Teams AND Up-Front Testing for Development of Safety-Critical Systems with Agile

H.Maria Maqsood, University of Florence
Eduardo Guerra, University of Bolzano
Xiaofeng Wang, University of Bolzano
Andrea Bondavalli, University of Florence

Categories and Subject Descriptors: D.2.7 [**Software Engineering**]: Documentation—*Evaluation/ methodology*; D.2.5 [**Software Engineering**]: Testing and Debugging—*Testing tools/ (e.g., data generators, coverage testing)*; B.8.2 [**Hardware**]: PERFORMANCE AND/ RELIABILITY—*Performance Analysis and Design Aids*; I.5.2 [**Pattern Recognition**]: Design Methodology—*Feature evaluation and selection*

General Terms: Agile Teams, Safety-Critical Systems,Testing

Additional Key Words and Phrases: Agile, Agile Teams, Autonomous agile teams, safety-critical systems, Testing, Up-font Testing, Test in Agile development, Testing Safety-Critical Systems

1. INTRODUCTION

There are multiple issues/challenges that must be addressed for making agile more suitable for safety-critical systems. In this paper we will address the issues relevant to team formation, communication and testing. Up-front testing goes well in line with agile principles, this is the reason we choose it for our proposed approach.

Safety-critical systems are defined as those systems whose failure can cause harm [Mwadulo 2016]. The system is considered safety-critical if its failure can lead to unacceptable circumstances such as loss of human lives or damage to the environment[Mwadulo 2016]. Development of these systems in an agile way can be very beneficial in terms of time and cost. The basic principles of agile say that there should be rapid development, strong communication among all stake holders and changes should be welcome at any stage of development [Alliance 2001]. As there is a lot of focus on people so every individual in team should be motivated and must be given suitable environment and support to perform their jobs [Stavru 2014]. Hence team building and communication is an important aspect for this approach.

[Stålhane-IDI] To build a successfully functioning team It is extremely necessary that whole team is on board regarding safety aspects of the system. [Leite 2017] Agile teams provide room for requirement changes during the development process. In case of safety-critical systems this needs to be handled with great care. Agile has major focus on team collaborations and interaction among all stakeholders[McHugh et al. 2011] and promotes self organizing and cross functional teams [Boehm 2002].

Along with team formation and communication testing is another crucial aspect of safety-critical systems. It poses many challenges to incorporate rigorous testing in short time span and within budget limits. Efficiency

in terms of cost and time are the principles that go hand in hand with agile. The critical nature of safety-critical systems requires that if not all , maximum risks are handled during development of these systems. However, it is not enough to just perform risk analysis, certain safety standards must be Incorporated during development of these systems, for this reason they must have testing strategy in place [Zimmermann et al. 2009]. There is a lot of room for improvement of this mechanism to fit perfectly to the needs of safety-critical systems especially in agile perspective [Tracey 2000]. In short we can say that

—Agile stresses on cross-functional autonomous teams [Chen et al. 2015] whereas, safety-critical systems need specific experts doing specific tasks.

—There is also a gap to be filled about how organizations should arrange teams to achieve the right level of autonomy in any particular scenario, in our case for safety-critical systems.[Hoda et al. 2012]

—It is a challenging task to create cross-functional teams and keep the size of the team small .(10-15) Persons on team, which is a standard agile team size.[Stray et al. 2018]

—[Holcombe et al. 1995] states that formal verification methods and test driven development can be used together and there can be many enhanced benefits of this approach. It can ensure safety of systems along with reliability. He further argues that safety-critical systems have to comply to certain safety standards, currently there are not many techniques of testing that can fit into this scenario.

This paper has two major sections. First section describes proposed patterns for Agile teams, their characteristics and communication for development of safety-critical systems. Second section suggests ways of up-front test planning for safety-critical systems.

2. AGILE TEAMS FOR SAFETY-CRITICAL SYSTEMS

In this section we present three patterns that will help in choosing team members, forming a team and communication among them. These patterns share the same context and forces.

2.1 Context

The process of building autonomous teams in agile software development is still not given the due share of importance. It is very important to carefully choose a team, assign roles to them and establish effective communication among them specifically when we are building safety-critical systems. People with different back grounds and expertise have different working norms [Stray et al. 2018]. Agile is less focused on documentation so teams should make only necessary documents to ensure that risks and vulnerabilities are managed. Agile starts with minimal documentation and contracts because otherwise these documents might be wasted as changes invalidate the upfront work. It is a challenging task to have rigorous communication while keeping documentation minimal. There are many challenges to come up with good working teams that have different skill sets and establish successful communication mechanism between them.

2.2 Forces

Agile suggests using cross-functional teams to keep size of team as small as possible. In safety-critical systems teams are divided and each team is focused on any one particular area hence generally it ends up with bigger number of teams. Agile suggests creating small teams generally consisting of 10-12 people whereas, Safety-critical system's development teams are large because they need experts of every area in the team working on specific aspects of project. Teams need to be defined carefully when developing safety-critical systems in terms of role identification especially when adapting agile approach for development. Agile culture is very different from safety-critical system's development team's culture. Some common grounds need to be defined to have a smoothly functioning team. Agile supports rapid development which makes it harder to have formal communication mechanisms in place, whereas safety-critical systems need formal ways of communication.

2.3 Complementary Teams

2.3.1 *Problem.* "What traits should an agile team have for developing safety-critical systems?"

2.3.2 *Solution.*

The first point to be considered is team size. It should be kept small ideally 10-15 people on each team. Every member of the team must have prior experience of safety-critical systems in terms of development. If agile is new to them there must be a workshop or training to have them on board with the idea. As for teams using traditional approaches for development this can be entirely new way of looking at things. There must be at least one person in each team with prior experience of working with agile.

2.3.3 *Consequences.*

(+) Having certain qualities in each member increases the chances of having an efficiently working team.
(-) Different cultural and work backgrounds can create issues.

2.4 Feature Based Teams

2.4.1 *Problem.* "How to create an agile team best suited for safety-critical system's development?"

2.4.2 *Solution.*

Create software according to its features. An ideal team should have people from all phases of development life cycle. In other words a single team should have requirement engineer, developer, tester, quality assurance person and team leader. According to the project in hand more than one person can be assigned to perform any specific task, for example there can be more than one tester on a team. The key point is everyone should be on board from day one. It should be a mutual decision to decide which features should be build first and which should be dealt later on. Some people can shuffle the roles e.g for any one particular feature a team member can be both a requirement engineer and developer, for second feature may be he is requirement engineer and tester. Roles assigned to every member of team would remain same until one feature is developed and deployed successfully. Creating one feature can take more than one iteration of SDLC (software development life cycle). Another aspect is to include people who are skilled enough to juggle their roles when needed.

2.4.3 *Consequences.*

(+) More insight into ongoing development for management.
(+) Team is cross-functional hence a small team performing all activities. (+) process is highly adaptive to updates or changes requested.
(-) The process is completely dependant on members of team, if anyone falls short there is a problem.

2.5 Tool Aided Communication in Teams

2.5.1 *Problem.* "How to achieve fast and reliable communication among team?"

2.5.2 *Solution.*

Team should use some tool for documenting requirements and progress during development. This should be a more formal step than writing user stories, as is normally done in agile development. Since safety-critical systems need to have an extensive documentation about details of each step. Using a tool can help the team in two ways, they can communicate faster, they can put daily log of tasks assigned to them and tasks completed, this should be done on daily basis. Work assigned can be on weekly or bi-weekly basis but every member of team should log his activity for that particular day on daily basis. Project manager can create chapters based on features of system. Each chapter can have many iteration cycles, numbering or codes can be used to identify these iterations. Using codes is more efficient as it increases the efficiency of communication, code can represent feature and

iteration cycle. For each feature, roles of members will remain same, but in every iteration there will be different outputs required from team. Usage of tool makes it easier to communicate, integrate and update the work of team in a time efficient manner. Also if there is any change of plans every one can be informed quickly and people can give their feedback on proposed change at collective platform. As agile proposes informal meetings with no formal documentation, this approach can stay in contact if outcome of meetings and processes adapted during development are gathered at collectively shared electronic platform. This will save the redundancy of work and people will be notifying and communicating in real time. However, any change in requirements or any kind of failure must be reported and approved by project manager before deciding on taking any counter measures. Client or User should be incorporated in meeting at start of every iteration, however once the iteration has begun it should be development team only communicating to each other.

2.5.3 *Consequences.*

(+) Better collaboration and communication among team.
(+) Using a tool makes communication faster and trackable.
(+) Each member of team feels increased responsibility.
(+) Better chances of improvement as continuous communication is maintained within team.
(-)Cultural differences of people can be hindrances in communicating and understanding each other which is frequently required.

2.6 Known Uses

NASA switched to agile software development for their mission called Orion Program. It was a human-rated mission with great complexity. They argue that the objectives of critical missions demand a new way of developing these systems and gave them motivation to adapt agile for this particular project. They indicated the short term interactions with deployment after every iteration and effective communication among major benefits. They used almost the similar approach as proposed in this paper for teams and communication. They argue that enhanced communication and meetings benefited the project to a great extent. This enabled the work progress smooth and made it easy to deal also they were able to take suggestions for improvement and incorporate them in more effective way as compared to traditional approaches. They further reported that this method of communication provided a clear picture of daily operations to higher management and based on this fact planning of responsibilities was made easy. Also the accountability measures were taken more effectively. They highly recommended it for projects who are having issues in traditional approach to create safety-critical systems, especially in terms of communication [Smith et al. 2019].

Gupta et al [Gupta and Reddy 2016] reported the journey of adopting agile methodologies for a project called Global Configuration Project (GCP). The team was using traditional approaches for building critical systems before this project. Here they adapted Scrum to be more specific. They particularly worked on creating agile teams, which were named as team 1, team 2 etc. Each team had people from different sections of development on board, they had tester, developer and client in every team. They created multi skilled and cross functional teams and communication was done as described by agile. They reported that this structure worked very well for them in terms of enhanced communication and shared ownership. This made whole team more responsible for their work. They used a digital tool (digital board) to document details of meetings which gave them "excellent environment for collaboration" as they quote. They further argue that they received positive feedback from customer also by switching to agile methods. They measured improvement in communication through a survey which reported positive feedback on the method.

3. UP-FRONT TESTING FOR SAFETY-CRITICAL SYSTEM

In this section we present two patterns for testing of safety-critical systems. These patterns share the same context and forces.

3.1 Context

Failure of safety-critical system can result in loss of lives, loss of huge financial amount or can be a threat to environment. In the light of criticallity of software, it is highly required to develop testing techniques that can give maximum test coverage and hence ensure that the system is safe. The approach to develop tests upfront have gained huge popularity among practitioners. This strategy proposes that tests should be written well in advance, as soon as requirements are defined. Tests must be written to fulfill those requirements[Janzen and Saiedian 2005]. Safety-critical systems are complex and there is need of rapid development in current time. Developers are implementing approaches that support the idea of testing in this context [Hause et al. 2010].

3.2 Forces

Agile promotes rapid development but safety-critical standards need prolonged and detailed procedures for testing. The concept of up-front testing is very much inline with agile principles and hence can be incorporated to build safety-critical systems in an agile way. Safety-critical systems need rigorous testing which is time consuming. [Holcombe et al. 1995] States that formal verification methods and test driven development can be used together and there can be many enhanced benefits of this approach. It can ensure safety of systems along with reliability.He further argues that safety-critical systems have to comply to certain safety standards, currently there are not many techniques of testing that can fit into this scenario.

3.3 Feature Based Testing

3.3.1 *Problem.* "How to inform developers about verification's required for safety-critical system, as early as possible?"

3.3.2 *Solution.*

Include testing resources in the requirements elaboration so there is a plan from the beginning on how to test. Each feature should have its own defined tests in the beginning. The team should identify the requirements concerning one particular feature of the system, then create iterations required to completely develop that feature, covering all the listed requirements for that feature. Consider safety standards and break safety standards into safety requirements with traceability to testing and test results. As soon as the requirements are finalized, in the start of the project, there must be tests written for every requirement. A requirement will be considered done when it passes its relevant test at end of iteration. This way tests will be designed up-front, before starting any iteration for development of any particular feature.

3.3.3 *Consequences.*

(+)Up-front testing goes hand in hand with safety-critical approach of up-front planning whereas, it reduces the time for testing which is required by agile principles.
(-)The approach is highly dependant on clearly and correctly defined requirements which is a challenging task on its own.

3.4 Requirement Changes and Testing

3.4.1 *Problem.* "How to handle testing for changed requirements during development of safety-critical systems?"

3.4.2 *Solution.*

In case there is any change in requirements, all the tests, directly or indirectly relevant to that requirement will be repeated to consider it "DONE" again. When more than one feature is developed, there will be tests at feature level too. All features must be tested individually and as whole after deploying any new feature. Tests for previous features will be automated whereas, new one will be tested manually. It will ensure that new functionality has

not effected any of the previously properly working features. This way the tests will increase as we move to the completion of software, making sure that each requirement and each feature is tested. Once tests are designed and performed, they can be automated for next cycles of testing, it will save time and effort of developers. This will provide a middle ground for agile and safety-critical principles to work together without neglecting any important aspect of the system. Still there is need of manual testers even after good automated test coverage. Incorporate security into the requirements backlog. This can be done in two ways: associate risk and failure modes and safety into the attributes of each requirement, and have safety requirements, both should be done in parallel. They are not alternatives to each other instead these are parallel running processes. They will operate the HIL (hardware in the loop testing) and they need to be domain people. They need to be able to poke holes in the operation of the system (and these guys are super to also be involved in the requirements!!).

3.4.3 *Consequences.*

(+) This approach will provide rigorous testing in comparatively less time frame as traditionally consumed for testing in safety-critical systems.
(+)This approach makes it easy to come up with automated tests once they are designed and performed for first time, it will save cost and effort of developers without compromising quality aspects.
(-)The approach is completely dependant on humans so inherently it is prone to errors and dependant on their skills.

3.5 Known Uses

The approach of doing up-front testing was adopted by Ericsson, they used an automation tool for the process. They used a typical Test Driven Approach along with automated testing tool. They did it at component level by exchanging XML data in the place of methods and classes. This methodology made it easier for them to automate tests. Their team reported that project's deployment time will be shortened with every new version of the project that uses the same tool and technique. They further added that it has decreased fault rates significantly.[Damm et al. 2005]

In this case study the approach of up-front automated tests is applied at IBM. They created automated tests after creating UML design. The team at IBM changed their practices from ad-hoc processes to test driven development. The team was not experienced with this methodology still they reported positive results. They particularly mentioned there was reduction in density of defects in new or changed code, as compared to experienced team who used ad-hoc processes. Another huge benefit with test driven development was its reuseability. So these tests became an asset of organization and further will be used again in other projects to enhance their quality without going through all the effort. They reported 40 percent less defects during testing and verification processes as compared to the product developed through traditional ways [Williams et al. 2003].

4. CONCLUSIONS

Patterns relevant to teams give a direction to use agile software development methodologies for safety-critical systems. They focus on team building and communication among that team in an agile way. We propose that agile can give much faster and efficient ways of communication, better insight to projects and enhanced team collaboration by our proposed method. This can be used in collaboration with traditional software development approaches for safety-critical systems for creating a hybrid approach that will result in fast paced work and better communication.

Testing patterns in this paper support the approach of test driven development for producing safety-critical systems. We have proposed the patterns that can be used with team patterns or can be used separately. Testing patterns provide a way to use the approach by keeping safety standards intact and pave a way to encourage the use of agile methods for development of safety-critical systems. This provides rigorous testing which is mandatory

for safety-critical systems in a time and cost efficient way, which is the basic principle of agile. In a nutshell, it provides a middle ground to use agile approach while honouring the rules of safety standards.

REFERENCES

Agile Alliance. 2001. Manifesto for agile software development. (2001).

Barry Boehm. 2002. Get ready for agile methods, with care. *Computer* 35, 1 (2002), 64–69.

Jiyao Chen, Donald O Neubaum, Richard R Reilly, and Gary S Lynn. 2015. The relationship between team autonomy and new product development performance under different levels of technological turbulence. *Journal of Operations Management* 33 (2015), 83–96.

Lars-Ola Damm, Lars Lundberg, and David Olsson. 2005. Introducing test automation and test-driven development: An experience report. *Electronic notes in theoretical computer science* 116 (2005), 3–15.

Rajeev Kumar Gupta and Prabhulinga Manik Reddy. 2016. Adapting agile in a globally distributed software development. In *2016 49th Hawaii International Conference on System Sciences (HICSS)*. IEEE, 5360–5367.

Matthew Hause, Andrew Stuart, David Richards, and Jon Holt. 2010. Testing safety critical systems with SysML/UML. In *2010 15th IEEE International Conference on Engineering of Complex Computer Systems*. IEEE, 325–330.

Rashina Hoda, James Noble, and Stuart Marshall. 2012. Self-organizing roles on agile software development teams. *IEEE Transactions on Software Engineering* 39, 3 (2012), 422–444.

Mike Holcombe, Florentin Ipate, and Andreas Grondoudis. 1995. Complete functional testing of safety critical systems. *IFAC Proceedings Volumes* 28, 25 (1995), 199–204.

David Janzen and Hossein Saiedian. 2005. Test-driven development concepts, taxonomy, and future direction. *Computer* 38, 9 (2005), 43–50.

Ana Isabella Muniz Leite. 2017. An Approach to Support the Specification of Agile Artifacts in the Development of Safety-Critical Systems. In *2017 IEEE 25th International Requirements Engineering Conference (RE)*. IEEE, 526–531.

Orla McHugh, Kieran Conboy, and Michael Lang. 2011. Agile practices: The impact on trust in software project teams. *Ieee Software* 29, 3 (2011), 71–76.

Mary Walowe Mwadulo. 2016. Suitability of Agile Methods for Safety-Critical Systems Development: A Survey of Literature. (2016).

Justin Smith, John Bradbury, Will Hayes, and Wes Deadrick. 2019. Agile Approach to Assuring the Safety-Critical Embedded Software for NASA's Orion Spacecraft. In *2019 IEEE Aerospace Conference*. IEEE, 1–10.

Tor Stålhane-IDI. Safety standards and Scrum–A synopsis of three standards. In *Nbl. SintefNo*.

Stavros Stavru. 2014. A critical examination of recent industrial surveys on agile method usage. *Journal of Systems and Software* 94 (2014), 87–97.

Viktoria Stray, Nils Brede Moe, and Rashina Hoda. 2018. Autonomous agile teams: challenges and future directions for research. In *Proceedings of the 19th International Conference on Agile Software Development: Companion*. 1–5.

Nigel James Tracey. 2000. *A search-based automated test-data generation framework for safety-critical software*. Ph.D. Dissertation. Citeseer.

Laurie Williams, E Michael Maximilien, and Mladen Vouk. 2003. Test-driven development as a defect-reduction practice. In *14th International Symposium on Software Reliability Engineering, 2003. ISSRE 2003.* IEEE, 34–45.

Fabian Zimmermann, Robert Eschbach, Johannes Kloos, and Thomas Bauer. 2009. Risk-based statistical testing: A refinement-based approach to the reliability analysis of safety-critical systems.