AARHUS UNIVERSITY

Coversheet

This is the accepted manuscript (post-print version) of the article.

Contentwise, the post-print version is identical to the final published version, but there may be differences in typography and layout.

How to cite this publication

Please cite the final published version:

Nielsen, P. A., & Tordrup Heeager, L. (2017). The Dynamics of Agile Practices for Safety-Critical Software Development: Short Paper. I *Proceedings of the XP2017 Scientific Workshops* Association for Computing Machinery. DOI: 10.1145/3120459.3120481

Publication metadata

Title:	The Dynamics of Agile Practices for Safety-Critical Software Development
Author(s):	Nielsen, P. A., & Tordrup Heeager, L.
Proceedings:	<i>Proceedings of the XP2017 Scientific Workshops</i>
DOI/Link:	10.1145/3120459.3120481
Document version:	Accepted manuscript (post-print)

General Rights

Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognize and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal

If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

The Dynamics of Agile Practices for Safety-Critical Software Development

Short Paper

Peter Axel Nielsen
Department of Computer Science
Aalborg University
Denmark
pan@cs.aau.dk

Lise Tordrup Heeager
Department of Management
Aarhus University
Denmark
lith@mgmt.au.dk

ABSTRACT

This short paper reports from a case study of the agile development of safety-critical software. It utilizes a framework of dynamic relationships between agile practices with the purpose of demonstrating the utility of the framework to understand a case in its context, and it shows significant dynamics. The study is concluded by pointing at which further research on the framework is required to use the framework in managing the agile development of safety-critical software.¹

CCS CONCEPTS

• **Software and its engineering** → **Agile software development** • **Software and its engineering** → **Software safety**

KEYWORDS

Agile software development, safety-critical software, agile processes, dynamism.

1 INTRODUCTION

We use safety-critical software and embedded software in abundance. For example, when diabetes patients use a digital insulin pump to control and fine-tune the level of blood

glucose, it is a safety-critical software system [18]. The benefits of these new devices are high, but the safety risks cannot be taken lightly. Other safety-critical software systems include railway signalling, self-driving cars, airbags, elevator control, flight control, medical diagnosis, early detection of pathogens, radiation therapy, defibrillators, etc. [1, 31].

The severity of potential safety fails has led to the institutionalisation of approval procedures and certification. In the USA the US Food and Drug Administration (FDA), for example, approves medical devices such as a digital insulin pump [22]. Other countries have similar agencies to approve and certify safety-critical products. There are several ISO standards [15], process models and process maturity models, [6, 19, 20] that one way or the other are part of stipulating how safety-critical systems must be developed. EN 50128, for example, is a European standard for the development of railway applications [21]. The development of these safety-critical products is highly regulated. The concern for risks and the well-being and lives of users subjects the products and their development processes to legislation, public interest, concerns of patients, consumers and citizens, and the responsibilities of the producers.

The traditionalist answer to how to develop safety-critical software has been to establish a formal development process molded over experience from project management and quality management [3, 33, 38]. The general process model is the waterfall model where the underlying idea is rational, i.e., first we think, and then we do. The traditionalist thinking was that this allows for thinking carefully about the system's features and how it will be used, and most importantly allow reasoning about system properties such as reliability, resilience, safety, and risk mitigation.

Agile development processes [7, 9] have been designed to alleviate problems with the rational processes by a careful experimentation process. The key features of agile development processes are that they contribute to the creation of change, that they embrace change, and that change in requirements should be taken to be a positive development. Many view this as counter to the processes for developing safety-critical systems. Others seek to find a path to connect.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from Permissions@acm.org.
XP '17 Workshops, May 22-26, 2017, Cologne, Germany
© 2017 Association for Computing Machinery.
ACM ISBN 978-1-4503-5264-2/17/05...\$15.00
<https://doi.org/10.1145/3120459.3120481>

In this paper, we present a case study to illustrate such a path. We analyse a case using a framework of relationships and show that these relationships are dynamic in nature. The implication is that managers and developers need to be aware of the dynamics and seek control rather than being pushed around by circumstances.

2 RELATED RESEARCH

The following is just a brief presentation on the topic as a short paper leaves little space. First, we will briefly present some key conjectures in existing research on the agile development of safety-critical software. Second, we will present a framework from a literature review of a large analysis of the existing research. The framework will be used as the analytical lens in the case analysis.

2.1 Agile Development of Safety-Critical Software

The understanding of agility in software development we subscribe to in this paper is by [9], namely “continual readiness ... to rapidly or inherently create change, proactively or reactively embrace change, and learn from change.” The understanding of what safety-critical software is stems from both [33] as systems that may “result in injury to people, damage to the environment or extensive economic loss” and from [7] making the distinction between loss of life, loss of essential money, and loss of comfort.

Several papers conclude that safety-critical systems can be developed using an agile method, e.g., [1], that agile practices do not contradict regulatory requirements [26], and that due to substantial compatibility, it is worthwhile to attempt adopting agile methods in safety-critical software development [30]. Agile methods and practices have been adopted in safety-critical software development for some years [23, 28]. Several studies on organisations using a variety of agile practices when developing safety-critical software, document that agility can be increased in safety-critical software development. Some even propose that agile methods are better suited for safety-critical software development and that using agile methods is advantageous [31]. However, most papers also conclude that to fulfil the regulatory requirements using agile methods; the agile methods need to be tailored, e.g., [11], and that there are several challenges in tailoring agile methods to safety-critical development, e.g., [5, 28]. Due to these challenges, it has been argued that agile methods should be regarded as complementary to plan-driven practices instead of being the replacement [13]. A similar conclusion is also reached by Boehm and Turner [4] when they take criticality to be one of five key dimensions to discriminate between plan-driven and agile processes; and again, the more critical the software, the more appropriate they claim it will be to choose a plan-driven process.

2.2 Framework

A recent literature review (2017) of 54 research articles published in leading conferences and journals [17] analyses how safety-critical software was or could be developed in an agile manner. The literature review is summarized in a framework depicted in Figure 1.

Figure 1: Framework of agile dimensions in development of safety-critical software, adapted from [17]

The framework shows the challenges of agile development of safety-critical software [17] concerning four areas of concern and five relationships found in the literature, cf. Figure 1. The four areas of concern found in the literature are:

- **Light documentation:** agile methods cannot abandon documentation entirely as it serves the purpose of proving that the software is safe, e.g., [25, 32], yet it should be kept at a minimum, e.g., [7].
- **Flexible requirements:** changes to requirements may invalidate safety arguments [12], yet safety requirements may be more stable [34] than other requirements.
- **Iterative & incremental lifecycle:** the key to embracing change either by being ready to iterate (rework) or splitting into increments [7, 8] is paved with barriers from the safety-critical challenge [25, 29] which perhaps is best met by incremental safety assurance [13].
- **Test-first strategy:** widely recommended in agile development [27] and it can be practiced for safety-critical software as well [10], but it is also less sophisticated than what is required [13, 21].

The framework, cf. Figure 1 and [17], focuses even more on five relationships (five, not six, are found in the literature):

- **Traceability of requirements:** is difficult as both requirements and even light documentation are in a state of change [24, 25].

- Securing safety with flexible requirements: hazard analysis conflicts with iterative processes [12, 16], but it may be possible to separate safety-critical requirements from other requirements [36].
- Iterative testing: using test-first can go well together with iterative processes as reported in [35], but longer iterations may be necessary [32], and testing in the target may be more difficult [30].
- Light test documentation: extensive testing of safety-critical software is at odds with light documentation [31], yet automated tests may be part of the documentation [14]
- Iterative and incremental validation & verification of safety: safety documentation may be produced incrementally [1, 21] and it may come with some difficulty as safety arguments often concern a whole system [12].

It is suggested in the framework that the areas of concern are mutually dependent and if one area is changing then it has implications for the other areas of concern through the relationships. The dynamics suggest that as we try to improve or change one area of concern of the agile development of safety-critical software this cannot be done without influencing or disturbing the area of concerns. It is also suggested that these dependencies are specific, for example, trying to change the *lightness* of documentation influences *flexibility* of requirements through *traceability*. It is this dynamism we will study in more detail through the case.

3 RESEARCH DESIGN

The research focus in this paper is on how we can connect agile software development with developing safety-critical systems. This can appropriately be studied as a contemporary phenomenon through a case study [2, 37]. This entails studying the phenomenon in its organisational context and with a clear unit of analysis. The case organisation is a pharmaceutical company that develops drugs for a specialised market, and the company also develops and manufactures the drug injection devices. In a recent project involving more than 100 developers, engineers and physicians and several diverse competencies in clinical aspects, mechanics, hardware, and software. The software team was small relative to the overall project organisation. The software team was agile while the overall project that it was part of was plan-driven.

The software team member, the software project manager, and managers of other subprojects, as well as the risk mitigation manager, were interviewed several times over a period of 8 months. At several occasions, one of the researchers was a participant observer covering design meetings, scrum meetings, and testing sessions. The interview and observation data were recorded and later transcribed and analysed in detail.

4 CASE ANALYSIS

The pharmaceutical case company developed a clinical device for dispensing drugs, for processing information, and for the patient to interact with to control the device. The project was large with 100+ developers. The software team had on average 17 developers, and the software project manager had close working relations with the other subprojects. Other subprojects had responsibilities covering: hardware, mechanics (including actuators and plastic casing and tubes), risk assessment and mitigation, and clinical development and testing. The software team had two separate key contributions to the overall project: software for a remote control to the dispenser and software for a controller to the actuator part of the dispenser.

It had been decided by the software project manager that the software team should utilise agile methods. The reason was that despite the overall requirements specification remained stable for a long time (more than two years) there were conveyed additional, and more detailed requirements to the software team and the set of software requirements was not stable. The overall project had to remain compliant with the regulatory requirements of which several applied directly to the software development processes, e.g., FDA process requirements as outlined in [22]. The software team maintained two faces: on the one side utilising Scrum and test-first strategies fully, and on the other side be compliant with FDA regulations.

The analysis of the case data has shown dynamics which we can get a better understanding of through the framework in Figure 1. The analysis starts with the top corner 'flexible requirements'. The work with the requirements showed varying degrees of flexibility. The overall requirements specification of the complete product was stable over a period of three years. That level of the specification was difficult to change as it required: top management decision-making, arguing against top management, and commitment and signature by more than 70 stakeholders company-wide. For that reason, the product requirement specification was viewed by the whole of the project as a framing within which all details should be developed. By implication, the software requirements had to be flexible to cater for problems and solutions arising outside the software. For example, with a risk analysis leading to well above 1000 product risks then some of these risks would have to be mitigated in the software thus leading to new or changed software requirements. Another example is that the hardware specified in the product specification had to be changed as the hardware technologies advanced over the three years and again leading to changed software requirements. The flexible software requirements led to a problem with the status of the software requirements documentation which we will come back to.

The flexible software requirements also led the software project manager to adopt an agile development process – that is the right corner in Figure 1 labelled 'Iterative & incremental

lifecycle’ – and here the first attempt at adopting an agile process happened through the utilization of Scrum to organize the backlog, to organize the development work, and to measure progress in a burndown chart. The process became iterative and included much rework of functionality. At this point, there was no concern for an incremental approach. The software team was fully occupied with keeping up the pace in 3-4 week sprints.

Gradually, the software team assisted by an agile coach began adopting a test-first strategy – the bottom corner of Figure 1. This was a wholly new way of working and the coach was used extensively to set up testing tools and automatic testing. This fitted well with the highly iterative nature of the development work as the tests (which did not have to be changed) proved to be very efficient. It is fair to state, that they would not have considered a test-first strategy had they not changed to an iterative process like Scrum. They would very likely have stayed with a traditional V-model as that fitted much better with the FDA process regulations.

At this point we have seen the following influencing changes:

1. Fixed, stable product requirements → Flexible software requirements
2. Flexible software requirements → Iterative process adopted
3. Iterative process adopted → Test-first strategy

This seems almost straightforward, but it became more complicated. The test-first strategy was only partly implemented. The unit testing was performed by the software developers, test cases were designed in connection with the programming, and automated regression testing was in place. Each piece of delivered software was also reviewed before delivered to the systems engineering group. This was fairly easy to document, and the collection of test cases was relatively easy to maintain despite the full coverage. Another part of the testing was more difficult. The regulatory requirements also led the systems engineering group to perform their tests at the level of the product as a whole. This was in part caused by the regulatory requirements and partly caused by the systems engineering group working in a plan-driven manner. This led to some delays in the testing procedures. While the unit testing was performed immediately in relation to the programming gaining most of the benefits of a test-first strategy with easy debugging and less rework; then the testing for integration into the hardware was performed later when the systems engineering group had it on their agenda. When they then eventually discovered errors these were reported in a bug tracking system, communicated to the developers, which in turn would address the debugging.

At the same time, the regulatory conditions for the documentation led the software project manager to great anguish. The strongly iterative nature of the software

development made it laborious to keep up with the documentation however light it was meant to be. The software project manager decided because of this to break the software development into more clearly defined increments rather succumbing to the less controlled iterativeness where it was less clear when a piece of software was providing a new fragment, or it was a rework of a previous piece of software. The stronger incremental development was less troublesome to deal with in providing documentation of the testing (both the unit testing and the integration testing).

We have now seen the additional influences between areas of concern in the framework, cf. also Figure 2:

4. Test-first strategy → Light documentation of unit testing
5. Difficult documentation of integration testing → Mix of test-first and test-later strategies
6. Troubled lightness of documenting iterations → reinforced increment definitions
7. Reinforced incremental process → More efficient documentation

The last two influences between areas of concern occurred as follows. The software team knew from experience that the software requirements had to be flexible, and they had worked with that all the way around the diamond of areas of concern. They also knew that they could not and should not develop full documentation of the requirements when they were still in the process of changing. Their documentation was, therefore, light, peer reviewed, but light. The idea was that when the product requirements specification had been modified, they would then update the software requirements to a full version in compliance with the regulatory requirements (influence 8 in Figure 2). At a much later point in time, this eventually led the overall project management to realise that the product requirements specification had to be changed however difficult that may have been (influence 9 in Figure 2).

Figure 2: The dynamics of influences in the unfolding of how areas of concern and their practical solutions

5 DISCUSSION AND CONCLUSIONS

We suggest that the case is an example of how dynamic the adoption of agile processes in developing safety-critical software can be – and perhaps often is. The case analysis shows how a concern in one corner can be solved and that it sometimes leads to necessary changes in another corner. This is outlined at a general level in Figure 1 and the framework, and it is demonstrated in a specific case in Figure 2.

We suggest further that this framework, as we have illustrated, will provide a more clear understanding of what happens when introducing agile process in safety-critical development. This goes well beyond the idealisation found in the existing guidelines on how to balance the plan-driven and the agile, e.g., in [4]. We also take the liberty to infer that our approach as we have outlined it here is addressing actual practical concerns and not only a theoretical view [27] of what might happen.

To keep this paper short we only conclude that with this case study we have shown the potential usefulness of the framework in [17]. Much research on this is still to come, and here we think that other studies should validate the findings we have suggested here and whether the framework is also perhaps useful as a managerial device in making decisions

about how to address the agile process issues of developing safety-critical software.

ACKNOWLEDGMENTS

We wish to thank the participants in the software team in the pharmaceutical company. For reasons of anonymity we cannot name the participants, but we are nevertheless thankful for their willingness to share their views and explain their practices.

REFERENCES

- [1] Abdelaziz, A.A., El-tahir, Y. and Osman, R. 2015. Adaptive Software Development for Developing Safety Critical Software. *International Conference on Computing, Control, Networking, Electronics and Embedded Systems Engineering (ICCNEEE)* (2015), 41–46.
- [2] Benbasat, B.I. 1987. The Case Research Strategy in Studies of Information Systems Case Research: Definition. *MIS Quarterly*. September (1987), 369–387.
- [3] Boehm, B. and Ross, R. 1989. Theory-W Software Project Management: Principles and Examples. *IEEE Transactions on Software Engineering*. 15, 7 (1989), 902–916.
- [4] Boehm, B. and Turner, R. 2003. *Balancing Agility and Discipline: A Guide for the Perplexed*. Addison-Wesley Professional.
- [5] Cawley, O., Wang, X. and Richardson, I. 2010. Lean / Agile Software Development Methodologies in Regulated Environments – State of the Art. *Lean Enterprise Software and Systems* (Heidelberg, 2010), 31–36.
- [6] Chrissis, M.B., Konrad, M. and Shrum, S. 2003. *CMMI: Guidelines for Process Integration and Product Improvement*.
- [7] Cockburn, A. 2006. *Agile software development: the cooperative game*. Pearson Education.
- [8] Cohn, M. and Ford, D. 2003. Introducing an agile process to an organization. *Computer*. 36, 6 (2003), 74–78.
- [9] Conboy, K. 2009. Agility from First Principles: Reconstructing the Concept of Agility in Information Systems Development. *Information Systems Research*. 20, 3 (Sep. 2009), 329–354.
- [10] Drobka, J., Noftz, D. and Raghu, R. 2004. Piloting XP on four mission-critical projects. *IEEE Software*. 21, 6 (2004), 70–75.
- [11] Fitzgerald, B., Stol, K.-J., O'Sullivan, R. and O'Brien, D. 2013. Scaling Agile Methods to Regulated Environments: An Industry Case Study. *Proceedings of the 2013 International Conference on Software Engineering* (2013), 863–872.
- [12] Ge, X., Paige, R.F. and McDermid, J.A. 2010. An iterative approach for development of safety-critical software and safety arguments. *Proceedings - 2010 Agile Conference, AGILE 2010*. (2010), 35–43.
- [13] Gorski, J. and Lukasiewicz, K. 2012. Assessment Of Risks Introduced To Safety Critical Software By Agile Practices — A Software Engineer's Perspective. *Computer Science*. 13, 4 (2012), 165–182.
- [14] Grenning, J. 2001. Launching extreme programming at a process intensive company. *IEEE Software*. 18, 6 (2001), 27–33.
- [15] Heeager, L.T. 2012. Introducing Agile Practices in a Documentation-Driven Software Development Practice: a Case Study. *Journal of Information Technology Case and Application Research*. 14, 1 (2012), 3–24.
- [16] Heeager, L.T. and Nielsen, P.A. 2009. Agile Software Development and its Compatibility with a Document-Driven Approach? A Case Study. *Australasian Conference on Information Systems* (Melbourne, 2009), 205–214.
- [17] Heeager, L.T. and Nielsen, P.A. 2017. *Agility in Development of Safety-Critical Software: A Conceptual Model*. Working paper, Aalborg University. <http://bit.ly/2tzjxZE>.
- [18] Heinemann, L., Fleming, G.A., Petrie, J.R., Holl, R.W., Bergenstal, R.M. and Peters, A.L. 2015. Insulin pump risks and benefits: A clinical appraisal of pump safety standards, adverse event reporting, and research needs. *Diabetes Care*. 38, 4 (2015), 716–722.
- [19] Humphrey, W.S. 1988. Characterizing the Software Process: A Maturity Framework. *IEEE Software*. 5, 2 (1988), 73–79.
- [20] Humphrey, W.S. 1989. *Managing the software process*. Addison-Wesley.

- [21] Jonsson, H., Larsson, S. and Punnekkat, S. 2012. Agile practices in regulated railway software development. *Proceedings - 23rd IEEE International Symposium on Software Reliability Engineering Workshops, ISSREW 2012*. (2012), 355–360.
- [22] Lin, W.L.W. and Fan, X.F.X. 2009. Software Development Practice for FDA-Compliant Medical Devices. *2009 International Joint Conference on Computational Sciences and Optimization*. 2, (2009), 388–390.
- [23] McCaffery, F., Trektore, K. and Ozcan-Top, O. 2016. Agile–Is it Suitable for Medical Device Software Development? *International Conference on Software Process Improvement and Capability Determination* (2016), 417–422.
- [24] McHugh, M., McCaffery, F. and Casey, V. 2012. Barriers to using agile software development practices within the medical device industry. *EuroSPI 2012* (2012), 1–7.
- [25] McHugh, M., McCaffery, F. and Casey, V. 2012. Barriers to Adopting Agile Practices when Developing Medical Device Software 2 Challenges to Developing Medical Device Software. *International Conference on Software Process Improvement and Capability Determination* (Heidelberg, 2012).
- [26] McHugh, M., McCaffery, F. and Coady, G. 2014. An agile implementation within a medical device software organisation. *Communications in Computer and Information Science*. 477, (2014), 190–201.
- [27] Nerur, S. and Balijepally, V.. 2007. Theoretical reflections on agile development methodologies. *Communications of the ACM*. 50, 3 (2007), 79–83.
- [28] Notander, J.P., Host, M. and Runeson, P. 2013. Challenges in Flexible Safety-Critical Software Development - An Industrial Qualitative Survey. *Product-Focused Software Process Improvement*. 7983, (2013), 283–297.
- [29] Paige, R.F., Charalambous, R., Ge, X. and Brooke, P.J. 2008. Towards Agile Engineering of High-Integrity Systems. *SAFECOMP* (Berlin, 2008), 30–43.
- [30] Paige, R.F., Chivers, H., McDermid, J.A. and Stephenson, Z.R. 2005. High-integrity extreme programming. *Proceedings of the 2005 ACM symposium on Applied computing - SAC '05*. (2005), 1518.
- [31] Rasmussen, R., Hughes, T., Jenks, J.R. and Skach, J. 2009. Adopting agile in an FDA regulated environment. *Agile Conference, AGILE 2009*. (2009), 151–155.
- [32] Rottier, P. a. and Rodrigues, V. 2008. Agile Development in a Medical Device Company. *Agile 2008 Conference*. (2008), 218–223.
- [33] Sommerville, I. 2015. *Software Engineering*. Pearson.
- [34] Stålhane, T., Myklebust, T. and Hanssen, G. 2012. The application of safe scrum to IEC 61508 certifiable software. *11th International Probabilistic Safety Assessment and Management Conference and the Annual European Safety and Reliability Conference 2012, PSAM11 ESREL 2012*. 8, (2012), 6052–6061.
- [35] VanderLeest, S.H. and Buter, A. 2009. Escape the Waterfall: Agile for Aerospace. *IEEE/AIAA 28th Digital Avionics Systems Conference* (2009).
- [36] Wolff, S. 2012. Scrum Goes Formal : Agile Methods for Safety-Critical Systems. *Proceedings of the First International Workshop on Formal Methods in Software Engineering: Rigorous and Agile Approaches* (Zurich, 2012), 23–29.
- [37] Yin, R.K. 2003. Case Study Research: Design and Methods. *SAGE Publications*.
- [38] Zultner, R.E. 1993. TQM for technical teams. *Communications of the ACM*. 36, 10 (1993), 79–91.