A Reflection on Agile Requirements Engineering: Solutions Brought and Challenges Posed

Irum Inayat
Faculty of Computer Science & Information Technology
University of Malaya
Kuala Lumpur, Malaysia
irum@siswa.um.edu.my

Lauriane Moraes
Computer Science School
PUCRS
Porto Alegre, Brazil
lauriane.moraes@acad.pucrs.br

Maya Daneva
Information Science Research Group,
University of Twente,
Enschede, The Netherlands
m.daneva@utwente.nl

Siti Salwah Salim
Faculty of Computer Science & Information Technology
University of Malaya
Kuala Lumpur, Malaysia
salwa@um.edu.my

ABSTRACT
The software development industry has rapidly accepted agile methods. Empirical studies suggest that due to their flexible and emergent nature, agile methods brought solutions to several chronic problems of traditional software development methods. One among the many is the acceptance of requirements changes at later stages of development. However, knowledge about the solutions that agile brought to requirements engineering (RE) is fragmented. Also, little is known about whether the agile philosophy, while introducing solutions to well-known RE problems from the past, has unintentionally opened new challenges. This paper offers a reflection on this matter. Based on the results of our recently published systematic review on agile RE, we reflect on the differences of 'traditional' and agile RE and the practices adopted by the latter, on the solutions and challenges of agile RE, and on some implications that agile RE might have posed for research and practice.

Categories and Subject Descriptors
D.2.1 [**Software Engineering**]: Requirements/Specifications – *elicitation methods (rapid prototyping, interviews, JAD, etc).*

General Terms
Documentation, Design, Human Factors.

Keywords
Agile Software Development, Requirements Engineering, Agile Requirements Engineering, Traditional Requirements Engineering, Requirements Engineering Practices, Agile Requirements Engineering Challenges.

1. INTRODUCTION
Software development is a social process relying on stakeholders' collaboration. Agile methods belong to the class of software development methods that incorporate frequent stakeholders' collaboration, iterative development and acceptance of requirements changes even at later stages of development [1].

Unlike in 'traditional' software development methods (e.g. waterfall), goals are defined for each iteration and are revisited once it is done in agile software development. Compared to traditional software development approaches, agile methods offer dynamicity by satisfying customer needs in less time and flexibility by welcoming requirements changes at later stages of the development cycle. Empirical evidence suggests that due to the flexible nature of the agile methods, projects that deployed them have outperformed those using traditional software development methods, e.g. higher performance and better product quality [2].

Though there are studies that describe requirements engineering (RE) practices that are also feasible for agile methods (e.g. [3]–[7]), the software development community still lacks comprehensive knowledge about the solutions that agile RE introduced to 'traditional' RE issues, nor if while introducing those solutions new challenges were opened along the way. Empirical evidence does exist (e.g. in [8]), but it often addresses a particular RE aspect, e.g. inter-iteration re-prioritization of requirements [8]–[10] or scaterred over several sources. Experiences on agile RE are fragmented and, in turn, it is hard to see what agile RE solutions are accompanied by new challenges and what solutions are not. In our recently published systematic literature review on agile RE [11], we identified (i) agile RE practices, (ii) traditional RE challenges resolved by agile RE and (iii) the challenges posed by agile RE to the software industry, contributing to compiling all these topics in a single source aiming to facilitate access to those interested in them. Our systematic literature review included literature published between 2002 and 2013. In sum, we identified 17 practices of agile RE, 5 traditional RE challenges that were overcome by agile RE, and 8 new challenges posed by agile RE to the software industry. Details on how we conducted the review and on the findings can be found in [11].

In this paper we use the insights gained in our review as the basis for our reflection on agile and 'traditional' RE. Our goal is to create awareness that the topic needs more empirical evidence and discussion if we want to help industry to ease their processes and improve performance. Our reflection is based in our experience as researchers and former practitioners who came to academia looking to learn how to solve issues like the ones discussed in this paper.

XP 2015 Workshops, May 25 - 29, 2015, Helsinki, Finland

DOI: http://dx.doi.org/10.1145/2764979.2764985

Table 1. Comparison of traditional and agile RE

Traditional Methods	Agile Methods
RE processes [12]	
Focused on gathering all the requirements and preparing the requirements specification before the design phase	Iterative requirements development throughout the development cycle
Models [2][13]	
Value fully specified problems, rigorous planning, pre-defined processes, and documentation to support the development of activities and to record decisions made	Value individuals and interactions over processes; working software over comprehensive documentation; customer collaboration over contract negotiation; and responses to change over following a plan
Planning [2][13]	
Process activities are planned in advance and progress is measured against the plan	Planning is incremental and is easier to change processes to reflect new decisions based on identified working needs

The remainder of the paper is organized as follows: Section 2 explains the key differences between traditional and agile RE. Section 3 describes the adopted practices of agile RE. Section 4 explains the challenges of traditional RE resolved by agile RE. Section 5 introduces the new challenges posed by agile RE to the software industry. Section 6 concludes the paper and defines implications for practitioners.

2. TRADITIONAL VS AGILE REQUIREMENTS ENGINEERING

In this section, traditional and agile RE are discussed with respect to their differences. Some of the main differences of traditional and agile RE are summarized in Table 1.

2.1 Traditional Requirements Engineering

Requirements reflect the needs of customers for a system that serves a certain purpose, e.g. controlling a device, placing an order, or finding information. The process of finding out, analyzing, documenting and checking these 'services' and their related constraints is called RE [13]. The aim of RE is to help the software team to know what to build before system development starts in order to prevent costly rework. RE refers to all software life-cycle activities concerning requirements [14] i.e. elicitation, analysis and negotiation, documentation, validation, and management [13][14]. RE when considered in such context and while using 'traditional' software development methods such as the waterfall model is termed 'traditional RE' [4].

In traditional requirements elicitation, users are involved in gathering requirements; plus, elicitation also includes understanding the application domain, business needs, system constraints, stakeholders and the problem the software system is expected to solve as a whole. Then, the elicited requirements are analysed for necessity, consistency (requirements should not be contradictory), completeness (no service or constraint is missing), and feasibility (requirements are feasible in the context of the budget and schedule available for the system development) through JAD sessions, prioritization, and modelling to resolve conflicts. The selected requirements are then written down in a requirements document to communicate what was defined to stakeholders and developers. Requirements are next validated to confirm the customers' real needs in order to avoid errors in a requirements document that can lead to extensive rework if discovered during development or after the system is in service. Last is the requirements management phase which deals with changes in requirements originated by changes on the stakeholders' understanding of what the system is expected to do that happen during the software development process. The system requirements must then also evolve to reflect this changed problem view and this is done through requirements management. Such changes are not welcomed and are often source of rework, delays and lack of satisfaction from the customer in the end.

However, in agile software development methods agile teams take a set of certain requirements, called user stories [16] in each iteration (called sprint) and develop a workable output to gather customer's feedback for rework if not satisfactory and start the next iteration with the rest of user stories. The fixes of the previous iteration with the development of the current one are done simultaneously. Therefore, we can see that agile methods are iterative, emergent and flexible to cater changes in requirements differently as compared to the linear approaches of traditional software development [17][18].

2.2 Agile Requirements Engineering

Agile methods are emergent and exploratory in which "knowing and action" are simultaneous [19] and are driven by business value and cost (return on investment) and not by the completeness of the implemented requirements [20]. This inseparable relation between eliciting user needs and developing them at the same time makes agile methods different from linear sequence following traditional methods [19] by offering more business value [21]. This difference in the theme of agile methods and traditional software development methods intrigues an interesting question: *How requirements are done in agile methods?*

In the past decade, the merging of agile methods and RE has been debated a lot. However, the question of which steps to follow to deal with user requirements in agile methods from scratch still needs attention. Though there are a handful of studies (e.g. JAD based requirements gathering in agile methods [22]) that describe how requirements are carried out in an agile way, still a consolidated view of agile RE to facilitate practitioners' understanding is needed. Therefore, we start by presenting a consolidated view of on how to perform RE activities in agile methods based on literature sources [12][23][24][25][26]. Agile methods involve continuous planning [27], i.e. release planning, iteration planning and task level planning. Iteration planning is done for each iteration that spans from 1 to 3 weeks. It involves user story estimation, acknowledgement of the accomplishments of the previous iteration and determining overall progress and goals for the next iteration. Release plan is done for each release in which iteration length is decided, developers and customers unanimously decide what will be in a particular iteration; velocity points are determined per iteration. Task level planning involves the breaking down of user stories into subsequent tasks, allocation of tasks among team members and focus is put on implementation issues. The overall agile RE process is explained below.

2.2.1. Role modeling

Role modeling of agile team comes first. It involves stakeholder identification, defining stakeholder involvement level, and building stakeholder trust [25]. It is foremost to identify stakeholders to ensure: (i) who is going to define the project scope, (ii) who is going to decide the budget and timing issues, (iii) who will maintain the business and development team

relationship, (iv) who will provide support to the teams, and (v) who will be the system users, among others. The stakeholders are then classified into categories based on their interaction level with the team and product. This creates user personas that eventually help in role modeling. Roles are defined and their needs are allocated with respect to potential system usage as user stories. It also involves dealing with proxy roles or customer representatives from the business stakeholders. Therefore, in this phase stakeholder roles, responsibilities, and needs are decided upon.

2.2.2. User story creation

User story is defined as a unit of functionality in agile methods. It specifies the customer requirements in a brief and concise way. User stories are not detailed like the traditional requirements specifications. They are short, direct and understandable by the stakeholders. The large and complex user stories are split into smaller ones based on ease of implementation. In this section, user story gathering, writing and acceptance testing of user stories are explained.

2.2.2.1. User story gathering

User story gathering goes on throughout the project in agile methods because user requirements keep on evolving, coming and going throughout the project. Therefore, user story evolving, elicitation, and development keep on going in an iterative manner. User stories are gathered through questionnaire, interviews, observations and workshops, and written in story cards. According to the proposed methods (e.g. [23]) non-functional requirements (NFRs) are also elicited from the customers side-by-side the functional requirements. So NFRs are also gathered through questionnaires, workshops and observation methods.

2.2.2.2. User story writing

User story writing workshops are arranged in which development team and customer jointly write the user stories. Such workshops are arranged in the beginning of each planned release. Estimation and priorities are not associated with the user stories at this stage. Low fidelity prototypes are designed to properly translate user demands into user stories for development.

2.2.2.3. Acceptance testing user stories

Tests are created and applied for each of the defined user stories. These tests confirm the correctness of the user stories. In addition, acceptance tests add to the usability and functionality of the user story. The acceptance tests make all the possible stones unturned where customer's imagination might go. These acceptance tests determine the completeness of a user story implementation. In practice, acceptance tests are small notes written at the back of story cards.

2.2.2.4. User story estimation

To measure the story points, agile teams allocate points to the stories and use these arbitrary values to measure the effort required to complete that user story. These points can be allocated in many ways based on the team's preferences. In most cases, project managers define a story point complexity range as a Fibonacci series (for example, 1, 2, 3, 5, 8). Another way is to pick a small reference story and estimate the other ones with reference to that using the Delphi estimation technique known as "planning poker" [27]. The stories are rated and reviewed before entering the next phase of prioritization.

2.2.2.5. User story prioritization

User stories are prioritized before iteration. The user stories are prioritized in terms of ranking (i.e. ordered as first, second and third) and also as group (i.e. 'high priority', 'low' and 'medium'). The high priority user stories are recorded in the product backlog and used as a guide to carry out the development work. Customers perform the prioritization task based on their understanding of the business value the user stories will bring using various techniques (e.g. dynamic reprortization of requirements [8], value-based requirements priortization [9] client-driven requirements prioritization [28], and risk-based requirements priortization [29]) but with due input from the development teams.

2.2.2.6. Disaggregating into tasks

The user stories are divided into smaller tasks for ease of development. The stories that are fairly small are highly interdependent are implemented as they are. Teams divide the user stories based on their own personal instinct and criteria.

2.2.2.7. User story allocation

The user stories divided into tasks are allocated to the developers. This is decided during planning meetings. Agile teams perform these allocations based on their consensus and discussion.

Our reflection on evidence published in empirical studies helped us create an overall picture of how agile RE happens across projects and organizations. Each iteration repeats the same process of user story creation, prioritization, estimation and implementation and the final output is shared with the customer to gather feedback and improve the product through retrospectives. Unlike in the phase-driven linear traditional RE, clients' changes can be accommodated at any time during the development lifecycle. Involving customer in user story prioritization also reduces the chances of incoming change requests that often. It also helps customers to make realistic expectations from the output of each iteration. Therefore, significant differences can be seen in the process of carrying out RE activities in agile methods as compared to the traditional methods. The flow of activities described in this paper defines the agile way of dealing with requirements. It invites the researchers to provide empirical results by conducting studies in agile based software development environments. In addition, the empirical evidence and experimental results will help shape up this process and flow of activities in a better manner.

3. ADOPTED PRACTICES OF AGILE REQUIREMENTS ENGINEERING

In our recent systematic review on agile RE [11] we identified 17 agile RE practices from literature. These practices are summarized in Table 2. In particular, we found several traditional RE practices particularly used for requirements gathering (i.e. observations, interviews, workshops) used for user story elicitation and creation in agile methods [3]. In addition, there are several other traditional RE practices used in agile RE such as customer involvement [17], face-to-face communication [7][8], requirements modeling [6] using techniques like goal-sketching [6]. Agile methods strongly advocate stakeholders' collaboration so customer involvement and face-to-face communication is an integral part of project development.

In particular, agile RE practices include user stories based on customer demands, user story prioritization [7][8] continuous planning, pairing [7][8], change management [4], cross functional teams, prototyping, testing before coding, review meeting, acceptance tests [30], and shared conceptualization [31]. Moreover, agile method propose iterative development of requirements to help making requirements less fragile [4].

Table 2. Summary of agile RE practices [11]

Practice	References
Acceptance tests	[4][5]
Change management	[4][5]
Code refactoring	[39]
Cross-functional teams	[32]
Customer involvement	[29][4][5]
Face-to-face communication	[4][5][17][33]
Iterative requirements	[4][5] [33]
Pairing for requirements analysis	[40]
Prototyping	[4][5][36]
Requirements management	[4][5]
Requirements modelling	[6][38]
Requirements prioritisation	[4][5][29] [33][35]
Retrospectives	[4][5] [33]
Review meetings acceptance tests	[4][5]
Shared conceptualisations	[31]
Testing before coding	[4][5][33][37]
User stories	[34][32]

Furthermore, agile methods proclaim iterative and gradual detailing of requirements [32] with continuous planning and retrospectives after each iteration. Prototyping helps the customer to provide feedback on requirements and enhances quicker feedback [5] . Therefore, we can say that these RE practices including traditional RE practices and practices particularly used in agile methods equally contribute to requirements development in agile project development.

4. TRADITIONAL RE CHALLENGES RESOLVED BY AGILE RE

Agile methods tend to solve several issues of traditional RE due to flexibility and dynamic workflow. The summary of several traditional RE challenges identified in our systematic literature review [11] and solutions to resolve them provided by the use of agile RE are shown in Table 3.

In traditional RE the major issue is the lapse of information exchange between relevant people, called communication gap [41]. Agile methods serve the purpose to resolve communication lapses among teams [32][30] through frequent face-to-face communication [42]. In traditional methods, customers usually get to see the output product after the completion of development and testing phase [30]. This minimal customer involvement in traditional software development like the waterfall model creates requirements changes at later stages. The frequent and face-to-face communication in agile methods solves this problem. Likewise, the gradual and iterative detailing of requirements in agile methods provides solution to the issue of over-scoped requirements in traditional RE [32]. The customer rectifies the product while in making after each iteration that helps to keep the customer's requirements realistic.

Lengthy requirements documentation, which is also considered unreliable at times, is another challenge of traditional RE [32][33]. After gathering requirements from the customer a detailed requirements specification document is prepared. This specification document is not only lengthy and complex but also incorporates technical details and language difficult to be understood for non-technical customers. Agile teams believe in exchanging information face-to-face, onsite customer presence and follow very less documentation which ultimately resolves this challenge.

Table 3. Summary of traditional RE challenges resolved by agile RE

Traditional RE challenge	Agile RE solution
Communication gap [32]	Collocated teams [43] [42] Face-to-face communication [4][5] On-site customer [7][25] [26]
Less customer involvement [30]	
Requirements documentation [32]	
Over-scoping of requirements [32]	Iterative detailing of requirements [32]
Requirements validation [30]	Requirements prioritization by customer Prototyping [4][5]

Requirements validation is another issue in traditional RE [30] which is resolved through constant requirements prioritization done by customer using various techniques (e.g. [28]) in agile methods. In addition, prototyping also helps the customer to visualize her demands and suggest changes if required at earlier development stages [4][5]. Prototype presentation sessions at the end of every iteration, also called "show and tell sessions" in agile methods help the customer to visualize step-by-step development of her idea and give feedback which serves as requirements validation or brings in rework in case of suggested changes.

Therefore, we can summarize that several detrimental challenges posed by traditional RE can be eradicated or minimized by using agile RE though this argument needs stronger empirical evidence and experience reports from industry to strength and generalize such conclusion.

5. NEW CHALLENGES POSED BY AGILE REQUIREMENTS ENGINEERING

In literature, traditional RE activities are mapped against agile methods' (i.e. Scrum) workflow to show that both are compatible (e.g. [23][12]). However, merging agile and traditional RE practices poses several challenges brought by agile (e.g. [5][11]) to the software industry. Our literature review [11] moved one step further aiming to reveal the possible solutions reported so far. Table 4 summarizes the list of identified solutions.

NFRs that determine the usability, security and performance of the system are to a certain extent neglected in agile methods [4]. Nevertheless, several methods to model NFRs are proposed recently to overcome this issue in agile methods (e.g. [46]–[50]). Some of the authors of the referred proposals put forward using modelling artefacts as part of agile RE in order to integrate NFR activities into the agile cycle. Other authors focus on a particular NFR, e.g. security [33][34] and suggest frameworks for assisting in the elicitation and evaluation of security requirements.

Following the life of a requirement, known as requirements traceability is another important issue to deal with in agile methods. However, methods are proposed to properly gather and store the requirements in agile methods to resolve incomplete requirements elicitation [23] and enable traceability at later stages (e.g. [32][33][53]).

Another important issue in agile RE is minimal or no documentation [4][5]. The story gathering promotes the accumulation of tacit knowledge across the team through knowledge exchange between customer and developers [16]. User requirements are documented as feature list, user stories or product backlogs in agile methods. Lengthy requirements

documents are simply not the agile way of dealing with requirements. The need to document requirements is resolved to some extent by consistent and frequent face-to-face communication among team members.

Table 4. Summary of agile RE challenges and their proposed solutions

Agile RE challenges	Proposed solutions
Neglecting NFRs [4]	Methods to tackle NFRs in agile methods [46][50]
Lack of requirements traceability	Methods to enable adequate requirements traceability [32][33]
Incorrect requirements prioritization [23]	Methods for value based requirements prioritization [35][37]
Minimal requirements documentation [4][5]	Collocated teams [43][42] Face-to-face communication [4][5] On-site customer [7][25][26]
Contractual issues [4]	Legal measures Fixed price contracts [29][54]
Customer availability [5]	Proxy customers [5]
Customer agreement [5][29]	Appointing appropriate customer representative [5]

In agile RE, clients perform business-value-based user story prioritization (e.g. [8][9][29]) depending upon their business needs and priorities. User stories are prioritized and reviewed before undergoing development [10]. However, incorrect requirement prioritization [23] can cause serious time and money loss in addition to rework.

The contractual terms of an agile based project are important issues in agile RE [4][12] that do not allow requirements' changes at later stages of product development. Changes at later stages of project involve cost that exceeds the pre-decided amount in contract. However, several studies (e.g. [29][54]) that focus on agile-based outsourced projects with fixed payment contracts state the solution for this issue.

Agile methods assume customer's availability [5][45] which itself is an issue due to time and budget allocation from client's side often resolved by introducing proxy or surrogate customers. However, customer's lack of domain knowledge, inability of decision-making, and lack of consensus on issues [5][29] can also cause serious issues in carrying out the project. This requires appropriate appointment of customer's representative with adequate domain knowledge from the client side.

In sum, it can be seen that agile RE also poses several serious challenges to the software industry. To counter these challenges some of the methods are proposed to carry out RE activities in agile methods smoothly (e.g. [23]). However, empirical evidence of the proposed solutions is required for implementation and generalization of results. Therefore, we can say that software industry lacks knowledge on how agile teams carry out RE activities in projects [55].

6. FINAL CONSIDERATIONS

Our reflection on the ways in which agile and traditional RE differ brought us to the following conclusions, which have some implications for other researchers and practitioners.

First, while consolidating the published ideas on how agile RE takes place in real life, we found a coherent small set of related activities that seem to be present in each agile RE process. Of course, we cannot generalize that this is the way of how all agile companies around the world approach their requirements and structure their working practices. More research on how exactly companies go about managing their agile requirements is therefore necessary.

Second, the community attempts to come up with approaches to include NFRs into the agile paradigm. While proposals have been published, no empirical evaluation took place in real-life settings. This clearly indicates a gap in our knowledge on what approach would work in what context. This in turn means the need for more empirical evaluation research.

Third, we provide a set of steps that guide the way to carry out RE activities in agile methods. Further research needs to be carried out to back up the proposed hierarchy with empirical evidence for generalization of results. Moreover, it can serve as a preliminary guideline for software industry practitioners to carry out requirements in an agile way. Also, it helps the tool manufacturing industry to design agile-RE-specific tools for requirements management and other alleged activities.

7. ACKNOWLEDGMENTS

We are highly grateful to Dr. Sabrina Marczak for her kind help with the idea, content and motivation for this paper . We also thank the PDTI Program, financed by Dell Computers of Brazil Ltd. (Law 8.248/91), for sponsoring the scholarship of one of the authors.

8. REFERENCES

[1]. Abdullah, N.N.B. et al., 2011. Communication Patterns of Agile Requirements Engineering. In *Proceedings of the 1st Workshop on Agile Requirements Engineering*. (Lancaster, United Kingdom, July 25-29, 2011).ARE'11. ACM. New York,NY,DOI=http://doi.acm.org/10.1145/2068783.2068784

[2]. Aurum, A. & Wohlin, C. 2005. Requirements Engineering: Setting the Context. In Engineering and Managing Software Requirements A. Aurum & C. Wohlin, Ed. Springer-Verlag, Berlin Heidelberg, 1-15. DOI= 10.1007/978-3-642-19858-8_18.

[3]. Bakalova, Z. et al., 2011. Agile requirements prioritization: What happens in practice and what is described in literature. Lecture Notes in Computer Science Ed. Springer-Verlag Berlin Heidelberg, 181–195.DOI= 10.1007/3-540-28244-0_1

[4]. Bang, T.J., 2007. An Agile Approach to Requirement Specification. In Agile Processes in Software Engineering and Extreme Programming G. C. et al. Ed. Springer-Verlag Berlin Heidelberg.193–197. DOI= 10.1007/978-3-540-73101-6_35.

[5]. Beck, K. et al., 2001. Manifesto for Agile Software Development. Available at: http://agilemanifesto.org/.

[6]. Berry, D.M., 2004. The Inevitable Pain of Software Development, Why there is no silver bullet,. In *Proceedings of the International Workshop on Time Constrained Requirements Engineering*. pp. 50-74.

[7]. Bjarnason, E., Wnuk, K. & Regnell, B., 2011. A Case Study on Benefits and Side-Effects of Agile Practices in Large-Scale Requirements Engineering. In *Proceedings of the 1st Workshop on Agile Requirements Engineering*. (Lancaster,

United Kingdom, July 25-29, 2011).ARE'11. ACM. New York, NY. DOI= 10.1145/2068783.2068786.

[8]. Bjarnason, E., Wnuk, K. & Regnell, B., 2011. Requirements are slipping through the gaps — A case study on causes & effects of communication gaps in large-scale software development. In *Proceedings of the 19th International Requirements Engineering Conference.* Lancaster, United Kingdom, July 25-29, 2011) ACM. New York, NY,USA. DOI=http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6051639.

[9]. Boness, K. & Harrison, R., 2007. Goal Sketching : Towards Agile Requirements Engineering. *In Proceedings of the International Conference on Software Engineering Advances*. (Cap Esterel, France, 25-31 Aug. 2007) IEEE DOI= 10.1109/ICSEA.2007.36.

[10]. Cao, L. & Ramesh, B., 2008. Agile Requirements Engineering Practices: An Empirical Study. *IEEE Soft*. 25,1,60–67.(Dec2010) DOI=http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4420071.

[11]. Carlson, D. & Matuzic, P., 2010. Practical Agile Requirements Engineering. *In Proceedings of the 13th Annual Systems Engineering Conference*. (San Diego, CA25-28 Oct 2010).

[12]. Cohn, M., 2005a. *Agile Estimation and Planning*, Prentice Hall , USA.

[13]. Cohn, M., 2009. *User Stories Applied For Agile Software Development*, Addison Wesley. Indiana, USA:

[14]. Cohn, M., 2005b. Writing Effective User Stories for Agile Requirements Mike Cohn — background Today ' s agenda Ron Jeffries ' Three Cs. In Software development best practices 2005. pp. 1–35.

[15]. Dagnino, A., Smiley, K., Srikanth , H., Antón , A., Williams, L., 2004. Experiences in applying agile software development practices in new product development. *In Proceedings of the 8th IASTED International Conference on Software Engineering and Applications*. (November 9 – 11, 2004 Cambridge, USA) MA, United States.

[16]. Daneva, M. et al., 2013. Agile requirements prioritization in large-scale outsourced system projects: An empirical study. J. Sys. Soft, 86, 5, (1333–1353). DOI= http://linkinghub.elsevier.com/retrieve/pii/S0164121212003536

[17]. Dybå, T. & Dingsøyr, T., 2009. What do we know about Agile Software development? *IEEE Soft.*,26, 5,DOI= 10.1109/MS.2009.145.

[18]. Eberlein, A. & Julio Cesar, S. do P.L., 2002. Agile Requirements Definition : A View from Requirements Engineering. *In Proceedings of the International Workshop on Time Constrained Requirements Engineering*. (Essen, Germany, 6–9 September 2002).doi=10.1.1.194.5553

[19]. Erickson, J., Lyytinen, K. & Siau, K., 2005. Agile Modeling , Agile Software Develpment , and Extreme Programming: The State of Research. *J. Database Manag* 16, 88–100.

[20]. Ernst, N. a. et al., 2014. Agile requirements engineering via para-consistent reasoning. *Info Syst*, 43, 100–116 DOI:http://linkinghub.elsevier.com/retrieve/pii/S030643791300077X .

[21]. Farid, W.M. & Mitropoulos, F.J., 2012a. NORMATIC: A visual tool for modeling Non-Functional Requirements in agile processes. 2012 *In the Proceedings of IEEE Southeastcon*, (Florida, US, 15-18 March 2012). DOI=: 10.1109/SECon.2012.6196989.

[22]. Farid, W.M. & Mitropoulos, F.J., 2013. NORPLAN: Non-functional requirements planning for agile processes. *In the Proceedings of IEEE Southeastcon*, (Florida, US, 4-7 April 2013). DOI= 10.1109/SECON.2013.6567463

[23]. Farid, W.M. & Mitropoulos, F.J., 2012b. Novel lightweight engineering artifacts for modeling non-functional requirements in agile processes. *In the Proceedings of IEEE Southeastcon*, (Florida, US 15-18 March 2012) DOI= 10.1109/SECon.2012.6196988.

[24]. Haugset, B. & Stalhane, T., 2012. Automated Acceptance Testing as an Agile Requirements Engineering Practice. *In the Proceedings of 45th Hawaii International Conference on System Sciences.* (Hawaii, US. 04 Jan - 07 Jan 2012) DOI= http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6149535.

[25]. Helmy, W., Kamel, A. & Hegazy, O., 2012. Requirements engineering methodology in agile environment. *International Journal of Computer Science Issue*s,9,5,293–300.

[26]. Inayat, I. et al., 2014. A systematic literature review on agile requirements engineering practices and challenges. *Computers in Human Behavior*. (Dec 2014) DOI=http://www.sciencedirect.com/science/article/pii/S074756321400569X .

[27]. Jun, L., Qiuzhen, W. & Lin, G., 2010. Application of Agile Requirement Engineering in Modest-Sized Information Systems Development. *In the Proceedings of the Second World Congress on Software Engineering*. (Wuhan, China, 19-20 Dec 2012). IEEE, DOI=http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5718379 [Accessed October 10, 2012].

[28]. Kotonya, G. & Sommerville, I., 1997. *Requirements Engineering*, Wiley.

[29]. Kumar, M., Shukla, M. & Agarwal, S., 2013. A Hybrid Approach of Requirement Engineering in Agile Software Development. *In Proceedings of the International Conference on Machine Intelligence and Research Advancement*. (Venice, Italy, 21-23 Dec 2013) DOI=http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6918885.

[30]. Lee, C. & Guadagno, L., 2003. FLUID: Echo–Agile Requirements Authoring and Traceability. *In Proceedings of the Midwest Software Engineering Conference*.

[31]. Lee, C., Guadagno, L. & Jia, X., 2003. An Agile Approach to Capturing Requirements and Traceability. *In Proceedings of the 2nd International Workshop on Traceability in Emerging Forms of Software Engineering*.(CA, USA 19 May)

[32]. Leffingwell, D., 2011. *Agile Software Requirements Lean Requirements Practices for Teams, Programs, and the Enterprise*, Addison Wesley.

[33]. Lucia, A. De & Qusef, A., 2010. Requirements engineering in agile software development. *Journal of Emerging Technologies in Web Intelligence*, 2, 3, 308–313.

[34]. Lundh, E. & Sandberg, M., 2002. Time Constrained Requirements Engineering with Extreme Programming – An

Experience Report. *In Proceedings of the International Workshop on Time Constrained Requirements Engineering*.(Essen, Germany, 9 May 2002).

[35]. Othmane, L. Ben, Angin, P. & Bhargava, B., 2014. Using Assurance Cases to Develop Iteratively Security Features Using Scrum. *In Proceedings of Ninth International Conference on Availability, Reliability and Security*. (Fribourg, Switzerland, 8-12 Sep 2014)

[36]. Paetsch, F., Eberlein, A. & Maurer, F., 2003. Requirements engineering and agile software development. *In Proceedings of Twelfth IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises* (Linz, Austria, 4-6 Sep 2003). IEEE Comput. Soc,

[37]. Paetsch, F. & Maurer, F., 2003. Requirements Engineering and Agile Software Development. *In Proceedings of Twelfth IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises*. (Linz, Austria, 4-6 Sep 2003). IEEE Comput. Soc,.

[38]. Pichler, M., Rumetshofer, H. & Wahler, W., 2006. Agile Requirements Engineering for a Social Insurance for Occupational Risks Organization: A Case Study. *In Proceedings of 14th IEEE International Requirements Engineering Conference* .RE'06.(Minnesota, U.S.A. 11-15 Sep 2006). IEEE Comput. Soc,.

[39]. Poonkundran, B., 2013. Agile Methodology in Fixed Price projects. *Global Advanced Research Journal of Enigneering, Technology and Innovation*, 2, 9, 243–249.

[40]. Pressman, R.S., 1996. *Software engineering A practitioner's Approach* 7th ed., Mc Graw Hill.

[41]. Racheva, Z., Daneva, M., Sikkel, K., et al., 2010. Do we know enough about requirements prioritization in agile projects: Insights from a case study In Proceedings of the 18th IEEE International Requirements Engineering Conference, (27 Sep- 1 Oct 2010, Sydney, Australia) IEEE Comput. Soc.

[42]. Racheva, Z., Daneva, M. & Buglione, L., 2008a. Supporting the dynamic reprioritization of requirements in agile development of software products. *In Proceedings of the International Workshop on Software Product Management*.(Barcelona, Spain, 9th Sep 2008).

[43]. Racheva, Z., Daneva, M. & Buglione, L., 2008b. Supporting the dynamic reprioritization of requirements in agile development of software products. *In Second International Workshop on Software Product Management*. ISWPM'08. Barcelona, Spain, 9th Sep 2008).

[44]. Racheva, Z., Daneva, M. & Herrmann, A., 2010. A Conceptual Model of Client-driven Agile Requirements Prioritization : Results of a Case Study. *In IEEE International Symposium on Empirical Software Engineering and Measurement* . *(*Bolzano-Bozen, Italy. 16-17 Sep 2010).

[45]. Ramesh, B., Baskerville, R. & Cao, L., 2010. Agile requirements engineering practices and challenges: an empirical study. *Information Systems Journal*, 20, 5, 449–480.

[46]. Schwaber, K., 2002. The Impact of Agile Processes on Requirements Engineering. *In Proceedings of the International Workshop on Time Constrained Requirements Engineering*. (Essen, Germany, 9 May 2002).

[47]. Sillitti, A. et al., 2005. Managing Uncertainty in Requirements : A Survey in Documentation-driven and Agile Companies. *In 11th IEEE International Software Metrics Symposium*.(Como, Italy, 19-22 Sep 2005).

[48]. Sommerville, I., 2011. *Software Engineering*, Pearson Education Inc.

[49]. Sonia & Singhal, A., 2011. Development of Agile Security Framework Using a Hybrid Technique for Requirements Elicitation. In *Advances in Computing, Communication and Control* S. Unnikrishnan, S. Surve, & D. Bhoir, Ed.. Springer-Verlag Berlin Heidelberg. 178–188..

[50]. Tarhan, A. & Yilmaz, S.G., 2014. Systematic analyses and comparison of development performance and product quality of Incremental Process and Agile Process. *Information and Software Technology*, 56, 5, 477–494.

[51]. Taromirad, M. & Paige, R.F., 2012. Agile requirements traceability using domain-specific modeling languages. In Proceedings of Extreme Modeling Workshop. (1 oct 2012, Innsbruck, Austria).

[52]. Tomayko, J.E., 2002. Engineering of Unstable Requirements Using Agile Methods. *In Proceedings of the International Workshop on Time Constrained Requirements Engineering*. .(Essen, Germany, 9 May 2002).

[53]. Wolfgang, E., 2011. Working with User Stories. *In Workshop on Agile Requirements Engineering*. (Lancaster, United Kingdom, July 25-29, 2011).ARE'11. ACM. New York,NY

[54]. Yu, Y. & Sharp, H., 2011. Analysing requirements in a case study of pairing. *In Workshop on Agile Requirements Engineering* (Lancaster, United Kingdom, July 25-29, 2011).ARE'11. ACM. New York,NY

[55]. Zhu, Y., 2009. Requirements Engineering in an Agile Environment. Master's Thesis, Uppsala University.