Post-print of: Katarzyna Łukasiewicz and Janusz Górski. 2018. Introducing agile practices into development processes of safety critical software. In Proceedings of the 19th International Conference on Agile Software Development: Companion (XP '18). Association for Computing Machinery, New York, NY, USA, Article 13, 1–8. DOI: https://doi.org/10.1145/3234152.3234174

Introducing Agile Practices Into Development Processes of Safety Critical Software

Katarzyna Łukasiewicz
Department of Software Engineering
Gdańsk University of Technology
Poland
katarzyna.lukasiewicz@pg.edu.pl

Janusz Górski
Department of Software Engineering
Gdańsk University of Technology
Poland
janusz.gorski@pg.edu.pl

Abstract

In this paper we present AgileSafe – a method which supports introduction of agile practices into safety-critical software development processes. To represent safety assurance constraints resulting from the regulatory context, AgileSafe uses assurance case patterns. The knowledge base of AgileSafe helps the user to select the agile practices relevant for the considered software development project. The corresponding assurance case patterns define the scope of the evidence to be collected to demonstrate that the project meets its safety constraints. The overview of the method is presented with reference to a case study - a project for continuous glucose monitoring-enabled insulin pump system.

CSS Concepts

Keywords

Agile development, Safety constraints, Safety-critical system, Safety arguments

ACM Reference format:

Katarzyna Łukasiewicz and Janusz Górski. 2018. Introducing Agile Practices Into Development Processes of Safety Critical Software. In *Proceedings of XXXX (XXXX). ACM, New York, NY, USA, 7 pages.* https://doi.org/10.1145/1234567890

1. INTRODUCTION

With an intensive progress and expansion of technology in the 20th and 21st centuries, devices and solutions in many domains have become increasingly software reliant. The safety-critical software domain will likely expand in the future, with dropping cost of hardware and new possibilities offered by the software [1]. As such, the importance of safety assuring solutions will grow, especially with time and cost playing an increasingly important role in this domain.

While plan-driven methodologies have proven their value and usefulness in safety-critical projects, the evolving market of software products of the last decade showed they could be considered as being too restrictive in some circumstances [2]. This is the case, in particular, while dealing with volatile requirements and ever-changing market demands. In such situation, heavyweight documentation and low flexibility associated with plan-driven approaches could have an impeding effect on a software development process [3], [2]. The need to deliver systems of acceptable quality, faster and at lower cost in comparison to competitors evoked seeking an alternative [4].

In response to these concerns agile methodologies have offered practices which value close relationship with customers, allow more relaxed approach towards documentation and provide a flexible development lifecycle based on short iterations [5]. A growing body of experience shows that successfully combined, agile practices can potentially reduce the cost of production as well as time to market [6], [7].

The objective of this paper is to introduce a new method, called AgileSafe, which supports introduction of agile practices into a software development process while still maintaining the compliance with the software assurance requirements imposed by the application domain. The crucial idea of this method is to employ evidence-based argument templates to explicitly represent the assurance requirements imposed by the safety context of the developed software. Such argument templates, if derived from the regulatory constraints relevant for a given development project, provide for identifying the scope of necessary assurance activities and for gathering the supporting evidence.

2. BACKGROUND

With growing competition in the domain, fast paced changes in technology and clients demanding innovations as well as the highest safety standards, safety-critical software companies are tempted to employ hybrid approaches where agility is combined with necessary safety assurance. As a result, in recent years researchers have been trying to propose such hybrid approaches. Some of the approaches relevant to our research have been described in this section.

XXXX, June, 2018, XXXX

https://doi.org/10.1145/1234567890

In 2009 Weiguo and Xiaomin [8] presented an agile based approach suitable for FDA compliant medical devices projects. The approach has been dedicated to a specific domain thus its applicability is limited. Another relevant approach is AV-Model [9] combining the traditional V-Model with Scrum and focusing on medical device software development and the IEC 62304 standard [10]. While the AV-Model presents some promising solution, its focus as well as potential applications are restrained and as such cannot be universally recommended.

A more comprehensible and practical solution has been proposed by a joint research group of SINTEF [11] and the Norwegian University of Science and Technology [12]. They proposed a method called SafeScrum [13], which concentrates on adapting Scrum into safety-critical software development. The method provides a well justified set of practices although the safety assurance aspect of the method is still a work in progress.

When releasing a piece of safety-critical software, the company responsible for it needs to be able to prove its safety in its target environment. Proving in this context means being able to convince the licencing bodies as well as the potential users that the software is acceptably safe and will not cause harm. In order to do that a sufficient evidence to back claims about safety should be presented. This is where assurance arguments can be of great use. Over a decade ago, Food and Drug Administration (FDA) and Software Engineering Institute (SEI) have begun an in depth analysis of the idea of safety assurance cases [14]. As a result, a series of documents presenting potential uses of assurance cases in FDA certification process have been issued [14], [15]. With the current FDA recommendation of assurance arguments for presenting compliance with safety regulations, this method is increasingly gaining recognition. On the other hand, as the Health Foundation report [16] states, there is little experience in the industry when it comes to the preparation of assurance cases. A need for special training and developing new methodologies in the matter is emerging. Templates and methods facilitating the use of arguments which the manufacturers can relate to can be of great value. This problem attracts the attention of the researchers and projects like AMASS [17] and SafeCer [18] have been focusing on an intelligible and comprehensive certification frameworks.

In AgileSafe we combine both, the guidance on introducing agile practices into a project and the assurance aspect of developing safety critical software,

3. AGILESAFE METHOD OVERVIEW

AgileSafe is a method of incorporating agile practices into critical software development while still maintaining the compliance with the software assurance requirements imposed by the relevant standards in the application domain.

There are two main uses of AgileSafe. The first is *applying* AgileSafe to obtain an advice on software development process, with suggestions on which practices to use and how to assert conformance with selected standards. The second is *improving* AgileSafe by updating the knowledge base of the method.

The high-level use case diagram of AgileSafe is presented in Figure 1.

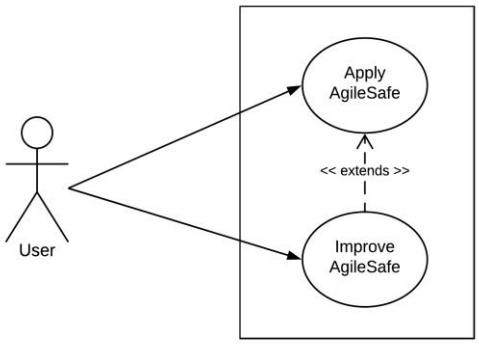

Figure 1 **The high-level use case diagram of AgileSafe**

The *Apply AgileSafe* use case is applied for a specific software development project (the Project) and User is a person or a team with a good knowledge of the Project. User needs to be able to specify the characteristics of the Project as well as to decide upon the final set of selected practices.

In order to follow the *Improve AgileSafe* use case, User should be an expert on agility and in addition a person with good knowledge of the standards and the safety aspects of software development.

Taking all of the above into consideration, the User could be a Project Manager, Process Engineer, Scrum Master (or the whole Team), RAMS Engineer or similar roles, depending on the company applying the AgileSafe method.

4. APPLYING AGILESAFE

The *Apply AgileSafe* use case is decomposed into more specific use cases, as shown in Figure 2. User needs to specify the characteristics of the Project and the regulations the Project needs to comply with. Based on these, the User is guided through two main aspects of AgileSafe: the process which selects the specifications of software development practices to be applied in the Project and the process which results in the set of assurance arguments corresponding to the regulations included in the regulatory context. Finally, the User applies the selected practices in the Project.

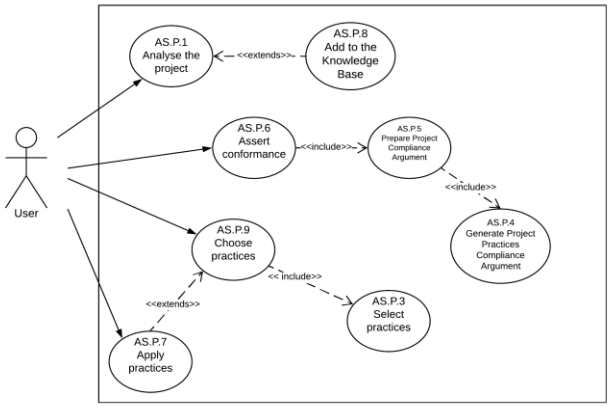

Figure 2 Apply AgileSafe use case diagram

In the further text we give a more detailed description of the use cases presented in Figure 2. During discussion, we will also call these use cases 'processes'.

4.1 AS.P.1 Analyse the Project

The first process of AgileSafe is *AS.P.1 Analyse the project*. In this process, the User is expected to gather information about the Project that is then being used as an input to the subsequent steps of AgileSafe. In order to determine the characteristics of the Project, AgileSafe uses an approach for scaling agile, presented by Scott W. Ambler [19], in particular because of its focus on context of the project. As Kruchten noted in [20], the context is vital in deciding how agile the software development and management in a given project can be. Ambler's scaling factors represent a broad spectrum of circumstances, both company and project related.

The factors are represented in the AgileSafe Practices Knowledge Base as Factors. Each Factor can be evaluated in a 5-point scale:

1. Team Size (based on Ambler's survey [21])
(Number of developers working in the project)
A – Under 10 developers; B – From 10 to 50 developers; C – From 50 to 100 developers; D – 100's of developers; E – 1000's of developers

2. Geographical Distribution (based on Ambler's survey [21])
(*Where are the team members located physically?)*
A – Co-located; B – Same building; C – Within driving distance; D – some working from home; E – Globally distributed

3. Domain Complexity
(How complicated is the target domain of the product?)
A – Straightforward; B - Predictable; C – Quickly changing; D – Complicated; E – Intricate/Emerging

4. Organisational Distribution
(What is the affiliation of the people working in the project, how is the work organised?)
A – Collaborative; B – Different teams; C – Different departments; D – Different partner companies; E – Contractual

5. Technical Complexity
(How complicates is the technological side of the project?)
A – Homogenous; B - Multiple technology; C – New technology; D - System/embedded solutions; E – Heterogeneous/Legacy

6. Organisational Complexity
(What are the structures of the company, how are they managed?)
A – Flexible, intuitive; B – Flexible, structured; C – Stable, evolutionary; D – Stable, planned; E – Rigid

7. Enterprise Discipline
(What lies in the centre of attention of the company management?)
A – Project focus; B – Mostly project focused; C – Balanced; D – Mostly enterprise focused; E – Enterprise focus

Note that the Regulatory Compliance factor of [19] has been omitted in the above list. This is because this particular factor gets special attention in AgileSafe and is being covered by other processes of .Figure 2.

To illustrate application of AgileSafe, we refer to a GlucoMet case study. The GlucoMet project concerns developing continuous glucose monitoring-enabled insulin pump system. The subject of the case study has been based on the remarks included in the following papers [22] [23] as well as the case study described in [24].Its characteristics are described in Table 1.

Table:1 GlucoMet project characteristics

Id	1	
Name	GlucoMet	
Description	Continuous glucose monitoring-enabled insulin pump system	
Regulatory Requirements	ISO 14971 [25]	
Characteristics	Factor	Values
	Team Size	B – From 10 to 50 developers;
	Geographical Distribution	B – Same building;
	Domain Complexity	D – Complicated;
	Organisational Distribution	B – Different teams;
	Technical Complexity	B - Multiple technology; D - System/embedded solutions;
	Organisational Complexity	B – Flexible, structured;
	Enterprise Discipline	A – Project focus;

These project characteristics, as presented in Table 1, have the potential to respond well to both, agile and disciplined practices. It is a safety-critical project, operating in a complicated domain but at the same time the size of the team, distribution and complexity of the organization open the possibility to introduce agility.

4.1.1 AS.P.8 Add to the Knowledge Base. AgileSafe maintains a Knowledge Base, which provides for matching the project characteristics with the suitable agile practices. The algorithms which suggest practices to the User are implemented in this Knowledge Base in the form of SWRL [26] rules. This is why the project characteristics need to be added to the Knowledge Base in order to obtain suggestions of practices suitable for this Project.

4.2 AS.P.6 Assert Conformance

In order to assert that the safety assurance requirements imposed by the appropriate regulatory documents have been built into the Project, AgileSafe uses assurance cases. The main idea is to provide assurance cases for the software development process as well as the end product itself. While the latter serves to demonstrate product conformance with a given standard or a guideline, the former is used to demonstrate that the selected software development practices provide sufficient assurance to the resulting product. By this combined approach the User can ensure that the selected practices are suitable for this particular Project with its safety requirements imposed by the standards.

The assurance argument patterns are derived from the relevant standards, regulations and guidelines. They follow the Trust-IT approach taken while applying argument structures to support application of standards. [27], [28], [29].

In order to increase usability of AgileSafe, the NOR-STA Argevide tool [30] has been chosen for managing the AgileSafe arguments set. NOR-STA offers a semi-graphical language (called TCL) which can be used to represent arguments and integrate them with external documents. A discussion of the relationship between TCL argument model and the OMG Structured Assurance Case Metamodel (SACM) [31] can be found in [32].

All of the arguments in the method are developed separately for each applicable standard in order to support separate certification processes and are based on the standard structure. There are three types of assurance arguments in AgileSafe: *Practices Compliance Argument*, *Project Practices Compliance Argument* and *Project Compliance Argument*. The first two focus on the software development practices being able to produce necessary conformance material and the last one presents the argumentation based on the actual artefacts collected in the Project.

4.2.1 AS.P.4 Generate Project Practices Compliance Argument . Project Practices Compliance Argument is a Practices Compliance Argument adapted to a specific Project and is determined by the Project Practices Set specific to this project. The Project Practices Compliance Argument refers only to the practices used in the particular Project along with the description of evidence they are providing. Its structure is defined in the Project Practices Compliance Pattern. The structure is similar to the Practices Compliance Argument Pattern. An example of Project Practices Compliance Argument is described in the Figure 4, based on the extract of ISO 14971 argumentation for GlucoMet project.

In TCL language, an argument is a hierarchical structure where the elements of the hierarchy are represented from left to right (as opposed to top-down representation). The following TCL elements are dedicated to representing elements of arguments. Argument conclusion is represented by a *claim* ( ) node. A node of type *argumentation strategy* (denoted) links the claim with the corresponding premises and uses a *rationale* node (denoted) to explain and justify the inference leading from the premises to the claim. A premise is a sort of assertion and can be in particular another claim to be further justified by its own premises, or a *fact* (denoted) represented by an

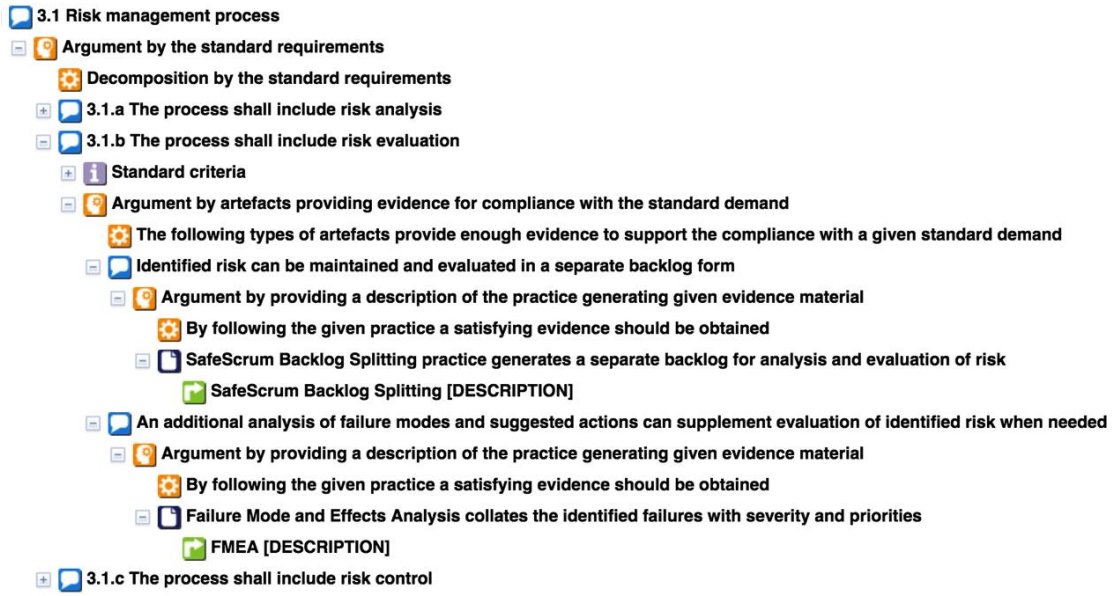

Figure 4 An excerpt of Project Practices Compliance Argument for ISO 14971 in NOR-STA tool

assertion to be demonstrated by the supporting evidence. Two elements of the TCL language serve as universal tools for structuring information: the *information* node (denoted) and the *reference* node (denoted). The reference nodes are used to integrate external documents, in particular the documents containing the evidence supporting the argumentation.

The User generates Project Practices Compliance Argument by limiting the Practices Compliance Argument only to the nodes he or she implements in his/her Project.

The Practices Compliance Arguments represent the connection between practices from Knowledge Base and standard requirements without any input from Project, they are easily reusable. Once prepared for a given standard, a Practices Compliance Argument should only be updated when the Knowledge Base gets updated with new practices.

4.2.2 AS.P.5 Prepare Project Compliance Argument . Based on the Project Practices Compliance Argument a Project Compliance Argument is being prepared. This is the actual argument, in which the User collects the evidence in the suitable nodes. Each evidence node has a connected Fact, taken form Project Practices Compliance Argument, explaining why the given artefacts serve as evidence for this particular standard requirement. The structure of this argument is outlined in the Figure 5, which depicts an excerpt of GlucoMet Project Compliance Argument for ISO 14971.

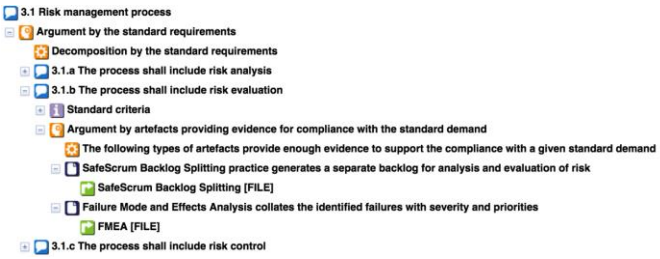

Figure 5 **An excerpt of Project Compliance Argument for ISO 14971 in NOR-STA tool**

4.3 AS.P.9 Choose Practices

As a result of using AgileSafe the User is presented with a Suggested Practices Set. This set contains software development practices which were deemed to be most suitable based on the Project Characteristics and safety requirements depending on the Project. This list may contain several Practices that concern the same aspect of software development or produce similar artefacts, making them redundant. A User makes the final choice which Practices should be used in a given Project and creates the Project Practices Set. These chosen Practices are then used to build Project Practices Compliance Argument and their artefacts fill the Project Compliance Argument.

4.3.1 AS.P.3 Select Practices. The AgileSafe *AS.P.3 Select Practices* process is the inner process of the method. Suggestions of practices for a given Project are prepared at this stage.

The *AS.P.3 Select Practices* process is based on two types of information about a Project: regulatory requirements it needs to meet and its Project Characteristics.

The suggesting algorithms are implemented in the form of SWRL rules in the Knowledge Base. If a given Practice satisfies any of the requirements of the standard needed by the Project and at the same time is suitable for the Project context as described in the Project Characteristics, such Practice is treated as potentially suitable for the given Project and is added to the Suggested Practices Set.

4.4 AS.P.7 Apply Practices

Upon determining Project Practices Set and developing assurance arguments, the User can implement the solution in his or her Project. By following the practices from Project Practices Set the User would produce artefacts - evidence that should then be placed in the corresponding nodes of the Project Compliance Argument.

5. IMPROVING AGILESAFE

The *Improve AgileSafe* use case implements a learning loop of AgileSafe. In order to further improve the method and tailor its advice to the User's needs more accurately, the knowledge stored in the method should be reviewed and updated on a regular base.

In particular, the User can introduce new software development practices to the pool of the practices from which the suggested practices are selected, as illustrated in Figure 6. Another activity focuses on preparing Practices Compliance Arguments for new standards, not yet covered by AgileSafe and/or updating already existing Practices Compliance Arguments to cover the new practices being introduced to the Knowledge Base.

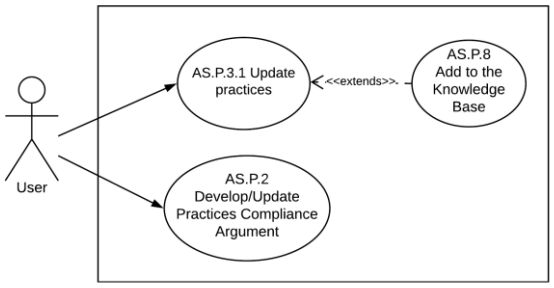

Figure 6 **Improve AgileSafe use case diagram**

5.1 AS.P.3.1 Update practices

In order to introduce a new practice to the Knowledge Base, the User should gather available information about the practice and analyze it according to the new practice template, as shown in the example of Backlog Splitting in Table 2.

Table 2 Example of practice description

Id	1		
Name	SafeScrum Backlog splitting		
Description	"In SafeScrum, all requirements are split into safety critical requirements and other requirements and inserted into separate product backlogs. Alternatively, the safety requirements are tagged. Adding a second backlog is an extension of the original Scrum process and is needed to separate the frequently changed functional requirements from the more stable safety requirements. With two backlogs we can keep track of how each item in the functional product backlog relates to the items in the safety product backlog, i.e. which safety requirements that are affected by which functional requirements. This can be done by using simple cross-references in the two backlogs and can also be supported with an explanation of how the requirements are related if this is needed to fully understand a requirement. The staffing of the Sprint team and the duration of the sprint (30 days is common), together with the estimates of each item decides which items that can be selected for development. Sometimes also e.g. the Safety responsible or the RAMS responsible takes part in the selection of which items have to be prioritized." [33]		
Discipline	Architecture	No	
	Deployment	No	
	Development	Yes	
	Environment	No	
	Project Management	Yes	
	Requirements	Yes	
	Test	No	
Capability	Factor	Values	
	Team Size	A – Under 10 developers; B – From 10 to 50 developers; C – From 50 to 100 developers; D – 100's of developers;	
	Geographical Distribution	A – Co-located; B – Same building; C – Some working from home; D – Within driving distance; E – Globally distributed	
	Domain Complexity	A – Straightforward; B - Predictable; C – Quickly changing; D – Complicated; E – Intricate/Emerging	
	Organisational Distribution	A – Collaborative; B – Different teams; C – Different departments; D – Different partner companies; E – Contractual	
	Technical Complexity	A – Homogenous; B - Multiple technology; C – New technology; D - System/embedded solutions;	
	Organisational Complexity	1 – Flexible, intuitive; 2 – Flexible, structured; 3 – Stable, evolutionary; 4 – Stable, planned;	
	Enterprise Discipline	1 – Project focus; 2 – Mostly project focused; 3 – Balanced; 4 – Mostly enterprise focused;	
Used in:	Name of the Regulation and regulatory requirement	General Practice	Fact
	ISO 14971 3.1 Risk management process 3.1.b The process shall include risk evaluation	Identified risk can be maintained and evaluated in a separate backlog form	SafeScrum Backlog Splitting practice generates a separate backlog for analysis and evaluation of risk
	Etc.		

A Discipline field contains a list of disciplines within which the practice operates (one or more). In the **Capability** field, for each factor, a list of predefined circumstances in which the practice works best (one or more, for each factor) is presented. These Factors are identical to the ones in the Project Characteristics. The **Used in** field presents a link to the Regulatory Requirement that the Practice (under General Practice) complies with. The Fact is the statement of the Practice's contribution to the compliance.

At this stage of AgileSafe maturity, it is up to the User who introduces the practice to decide how well a given practice can respond to the specific project Factors and regulatory requirements of standards.

5.2 AS.P.2 Develop/Update Practices Compliance Argument

The *AS.P.2* process focuses on preparing the Practices Compliance Arguments for the standards that are required for the given Project. The User needs to check whether Practices Compliance Arguments for the needed standards are already available in the AgileSafe Knowledge Base. If so, they should be revised and updated, if needed. Otherwise, a new Practices Compliance Argument for a needed standard should be developed and inserted to the Knowledge Base.

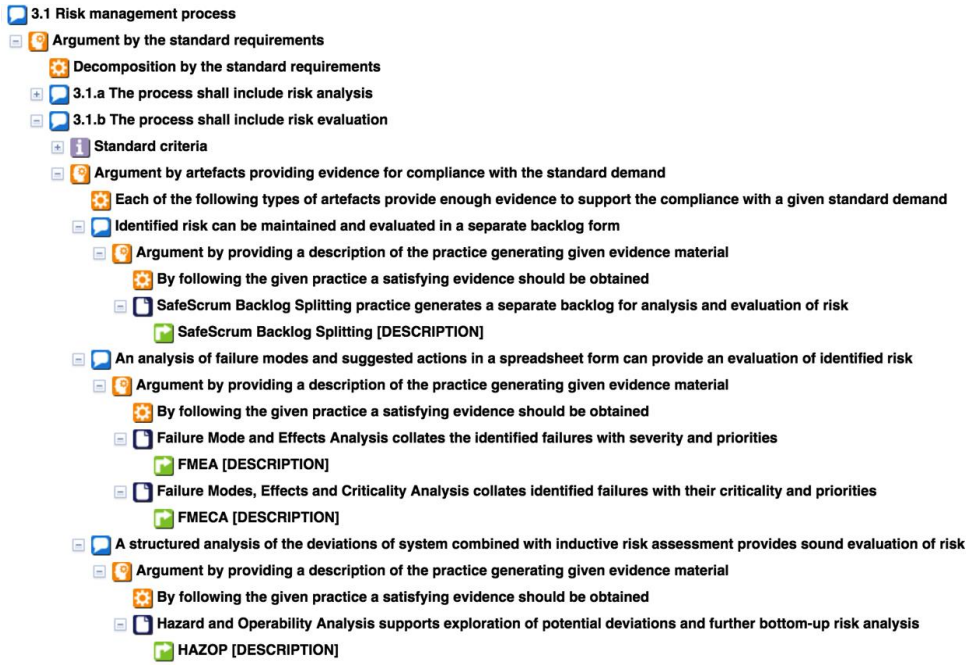

Figure 7 **An excerpt of Practices Compliance Argument for ISO 14971 in NOR-STA tool**

The structure of Practices Compliance Arguments is based on the standard requirements. The Practices Compliance Arguments are generic and focus on the conformance of practices from Knowledge Base with particular requirements of the considered standard. For each such requirement they propose an argumentation strategy and the range of software engineering practices used for collecting evidence demonstrating the compliance. They also contain explicit justification that the argumentation strategy is adequate on the condition that the evidence is collected and integrated with the argument. A list of claims concerning different types of practices, which may contribute to satisfying the standard demand, is presented, each claim postulating the potential of a given practice to generate the evidence needed to demonstrate compliance.

Figure 7 presents an excerpt of Practices Compliance Argument for ISO 14971 represented in the NOR-STA tool.

6. CONCLUSIONS

The AgileSafe method presented in this paper is a method, which allows an introduction of agile practices into the safety-critical software development while importantly delivering a solution for asserting conformance with applicable regulatory requirements.

In order to facilitate the implementation of the method, AgileSafe uses tool support: NOR-STA tool [27] for managing and developing assurance arguments and Protégé [34] for Knowledge Base operations.

To this date, elements of AgileSafe have been implemented in student projects. The method has been also evaluated by a group of experts from Poland and Norway, as well as two developers/Scrum Masters from a safety-critical company in Norway.

The results are positive and indicate AgileSafe potential of bringing added value to safety critical software development processes.

Nevertheless, some further work is needed to make the method more readily applicable for the industry. The directions for further progress include expansion of a starting pool of practices available in the Knowledge Base, availability of ready to use Practices Compliance Arguments for most commonly used standards as well as incorporating some experts and Users feedback as a part of teaching of the method.

References

[1] John C. Knight. 2002. Safety critical systems: challenges and directions. In Proceedings of the 24th International Conference on Software Engineering (ICSE '02). ACM, New York, NY, USA, 547-550..

[2] 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..

[3] Boehm and Richard Turner. 2003. Balancing Agility and Discipline: A Guide for the Perplexed. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA..

[4] Petersen, K., & Wohlin, C. 2010. The effect of moving from a plan-driven to an incremental software development approach with agile practices. Empirical Software Engineering, 15(6), 654-693

[5] Abrahamsson, P., Salo, O., Ronkainen, J. & Warsta, J. 2002. Agile software development methods: Review and analysis, VTT publication 478, Espoo, Finland, 107p..

[6] Drobka, J., Noftz, D. and Raghu R. 2004. Piloting XP on four mission-critical projects. IEEE Softw., 21(6), pp.70-75

[7] Lindvall M., Muthig D., Dagnino A., Wallin C., Stupperich M., Kiefer D., May J. & Kähkönen T. 2004. Agile Software Development in Large Organizations. Computer, 37(12), pp. 26-34
[8] Weiguo L., Xiaomin F. 2009. Software Development Practice for FDA-Compliant Medical Devices. Proceedings of the 2009 International Joint Conference on Computational Sciences and Optimization, Sanya, China
[9] McHugh, M., McCaffery, F. and Coady, G. 2014. An Agile Implementation within a Medical Device Software Organisation. Communications in Computer and Information Science, pp.190-201.
[10] International Organization for Standardization 2006. IEC 62304:2006 Medical device software -- Software life cycle processes, [online] Available at: https://www.iso.org/standard/38421.html (Visited: 1/03/2018)
[11] SINTEF 2017, [online] Available at: http://www.sintef.no/ (Visited: 20/03/2018)
[12] Norwegian University of Science and Technology 2017, [online] Available at: https://www.ntnu.edu/ (Visited: 20/03/2018)
[13] Mycklebust, T., Stålhane, T. and Hanssen, G. 2016. Use of Agile Practices when developing Safety-Critical software. In: Proceedings of The 34th International System Safety Conference (ISSC)
[14] Weinstock C. and John B. Goodenough. 2009. Towards an Assurance Case Practice for Medical Devices. TECHNICAL NOTE Software Engineering Institute October.[online] Available at: http://www.sei.cmu.edu/reports/09tn018.pdfAvailable at: http://www.sei.cmu.edu/reports/09tn018.pdf (Visited: 20/03/2018)
[15] Food and Drug Administation. 2015. A Mobile Medical Applications. Guidance for Industry and Food and Drug Administration Staff
[16] Bloomfield, R., Chozos, N. and Cleland, G. 2012. Supplement G: Safety case use within the medical devices industry. Supplements to: Using safety cases in industry and healthcare. [online] London: The Health Foundation, pp.G2-G17. Available at: http://www.health.org.uk/sites/default/files/UsingSafetyCasesInIndustryAndHealthcare_supplements.pdf. (Visited: 10/03/2018)
[17] AMASS (Architecture-driven, Multi-concern and Seamless Assurance and Certification of Cyber-Physical Systems) 2018. [online] Available at: https://www.amass-ecsel.eu/ (Visited: 30/05/2018)
[18] SafeCer 2018 [online] Available at: https://www.sp.se/en/index/research/dependable_systems/safecer/Sidor/default.aspx (Visited: 30/05/2018)
[19] Ambler, S. 2010. IBM agility@scale: Become as Agile as You Can Be. IBM Global Services
[20] Kruchten, P. 2011. Contextualizing agile software development. Journal of Software: Evolution and Process, 25(4), pp.351-361
[21] Ambler, S. 2012. Summer 2012 DDJ State of the IT Union Survey. [online] Available at: http://www.ambysoft.com/surveys/stateOfITUnion201209.html (Visited: 10/03/2018)
[22] Chen Y., Lawford M., Wang H..2014. Insulin Pump Software Certification. Foundations of Health Information Engineering and Systems. FHIES 2013. Lecture Notes in Computer Science. 8315().pp 87-106
[23] Zhang, Y., Jones, P. L., & Klonoff, D. C..2010. Second Insulin Pump Safety Meeting: Summary Report. Journal of Diabetes Science and Technology, 4(2), pp 488–493..
[24] Górski J., Łukasiewicz K..2013.Towards Agile Development of Critical Software. Software Engineering for Resilient Systems. SERENE 2013. Lecture Notes in Computer Science, 8166(), Springer.
[25] International Organization for Standardization .2007. ISO 14971:2007 Medical devices -- Application of risk management to medical devices, [online] Available at: https://www.iso.org/standard/38193.html (Visited 01/03/2018)
[26] SWRL: A Semantic Web Rule Language Combining OWL and RuleML. 2004. [online] Available at: https://www.w3.org/Submission/SWRL/https://www.w3.org/Submission/SWRL/
[27] NOR-STA project Portal. 2017. [online] Available at: www.nor-sta.eu (Visited: 30/05/2018)
[28] Cyra, L. and Górski, J. 2011. Support for argument structures review and assessment. Reliability Engineering & System Safety, 96(1), pp.26-37
[29] Górski, J., Jarzębowicz, A. and Miler, J. 2012. Validation of Services Supporting Healthcare Standards Conformance. Metrology and Measurement Systems, 19(2), pp.269-282
[30] Argevide. 2017. [online] Available at: http://www.argevide.com/ (Visited: 30/05/2018)
[31] Structured Assurance Case Metamodel (SACM), version 2.0, Object Management Group. 2017.
[32] Relationship between TCL and SACM 2.0. https://www.argevide.com/documents/ (Visited: 02/06/2018)
[33] Stålhane, T., Myklebust, T., and Hanssen, G.K. 2012. The application of Safe Scrum to IEC 61508 certifiable software. ESREL 2012, Helsinki, Finland.
[34] Musen, M.A. 2015. The Protégé project: A look back and a look forward. AI Matters. Association of Computing Machinery Specific Interest Group in Artificial Intelligence, 1(4), June 2015.