Computer Science • 13 (4) 2012 http://dx.doi.org/10.7494/csci.2012.13.4.165

Janusz Górski
Katarzyna Łukasiewicz

ASSESSMENT OF RISKS INTRODUCED TO SAFETY CRITICAL SOFTWARE BY AGILE PRACTICES — A SOFTWARE ENGINEER'S PERSPECTIVE

Abstract *In this article we investigate the problem of applying agile practices into safety-critical projects. The goal of our research is to investigate potential benefits from introducing agile practices into safety-critical environment and to present a solution providing for balancing agile approach with more disciplined assurance techniques, bringing the best of the two worlds together. In this article we present the supporting ideas such as assurance argument patterns along with a case study. The case study investigates how software engineers perceive risks associated with the introduction of agile practices and collect their ideas on how these risks could be mitigated.*

Keywords software development, safety-critical projects, agile practices, experimental assessment, TRUST-IT

1. Introduction

Agile methodologies have grown in popularity since the presentation of the Agile Manifesto in 2001 [1]. They were introduced as an alternative to plan-driven methodologies, which were considered as being too restrictive in some circumstances, in particular, while dealing with volatile requirements and ever changing market demands. In such situations, heavy-weight documentation and low flexibility associated with plan-driven approach could have an impeding effect on software development process [13]. In response to these concerns agile methodologies have offered practices which value a close relationship with clients, allow a more relaxed approach towards documentation and provide a flexible development lifecycle based on short iterations. While agile approaches have gained an almost immediate acclaim among SMEs and companies involved in non-critical projects, they were received with much distrust by larger companies and software engineers engaged in long-term and/or critical projects. In the case of such projects, the predictability and stability of plan-driven methodologies can bring expected profits [27] by facilitating the certification processes and establishing a repetitive quality. The up-front analysis and rigorous documentation provide valuable foundations for further risk management and process traceability [12]. What is more, some companies have years of experience in managing their projects following plan-driven practices, therefore they have acquired the know-how that increases trust towards this approach [17], [32]. In [26], major challenges related to moving from plan-driven to agile practices were summarized from a large corporate company's perspective.

While plan-driven methodologies have proven its value and usefulness in safety-critical projects, the evolving market of software products of the last few years puts this approach to the test. A growing competition, ever changing technologies and more diverse groups of clients have changed the expectations towards software development methods. The need to deliver systems of acceptable quality, faster and at lower cost in comparison to competitors evoked seeking an alternative [28].

In this paper we discuss the models of combining agile and plan-driven practices as well as their potential applications in safety-critical projects that were introduced in the literature thus far. The main objective of our research is to develop a method that would allow to combine agile and more disciplined practices in safety-critical projects while keeping the acceptable level of safety assurance in accordance with the norms and the standards applicable for a given product. The solution will be evaluated in terms of added value for manufacturing processes of safety-related software. In this paper we present the supporting ideas such as *assurance argument patterns* along with a case study.

2. Balancing agility and discipline

For many years it has been prejudicially believed that agile methodologies are at odds with maturity models and certification schemes [18]. This point of view might stem from the outdated notions based on inflexible models such as the Capability Maturity Model (replaced with the Capability Maturity Model Integration [4]) and an oversimplification of agile values [18]. In fact, present-day models provide more freedom than their predecessors while agile methodologies are not about the lack of documentation and recklessness.

Attempts to combine the best of the two approaches, the agile and the disciplined one, have been presented as early as in 2003 [13] and initiated a global discussion about the need and applicability of such hybrid methodologies. Successfully combined, agile practices can potentially reduce the cost of production as well as time to market while more mature practices could improve the quality of the product and enable better control of the development process. As a result, some models adapting agile practices into maturity models such as CMMI have been introduced since then [16], [25], [15], [14], providing new possibilities for companies engaged in long-term and critical projects. What is more, case studies and evidence of successful applications of such balanced approaches have been provided as well [18], [23], [29], [11], [30], [24].

3. Introducing agile practices to the safety-critical domain

In recent years a growing competition in the IT market has also influenced the safety-critical domain. When it comes to medical software companies, they have evolved from supporting only hospitals and doctors to providing personalized e-health software solutions and equipment for individual patients. It has become crucial to offer better software, more appealing to a client while keeping the costs low, in order to compete in this fast changing and growing market. Consequently, there is a strong demand for increasing efficiency of software development processes (in terms of effort, user satisfaction and time) while still respecting the safety requirements imposed by relevant standards, regulations and recommendations[1].

3.1. Applications of the hybrid approaches

There is a growing body of evidence supporting the theory that incorporating agile practices into safety-critical projects is not only feasible but also potentially profitable. In 2003 Alleman et al. [10] presented an approach combining eXtreme Programming practices [7] with Earned Value Management [6] that was successfully implemented in a government contracted project. In their article they described their experiences and improvements obtained by using this approach although the approach itself was not sufficiently illustrated and little was mentioned about which features of the product and its certificates had influenced the choice of practices.

Consecutive reports were more specific about successful implementations of their hybrid approaches. Rasmussen et al. [31] described the application of a tailor-made

[1]For instance, US Food and Drug Administration (FDA) recommends explicit assurance cases for medical devices which will impact on the whole related suppliers market

agile approach in a Food and Drug Administration (FDA) regulated project in Abbott company. As a result of the rapid expansion of the market, which demanded responding to the changing requirements as well as reducing production cost, the company decided to employ a software engineering organization AgileTek [2], which developed the Agile+ solution. They managed to improve their processes and achieved the financial goals by introducing a more agile approach while still maintaining the acceptable level of FDA safety assurance. Unfortunately, because of a commercial aspect of the Agile+ it was only briefly described in the article, making it difficult for other companies to benefit from Abbott experiences without external help.

Another interesting case study was presented in an article by Petersen and Wohlin [28] in which they described the application of agile practices at Ericsson AB, which is certified with ISO 9001:2000. They focused on comparing how the perceived impediments have changed owing to the introduction of agile practices. The improvements were noticeable as most of the concerns raised in relation to the plan-driven methodologies were alleviated as well the number of perceived impediments was reduced by introducing a more agile approach.

3.2. Models for adopting agile to safety-critical development

The increasing number of reports suggesting that adapting agile practices to suit safety-critical processes can bring measurable profits provoked the need for a model of such adaptation.

In the literature we can find some attempts to propose such models. Weiguo and Xiaomin [35] presented an approach suitable for FDA compliant medical devices projects. Their method was based on the idea of combining an incremental character of developing code with a classic, waterfall-like way of preparing project documentation. Unfortunately, the model was insufficiently described and we know very little about its possible implementations. Stephenson, McDermid and Ward [34] presented another model for tailoring agile practices to suit safety-critical systems. They called it the Agile Health Model and it was built around the idea of modular structures and risk management techniques known from plan-driven methodologies. However, the model was in the preliminary stage, introduced mainly in order to prove the possibility of applying agile practices into safety-critical projects, and no case study was provided as well.

A more complex and well described model was presented by Paige et al. [27]. They collated agile and safety principles and demonstrated the key challenges that need to be addressed when formulating a hybrid approach. Their main concerns were different approaches towards communication, documentation, customer participation, multiple-domain engineering, testing and incrementality. Indeed, as much as agile methodologies value an active customer participation it is often impossible in safety-critical systems to keep in touch with every group of stakeholders, in particular if we take external certification organizations into consideration as well. Testing and incrementality, both crucial to the agile development, are also difficult to reconcile with safety requirements. Agile testing strategy is based on unit and black-box tests while in order to satisfy certification bodies it is important to incorporate costly and time consuming white-box and acceptance tests. What is more, the incremental product development, one of the key attributes of all agile methodologies, can impede the process of certification and preparation of safety arguments as they should be addressed up-front with all of the requirements and risks known beforehand.

Paige et al.'s solution concentrates on the following ideas: pair-programming of software and system engineers, the introduction of risk management techniques, usage of tools for generating documentation from source code and tackling the incrementality by using "pipelined iterations" consisted of the minor and major iterations with acceptance tests at the end of every major one. Their model was implemented in a case study in which an Integrated Altitude Data Display System (IADDS) for planes was developed. As a result, their approach was put to the test as their solutions proved to be insufficient in some aspects. They concluded that although "XP and Aps (Agile practices) in general were not designed with safety-critical systems development in mind, they can be adapted to that sort of development", "it is rather unlikely that level A software can be produced in the near future with the modifications made to the process so far" [27].

While these models of adapting agile practices to suit safety-critical projects are valuable sources of knowledge, there is still a need to develop a more easy to use and thorough set of guidelines for safety-critical software companies that would like to adapt agile practices into their project development.

4. Assurance argument patterns

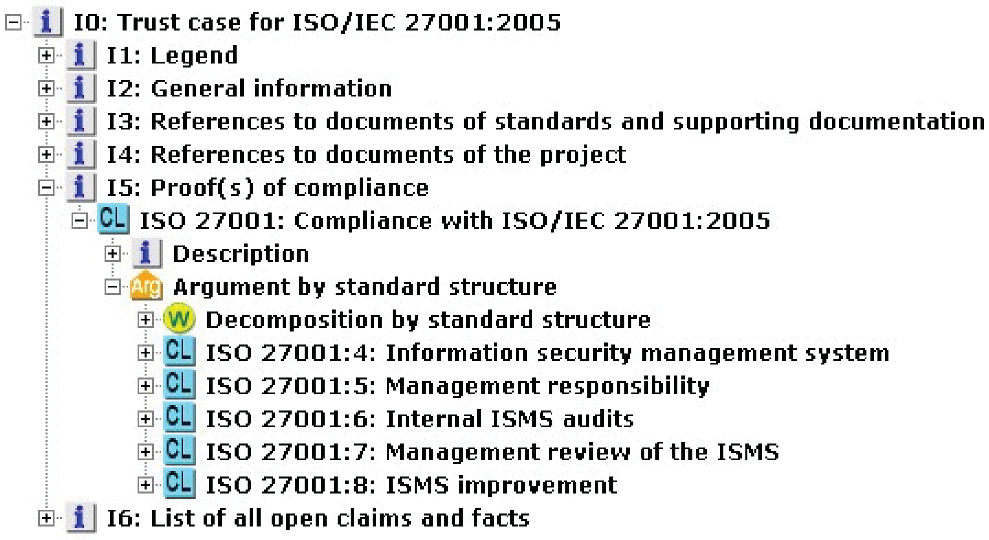
In [22] we have proposed an approach in which agile practices are introduced to a development process in a controlled way, by applying *assurance argument patterns*. This leads to a method which supports combining agile and more disciplined practices in safety-critical projects while keeping the acceptable level of safety assurance, in accordance with the assurance criteria chosen for a given product. The source of these criteria are the relevant guidelines, norms and standards related to a given application domain.

An argument is a data structure that includes *claims*, *assumptions*, *facts* and *evidence*. A claim can be related to other claims, facts and/or assumptions through an *inference*. In such cases this claim is called *conclusion* and the claims, facts and assumptions linked to it by the inference are called its *premises*. The inference linking the promises with the conclusion justifies why while accepting the premises we are entitled to accept the conclusion. An inference is associated with a *warrant* which justifies why the related conclusion should be accepted from the premises. If a given premise is a claim, it is considered as a conclusion to be further justified by its own premises. If a premise is an assumption, it does not require further justification. If a premise is a fact, it is required that it is supported by evidence submitted in external documents. Such documents are linked to the fact by means of *references*.

To provide for more structure, the arguments use also the *information* nodes which can occur in any place of the argument structure. This model of an argument belongs to the TRUST-IT methodology [21], [19], [20] which also provides for graphical representation of arguments. The methodology is supported by a platform of software services, called NOR-STA, which provide for arguments creation, editing, publishing, integration with evidence and assessment [9].

An assurance argument pattern is an incomplete argument. A pattern has 'dangling references' which do not point to any evidence – the evidence is missing in the pattern. It is also possible, that the pattern contains unjustified claims which need to be argued for in a more detailed way. After providing the evidence and completing unjustified claims, the pattern is converted into an argument.

An example argument pattern is given in Figure 1.



```
⊟ i I0: Trust case for ISO/IEC 27001:2005
    ⊞ i I1: Legend
    ⊞ i I2: General information
    ⊞ i I3: References to documents of standards and supporting documentation
    ⊞ i I4: References to documents of the project
    ⊟ i I5: Proof(s) of compliance
        ⊟ CL ISO 27001: Compliance with ISO/IEC 27001:2005
            ⊞ i Description
            ⊟ Arg Argument by standard structure
                ⊞ W Decomposition by standard structure
                ⊞ CL ISO 27001:4: Information security management system
                ⊞ CL ISO 27001:5: Management responsibility
                ⊞ CL ISO 27001:6: Internal ISMS audits
                ⊞ CL ISO 27001:7: Management review of the ISMS
                ⊞ CL ISO 27001:8: ISMS improvement
    ⊞ i I6: List of all open claims and facts
```

Figure 1. An example argument pattern following the TRUST-IT argument model.

The pattern presented in Figure1 represents the top level decomposition of an argument tree related to conformity with ISO/IEC 27001 standard on information security management. The tree is represented from left to the right (just like the file directory trees in an operating system interface). Figure 1 shows the information nodes (labeled 'I') structuring the argument pattern. The information node labeled 'I5' contains the claim (denoted 'CL ISO27001: Compliance with ISO/IEC 27001:2005') which predicates conformity with the standard ISO/IEC 27001. The conformity is validated by the claim's decomposition into more specific claims and finally ends with asserting facts. Figure 1 presents the decomposition of the argumentation to five more specific claims. The inference (represented by the node labeled 'Arg' and the associated warrant labeled 'W') relating these more specific claims (the premises) to the higher level claim (the conclusion) is based on structure of the standard (its decomposition into chapters).

We assume that such argument patterns can be built for a given software development project by analyzing the requirements of the relevant guidelines and standards related to software assurance in a given domain. The requirements can address both, the (software) product to be developed and the development process leading to this product. In our research, the primary focus is on the development process and on the assurance that is required in case agile practices are incorporated to the process.

In order to better understand the safety risks introduced by the agile software practices we plan for a series of experiments and cases during which we will attempt to assess these risks. The remainder of this paper describes a case study in which junior software engineers were involved in the risk assessment process.

5. Case study

The domain of medical safety-critical software has been developing at an unprecedented speed over the past few years. In addition to supplying hospitals and providing solutions for medical staff it now brings a wide variety of e-health technologies, including personal medical equipment and devices. In order to not be left behind and to reach bigger and more diverse customer groups, medical companies might look for new software development processes to reduce costs, accelerate time to market and improve product quality. For this reason we have decided to use that domain for our case study.

The case study was carried out between the beginning of March 2012 and the end of May 2012. For this study we have selected a group of 31 postgraduate students from our university that specialize in the field of software engineering. Some 67% of them have already began to work as part-time employees in various software companies and the majority of this group already have applied agile practices in their industrial work. Although they did not yet have real experience with safety critical software development, all participants have attended courses on plan-driven and agile methodologies and completed the course on high integrity systems.

5.1. The case study objectives

The main objective of the case study was: *to investigate how (junior) software engineers identify and assess risks associated with applying selected agile practices to critical software development and what are their suggestions concerning risk mitigation.*

It is expected that the results of the case study will help to devise a checklist of hazards as well as risk estimates and the suggestions for additional risk mitigation practices for incorporating agile practices into safety-critical software development.

The metrics to be collected in the case study include:

- List of hazards for an insulin pump applied in its target environment.
- A complete list of agile practices.

- A list of hazard scenarios explaining how applied agile practices contribute to software hazards.
- Risk assessment (risk levels) associated with each agile practice used.
- Agile practices which carry the highest risk.
- A list of risk mitigation recommendations.

5.2. The case study domain description

We devised a fictional company called MediSoft and specified its operational activity as producing software for insulin infusion pumps. After observing an increasing role of agile methodologies the company's management team became interested in the possible benefits that might be gained by utilizing such methodologies in their workplace. As a way to investigate the effects of the introduction of agile approaches into software development, MediSoft chose to carry out a pilot project whose aim is to prepare software for insulin infusion pump. They would like to employ eXtreme Programming [7] and Scrum [33] methodologies.

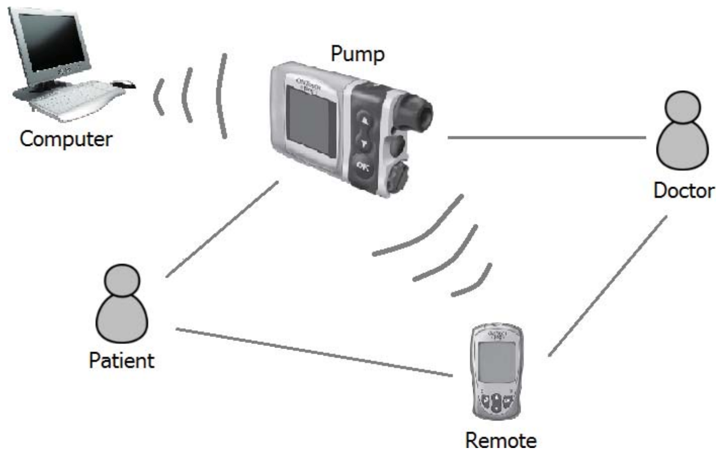


Figure 2. A context diagram of the insulin pump.

Students were divided into 11 project groups, each group consisting of not more than 3 members. Every group was given a short description of MediSoft company as well as product specification for standard infusion pump and guidelines on hazards analysis. They were also supposed to use the Designsafe tool [5] for risk assessment and MS Visio for fault trees editing.

An insulin pump is a device for patients with diabetes who need to control their blood sugar level by administrating insulin. The pump is attached to the patient's body along with a small container filled with insulin. At the proper times, small and precisely calculated amounts of insulin are released from the container into the patient's bloodstream. It helps to keep blood glucose levels steady between meals and during sleep.

At meal time, you can instruct the pump to deliver the amount of insulin needed to match the grams of carbohydrate in the food that is eaten.

The insulin pump description used in the project is based on the Animas One-Touch Ping [3], a real pump available on the market, characterized by the following features:

- Calculator for carbohydrates, blood glucose corrections and insulin.
- Insulin bolus very precise, should allow dosing even the lowest amounts of insulin in order to respond to every glucose deviation.
- Reminders for when to perform blood glucose checks.
- Easily available insulin dose corrections.
- Measuring the level of active insulin in the body.
- Wireless communication, the pump can be controlled with a wireless remote.
- Wireless bolus calculation and delivery.
- Uploading data from the pump to a computer using dedicated software.
- Information about the state of the body shown on the screen.
- Waterproof.

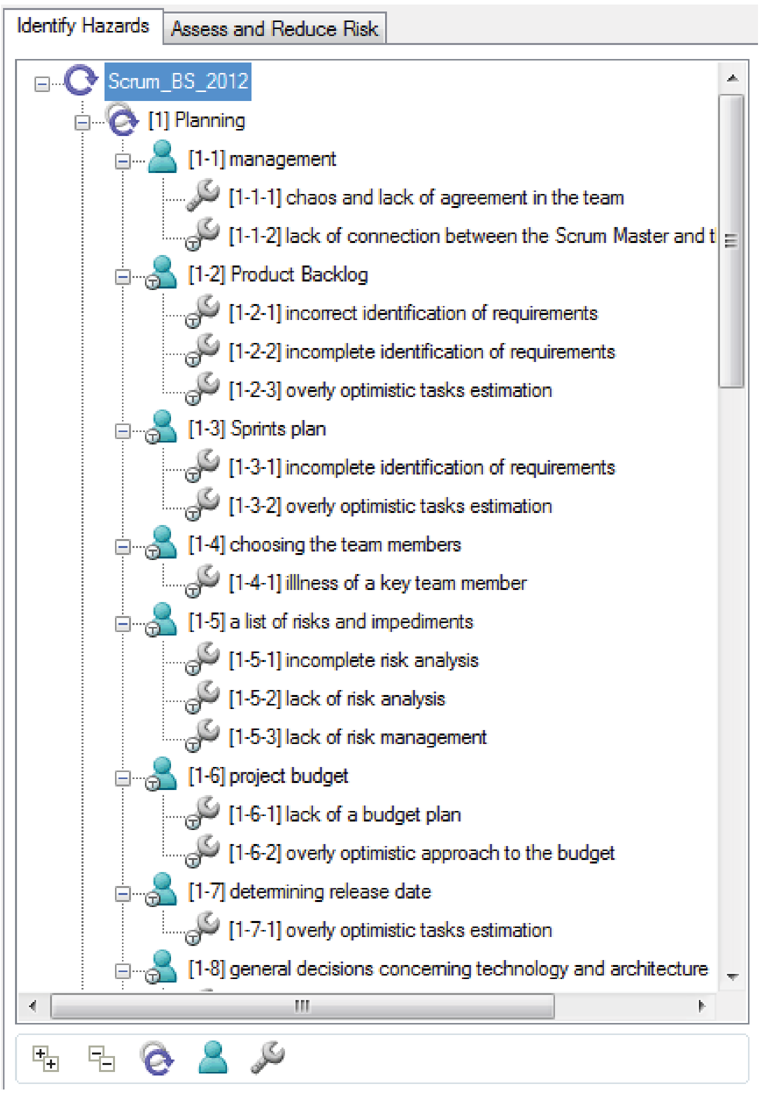
5.3. Participant' tasks and the results.

The case study has been divided into three tasks. After completing each task, the participants submitted the results using Moodle course management system [8].

Task 1. Preparing a list of hazards and hazard scenarios. Based on a documentation of the insulin pump and their own knowledge and imagination each group prepared a list of hazards connected with using the insulin pump in its target environment. This hazards identification process was supported by a guideline obtained at the beginning of the case study. The groups were also provided with an inventory of agile practices represented in the Designsafe tool [5]. After identifying hazards, they were asked to analyze the potential causes of each hazard and to document the result as a fault trees (FTA diagrams in MS Visio). Of particular interest were these fault trees for which the represented hazard scenarios were 'anchored' in the software development practices. Some groups had their Designsafe tool 'populated' with Scrum practices (see Figure 3) and the others with eXtreme Programming ones. The groups were free to extend this initial seeding, if they found it necessary.

Task 2. Conducting risk assessment. Each group performed, with the help of Designsafe tool, a risk assessment for the introduction of agile practices to the project developing software for the insulin pump.

Based on the fault trees prepared in Task 1, the groups analyzed the connection between the development related impediments and the identified hazards. This work resulted in refining the fault trees and possibly with refining the lists of development process impediments submitted initially.



Scrum_BS_2012
[1] Planning
[1-1] management
[1-1-1] chaos and lack of agreement in the team
[1-1-2] lack of connection between the Scrum Master and t
[1-2] Product Backlog
[1-2-1] incorrect identification of requirements
[1-2-2] incomplete identification of requirements
[1-2-3] overly optimistic tasks estimation
[1-3] Sprints plan
[1-3-1] incomplete identification of requirements
[1-3-2] overly optimistic tasks estimation
[1-4] choosing the team members
[1-4-1] illness of a key team member
[1-5] a list of risks and impediments
[1-5-1] incomplete risk analysis
[1-5-2] lack of risk analysis
[1-5-3] lack of risk management
[1-6] project budget
[1-6-1] lack of a budget plan
[1-6-2] overly optimistic approach to the budget
[1-7] determining release date
[1-7-1] overly optimistic tasks estimation
[1-8] general decisions concerning technology and architecture

Figure 3. A tree of Scrum process stages along with tasks and possible impediments.

Then, the groups assessed the risk, assessing the severity of each hazard and the likelihood of its materialization. An example of such a risk analysis is given (Table 1).

Table 1
An example of a simple hazard analysis (based on Designsafe layout).

Name	Insulin overdose
Hazard category	Software fault
Description	The pump infuses a too high dosage of insulin due to insulin overflow or air pressure in lines.
Users	All
Connected Tasks	Product Backlog/Requirements Managements, Risk management, Technical analysis, Dose calculation algorithm design, Implementation, User Interface project, User's guide preparation
Cause	Faulty dose calculation algorithm, wrong requirements analysis, error in the implementation, misleading user interface or manual, database error
Severity	Catastrophic
Probability	Likely
Risk Level	High
Reduce Risk	Extensive testing, up-front technical analysis, good contact with the client representative, regular iterations, testing user interface and manual with real patients and doctors

Table 2 presents the risk assessment matrix used in this task.

Task 3. Proposing a list of risk mitigation practices. The groups were asked to propose a list of additional practices that would mitigate the risks found in Task 2 and that could be used as an extension to the agile methodology they have been working with.

Upon completion of the tasks, a common session was organized for all participants were they were discussing on how incorporating agile practices into safety-critical projects can affect the product risk and what mitigation actions could be employed during the development process to lower the risk.

5.4. Results

Results of the tasks are summarized below. As the experiment was completed at the beginning of June 2012, the raw data collected are not yet fully processed at the time of writing this text. Nevertheless, some interesting observations can be made.

Table 2
The risk assessment matrix.

	Severity of harm			
Probability of harm	**Catastrophic**	**Serious**	**Moderate**	**Minor**
Very Likely	High	High	High	Medium
Likely	High	High	Medium	Low
Unlikely	Medium	Medium	Low	Negligible
Remote	Low	Low	Negligible	Negligible

Table 3
A template for the additional practices description.

No.	*Name of the practice*
Description	*A description of the proposed practice – what activities it includes, how should they be performed, by whom, at what stages of the project.*
Related hazards	*Which hazards (from your risk analysis) the practice is expected to have influence on.*
Expected influence	*What is the expected result of implementing the practice, in what way it could reduce the risk, to what extent.*
Agility/discipline balance	*How the practice will affect the agility of the methodology ie. will it require some alterations in project roles or additional project stages etc.*

Results of Task 1. Overall, the 11 teams have distinguished 124 hazards and potentially hazardous situations in total, at different levels of detail. The hazards were not independent, often were synonymous, the differences were also in the scope and level of detail. After rough processing, they can be grouped into the following categories:

- User errors (adjusted dose, incorrect configuration, etc.).
- Error in measuring the level of insulin or sugar.
- Physical / hardware errors.
- Missing or incorrectly administered insulin doses.
- Lack of measurement of insulin or sugar within a prescribed period.
- Errors in alerting system (sugar level, the needle slipped, discharging, etc.).
- Unauthorized use of the device via radio waves.
- Interruption of system normal activity.
- Incorrect display of data.

Most of the hazards were further analyzed in a form of fault trees. An example is given in 4. The fault trees delivered as the result of this task were treated as an input to the next task where the groups were aiming at assessing risks. They were asked to concentrate on hazards connected with the software and human error

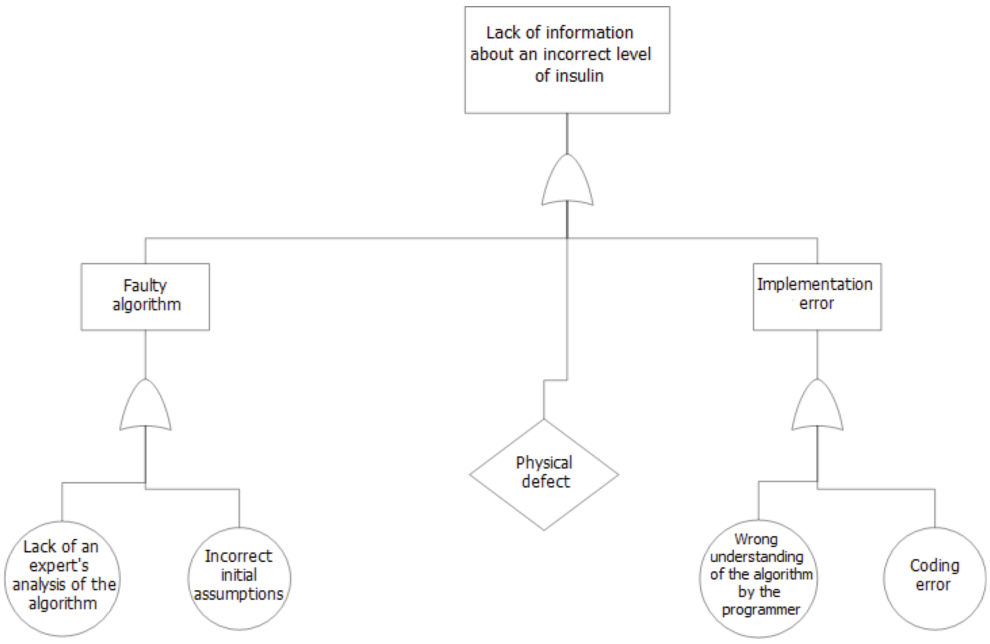

Lack of information about an incorrect level of insulin
Faulty algorithm
Implementation error
Physical defect
Lack of an expert's analysis of the algorithm
Incorrect initial assumptions
Wrong understanding of the algorithm by the programmer
Coding error

Figure 4. An example Fault Tree from one of the groups (Kalenik, Kurszewski, Karewska).

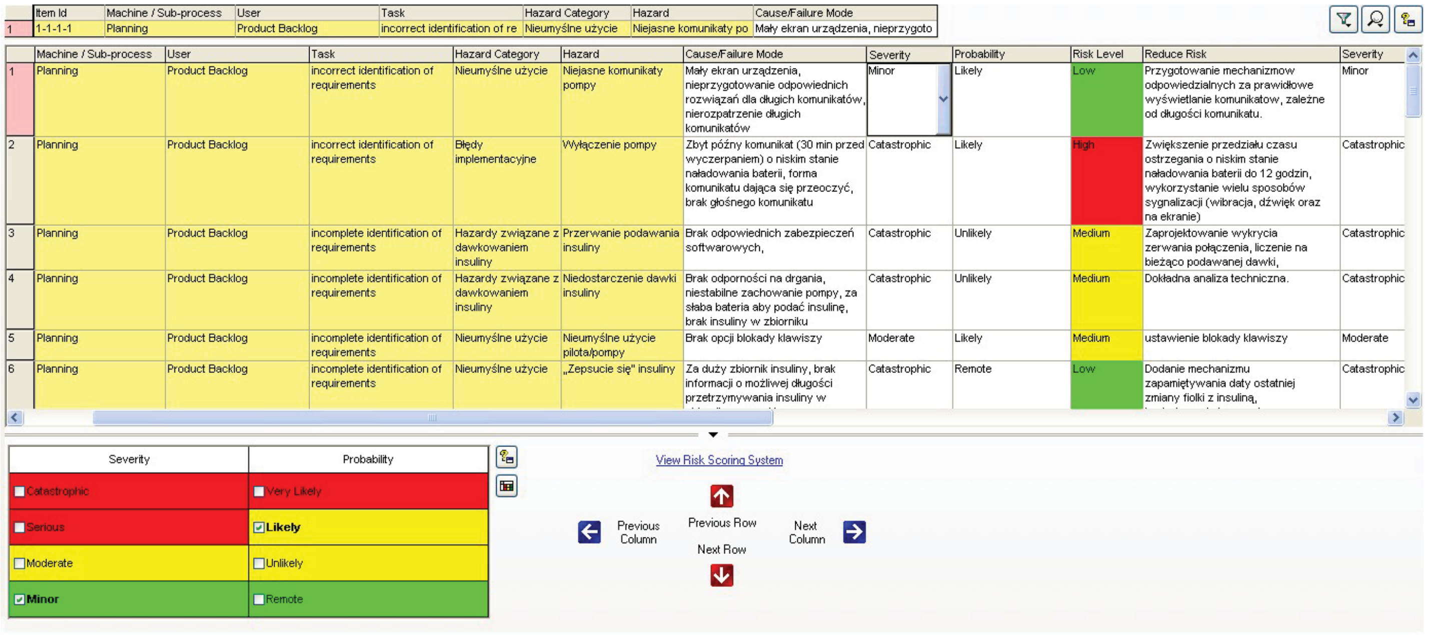
rather than hardware or misfortunes, to drive them towards events originating in the software development process. They were also encouraged to consider latent conditions related to organizational structure and management practices, having in mind the agile project development practices.

Results of Task 2. Risk assessment was performed with the help of Designsafe tool. During this task the groups were also refining the fault trees developed in Task 1. The results of risk assessment were presented in the tool, an example is given in Figure 5.

The project tasks associated with the highest risk together with their impediments are listed below.

For SCRUM:

- *Product Backlog* – incorrect identification of requirements.
- *Sprints Plan* – incomplete identification of requirements.
- *General decisions concerning technology and architecture* – lack of architecture plan and crucial implementation decisions.
- *General decisions concerning technology and architecture* – incomplete architecture plan and lacking crucial implementation decisions.
- *Providing the requirements (client)* – incorrect identification of requirements.
- *Providing the requirements (client)* – incomplete identification of requirements.



	Item Id	Machine / Sub-process	User	Task	Hazard Category	Hazard	Cause/Failure Mode
1	1-1-1-1	Planning	Product Backlog	incorrect identification of re	Nieumyślne użycie	Niejasne komunikaty po	Mały ekran urządzenia, nieprzygoto

	Machine / Sub-process	User	Task	Hazard Category	Hazard	Cause/Failure Mode	Severity	Probability	Risk Level	Reduce Risk	Severity
1	Planning	Product Backlog	incorrect identification of requirements	Nieumyślne użycie	Niejasne komunikaty pompy	Mały ekran urządzenia, nieprzygotowanie odpowiednich rozwiązań dla długich komunikatów, nierozpatrzenie długich komunikatów	Minor	Likely	Low	Przygotowanie mechanizmow odpowiedzialnych za prawidłowe wyświetlanie komunikatow, zależne od długości komunikatu.	Minor
2	Planning	Product Backlog	incorrect identification of requirements	Błędy implementacyjne	Wyłączenie pompy	Zbyt późny komunikat (30 min przed wyczerpaniem) o niskim stanie naładowania baterii, forma komunikatu dająca się przeoczyć, brak głośnego komunikatu	Catastrophic	Likely	High	Zwiększenie przedziału czasu ostrzegania o niskim stanie naładowania baterii do 12 godzin, wykorzystanie wielu sposobów sygnalizacji (wibracja, dźwięk oraz na ekranie)	Catastrophic
3	Planning	Product Backlog	incomplete identification of requirements	Hazardy związane z dawkowaniem insuliny	Przerwanie podawania insuliny	Brak odpowiednich zabezpieczeń softwarowych,	Catastrophic	Unlikely	Medium	Zaprojektowanie wykrycia zerwania połączenia, liczenie na bieżąco podawanej dawki,	Catastrophic
4	Planning	Product Backlog	incomplete identification of requirements	Hazardy związane z dawkowaniem insuliny	Niedostarczenie dawki insuliny	Brak odporności na drgania, niestabilne zachowanie pompy, za słaba bateria aby podać insulinę, brak insuliny w zbiorniku	Catastrophic	Unlikely	Medium	Dokładna analiza techniczna.	Catastrophic
5	Planning	Product Backlog	incomplete identification of requirements	Nieumyślne użycie	Nieumyślne użycie pilota/pompy	Brak opcji blokady klawiszy	Moderate	Likely	Medium	ustawienie blokady klawiszy	Moderate
6	Planning	Product Backlog	incomplete identification of requirements	Nieumyślne użycie	„Zepsucie się" insuliny	Za duży zbiornik insuliny, brak informacji o możliwej długości przetrzymywania insuliny w	Catastrophic	Remote	Low	Dodanie mechanizmu zapamiętywania daty ostatniej zmiany fiolki z insuliną,	Catastrophic

Severity	Probability
☐ Catastrophic	☐ Very Likely
☐ Serious	☑ Likely
☐ Moderate	☐ Unlikely
☑ Minor	☐ Remote

Figure 5. An example risk assessment screen from Designsafe tool.

For XP. Tasks along with their impediments which were associated with the highest risk:

- *User Stories* – incomplete identification of requirements.
- *Prototyping* – too general plan for architecture and methods of implementing of the system.
- *Release scope*: functionalities from previous iteration – large load on errors from the previous iteration.
- *Tests preparation* – incomplete test plan.
- *Unit tests* – low coverage.
- *Aacceptance tests* – low coverage.

Results of Task 3. In this task the groups were asked to consider possibilities of risk mitigation, especially for the hazards with the highest risk. The most commonly proposed mitigation practices included:

- Introducing an expert knowledge into the project.
- Extensive testing (i.e. enhanced acceptance tests, Test Driven Development).
- Introducing safety standards.
- Improving quality assurance in relation to the artefacts different than the code (i.e. preparing really 'good' User Stories).
- Keeping high coding standards.

6. Conclusion

The paper introduced our approach to the problem of introducing agile practices into critical software development processes and then reported on the experiment during which we attempted to investigate how this problem is perceived from the software engineer's perspective.

The data collected during the experiment were not yet finally processed, therefore what is presented in this paper reports more on the scope of the raw data collected than presents the final conclusions derived from these data. Nevertheless, even now we can conclude that the participants of the experiment were generally optimistic towards agile methodologies and their applicability to safety-critical projects. Their opinions can be summarized as follows:

- Agile methodologies should be regarded as complementary to plan driven practices instead of being the replacement.
- Extensive testing and good identification of requirements are vital.
- Contact with domain experts and potential users is crucial.
- Safety assurance should be incremental – in parallel to iterations in development.

However, as the participants of the experiment were students we do not expect that they performed their analyses at the expert level. Their output reflects the opinions of junior software engineers, their concerns regarding adapting agile practices into critical software development and their views on possible improvements and compromises.

In further research we will aim at incorporating other views (for instance, software developing companies, project managers, assessors, etc.). in order to learn which practices cause most uneasiness and distrust and how they could be revised in order to make them more compliant with safety standards. In parallel to this we will investigate safety standards, guidelines and regulations (focusing on the e-health domain) to reflect them in the assurance argument pattern represented with the help of TRUST-IT methodology and NOR-STA services.

References

[1] Agile manifesto. `http://agilemanifesto.org`.

[2] Agiletek. `http://www.agiletek.com/`.

[3] Animas one touch ping. insulin pump. `http://www.animas.com/animas-insulin-pumps/onetouch-ping`.

[4] Capability maturity model integration. `http://www.sei.cmu.edu/cmmi`.

[5] Designsafe tool. `http://www.designsafe.com/`.

[6] Earned value management. `http://www.earnedvaluemanagement.com/`.

[7] Extreme programming: A gentle introduction. `http://www.extremeprogramming.org/`.

[8] Moodle. `http://moodle.org/`.

[9] Nor-sta services portal. `www.nor-sta.eu/en`.

[10] Alleman G. B., Henderson M., Hill C. H. M., Seggelke R.: Making Agile Development Work in a Government Contracting Environment Measuring velocity with Earned Value. *Proc. of the Agile Development Conference 2003*, Salt Lake City, Utah, June 2003.

[11] Babuscio J.: How the FBI learned to catch bad guys one iteration at a time. In *2009 Agile Conference Proceedings, Chicago, USA*, pp. 96–100. August 2009.

[12] Boehm B.: Get ready for agile methods, with care. *IEEE Computer*, 35, 2002.

[13] Boehm B., Turner R.: *Balancing Agility and Discipline: A Guide for the Perplexed.* Addison Wesley, 2003.

[14] Bulska K. M. J.: Łączenie zwinności metodyki SCRUM z dojrzałością modelu CMMI. (Integration of the agile SCRUM practices with the maturity of CMMI). In *National Conference of Software Engineering (KKIO)*, 2010.

[15] Diaz J., Garbajosa J., Calvo-Manzano J.: *Mapping CMMI Level 2 to Scrum Practices: An Experience Report.* Proc. EuroSPI 2009 Alcala, Spain, September 2009.

[16] Fritzsche M., Keil P.: *Agile methods and CMMI: Compatibility or conflict?*, vol 1(1). 2007.

[17] Ge X., Paige R., McDermid J.: *An Iterative Approach for Development of Safety-Critical Software and Safety Arguments.* IEEE, 2010.

[18] Glazer H., Anderson D., Anderson D. J., Konrad M., Shrum S.: *CMMI or Agile : Why Not Embrace Both ! Software Engineering Process Management.* Technical Note for Software Engineering Institute.

[19] Górski J.: Trust Case — a case for trustworthiness of IT infrastructures. 196:125–142, 2005.

[20] Górski J.: Trust-IT – a framework for trust cases. *Workshop on Assurance Cases for Security — The Metrics Challenge. Proc. of DSN 2007, June*, 25-28:204–209, 2007.

[21] Górski J., Jarzebowicz A., Leszczyna R., Miler J., Olszewski M.: Trust case: Justifying trust. it solution. *Reliability Engineering and System Safety*, 89:33–47, 2005.

[22] Górski J., Łukasiewicz K.: *Agile development of critical software, can it be justified?* Proc. 7th ENASE Conference, Wroclaw, Poland, 2012.

[23] Lindvall M., Muthig D., Dagnino A., Wallin C., Stupperich M., Kiefer D., May J., Kahkonen T.: Agile software development in large organizations. *Computer*, 37(12):pp. 26–34, 2004.

[24] M. P., Mantyniemi A.: *An Approach For Using CMMI in Agile Software Development Assessments: Experiences From Three Case Studies.* Proc. of SPICE Conference, Luxembourg, May 2006.

[25] Marsal A., de Freitas B., Furtado Soares F., Furtado M., Maciel T., Belchior A.: Blending SCRUM practices and cmmi project management process areas. *Innovations in Systems and Software Engineering*, 4(1):17–29, 2008.

[26] P. F.: *How can you be Agile in "Rough Terrain" and under "Tight Boundary Conditions".* Proc. 7th ENASE Conference, Wroclaw, Poland, 2012.

[27] Paige R., Charalambous R., Ge X., Brooke P.: *Towards Agile Engineering of High- Integrity Systems.* Proc. of 27th International Conference on Computer Safety, Reliability and Security (SAFECOMP), September 2008.

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

[29] Poppendieck M.: *Lean software development: an agile toolkit.* Addison-Wesley, 2003.

[30] Potter N., Sakry M.: Implementing Scrum (Agile) and CMMI together. *Process Group Post Newsletter*, 16(2), January 2012.

[31] Rasmussen R., Hughes T., Jenks J. R., Skach J.: *Adopting Agile in an FDA Regulated Environment.* Agile Conference Proc. , Chicago, USA, August 2009.

[32] Rottier A., Rodrigues V.: *Agile Development in a Medical Device Company.* IEEE, 2008.

[33] Schwaber K., Beedle.: *Agile Software Development with Scrum.* Prentice Hall, 2001.

[34] Stephenson Z., McDermid J., Ward A.: *Health Modelling for Agility in Safety-Critical Systems Development*. Proc. of the First IET International Conference on System Safety Engineering, June 2006.

[35] Weiguo L., Xiaomin F.: *Software Development Practice for FDA-Compliant Medical Devices*. Proc. of the 2009 International Joint Conference on Computational Sciences and Optimization, 2009.

Affiliations

Janusz Górski
Gdańsk University of Technology, Department of Software Engineering, Faculty of Electronics, Telecommunications and Informatics, jango@eti.pg.gda.pl

Katarzyna Łukasiewicz
Gdańsk University of Technology, Department of Software Engineering, Faculty of Electronics, Telecommunications and Informatics, katarzyna.lukasiewicz@eti.pg.gda.pl

Received: 30.04.2012
Revised: 4.07.2012
Accepted: 3.09.2012