AGILE METHODS AND REQUIREMENTS ENGINEERING IN CHANGE INTENSIVE PROJECTS

Martin Fritzsche
Institut für Informatik, Technische Universität München, Boltzmannstr. 3, Garching b. München, Germany

Keywords: Agile software development, requirements engineering, eXtreme Programming, Scrum, Crystal, Dynamic Systems Development Method, Adaptive Software Development.

Abstract: In this paper we discuss how well agile methods can deal with requirements related issues in change intensive projects. Five agile methods are considered: eXtreme Programming, Scrum, Crystal, Dynamic Systems Development Method and Adaptive Software Development. We analyze how well these methods implement the basic goals of requirements engineering, how they counteract or support the occurrence of requirements changes and how they cope with problems arising from changing requirements. We show that agile methods provide a valid approach for requirements related issues, but also identify their weaknesses.

1 INTRODUCTION

Agile methods have become increasingly popular in recent years. They are mostly used for development in small projects focusing on web-based or mobile applications. However they have been reported to be also successful in larger projects (Elssamadisy, 2001, Cockburn and Highsmith, 2001). Agile methods claim to be able to cope with changing customer needs. In fact changing requirements are a widespread problem and dealing with them is deemed a critical success factor (Standish Group, 1995). This makes agile methods even more interesting.

In this paper we focus on the question how agile methods deal with requirements, particularly unstable requirements. We show that they provide specific practices that address requirements related issues in general and the problem of changing requirements in particular. However we also point out where their shortcomings are and in which areas adjustments would have to be made. In our analysis we focus on five methods: eXtreme Programming (XP) (Beck, 2000, Beck and Fowler, 2001), Scrum (Schwaber, 1995, Schwaber and Beedle, 2002, Schwaber, 2004), the Crystal Methodologies (Crystal) (Cockburn, 2002), Dynamic Systems Development Method (DSDM) (Stapleton, 1997) and Adaptive Software Development (ASD) (Highsmith, 2000).

The next section gives a brief overview on related work. In section 3 we present our approach to analyze agile methods in respect to their fitness for requirements engineering in a change intensive environment. We then perform the analysis. The last section summarizes our findings.

2 RELATED WORK

There are some contributions to the topic of the agile methods' fitness for requirements engineering worth mentioning. Leite (2001) extracts the requirements related elements from XP. He then identifies several aspects where XP has shortcomings in regard to requirements engineering and proposes possible solutions. However, while stating, that there are other problems, he doesn't list them. Leite's work provides a good starting point, but we want to carry out a thorough examination of the topic and consider also the problem of changing requirements.

Tomayko (2002) identifies several agile practices for requirements engineering. He concentrates on the problem of not being able to make correct estimates when confronted with unstable requirements.

Paetsch, Eberlein and Maurer (2003) provide a good overview of the requirements part of XP, Scrum, Crystal, DSDM and ASD. They then propose several requirements engineering techniques which should be incorporated by agile methods.

Fritzsche M. (2008).
AGILE METHODS AND REQUIREMENTS ENGINEERING IN CHANGE INTENSIVE PROJECTS.
In *Proceedings of the Third International Conference on Evaluation of Novel Approaches to Software Engineering*, pages 81-88
DOI: 10.5220/0001762900810088
Copyright © SciTePress

They explain how they can improve agile methods, but don't explicitly list the problems they see in the way agile methods operate. The same is done in an earlier work by Eberlein and Leite (2002). While these papers can be part of the solution, we first want to exactly define the problem that has to be solved. In a later step it would be useful to compare the solution proposed in (Paetsch, Eberlein, Maurer, 2003) with our work and analyze which problems can be solved by their proposal and which can't.

3 FITNESS OF AGILE METHODS FOR REQUIREMENTS ENGINEERING

Our goal is to determine in how far agile methods provide guidance concerning requirements engineering issues in change intensive projects, where their strengths and weaknesses lie, and where adjustments have to be made. In this chapter we suggest a qualitative approach for analyzing a method's fitness, apply the approach to XP, Scrum, Crystal, DSDM and ASD and discuss the findings.

3.1 An Approach to Analyze the Fitness of Agile Methods for Requirements Engineering

To analyze the fitness of a method for requirements engineering in change intensive projects several dimensions have to be considered. First we have to investigate in how far the method achieves the general goals of requirements engineering, i.e. issues that always have to be addressed, independent of change intensity. Second we have to analyze how the method mitigates requirements changes or whether it augments the occurrence of changes. We only consider sources of changes that can be influenced by the method. External changes like the introduction of new technology, changing laws or business strategies are not regarded. Third we have to discuss whether and how the method deals with problems arising from changing requirements.

For each of the three dimensions in our analysis we developed a catalogue of criteria that have to be considered. In each of the categories there are items of varying degree of detail as their numeration suggests. Subordinate items are issues that complement the associated superordinate items.

To each of the criteria we discuss the common aspects of the five agile methods and point out the specifics of individual methods.

3.2 Applying the Approach

3.2.1 Goals of Requirements Engineering

1. Discover the goals which are pursued by developing the system: Agile methods don't try to gather all goals of the system at the beginning of the project. Instead they develop only a rough sketch of the goals and refine them during the course of the project. They assume that not all goals can be known ad initio and therefore rely on learning processes during development. They employ iterations and frequent releases to support learning and feedback cycles. A project vision is explicitly developed by XP, DSDM and ASD. DSDM and ASD make use of prototypes to support the discovery of goals.

1.1. Discover all Stakeholders' Needs: The stakeholders' needs aren't collected completely at the beginning of the project but developed iteratively. Agile methods rely heavily on feedback from development and from the usage of the system to discover the real needs of the stakeholders. Stakeholders are involved in the project regularly or even continuously in the case of XP and Crystal. Scrum and DSDM address all stakeholder classes. DSDM even demands that workshops for stakeholder discovery are held. XP and Crystal strongly focus on users and customers, respectively. They lack in covering this goal because they don't consider all different stakeholder classes.

1.1.1. Define Requirements Necessary to meet the stakeholders' needs: Requirements are defined and refined iteratively during the course of the project. Their necessity and sufficiency can be evaluated through iterations and feedback from the use of the system. By intensively integrating the stakeholders into the project and giving them the competence to make decisions about requirements agile methods are able to relate the requirements to the stakeholders' needs.

1.1.2. Identify Rationale for Requirements: The analyzed methods don't explicitly address the issue of identifying and documenting the rationale for requirements. Since the stakeholders are involved throughout the project they can be asked about the rationale for the requirements if necessary. Though, usually they won't be able to remember every requirement's rationale.

1.1.3. Consider Changes to the Stakeholders' needs: Agile methods don't try to anticipate changes to the stakeholders' needs. Instead they integrate stakeholders into the project, so that they quickly are informed about changes to the stakeholders' needs. The methods can react flexibly to changes. At the

end of each iteration it is possible to change the direction of the project.

1.2. Gain a Broad Understanding of the Domain, the Organization and the Business Processes: Agile methods don't conduct a systematic analysis of the domain at the beginning of the project. Instead the domain knowledge is gained through the integration of domain experts and continuous learning due to iterations and feedback from the use of the system. XP and DSDM make a small domain analysis at the project beginning. DSDM and ASD feature prototyping to increase the understanding of the domain.

1.3. Understand the System's Impact on Business Processes: Agile methods don't conduct extensive studies to understand the system's impact on business processes at the beginning of the project. The impacts are directly observed using early and frequent releases. Led by the feedback gained from the system's use, the development can be adjusted accordingly. XP additionally employs spikes, while DSDM and ASD use prototypes to increase the understanding of the system's impact.

1.4. Assure the System's Profitability: After each release the project can be ended if future development isn't expected to be profitable. Scrum and XP make cost estimates. XP additionally demands that each story has to provide business value.

1.4.1. Determine Return on Investment of the System and Individual Requirements: Scrum and XP provide cost estimates. The business value each requirement provides influences the priority the customer assigns to them. A detailed analysis of the return on investment of the individual requirements is not carried out.

1.5. Define System Scope: This goal isn't explicitly addressed by agile methods.

2. Achieve the Most Important Goals which are Pursued by Developing the System: The order in which the goals are realized is set by their priorities. Therefore mainly unimportant goals remain unrealized if the project is cancelled before all goals are implemented.

2.1. Select the Necessary Requirements to achieve the Most Important Goals: XP's customer decides which stories should be implemented next. The developers have to discuss with the customer which steps are necessary to realize them. Scrum demands that the goals are broken up into detailed requirements. Thereby the method enforces the relation of the detailed requirements to the goals. Crystal and DSDM don't define how requirements are selected. ASD features JAD workshops for the selection of requirements. How exactly the requirements are selected isn't specified.

2.2. Realize the Necessary Requirements: XP applies a test-first approach. Tests are created before the code. When the tests work, the implementation stops. Therefore only that which is necessary is implemented. The other agile methods don't address this topic.

2.2.1. Consider Only Viable Requirements:

2.2.1.1. Understand which Requirements cannot be realized: Apart from DSDM the agile methods don't undertake a thorough feasibility study. Whether requirements are feasible is often discovered only during development. Though, there are some single techniques, agile methods employ. XP, DSDM and ASD make risk analyses. XP and Scrum make cost estimates. DSDM and ASD use prototypes.

2.2.1.1.1. Resolve Conflicts between Requirements Favouring High-priority Requirements: Agile methods don't perform a thorough analysis of conflicts between requirements. Conflicts may be discovered by chance during development. A fast reaction to identified conflicts is possible, due to short iterations. XP demands that stories should be independent, which lessens the probability of conflicts a bit. Conflicts may be discovered using spikes and conflicting tests. DSDM and ASD employ prototyping during which conflicts can be found.

2.2.1.1.1.1. Determine which Requirements Take Precedence in the Development: XP lets the customer decide on requirement priorities. In Scrum based projects priorities are set by the Product Owner in conjunction with the stakeholders. ASD demands that poorly understood but critical requirements take priority. Regarding other priorities no explicit statement is made.

2.2.1.1.2. Check Budget: Agile methods utilize time boxing to fix the costs within an iteration. The use of iterations makes it possible to flexibly adapt the length of the project, if the contracting allows it. Therefore one can react to budget problems accordingly. Long term planning of the budget is difficult because of unclear requirements. XP and Scrum make cost estimates to control the budget. In addition Scrum demands that the budget is managed empirically.

2.2.1.1.3. Check Schedule: Time boxing and iterations allow a rigorous control of the schedule and easy corrections. While short term plans are relatively well controlled, the long term planning of the schedule is difficult because of unclear requirements. XP's and Scrum's use of cost estimates makes a more realistic schedule possible.

2.2.1.2. Communicate the Feasibility of the Requirements to the Stakeholders: Apart from DSDM agile methods don't analyze the feasibility of the requirements at the beginning of the project. Therefore they can only communicate it to the stakeholders later, when it is known. Iterations and releases provide the opportunity to communicate with the stakeholders. DSDM communicates the feasibility of the requirements at the beginning of the project to the stakeholders during the feasibility study. Using XP or Crystal the communication is easier because customers are integrated into the team. In an XP project the developers consult the customer during requirements elicitation by estimating the feasibility of requirements.

2.2.2. Communicate Requirements to the Developers: XP employs story cards, acceptance tests and direct communication to communicate requirements to the developers. The method is very dependent on how good the customer can be integrated into the team to facilitate communication about requirements details. In addition it can be difficult to capture non functional requirements in test specifications. Scrum uses the Product, Release and Sprint Backlogs to communicate requirements. Crystal features use cases as requirements specification and integrates the customer continuously into the team to enable constant direct communication. DSDM communicates requirements using requirements lists and direct communication. ASD also employs direct communication via reviews and JAD workshops.

2.2.2.1. Assure that the Requirements are Comprehensible for the Developers: Agile methods employ direct communication to ensure that the requirements are comprehensible. The developers have the possibility to ask the stakeholders about things they don't understand. Misunderstandings can be discovered and rectified at the end of each iteration. XP specifies the requirements in an understandable way using tests. Scrum lists the details about requirements in the Product Backlog. Crystal's use cases can be written in an understandable way. DSDM and ASD employ prototypes to increase requirements understanding.

2.2.2.2. Define the Requirements Correctly: Direct communication and lists of requirements lack formal precision. Therefore the correctness is not assured. XP and Crystal offer with test specifications and use cases, respectively, notations which can provide enough precision if employed accordingly.

2.3. Assure Compliance with Requirements: Compliance with requirements is controlled through reviews at the end of iterations and through the use of the system. Apart from ASD all methods stress the importance of tests to control the compliance with requirements.

2.3.1. Identify Development Risks and Provide Preventive Actions and Contingency Plans: Agile methods don't try to anticipate problems and don't create risk mitigation and contingency plans. Instead they try to flexibly react to occurring problems. Iterations are an important instrument in this regard. XP, DSDM and ASD perform risk analyses. To explore risks XP employs spikes while DSDM and ASD make use of prototypes.

2.3.2. Communicate Possible Problems and Risks in the Development to the Stakeholders: Review meetings at the end of iterations provide a platform where problems and risks in the development can be communicated to the stakeholders. The risk analyses employed in XP, DSDM and ASD make communication of possible risks early in the project possible. XP, Crystal and DSDM can communicate problems and risks continuously because of the customer's constant integration into the team.

3. Enhance the Quality of the Product: Agile methods employ frequent stakeholder reviews to control the quality of the product. XP, Crystal and DSDM stress the importance of tests for quality control.

3.1. Assure Quality in the Process: XP and Scrum employ regular meetings where the team discusses how the process can be improved.

3.2.2 Reasons for Intensive Change of Requirements

1. Project internal factors:

1.1. Factors Concerning the Understanding of the Requirements: Agile methods don't strive for a complete understanding of the requirements at the beginning of the project. They rely on learning processes facilitated by feedback through frequent iterations and releases. They use simple and easy to understand communication channels. These are however not precise. Therefore misunderstandings may happen.

1.1.1. Requirements were Misunderstood at the Beginning of the Project: Requirements aren't analyzed in detail at the beginning of the project. Therefore they will be misunderstood often.

1.1.2. Conflicts between Requirements are found: Agile methods don't analyze requirements regarding conflicts at the beginning of the project. Therefore conflicts are discovered later during development and cause late requirements changes.

1.1.3. Priorities of Requirements Change: As requirements become more and more understood their priorities can change.

1.1.4. New Stakeholders are Introduced: Agile methods state that the stakeholders should be integrated right from the start of the project. However, only DSDM demands a systematic search for stakeholders. Crystal and XP focus on users and customers, respectively, and are likely to disregard other stakeholder classes.
1.2. Factors Concerning the Realization of the Requirements:
1.2.1. Requirements cannot be realized: Since requirements remain poorly understood for a long time, it will often happen, that late in the project the team discovers, that certain requirements cannot be realized. XP, DSDM and ASD provide with risk analyses a rudimentary way to identify requirements which cannot be realized. DSDM and ASD additionally employ prototyping.
1.2.2. Requirements cannot be realized within the given Schedule and Budget: The short planning horizon increases the precision of the plans. However due to the unclearness in requirements agile methods have difficulties to correctly estimate the cost of individual requirements and the project as a whole.
1.3. Factors Concerning the Organization:
1.3.1. Budget and Schedule Changes: Due to the short planning horizon short term budget and schedule are relatively precise. Because of unclear requirements the correct estimation of budget and schedule for the whole project, however, is very difficult. Requirements changes therefore arise because of wrong estimates.

3.2.3 Difficulties with Changing Requirements

1. Additional Effort is generated:
1.1. Stakeholders have to be Assembled again in Order to decide about the Acceptance of Changes: No additional effort is generated, because agile methods already assemble the stakeholders at the end of iterations. XP, Crystal and DSDM even involve the stakeholders continuously.
1.2. Artifacts have to be adjusted:
1.2.1. Identify Artifacts that have to be adjusted: Agile methods create only as much artifacts as deemed absolutely necessary. However as Leite (2001) points out for the example of XP, they don't provide traceability between the requirements and the parts of the artifacts that depend on them.
1.2.2. Modify, discard or add Artifacts: The associated effort is small in comparison to other methods, because agile methods create fewer artifacts.
1.2.3. Artifacts should be Extendable und Modifiable: Changes are everyday business for agile methods. Therefore artifacts are designed in a way that makes such changes easy. In addition only few artifacts are created, in order to cut the effort in updating multiple artifacts. XP and DSDM stress the importance of regression tests for reducing the effort necessary for incorporating changes.
1.2.4. Adjustments are Often Complicated and Error-prone: Artifacts are held simple and easy to change. Therefore agile methods mitigate this problem. XP and DSDM furthermore employ regression tests. XP also demands constant refactoring to improve the structure of the code and prevent it from decaying to the point adjustments get too complicated.
1.3. Initial Effort may be wasted when Requirements Change: Agile methods invest less initial effort in comparison to other methods. Therefore, if requirements change less, effort is wasted.
1.4. Requirements Changes may get denied because People Shy the Additional Effort: Changes are expected to happen and welcomed. Therefore it is not probable that teams using agile methods shy the effort to include changes.
2. Not all Conflicts can be detected at Project Beginning: Agile methods don't perform a conflict analysis at the beginning of the project. This is a problem they encounter even without changes.
3. Not Documented thoughts about Requirements are no Longer Present but needed when Requirements Change: Agile methods create very little documentation. Therefore this is a widespread problem. It is remedied a bit because stakeholders are involved throughout the project and still can be asked about their thoughts on requirements changes.
4. Project Plan has to be adjusted: Agile methods only put up detailed plans for the immediate future, so that less replanning is necessary. In addition the iterations make adjustments to plans easy.
4.1. Estimates have to be updated: XP, Scrum and DSDM update the estimates regularly. Crystal and ASD don't define which estimates are made and when they are updated.
4.2. Content of Releases and Iterations can Change: The short planning horizon minimizes the effort necessary to adjust the content of releases and iterations.
4.3. Schedule and Budget can Change: Due to the use of time boxing and fix team sizes the schedule and budget for a given iteration practically don't change. Agile methods have problems with long term planning though. They have great difficulty in fixing schedule and budget for the whole project.
5. Problems with costs arise:
5.1. System Costs are not known at Project Beginning: The requirements are not completely

understood at the beginning and no thorough analysis of the requirements is made. Agile methods can't correctly estimate the system cost at project beginning. (Tomayko, 2002).

5.2. Changes possibly cannot be realized within the Budget: The use of iterative planning allows for a flexible extension of the project if the budget can be increased to realize all the changes. XP, Scrum and DSDM make use of priorities to ensure that the most important requirements changes will be realized.

5.3. Late changes can be particularly costly: If the costs of late changes cannot effectively be controlled then agile methods cannot successfully be applied. Only XP offers concrete methods to counteract increasing costs: Object orientation, simple design, automated tests and refactoring.

3.3 Summary of the Findings

In Table 1, Table 2 and Table 3 we rate how the agile methods perform based on our findings. We differentiate three ratings: good coverage of the criterion (+), medium coverage (0) and bad coverage (-).

Table 1: Ratings of agile methods regarding their ability to reach general goals of requirements engineering.

	XP	Scrum	Crystal	DSDM	ASD
1. Discover the goals which are pursued by developing the system	+	+	+	+	+
1.1. Discover all stakeholders' needs	0	+	0	+	0
1.1.1. Define requirements necessary to meet the stakeholders' needs	+	+	+	+	+
1.1.2.. Identify rationale for requirements	-	-	-	-	-
1.1.3. Consider changes to the stakeholders' needs	+	+	+	+	+
1.2. Gain a broad understanding of the domain, the organization and the business processes	+	+	+	+	+
1.3. Understand the system's impact on business processes	+	+	+	+	+
1.4. Assure the system's profitability	+	+	0	0	0
1.4.1. Determine return on investment of the system and individual requirements	0	0	-	-	-
1.5. Define system scope	-	-	-	-	-
2. Achieve the most important goals which are pursued by developing the system	+	+	+	+	+
2.1. Select the necessary requirements to achieve the most important goals	+	+	-	-	0
2.2. Realize the necessary requirements	+	-	-	-	-
2.2.1. Consider only viable requirements					
2.2.1.1. Understand which requirements cannot be realized	0	0	-	+	0
2.2.1.1.1. Resolve conflicts between requirements favouring high-priority requirements	-	-	-	-	-
2.2.1.1.1.1. Determine which requirements take precedence in the development	+	+	-	-	0
2.2.1.1.2. Check budget	0	0	0	0	0
2.2.1.1.3. Check schedule	0	0	0	0	0
2.2.1.2. Communicate the feasibility of the requirements to the stakeholders	+	+	+	+	+
2.2.2. Communicate requirements to the developers	+	+	+	+	+
2.2.2.1. Assure that the requirements are comprehensible for the developers	+	+	0	0	0
2.2.2.2. Define the requirements correctly	0	-	-	-	0
2.3. Assure compliance with requirements	+	+	+	+	0
2.3.1. Identify development risks and provide preventive actions and contingency plans	-	-	-	-	-
2.3.2. Communicate possible problems and risks in the development to the stakeholders	+	0	0	+	0
3. Enhance the quality of the product	+	0	+	+	0
3.1. Assure quality in the process	0	0	-	-	-

Table 2: Ratings of agile methods regarding their ability to counteract reasons for intensive change of requirements.

	XP	Scrum	Crystal	DSDM	ASD
1. Project internal factors					
1.1. Factors concerning the understanding of the requirements	-	-	-	-	-
1.1.1. Requirements were misunderstood at the beginning of the project	-	-	-	-	-
1.1.2. Conflicts between requirements are found	-	-	-	-	-
1.1.3. Priorities of requirements change	-	-	-	-	-
1.1.4. New stakeholders are introduced	-	-	-	+	-
1.2. Factors concerning the realization of the requirements					
1.2.1. Requirements cannot be realized	0	-	-	0	0
1.2.2. Requirements cannot be realized within the given schedule and budget	-	-	-	-	-
1.3. Factors concerning the organization					
1.3.1. Budget and schedule changes	-	-	-	-	-

Table 3: Ratings of agile methods regarding their ability to deal with difficulties because of changing requirements.

	XP	Scrum	Crystal	DSDM	ASD
1. Additional effort is generated					
1.1. Stakeholders have to be assembled again in order to decide about the acceptance of changes	+	+	+	+	+
1.2. Artifacts have to be adjusted					
1.2.1. Identify artifacts that have to be adjusted	0	0	0	0	0
1.2.2. Modify, discard or add artifacts	+	+	+	+	+
1.2.3. Artifacts should be extendable und modifiable	+	+	+	+	+
1.2.4. Adjustments are often complicated and error-prone	+	0	0	0	0
1.3. Initial effort may be wasted when requirements change	+	+	+	+	+
1.4. Requirements changes may get denied because people shy the additional effort	+	+	+	+	+
2. Not all conflicts can be detected at project beginning	-	-	-	-	-
3. Not documented thoughts about requirements are no longer present but needed when requirements change	0	0	0	0	0
4. Project plan has to be adjusted	+	+	+	+	+
4.1. Estimates have to be updated	0	0	-	0	-
4.2. Content of releases and iterations can change	+	+	+	+	+
4.3. Schedule and budget can change	-	-	-	-	-
5. Problems with costs arise					
5.1. System costs are not known at project beginning	-	-	-	-	-
5.2. Changes possibly cannot be realized within the budget	+	+	0	+	0
5.3. Late changes can be particularly costly	+	-	-	-	-

Agile methods offer a different approach to requirements than traditional methods. They make use of frequent or continuous stakeholder involvement and intensive feedback and profit from learning processes to iteratively elicit, analyze and validate requirements. Instead of putting up detailed plans at the beginning of the project they plan iteratively and only for the immediate future.

While fulfilling many of the goals of requirements engineering their approach poses several problems. First of all some agile methods fail to recognize that besides customers and users there are other classes of stakeholders that also have to be considered. DSDM and Scrum however prove that even in an agile environment different stakeholder classes can be considered. The main problem of agile methods regarding requirements is that they gain only little understanding about the requirements at project beginning. Hence they can't define the system scope, conflicts can't be detected, estimates are likely to be wrong, many risks remain unknown and the correct planning of schedule and budget for the project is very difficult. All these factors contribute to later changes in the project. Agile methods acknowledge the fact that there always are changes that can't be prevented. Therefore they postpone many necessary

requirements engineering activities and thus cause additional changes. In our opinion they in fact are able to cope with changes reasonably well, so theirs is a valid approach. However they rely on two very critical points. On the one hand their success is dependent on their ability to effectively involve the stakeholders throughout the project, which isn't always possible. On the other hand they rely on changes not getting too costly. Changes that have great impact on the system can be very costly and can prove to be critical for the project, as is shown by Boehm (1981).

We suggest differentiating between different types of requirements, rating them according to their probability to change and their impact on the system. We think it would be advantageous for agile methods to follow a higher ceremony traditional requirements approach for requirements that are not likely to change and those who have great impact on the system while following their lightweight approach for other requirements. However this is just an assumption and remains to be proven empirically.

One still has to remember that another project characteristic that is often found in an agile environment is strong time constraints. The effort invested in requirements activities has to be balanced with their benefit. Though, if some changes in requirements can effectively be prevented by a little effort early in the project it may well be worth it.

4 CONCLUSIONS

Our aim in this paper was to show whether and how agile methods deal with requirements related issues in a change intensive environment. We showed that agile methods provide a valid approach but we also identified several weaknesses. Our work gives an overview over the problems agile methods have concerning requirements and can serve as a basis for further discussion. However we performed our analysis only qualitatively. Studies based on empirical data are necessary to solidify our findings. Furthermore, solutions for the identified problems need to be found and empirically evaluated.

ACKNOWLEDGEMENTS

This work has been supported by the German ministry of education and research (BMBF) within the project "Interorganisationale Softwareentwicklung unter dem Aspekt der Wandlungsfähigkeit und der Wiederverwendung" (IOSE-W^2), grant no. 01/SF 03.

REFERENCES

Beck, K., 2000. *Extreme Programming Explained – Embrace Change*, Addison-Wesley, Boston.

Beck, K., Fowler, M., 2001. *Planning Extreme Programming*, Addison-Wesley, Boston.

Boehm, B.W., 1981. *Software Engineering Economics*, Prentice-Hall, Englewood Cliffs.

Cockburn, A., Highsmith, J.A., 2001. *Agile Software Development*, Prentice-Hall, Englewood Cliffs.

Cockburn, A., 2002. *Agile Software Development*, Addison-Wesley, Boston.

Eberlein, A., Leite, J.C.S. do Prado, 2002. Agile Requirements Definition – A View from Requirements Engineering. In *TCRE'02, International Workshop on Time-Constrained Requirements Engineering (TCRE'02)*.

Elssamadisy, A., 2001. XP on a Large Project – A Developer's View. In *Proceedings of the XP Universe Conference*. Object Mentor Inc.

Highsmith, J.A., 2000. *Adaptive Software Development – A Collaborative Approach to Managing Complex Systems*, Dorset House Publishing, New York.

Leite, J.C.S. do Prado, 2001. Extreme Requirements. In *Jornadas de Ingenieria de Requisitos Aplicadas.*

Paetsch, F., Eberlein, A., Maurer, F., 2003. Requirements Engineering and Agile Software Development. In *Proceedings of the International Workshop on Enabling Technologies – Infrastructure for Collbaborative Enterprises*.

Schwaber, K., 1995. Scrum Development Process. In *OOPSLA'95, Workshop on Business Object Design and Implementation*. Springer Verlag.

Schwaber, K., 2004. *Agile Project Management with Scrum*, Microsoft Press, Redmond.

Schwaber, K., Beedle, M., 2002. *Agile Software Development with Scrum*, Prentice-Hall, Upper Saddle River.

Standish Group, 1995. The Chaos Report. http:// www.standishgroup.com.

Stapleton, J., 1997. *Dynamic Systems Development Method – The Method in Practice*, Addison-Wesley, Boston.

Tomayko, J.E., 2002. Engineering of Unstable Requirements Using Agile Methods. In *TCRE'02, International Workshop on Time-Constrained Requirements Engineering (TCRE'02)*.