be considered "DONE" if there is any feature of the system directly or indirectly related to the requirement that is still in the process of development. Then it proposes to have upfront design, enough to start the project but allows space to change or update it during the whole development life cycle. It can fit into the iterative nature of agile while carefully keeping track of changes relevant to safety without compromising safety standards.

2.4.3 Consequences. (+) A list of upfront high-level safety requirements makes it compliant to many safety standards.
(-) An important requirement may not be identified and will not be traced.

2.4.4 Problem. "How to trace safety requirements throughout software development?"

2.4.5 Solution. This is a pattern for traceability of selected requirements throughout the software development life cycle. The method is a hybrid approach of traditional safety-critical development approaches and agile principles. Identify safety requirements and trace only selected requirements during the whole development process. Use an iterative approach to develop the project as it will pave a way for more validated short and timely outputs, also team does not need to plan everything upfront. However, it is important to note that the team must have enough knowledge of requirements in hand that they can sketch an idea of the whole system to start with.

The method to keep track of requirements must be formal. It can be a document, database or some online tool that facilitates such tasks. Each safety requirement will have a code attached to it which will represent some basic information like relevant features, unique id, and version. Codes for features can be selected by the team working on a project. If there is any update/change in the requirement, its updated version will be created. If there is any update or change in any requirement, all of its mentioned "relevant features" must be checked for updates and tested for validity and verification.

This will balance out the problems that teams face when adopting agile process models for the development of safety-critical systems. It will provide a form of upfront planning but can deal with the iterative nature of agile. It will not create too heavy documentation to handle, as not all requirements are accounted for. The time required to perform these steps at the beginning of the project and start of every iteration is not very long and hence the principle of "working software in short iterations" is not affected. It will move the team from the ad-hoc approach of taking requirements to formal documentation but only for a particular set of requirements, safety requirements in this case. Since it is only for specific kinds of requirements it will not add burden on a process in terms of time or documentation.

At the end of each iteration, the whole team will discuss all requirements for next iteration, as is normal practice in agile, but now they will select safety requirements that will be traced throughout

the process OR will update the trace document of safety requirements. This update can be in terms of adding a new requirement, modifying any current requirement, giving the status of "DONE" to any requirement OR selecting a set of requirements for the next iteration. The second idea addresses the complete goal in terms of short, small goals. By complete the goal, we refer to the final product required. A team can collectively decide the goals to be safety-critical. So when there is any requirement change is requested that is directly related to one of these goals or it has an impact on these goals, we need to include it in requirements that must be traced. These values can be mapped to create a matrix that can be evaluated for each iteration.

This pattern can bridge the gap between agile principles and safety standards. It can work with comparatively less documentation but at the same time it maintains log for safety requirements.

2.4.6 Consequences. (+) The maintenance of the traceability documentation is reduced because of a short number of requirements to be traced.
(+) The proposed pattern compliments the meetings of agile teams before each iteration, it will help to decide and update traceability document for safety requirements.
(+) The pattern supports the idea of using automated tools OR document self-created by the team.
(-) With iterative development, it is difficult to validate and verify system-level behavior.

2.5 Known Uses

Wang et al. [28] applied Scrum, an agile process model to the Safe Home project, a safety-critical system. They applied an approach to safety stories and STPA for developing this safety-critical system. Their research provided the first practical and empirical view of applying agile to safety-critical systems. With this approach, they found that there were challenges in communication, priority management, and acceptance criteria for safety requirements. They suggested looking for solutions in terms of safety stories, pre-planning meetings or regular safety meetings [29].

Implemented [23]the agile practices within Abbott Diagnostics. The company completed two projects side by side one with agile methodology and another with the plan-driven approach. They concluded that the projects they completed with agile have cost savings of 35 percent to 50 percent as compared to plan-driven projects. They also faced some challenges with an agile approach, which included accommodating changes, applying the agile approach completely. They had to adopt a hybrid approach by combining VModel and Scrum. They reported having traceability issues, lack of upfront planning and managing multiple releases with some other ones.

Applied [11] agile approach to image-guided surgical toolkit development. He reports among major issues the problem to perform hazard analysis, specifying and analyzing safety requirements, testing those requirements and compliance with safety standards and certification.

All of the above mentioned have used Scrum from agile process models and have reported problems in the tracking of requirements and analyzing critical requirements in the process, this is the issue addressed by our proposed pattern.

3 PATTERN FOR TEST AUTOMATION OF SAFETY-CRITICAL SYSTEMS IN AN AGILE WAY

3.1 Context

In recent years agile methodology has proven to be useful in the process of software development for different kinds of software. Here we are particularly concerned with safety-critical software development. Testing is another huge area that needs to be addressed when we talk about safety-critical systems. Testing can be manual or automatic. Safety-critical systems go through rigorous testing as a part of their basic development process. When we try to address the development of safety-critical systems automated testing can be helpful to adapt agile for their development as in agile we are looking forward to reduce documentation [31] [16] and time. Dustin et al. [9] gives definition of automated testing as the automation of tests include the usage of automated tools for testing, execution of scripts for testing and verification process for testing of requirements.

3.2 Motivation

Karhu says [20] that manual testing takes a lot more time as compared to automated testing. Automation of tests also increases the efficiency for performing the steps to produce same functionality of system repeatedly. This is in particular a huge help in performing the tests repeatedly and iteratively when there are any changes to the software. Automation of tests can take up to 50 percent of total project effort, but it is worth the investment.

Testing can be tough task to handle especially when developing with agile approach. Testing needs to be handled in same way and given same importance as other artifacts during the process, or even more. Preparing a right suit of tests can save a lot of time and effort; carefully developed test strategy can be used repeatedly and incrementally. In today's world it even has more importance due to rapidly changing nature of software requirements and speedy integration needs. This brings in the demand of efficiency and reusability. According to [1] Almog testing is the most expensive part of development. Automation should be applied in testing wherever possible to reduce expenses; at least repetitive tasks can be handled through test automation. In case of safety-critical systems, a thorough approach is required to adapt the procedure.

3.3 Problem

It is evident that testing is one of the major areas of concern for safety-critical systems. When we try to develop such complex systems in an agile way, automation of testing needs to be handled with extreme care.In this pattern we address the problem of

"How to perform automated testing in order to achieve the reduced documentation and time in the process of developing safety-critical systems by agile approaches?"

3.4 Forces

- **Documentation**: Agile suggests using lean documentation, testing needs detailed documentation that needs regular updates.

- **Traditional Testing**: In traditional approaches, testing is not an iterative or incremental process, instead, it is considered a linear process. In agile, testing must be handled iteratively. Since all agile approaches work to deliver working software in small iterations. For example, scrum works in sprints; each sprint is an iteration and at the end generates some output. The idea is to deliver running modules in short sprints. Similarly XP, another widely used agile process model, advocates small releases of running software
- **Cost and Time**: Testing is most expensive part of project, needs a lot of time and cost to perform, whereas agile focus on rapid development in minimum cost possible.

3.5 Solution

We present a testing strategy as continuation of requirement traceability pattern defined above. As we are grouping requirements on the basis of system features, we propose to do the same with testing. We propose to have automated tests to check each feature of the system. As the process is iterative and incremental, with each iteration new changes can be checked manually, whereas the whole feature can be tested automatically to see the effect of change. And to make sure that even after the proposed change system's particular feature is behaving appropriately. In safety-critical systems we have to keep track of every requirement, along with any change requested in the requirements, testing in this scenario can be automated to certain extent only and with great care. In the light of critical nature of safety systems, we propose to perform manual testing of each requirement when it's implemented for first time. When the second requirement belonging to same feature is implemented, first is tested again automatically. In this way the automated testing is performed incrementally along with requirement implementation.

Each new requirement is tested manually, and all previous requirements are tested automatically related to one particular feature of the system. When one feature is marked as complete, the same procedure is repeated. Let's say we have feature 1 completely developed and tested, now we are implementing requirement 1 of second feature, this requirement will be manually tested whereas feature one will be automatically tested during testing phase. When any change comes in a requirement of any feature, the changed requirement's implementation will be tested manually and all completed features will be tested automatically in testing phase, to make sure that there is no harm done to already completed features of the system. The proposed approach will be repeated with every new requirement, any change in current requirements and every iteration of development. This will decrease the load of manual testing, although we cannot eliminate it completely, but this hybrid approach paves a way to develop safety-critical systems with agile approach incorporated by finding a middle ground to suit the needs of both.

3.6 Consequences

(+)Through automation of testing, there is no compromise on testing at any stage, automation only helps in reduced documentation and effort.
(+)The proposed pattern fits well with the current team structure of agile where teams are multi-functional. The same team can work on requirements and testing to consider the requirement as "DONE".
(+)This pattern supports the idea of using automated tools to shorten the time span of testing.
(+)The proposed approach of testing will enable to test multiple features at every iteration; hence can be useful to test system level behavior.
(-)The process of testing is dependent on identification and implementation of requirements if an important requirement is not identified it will not be tested and traced.
(-)The process is completely dependent on people working in a team and hence chances of human error cannot be neglected.

3.7 Known Uses

[19] Have presented a case study on testing of software for nuclear reactor. These systems are safety-critical systems and hence highly fragile. The process adapted for development must be thoroughly traceable and it must conform to certain safety standards. Jee et. al [19] have presented an approach for automation of testing for this system. Their tool provides test coverage for function block diagrams, which are intensively used in such systems. They took test cases prepared by experts of this domain for protection system software. By using automated testing through the tool they developed they found out that there were many paths not covered by manual testing performed by expert testers. Further they have demonstrated in a quantitative way a detailed analysis of results generated by tests. Their case study promisingly conveys that the idea of automated testing is extremely effective in safety-critical systems. Further they argue that this approach is more effective in terms of accuracy. It is highly intuitive and provides continuous monitoring progress.

They [17] applied the software based self-tests referred to as SBST frame work for automation of tests to support the design of an industrial programmable logic controller. This unit was designed for hydro-electric power plants. The approach provided extremely efficient feedback on software based self-tests in terms of detection of faults and summary of the diagnostic coverage. They further argue that this approach can be used in up front design, since it does not require hardware. Up front design is one of the main requirements for development of safety-critical systems. It works by injecting faults in the system for testing and hence approach can be used in an agile manner of development too by introducing new faults to check newly developed features of the system. Both of the case studies presented above argue the importance and usefulness of automated testing in safety-critical systems. Furthermore, it can be observed that the approach used for automation of tests in these cases can be adapted easily in agile way of development and is in line with our both patterns.

4 CONCLUSIONS

The first set of patterns for requirement traceability paves a way to develop safety-critical systems with a hybrid approach that finds a middle ground between traditional safety approaches and agile development principals. It is a fact that safety-critical systems cannot be developed completely with agile approaches, however, since

we have proven advantages of agile process models for development we have given a method to include them in the requirement engineering phase of development for safety-critical systems.

Test pattern describes a way of automation of testing for safety-critical systems. Testing can be a hybrid approach of traditional testing strategies for safety-critical systems and using automated tools where possible. Our proposed pattern pin points the areas and applications of automated testing in the agile iterations while keeping the rules of safety standards in tact in the whole procedure.

These patterns can be used as complimentary patterns with each other or can be used separately also. This leads to a complete approach of designing iterations in agile manner with complete traceability of requirements and testing in time efficient manner through careful selection of safety requirements and test automation.

REFERENCES

[1] Dani Almog and Yaron Tsubery. 2015. How the Repository Driven Test Automation (RDTA) will make test automation more efficient, easier & maintainable. In *Proceedings of the 8th India Software Engineering Conference*. 196–197.

[2] Paul Arkley and Steve Riddle. 2005. Overcoming the traceability benefit problem. In *13th IEEE International Conference on Requirements Engineering (RE'05)*. IEEE, 385–389.

[3] Kent Beck, Mike Beedle, Arie Van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, et al. 2001. The agile manifesto.

[4] Jane Cleland-Huang. 2006. Just enough requirements traceability. In *30th Annual International Computer Software and Applications Conference (COMPSAC'06)*, Vol. 1. IEEE, 41–42.

[5] Jane Cleland-Huang, Adam Czauderna, Alex Dekhtyar, Olly Gotel, Jane Huffman Hayes, Ed Keenan, Greg Leach, Jonathan Maletic, Denys Poshyvanyk, Youghee Shin, et al. 2011. Grand challenges, benchmarks, and TraceLab: developing infrastructure for the software traceability research community. In *Proceedings of the 6th international workshop on traceability in emerging forms of software engineering*. 17–23.

[6] Jane Cleland-Huang, Orlena CZ Gotel, Jane Huffman Hayes, Patrick Mäder, and Andrea Zisman. 2014. Software traceability: trends and future directions. In *Proceedings of the on Future of Software Engineering*. 55–69.

[7] Mike Cohn. 2010. *Succeeding with agile: software development using Scrum*. Pearson Education.

[8] National Research Council et al. [n.d.]. Committee for Advancing Software-Intensive Systems Producibility.(2010). *Critical Code: Software Producibility for Defense* ([n. d.]).

[9] Elfriede Dustin, Jeff Rashka, and John Paul. 1999. *Automated software testing: introduction, management, and performance*. Addison-Wesley Professional.

[10] René Freude and Alexander Königs. 2003. Tool integration with consistency relations and their visualization. In *Proc. Workshop on Tool-Integration in System Development (TIS 2003)*. Citeseer, 6–10.

[11] Kevin Gary, Andinet Enquobahrie, Luis Ibanez, Patrick Cheng, Ziv Yaniv, Kevin Cleary, Shylaja Kokoori, Benjamin Muffih, and John Heidenreich. 2011. Agile methods for open source safety-critical software. *Software: Practice and Experience* 41, 9 (2011), 945–962.

[12] Sebastian Gayer, Andrea Herrmann, Thorsten Keuler, Matthias Riebisch, and Pablo Oliveira Antonino. 2016. Lightweight traceability for the agile architect. *Computer* 49, 5 (2016), 64–71.

[13] Arbi Ghazarian. 2008. Traceability patterns: an approach to requirement-component traceability in agile software development. In *Proceedings of the 8th conference on Applied computer scince*. World Scientific and Engineering Academy and Society (WSEAS), 236–241.

[14] Janusz Górski and Katarzyna Łukasiewicz. 2012. Assessment Of Risks Introduced To Safety Critical Software By Agile Practices—A Software Engineer's Perspective. *Computer Science* 13, 4 (2012), 165.

[15] Orlena CZ Gotel and CW Finkelstein. 1994. An analysis of the requirements traceability problem. In *Proceedings of IEEE International Conference on Requirements Engineering*. IEEE, 94–101.

[16] Rashina Hoda, James Noble, and Stuart Marshall. 2012. Documentation strategies on agile software development projects. *International Journal of Agile and Extreme Software Development* 1, 1 (2012), 23–37.

[17] Andrea Höller, Gerhard Schönfelder, Nermin Kajtazovic, Tobias Rauter, and Christian Kreiner. 2014. FIES: a fault injection framework for the evaluation of self-tests for COTS-based safety-critical systems. In *2014 15th International Microprocessor Test and Verification Workshop*. IEEE, 105–110.

[18] Marcus Jakobsson. 2009. *Implementing traceability in agile software development*. Department of Computer Science, Lund University.

[19] Eunkyoung Jee, Suin Kim, Sungdeok Cha, and Insup Lee. 2010. Automated test coverage measurement for reactor protection system software implemented in function block diagram. In *International Conference on Computer Safety, Reliability, and Security*. Springer, 223–236.

[20] Katja Karhu, Tiina Repo, Ossi Taipale, and Kari Smolander. 2009. Empirical observations on software testing automation. In *2009 International Conference on Software Testing Verification and Validation*. IEEE, 201–209.

[21] Christopher Lee, Luigi Guadagno, and Xiaoping Jia. 2003. An agile approach to capturing requirements and traceability. In *Proceedings of the 2nd International Workshop on Traceability in Emerging Forms of Software Engineering (TEFSE 2003)*, Vol. 20.

[22] Martin McHugh, Fergal McCaffery, and Valentine Casey. 2012. Barriers to adopting agile practices when developing medical device software. In *International Conference on Software Process Improvement and Capability Determination*. Springer, 141–147.

[23] Martin McHugh, Fergal McCaffery, and Garret Coady. 2015. Adopting Agile Practices when Developing Medical Device Software. *J Comput Eng Inf Technol 4: 2. doi: http://dx. doi. org/10.4172/2324* 9307 (2015), 2.

[24] J Rasmussen. 2003. Introducing XP into greenfield projects: Lessons learned. *Ieee Software* 20, 3 (2003), 21–28.

[25] RTCA SC. 2011. 205/EUROCAE WG-71. Software Considerations in Airborne Systems and Equipment Certification. *No. RTCA DO-178C, RTCA Inc* 1140 (2011).

[26] Christoph Johann Stettina and Werner Heijstek. 2011. Necessary and neglected? An empirical study of internal documentation in agile software development teams. In *Proceedings of the 29th ACM international conference on Design of communication*. 159–166.

[27] Sulabh Tyagi, Ritu Sibal, Bharti Suri, Bimlesh Wadhwa, and Shanuj Shekhar. 2018. Development of Reusable Hybrid Test Automation Framework for Web Based Scrum Projects. *Journal of Applied Science and Engineering* 21, 3 (2018), 455462.

[28] Yang Wang, Ivan Bogicevic, and Stefan Wagner. 2017. A study of safety documentation in a Scrum development process. In *Proceedings of the XP2017 Scientific Workshops*. 1–5.

[29] Yang Wang, Jasmin Ramadani, and Stefan Wagner. 2017. An exploratory study on applying a scrum development process for safety-critical systems. In *International Conference on Product-Focused Software Process Improvement*. Springer, 324–340.

[30] Rebecca Wirfs-Brock, Joseph Yoder, and Eduardo Guerra. 2015. Patterns to develop and evolve architecture during an agile software project. In *Proceedings of the 22nd Conference on Pattern Languages of Programs*. 1–18.

[31] Zheying Zhang, Mike Arvela, Eleni Berki, Matias Muhonen, Jyrki Nummenmaa, Timo Poranen, et al. 2010. Towards lightweight requirements documentation. *Journal of Software Engineering and Applications* 3, 09 (2010), 882.