A MODEL FOR RISK MANAGEMENT IN AGILE SOFTWARE DEVELOPMENT

Author

Ville Ylimannela
Tampere University of Technology
ville.ylimannela@tut.fi

Abstract

This paper researches risk management in agile software development. In traditional waterfall-model risks were usually managed by using project risk management frameworks.

Nowadays agile methods have started replacing the traditional models. One of the reasons was old models inability to respond constantly changing business requirements. Agile development is based on short iteration cycles, which allow them to respond to changes in business environment. Using agile development is itself risk management at project level.

Problems started arising, when people tried to merge the old school heavy project risk management models with agile models. One of the key principles in agile development is to avoid unnecessary bureaucracy and documentation. The traditional risk management is heavily centered around documentation.

Two companies were interviewed during the making of this paper. The goal was to address the issues which arose during the interviews. The suggested model is based on existing models and interviews.

Keywords: risk management, agile, scrum, software development

1 Introduction

Risk management has traditionally been an integral part of software development. The change from traditional models, such as the waterfall model, to agile methods has created new challenges in the field of risk management. This paper will discuss the issues regarding risk management in agile software development.

In modern software projects, security and risk management are not just something one might do if there are time and resources. Security has become an important part of the end product. This means risk management must be introduced at the beginning of the project, and risks must be evaluated and assessed during the whole development cycle. Agile software development methods are focused around delivering the maximum benefit to a product owner. Problems arise when the team is too focused on achieving their goals for the sprint, leaving very little time for increasing security of the product.

By using the existing models and adding some new ideas, the purpose is to create a solution to problems software developers are facing when trying to manage risks in agile environment. Two companies were interviewed during making of this paper. They were asked questions about their risk management practises and how were they coping in the agile environment.

2 Limitations and challenges

Agile development is based on short iteration cycles and providing constant flow of program code to the product owner. Traditional risk management is a slow and comprehensive process. Agile processes are not meant to create heavy documentation, which is usually what risk management produces. When creating a new model to merge agile and risk management worlds, it is important to stay loyal to agile manifesto and lean principles.

Two companies were interviewed for this thesis. Both were using an agile software development process. Companies were questioned about their risk management in agile software development. The semi-structured interviews brought up shortcomings and slight problems. The biggest problems when conducting risk management in agile project environment were the following:

- Resources are a problem. Since the goal is to create a working increment each sprint, it might be hard to find time and people to conduct risk management each sprint. The general consensus was, that risk management takes five percent or less of the total time available each sprint.
- There was no clear risk owner.
- Acceptance of residual risks. Who has the power to accept residual risk at different abstraction levels?
- There was very little security training. Technical understanding of some managers might be lower than Scrum teams'.
- Risk communication was a problem. When is a risk high enough to be reported for a product owner for approval?
- If a risk is not directly related to a backlog item, it might go completely ignored.
- Sprint's DoD (Definition of Done) lacked security aspect.

The interviews showed that risk management is considered as an important part of agile software development.

Another thing that was brought up during the interview was maturity of the software. There is less time used on risk management as the software develops further. This is natural part of software development. Risks identified early in the process will not cause as much financial loss as the ones found later on. What this means is that a good model for integrating risk management to agile needs to address this issue.

3 Suggested model

This chapter defines a model for managing risks in agile environment. All major risk management phases described in PMBOK (project Management Body of Knowledge), from planning risk management to monitoring and re-evaluating risks, are included in the model [4]. The model revolves around a risk board, which is used and updated constantly throughout the software development cycle. The model defined in this chapter has not been tried in any real-life software development scenario. Despite the fact that Scrum terms are used and the model is initially thought to be part of Scrum, there is no reason why it could not be applied to other agile frameworks.

3.1 Risk board and risk notes

The board used to display risks is a modified version of the board suggested in [1]. The difference is that the board is in a form of a risk matrix. The matrix form of risk board allows the team and product owner to get a good idea about the general risk level in the project. The board below is a 3x3 risk matrix, however, there is no reason why a larger matrix could not be used. The board itself is a whiteboard. Fig. 1 shows the template for the risk board.

Risk table	Impact 1	2	3
Probability 1			
2			
3			
Avoided: Transferred:			

Figure 1: 3x3 Risk Board.

The size of the risk is calculated by multiplying impact and probability. The problem with the board might be the estimation of probability. In some scenarios the probability might be hard to assess. If the team feels this problem is a constant occurrence, they could start using simpler board, with just low, medium and high risks.

If a risk is completely avoided or transferred to someone else, the sticky notes should be placed away from the matrix to a dedicated area below. The template for risk notes is defined in Fig. 2. The risk notes should have at least the following information: feature ID to which the risk is related, who is responsible for implementing the feature and a short description of the risk.

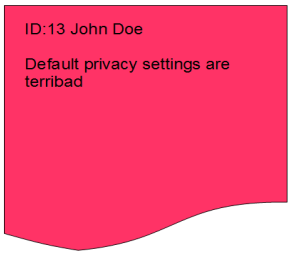

Figure 2: Risk note template.

There are in total two colours of sticky notes, red and yellow. The red notes are risks and yellow ones response solutions. When there is a large cross over the yellow note, it means that the solution has already been implemented. The Fig. 8 defines the template for solution notes. When a solution is suggested, it is written on a yellow note and attached to the risk it relates to. Information on the note should include at least the suggestion and maybe a rough cost estimate in implementation time or in other resource.

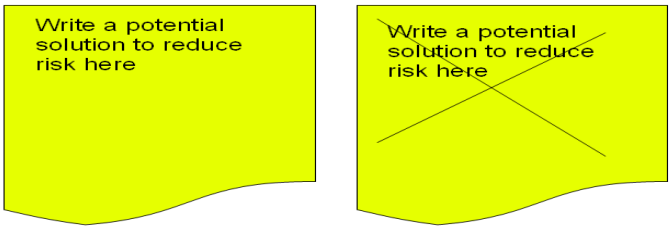

Figure 3: Risk response notes. The one on the left is an unattended solution and the one on the right is an implemented solution.

When a solution is crossed out, the red risk note should also be moved to a new position on the board. This is done to avoid confusion regarding current risks and to prevent one risk being moved twice, which might drastically affect the accuracy of the process. The strength of the risk board is that it makes risk management highly visible and hopefully easy to understand.

3.2 Checklists

To help in identification of high-risk components, checklists should be used. In this suggestion there are two checklists. One is based on the past experience and high-risk components. This list is more technical by nature. The other is based on security and abuse cases suggested in [2], and it is used as a reference card for general security requirements.

One checklist should be based on past experience or already existing listing of a high risk code, such as [3]. It is not always easy to tell what feature is a high-risk one, thus all features should receive a quick risk identification. It could be in the form of a brainstorm or some other quick and light-weight method. The checklist that includes high-risk components should be updated from time time.

Security and abuse cases should be another checklist, which sets the general requirements for feature security. As suggested in [2], the security and abuse cases are a great way to approach security from an outer perspective. Security cases approach the security from a users' perspective, so it is important to have a clear vision about the users' security needs. The baseline for security is set during planning phase. The idea is that the security and abuse cases are not booked as backlog items. They are used as a general reference that is compared to every feature. The person responsible for implementation of the feature should ask:

- What needs to be done in order to meet client's security requirements?
- Does the feature meet the requirements at the end of the sprint?

The latter could be part of the sprint DoD. In order to use security and abuse cases effectively, the team must have understanding of the security requirements and potential attackers. This allows them to direct risk identification to correct software components.

3.3 Process

The flowchart in Fig. 4 presents the idea how risks are identified, assessed and approved in the model.

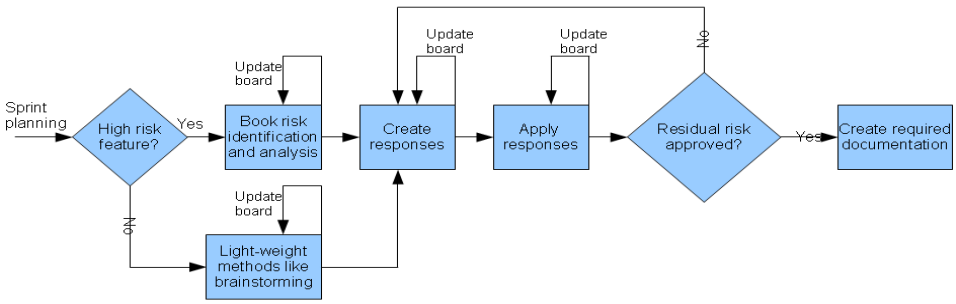

Figure 4: Flowchart of the process.

The flowchart does not include risk management planning, which is only done once as a part of project visioning. It also doesn't include risk monitoring. The table 1 describes what additional things are done during the Scrum meetings. This does not mean that risk identification, assessment and responses would only be done during these meetings, most of the work will be done between the meetings. X means that it should be done every time, P means it is possible if needed.

Table 1. Additional risk management activities in meetings.

Risk management in meetings	Identification	Assessment	Create response	Apply response	Risk approval
Sprint planning	X	X	X		
Daily Scrum	P	P	P		
Sprint review	P	P	P		X
Sprint retrospective					

Risks can be identified during all meetings apart from retrospective. As a risk is identified, one should also suggest the impact and probability it might have. Even if a feature receives a quick risk identification during sprint planning, this still means that further identification and assessment should be done by the person responsible for the feature. Identifying and assessing new risks during the sprint review is possible, but applying any responses will be a task for the next sprint.

3.3.1 Planning

This is only done once during the project lifetime. While it is not possible at this point to tackle the actual security issues in the software, this is still an important part of risk management. The planning is done when establishing a vision about what the software should be like. From security and risk management perspective it is important to set requirements and goals for security.

Security training is an important part of creating secure software. Security training and methods should also be discussed before starting to develop the actual software. Everyone

involved with the software should receive security training, including product owner and scrum master. The people responsible for accepting high-level risks should receive an adequate amount of security training in order to be able to make proper decisions.

Depending on the security requirements, general security level wanted and attacker profiles the teams can adjust the time used on risk management and guide the risk identification to items that are most likely attacked. The definition of done it should be discussed here. The question is, when is a risk so high that it prevents the deliverables from being shipped? This obviously depends on the security requirements of the software and will vary from project to project.

3.3.2 Risk identification, assessment and response

All features in the backlog deserve at least a quick risk identification. However, more in-depth identification and analysis should be done to those features that are considered having an elevated risk level. Just by reading the feature it is not easy to tell whether the code is high risk. The technical checklist should be used to identify high-risk components, which might require a comprehensive risk identification and assessment.

Risk management should officially be part of at least sprint planning and review. New risks should be identified when choosing the features the team will commit to. This is also great timing for brainstorming, since all the members should attend. If risk management is added to the agenda of these meetings, then the meetings will be slightly longer. Risks can be identified during any Scrum meeting, however daily scrum being only maximum of 15 minutes, it might end up consuming a large portion of the time.

The person responsible for implementing the feature is also responsible for making sure it receives risk identification, and if risks are found, assessment and risk response implementation. As soon as a risk is identified, it is written on a red note described in Fig. 2 and placed to the risk board. The same is done with risk solutions, however, yellow note is used and it is attached to risk it potentially can solve or mitigate.

When a feature is considered a high-risk, it means a new task is booked for the risk identification and analysis. This task is added to the sprint backlog. This is done before starting to work on the actual feature itself. It should be done by the same person or people who are responsible for implementing the feature.

The person responsible for a feature is also responsible for applying necessary risk responses, meaning he or she is the risk owner. The lack of clear risk ownership was considered as a problem during the interviews. Choosing which responses to apply is a crucial part of effective risk management. The following should be considered when creating and applying risk responses:

- Size of the risk.
- Risk response should be in proportion to the risk and feature.
- How important is the feature the risk is related to?

There might be large and problematic risks related to a feature which is not high priority. Creating and applying responses would take a large amount of time to complete a feature which is not important.

After a response has been applied, the note will be crossed over and the risk itself is moved to a new spot on the board. In a case where there are several responses to a risk, it is up to the person responsible of the feature to decide which ones are implemented and which ones are ignored. The best response depends on the security requirements of the software and the efficiency, resources versus mitigation, of the response.

3.4 Risk approval and monitoring

One of the problems that came up during the interviews was that it was not always clear who is responsible for accepting residual risks. This sub-chapter presents one solution. When accepting risks two things need to be considered, the nature and size of the risk.

At this point all risks and applied responses should be visible on the risk board. The board tells how high is the residual risks after the crossed over responses have been applied. The idea is, that if risk is 2 or smaller then the person who implemented the feature and managed its risks can approve the residual risk. In case of risk being over 2, the product owner must approve the risks.

If the product owner feels like he or she can't make the decision about accepting the risk, then the next approval depends on the organizational structure. It could be internal or external client, business decision maker or SoS-board (Scrum of Scrums). Problem with getting approval from them is that it is "unnecessary bureaucracy", which is something lean software development should avoid.

When the risk has been accepted, there is no longer any reason for it to be on the board. The accepted risks should be stored in a digital form for potential future use and re-evaluation. The larger risks accepted should be re-evaluated from time to time.

3.5 Summary

This suggestion included all traditional risk management steps. Summary of all the activities can be seen in table 2.

Table 2: Summary of security activities.

Activities	Suggestion	Done during
Plan	Identify security needs, make sure the team receives adequate training, create security cases and create attacker profiles. Create a checklist for high-risk features.	Once before the project starts
Identify	Decide whether the feature is high-risk or not. If the feature is a high-risk, book risk identification to sprint backlog, otherwise use light methods like brainstorming or delphi technique. Add identified risks to the risk board.	Before implementation of the feature
Assess	Try to assess impact and probability of the identified risks. Update the risk.	After identification
Respond	Create responses, make sure they're in proportion to total risk and	During feature

	the importance of the feature. Estimate the time required to implant the feature with the added time from response. Choose the best responses.	implementation
Accept	Make sure all the risks are accepted by the risk owner, product owner or client.	Sprint review
Monitor	Digitalize the risks and remove the risk from board. Make sure to include all the information. Re-evaluate larger risks between few sprints.	Between 2 to 5 sprints

4 Conclusion

Using traditional risk management methodologies such as PMBOK in an agile environment requires adjustments. Also the methodologies used to identify security risks in software are very different from ideas given in PMBOK. When this paper was written very little research had been done in the field of agile risk management. The best methods for risk management are always dependant on the whole infrastructure used to develop the software, so it is impossible to create one risk management framework which would meet everyone's requirements.

This paper created a model which can be used to manage risks in agile development environment. The results have yet to be verified by testing them in a real-life development scenario. Further studies could include testing the model and developing it further based on the results.

References

1. Claude J.(23.7.2009)., Agile risk board, a way to manage risks in agile enviroment. Agile-ux. (WWW) (Cited 29.11.2010). Retrieved from: http://www.agile-ux.com/2009/07/23/agile-risk-board/
2. Vähä-Sipilä A.(2010)., Product Security Risk Management in Agile Product Management. (WWW). (Cited 7.10.2010). Retrieved from: http://www.owasp.org/images/c/c6/OWASP_AppSec_Research_2010_Agile_Prod_Sec_Mgmt_by_Vaha-Sipila.pdf
3. Microsoft, SDL-Agile high risk code. (WWW). (Cited 30.11.2010). Retrieved from: http://msdn.microsoft.com/en-us/library/ee790613.aspx
4. Project Management Institute, inc. PMBOK: A Guide to the Project Management Body of Knowledge, fourth edition.