Agile Requirements Definition: A View from Requirements Engineering

Article · September 2002

Source: CiteSeer

CITATIONS
45

READS
297

3 authors:

Armin Eberlein
University of Regina
126 PUBLICATIONS **1,820** CITATIONS

SEE PROFILE

Julio Cesar
Center for the Study of State and Society
11 PUBLICATIONS **87** CITATIONS

SEE PROFILE

Julio Cesar Sampaio do Prado Leite
Pontifícia Universidade Católica do Rio de Janeiro
256 PUBLICATIONS **3,717** CITATIONS

SEE PROFILE

Agile Requirements Definition: A View from Requirements Engineering

Armin Eberlein
Department of Elec.&Comp. Engineering
University of Calgary
Calgary, AB T2N 1N4
Canada
eberlein@enel.ucalgary.ca

Julio Cesar Sampaio do Prado Leite
Departamento de Informática - PUC-Rio
Rua Marquês de São Vicente 225 - Gávea
Rio de Janeiro – RJ 22453-900
Brasil
julio@inf.puc-rio.br

Abstract

Agile methods are an attractive alternative for those pressured to produce code fast. Many programmers like the hands-on strategy of these approaches which also help them avoid some of the less exciting tasks, such as specification. On the other hand, some people appear to welcome agile methods as an excuse to throw overboard everything that requirements engineering has been teaching. This position paper looks at numerous aspects of requirements engineering and argues about their suitability for agile approaches. The aim is to elicit lessons from requirements engineering that agile methods might consider, if quality is a major concern.

1. Introduction

There are several arguments for the necessity of good requirements practice. One of the simplest ones was stated by von Neumann: "There is no sense in being precise about something when you do not even know what you are talking about." Another argument is related to error correction. It is intuitive that is less expensive to repair an error as we define a product than it is to fix it when the product is ready. The literature have been reporting rates that range from 1 to 20 up to 1 to 200 on error correction cost as we move from the definition to the implementation and later to maintenance [3].

These arguments are substantiated by industry reports. The Standish and Gartner groups have, several times, conducted surveys that point out that project failures are perceived as caused by a lack of proper attention on requirements processes. One of the most recent surveys from the year 2000 reports that only 26% of software projects are considered successful, so 74% are not. Detailing the causes of success or failure the keyword that is more frequent is requirements. The CHAOS report published in 1995 [5] shows that almost half of the cancelled projects failed due to a lack of requirements engineering effort and that a similar percentage ascribes good requirements engineering as the main reason for project success. Successful projects do manage requirements, failed ones lack requirements processes. A recent survey from McPhee [15] showed that senior software developers and project managers believe that requirements activities should account for 25% of the total development effort. This was the case although the survey focused on projects that have a critical time-to-market.

If we agree that the arguments above are sound, how can a software process be successful if it does not tackle requirements? Nevertheless, there are many companies that do not practice good requirements engineering. One of the main reasons given for skipping the requirements engineering phase is lack of time. Too often, there is high pressure to deliver something to the customer as soon as possible to keep him/her happy. For this reason, a lot of companies have recently shown great interest in the newly emerged agile methods, such as Extreme Programming.

However, several factors do pose an obstacle to the overall dissemination of requirements engineering practices by industry. Leite [12] has enumerated four of them: a) lack of formal education, university level, on requirements engineering; b) more and more customers demand constant evolution on very tight schedules; c) the model "everything" mentality; d) the technology transfer problem, mainly with respect to packing.

The agile community claims that they do tackle requirements, but we think that this is poorly performed requiring more validation cycles than necessary and relying too much on individuals. This, in the long run, may bring severe problems to the software organization responsible for software built following an agile method.

However, there is anecdotal evidence that projects that use agile methods have a high productivity. However, this evidence is not yet substantiated and it is uncertain what the real reasons are for the improved productivity. Is it because the team consisted only of the best programmers of the company? Or maybe productivity improved because one of the best consultants available was hired?

In fact, pretty much all projects that used agile methods and experienced higher productivity have to answer these questions with "yes".

We will briefly describe four good practices of requirements engineering that we believe could enrich agile approaches. We will cover Customer Interaction, Analysis (Validation and Verification), Non-Functional Requirements and Managing Change.

2. Customer interaction

The importance of involving customers in the development process has long been recognized. Reviews and feedback from practitioners show that direct and regular interaction with the customer is one of the key factors for project success. Projects that fail tend to have a great lack of early customer feedback. The Requirements Engineering (RE) community has therefore been involved in the establishment of techniques that improve this interaction. We use the word customer in order to distinguish between two types of customers: users and clients. Users are individuals that interact directly with the information system and clients are individuals that have needs that will be fulfilled by the software system. Customers as well as developers are stakeholders in the process of software construction.

Work in this area has been inspired by cognitive psychology, anthropology, sociology and linguistics [16]. Cognitive psychology helps understand the difficulties people have in expressing what they want to get from a system. Anthropology has shown the importance of observing people in order to find out how a system can help them in their work. Sociology can help in the interaction between stakeholders. Linguistics addresses the important issue of communication and the clear expression of requirements.

In a first step, all relevant stakeholders have to be identified. Stakeholder identification needs to go beyond the selection of one vocal client. It is important to identify all the individuals or organizations that have a say in the development of the system or are affected by its success or failure. Even stakeholders, like the users, are not homogeneous and it is important to determine the requirements of these different actors. Leite in [12] has pointed out that XP [1] fails to take this into consideration, i.e., there are several actors acting as customers, not just one. This, in turn, gives rise to issues related to viewpoints [22].

After the identification of stakeholders, requirements have to be elicited. There are several techniques that can be used, like interviews, questionnaires, requirements recovery, prototyping, discourse analysis, ethnography and reuse. It is during this elicitation process that intense customer interaction has to occur.

The agile movement is very strong in its focus on customer interaction. However, most of this interaction is done by prototyping. Several agile methods, XP for instance, assume that the client already knows everything that needs to be known. This is unfortunately not often the case. However, it is a question of how much risk is involved by not spending sufficient time selecting the right customers and prioritizing them and their respective requirements. For instance, your system end user might not bother about any safety regulations that a system might have to meet. Of course, as we pointed out before, it is almost impossible to have all your requirements from just one person. So it is true that there is still the need to consider elicitation as a process that gathers requirements from different viewpoints [14].

The agile community can also learn from various interviewing techniques that have been in use in the RE community. The importance of context-free questions, open questions and meta questions is also valid in an agile environment. If there are conflicts between stakeholder requirements, the use of Analytical Hierarchy Process (AHP) [21] or other prioritization tools [19] can help. The same is valid for Joint Application Development (JAD) [25] which promotes the use of a professional facilitator who can help resolve conflicts. Boehm's work on WinWin [2] can also be beneficial. Distributed development is another point that is not really addressed in agile processes. The agile community could benefit from work on distributed requirements negotiation and the impact of the communication medium on the quality of negotiation as is done by Damian et al [8].

3. Analysis (Verification & Validation)

Agile methods rely fundamentally on the use of validation (testing), not only as a way of achieving quality by getting rid of errors, but also as a way of identifying requirements, that is by prototyping. It is particularly interesting that agile processes do not address aspects of verification[1].

Requirements Engineering (RE) has inherited several ideas from software engineering with respect to applying formalisms for the modeling of requirements. Using formal modeling RE can provide the basic infrastructure for analysis.

[1] We should keep in mind the classical distinction [23] (p. 446) of verification "are we building the product right?" and validation "are we building the right product?". So test (validation) is different from inspection (verification), for example.

A major advantage of having early formalisms is the possibility of performing automated consistency checking, either intra-requirements or inter-requirements, that is checking consistency based on the semantics of the modeling language or checking consistency by comparing, for instance, two similar descriptions for the same set of requirements.

The work of van Lamsweerde [24] is an example of applying consistency checking in a given requirements model. The work of Nuseibeh et al [17] is an example of checking consistency between different descriptions of the requirements, and the work of Reubenstein is an example of checking the requirements against those previously defined and contained in a knowledge base [20].

Besides the use of automation for verification of requirements, there is the possibility of applying the concepts of inspections [9] to requirements. In this case, verification is performed by using well established checklists (meta knowledge) that should be applied to the requirements by a well established process. An example of this is the work performed by Gough et al in scenarios inspection [11].

We are convinced that agile methods would benefit if they additionally included verification together with validation. Simple tools to check early requirements descriptions associated with effective management practices for applying inspections, of course upon availability of check-lists, can improve the quality of agile processes, which usually only rely on validation. An example of such tools could be a syntax driven editor for user stories. Another example is a test case generator based on user stories.

4. Non-functional requirements

Unfortunately, it is common to read about failures due to the lack of consideration of non-functional requirements (NFRs) [4]. Considering non-functional aspects only at the design level has caused several problems. However if the non-functional aspects are considered only at the implementation level, as we understand is the case for agile methods, then the problems will be even greater.

Considering non-functional aspects with customers is a tricky business. It is seldom that clients and users do refer to the specifics of transaction time, ease of use, levels of confidentiality, and so on. Cysneiros [7] has been reporting this kind of difficulty in the realm of health care applications, and others as well have experienced similar difficulties when eliciting non-functional requirements from customers in other domains.

The work of Chang et al [6] on the NFR framework has provided important insights for researchers on how to deal with NFRs in the context of requirements definition. In particular the NFR graph that makes it possible to refine non-functional goals into operations at the functional level. Using NFR graphs also helps identify conflicts between non-functional requirements.

We understand that there is a need and an opportunity for agile methods to include techniques that make it possible to identify non-functional requirements early on and to describe them in such a way that an analysis may happen before implementation. Ideally these NFRs would have to be dealt with at the same time that User Stories [1] are scoped with customers.

5. Managing Change

One of the most important aspects of agile methods is that change is a built-in aspect of the process. Since you develop software in small releases and use refactoring as a frequent practice, change is intrinsic to agile methods. We believe this is an important characteristic to this kind of processes. However, managing change is believed to be a fundamental practice in software engineering. The CMM model states explicitly that an organization can only be considered level 2 if it has implemented the key area of requirements management.

Requirements management is a precondition of managing change based on the customers' expression of needs, i.e., the requirements. Central to requirements management is the concept of requirements traceability [18][10]. As such it is important to have descriptions of the requirements and the paths that link these requirements to their source as well as to code. We believe that agile methods cannot afford not to have requirements management, even if they aim at having the code as the main document of the software process. In XP there are User Stories, but XP reports fail to use these User Stories as a baseline for requirements management.

As such, we believe that agile methods should not only rely on configuration management practices, aimed at the code level. We must find ways of adapting requirements management practices for agile processes, as a way of providing traceability from customers' intentions to actual implementation.

6. Conclusion

Our aim in this position paper was to list some aspects that we see of major importance in any requirements engineering process and that appear to be ignored in the agile movement. Our idea was to ignite a debate and to sketch out a path how essential RE practices

can be adapted in the context of agile main ideas (simplicity, short releases, continuing validation and frequent refactoring).

7. Bibliography

[1] [Beck 00] Beck, K., Extreme Programming Explained Embrace Change, Addison Wesley Longman, Inc., (2000).

[2] [Boehm 01] Boehm B.W., Grünbacher P., Briggs B.: Developing Groupware for Requirements Negotiation: Lessons Learned, IEEE Software, May/June 2001, pp. 46-55 (2001)

[3] [Boehm 81] Boehm B.W., Software Engineering Economics, Prentice Hall (1981)

[4] [Breitman 99] Breitman, K.K., Leite, J.C.S.P, Finkelstein, A The World´s at Stage: A Survey on Requirements Engineering using a Real-Life Case Study Journal of the Brazilian Computer Society: Vol.6, N. 1, Pags. 13 - 37 (1999)

[5] [Chaos 95] CHAOS, Software Development Report by the Standish Group (1995).

[6] [Chung 99] Chung L., Nixon B. A., Yu E. and Mylopoulos J.v Non-Functional Requirements in Software Engineering, Kluwer Academic Publishing (1999).

[7] [Cysneiros 02] Cysneiros, L.M. Requirements Engineering in Health Care Domain, In Proceedings of the RE02 International Conference on Requirements Engineering, IEEE Computer Society Press (to appear).

[8] [Damian 00] Damian D., Eberlein A., Shaw M. and Gaines B. Using different communication media in requirements negotiation, IEEE Software 17, (3), pp. 28-35 (2000)

[9] [Fagan 76] Fagan, M.E. Design and Code Inspections to Reduce Errors in Programming Development", IBM System Journal, Vol. 15, N. 3, Jul- 1976, pp. 182-211. (1976)

[10] [Gotel 94] Gotel, O. Finkelstein, A. An Analysis of the Requirements Traceability Problem. In First IEEE International Conference on Requirements Engineering (ICRE), IEEE Computer Society Press, pp.94-101. (1994)

[11] [Gough 95] Gough, PA., Fodemski, F.T., Higgins, S.A, Ray, S.J. Scenarios – an Industrial Case Study and Hypermedia Enhancement, in Proceedings of the IEEE International Symposium on Requirements Engineering, IEEE Computer Society Press, pp. 10-17, (1995)

[12] [Leite 00] Leite, J.C.S.P "Is there a Gap between RE Research and Re Practice?" in Proceedings of the Fourth Intl. Conference on Req. Engineering, IEEE Computer Society Press, 73-74, (2000)

[13] [Leite 01] Extreme Requirements in Jornadas de Ingeniería de Requisitos Aplicada, Universidad de Sevilha, Sevilha,Spain, (http://www.lsi.us.es/~amador/JIRA/Ponencias/ JIRA_Leite.pdf) (2001)

[14] [Leite 91] Leite, J.C.S.P, Freeman, P. A. "Requirements Validation Through Viewpoint Resolution" IEEE Transactions on Software Engineering: Vol. 17, N. 1, pp: 1253 -- 1269, (1991).

[15] [McPhee 02] McPhee C. and Eberlein A. Requirements Engineering for Time-to-Market Projects, Proceedings of the 9th Annual IEEE International Conference on the Engineering of Computer Based Systems (ECBS2002), Lund, Sweden (2002)

[16] [Nuseibeh 00] B. Nuseibeh and S. Easterbrook. Requirements Engineering: A Roadmap. In A. Finkelstein, editor, "The Future of Software Engineering", Special Volume published in conjunction with ICSE 2000 (2000)

[17] [Nuseibeh 94] Nuseibeh, B. Kramer, J., Finkelstein, A.C.W. A Framework for Expressing the Relationships Between Multiple Views in Requirements Specification. IEEE Transactions on Software Engineering, 20(10): 760-773, (1994)

[18] [Ramesh 01] Ramesh, B., Jarke, M. Towards Reference Models of Requirements Traceability, IEEE Transactions on Software Engineering, 27(1) 58-93, (2001)

[19] [Regnell 01]: Björn Regnell, Martin Höst, Johan Natt och Dag, Per Beremark and Thomas Hjelm An Industrial Case Study on Distributed Prioritization in Market-Driven Requirements Engineering for Packaged Software, Requirements Engineering Journal 6:51–62 (2001)

[20] [Reubenstein 91] Reubenstein H.B. and Waters R.C. The Requirements Apprentice: Automated Assistance for Requirements Acquisition, IEEE Transactions on Software Engineering, 17, (3), pp. 226-240 (1991).

[21] [Saaty 80] Saaty T.L.: The Analytic Hierarchy Process, Wiley, New York, (1980)

[22] [Silva 02] Silva, A. Requirements, domain and specifications: a viewpoint-based approach to requirements engineering. In Proceedings of ICSE (IEEE International Conference on Software Engineering), IEEE Computer Society Press, (2002)

[23] [Sommerville 95] Sommerville, I. "Software Engineering", Addison-Wesley, (1995)

[24] [van Lamsweerde 98] van Lamsweerde, A. Darimont, R. Letier, E. Managing Conflicts in Goal-Driven Requirements Engineering, IEEE Transactions on Software Engineering, 24(11), 908-926 (1998)

[25] [Wood 95] Wood, J. and Silver, D. Joint Application Development, Wiley (1995)