Journal of Software Engineering and Applications, 2016, 9, 63-70
Published Online March 2016 in SciRes. http://www.scirp.org/journal/jsea
http://dx.doi.org/10.4236/jsea.2016.93005

Quintessence of Traditional and Agile Requirement Engineering

Jalil Abbas

School of Computing and Information Sciences, Imperial College of Business Studies, Lahore, Pakistan
Email: sjshah786@gmail.com

Received 17 December 2015; accepted 14 March 2016; published 17 March 2016

Abstract

Requirement gathering for software development project is the most crucial stage and thus requirement engineering (RE) occupies the chief position in the software development. Countless techniques concerning the RE processes exist to make sure the requirements are coherent, compact and complete in all respects. In this way different aspects of RE are dissected and detailed upon. A comparison of RE in Agile and RE in Waterfall is expatiated and on the basis of the literature survey the overall Agile RE process is accumulated. Agile being a technique produces high quality software in relatively less time as compared to the conventional waterfall methodology. The paramount objective of this study is to take lessons from RE that Agile method may consider, if quality being the cardinal concern. The study is patterned on the survey of the previous research reported in the coexisting literature and the practices which are being pursued in the area.

Keywords

Requirement Engineering, Waterfall, Software Development Life Cycle, Agile Software Development, Elicitation

1. Introduction

In software engineering, software development methodology known as software development life cycle (SDLC) is a sectionalisation of software development work. Common methodologies include Waterfall, Prototyping, Iterative and Incremental development, Spiral development, Rapid application development, Extreme Programming and other different kinds of Agile methodology. All these methods comprise of multiple phases and a variety of different activities. For instance design, re-factor, reuse, re-engineering and maintenance are some common activities, employed to complete software solutions. A wide variety of such frameworks have evolved over the years, each with its own recognized strengths and weaknesses. One software development methodology framework doesn't adequately suffice for all projects.

How to cite this paper: Abbas, J. (2016) Quintessence of Traditional and Agile Requirement Engineering. *Journal of Software Engineering and Applications*, **9**, 63-70. http://dx.doi.org/10.4236/jsea.2016.93005

Over the years, most of the software development methods have been made immaculate and then referred to as traditional methods. One of the oldest of these traditional methods is waterfall which was firstly explained by Winston Royce in 1970 [1]. It is still very much in vogue widely practiced both in large and small projects [2]. The Waterfall model is a sequential design process which is used in software development processes where progress palpably is flowing downwards like a Waterfall through the phases of requirement gathering and analysis, design, coding, testing and maintenance. Every stage is to be treated separately at an opportune moment so you cannot jump stages. Documentation is done at every stage of a Waterfall model, providing an opportunity to the people to decipher as what has been done. Similarly testing is carried at every stage. Waterfall method is understood for its concrete and complete requirements and these features make this approach more viable and stable. It is often said about this method that spending more time early in the cycle can pave way to greater success at later stages.

The Agile development method came to limelight as the result of gathering of seventeen representatives from the software development industry in snowbird, Utah in 2001 [3]. Their intention was to develop innovative approaches to software development that would make organization react rapidly and adapt to volatile requirements and technologies.

In Agile Manifesto [3] they gave the identification of the following four priorities:

Priorities in Agile Manifesto

There exist multiple types of Agile methods as extreme programming, scrum, feature-driven development, dynamic system development method, adaptive software development, crystal and lean software development. What is common to all methods is the division of client's requirements into multiple release cycle which are available in smaller portions regarding to their business value [4]. These methods comprise of most recognizable quality factors such as cost effectiveness, efficiency, extendibility, maintainability, portability, reusability and robustness [5].

The remainder of the paper is organized as follows: Section 2 gives a comparison between traditional and Agile software development methodologies. Section 3 explains the Requirement Engineering process. Section 4 describes the RE in waterfall and RE in Agile, also the challenges of traditional RE resolved by Agile RE are discussed. Conclusion is given at the end of paper.

2. Comparison of Agile and Waterfall Development Methods

Agile and waterfall methods stand apart so far as their activities are concerned, as they are put to use within the development process [6]-[9]. To understand clearly the difference between Waterfall and Agile the comparison is made in a tabular form and is provided in **Table 1**.

3. Requirement Engineering

Software Requirements describe features and functionalities of the target system it also tells the expectations of the users from the software product .The requirements can be obvious or occult, either it is known or not known, expected or unexpected from client's point of view. The formidable single part of making a software system is deciding clearly as what to build. No other part of the conceptual work is as formidable as making the detailed technical requirements. The process to glean the software requirements from client, analyze and document them is named requirement engineering. It is sometime overlooked or assumed to be a straight and undistorted task [26], requirements collecting for software development projects is the most difficult phase of any software development methodology. To determine software requirements is the fulcrum to any successful project. Requirements cannot be easily defined and estimated for managing any project [27]. Some studies have exposed that around 37% of the problems occurred during the development of system related to the requirement phases [28] and is graphically depicted in **Figure 1**.

Requirement engineering stresses the use of systematic and repeatable techniques that ensure the completeness, consistency and relevance of the system requirements. The process used for RE changes widely depending on the

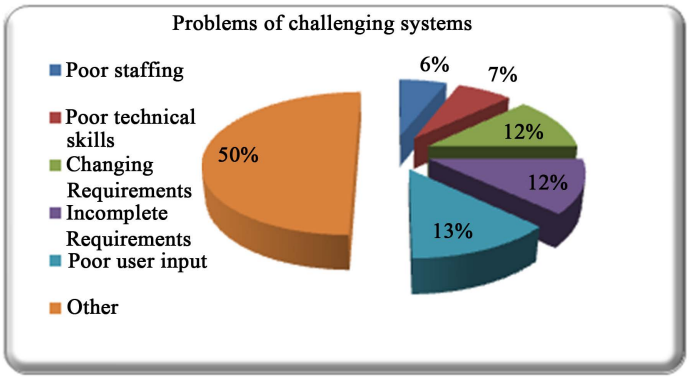

Figure 1. Problems of challenging system.

Table 1. Comparison of Waterfall and Agile development methods.

Metric	Traditional development process	Agile development process	Study that reported
SDLC	Linear	Iterative	[9] [10]
Development style	Traditional methods are predictive (plan–driven)	Agile methods are adaptive	[11]-[14]
Documentation	Enough documentation to be able to answer all questions that might be asked in the future.	Light (replaced by face to face communication)	[7] [15]
Customer involvement	in traditional approaches the customer is mainly engaged during the early phase of the project	Agile methods engage the customer throughout the whole development process.	[13] [16]-[18]
Change	Resistance to change	Welcoming to change, even changes are brought in late in the project.	[19] [20]
Size	traditional methods are able to manage effectively large project	Manage effectively requirements in small projects but not in large ones.	[13]
Planning scale	Long term	Short term	[13]
Management	process oriented, command and control	People oriented, leadership and conformity.	[11] [12] [15] [21] [22]
Team organization	Pre- structured teams	Self organizing teams	[23]
Ownership	Ownership belongs to only project manager	Shared ownership	[19]
Prioritization	Requirements are typically prioritized once.	Prioritize feature lists repeatedly during development	[24]
Customer feedback	At the termination of the project	At the completion of every sprint	[25]
Risk identification	No risk identification.	Early identification and mitigation in every sprint.	[25]
Time between specification & implementation	Long	Short	[13]
Delivery	Delivering artifacts phase wise and delivery of working software at the end of project.	Demonstration and delivering working software et the end of every sprint.	[25]
Measure of Success	Conformance to plan	Business value delivered	[3]

application domain, the people involved and organization developing the requirements. There exists a plethora of generic activities common to all processes. So RE process can be split up into 2-main assortments:

- Requirement Development
- Requirement Management

The goal of requirement development is to identify, capture and agree upon a set of functional requirements and product characteristics that will gain the stated business objectives. It contains four kinds of activities as shown in **Figure 2** [29].

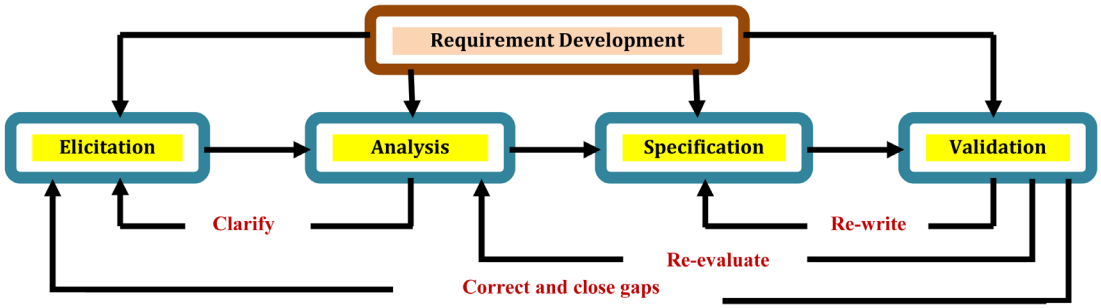

Figure 2. Requirement development activities.

- Elicitation: It is process discovering, reviewing, documenting, knowing user needs and constraints for the system.
- Analysis: It provides feedback loop to refine user's needs and restraints.
- Specification: It is process of documenting the user's needs and restraints.
- Validation: This ensures that the system requirements are complete, correct, consistent and clear.

However, the requirement management is the process of documenting, analyzing, tracing, prioritizing and agreeing on requirements and then controlling change and communicating to relevant stakeholders. It is a continuous process throughout a project. A requirement is a capability to which a project outcome (product or service) should conform.

4. Requirement Engineering in Agile and Waterfall

4.1. RE in Waterfall

Requirement engineering involves a number of processes for gathering requirements in accordance with the needs and demands of users and stakeholders of the software product. Waterfall Requirement Engineering involves some important features that are elicitation, analysis, documentation and managing of the requirements [30] [31] as already mentioned in Section 3. In the waterfall model requirements engineering is presented as the first phase of the development process. This traditional approach to the RE process focuses on gathering all the requirements and preparing the requirements specification document up front before proceeding to the design phase [32]. In the waterfall method the project is separated into stages distinctly and commitments must be made at an early stage, which makes it hard to alter the requirements if customers change their minds. So waterfall is more suitable when the requirements will probably not be changed during the implementation time. In conclusion, the waterfall model takes a static viewpoint of Requirements Engineering by ignoring issues such as the volatility of requirements and its impact on earlier and later phases of development [33].

4.2. RE in Agile

According to various researchers Agile methodology and its family members are based on the following principles also known as Agile manifesto [34]-[37]:

Customer Satisfaction	Frequent Delivery	Motivated Team
Technical Excellence	Emergent Design	Incremental development
Embrace Change	Collaboration	High Bandwidth
Sustainable Pace	Simplicity	Continuous Improvement

These principles are fairly simple in concept, but are profoundly deep in practice.

Agile assumes that requirements engineering continues through the lifetime of a system. In Agile, RE is achieved through continuous collaboration while requirement gathering, developing and testing may happen at the same time. This is achieved by applying the practice of evolutionary requirements which suggests that requirements should evolve over time. In Agile, the business requirements are elicited and documented in the form

of user stories, which are from portrays user's perspective [38]. These user stories are used as a primary unit of work and continue to grow during the lifecycle of the project. Agile methods involve continuous planning, *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 [39]-[42]. After the literature survey of RE in Agile the overall Agile RE process is accumulated and described in **Figure 3**.

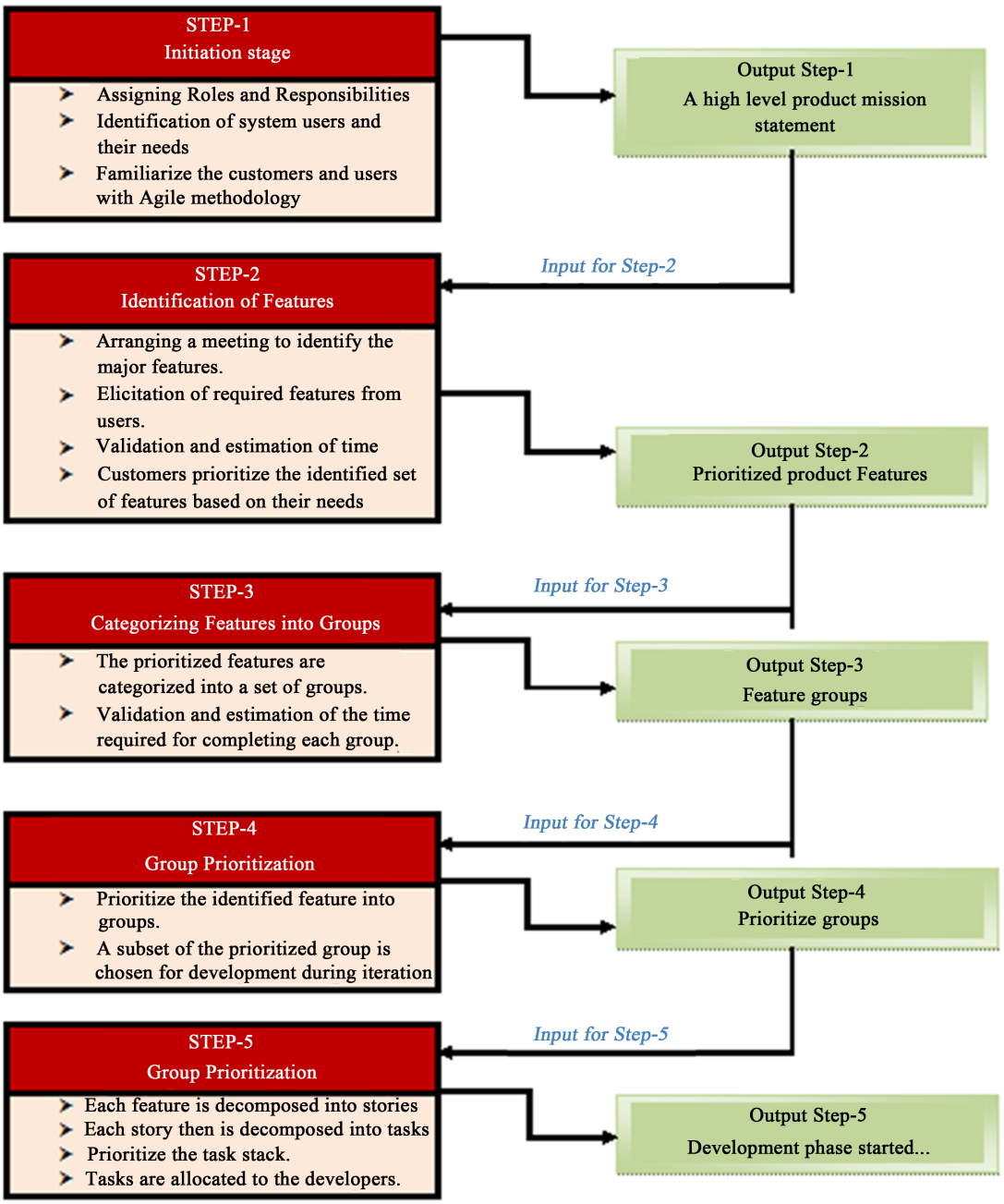

Figure 3. RE process in agile.

Table 2. Issues of RE in waterfall resolved by agile RE.

Issues of RE in Waterfall	**Resolved by RE in Agile**
Customer involvement: Customers are involved only during the beginning of requirement gathering and analysis [40].	Customers are involved throughout the complete process.
Prioritization of requirements: Complete requirements for the full project are prioritized upfront and the prioritization is kept up through the project lifecycle, and reprioritization is arduous [47].	Priorities are setup for all iterations that offer opportunities for getting desirable results and customer satisfaction [48] [49].
Documentation: Totally emphasizes at properly gathering organizing and documenting all requirements and excludes any live meetings/conferences [14].	User stories are concise and provide to-the-point explanation of user demands, obviate the need for maintaining long SRS documents.
Requirements validation: Validation happens late in the life cycle.	Prototyping helps in providing the customer with a blueprint of the product, and therefore helps in validating the requirements [50].
Communication: It is a major factor in the delay and failure of software projects [40].	It provides regular interaction with customer and among teams.
Over-scoping of requirements: It is the cause of rework, which in turn causes further investment.	Developers receive a list of features that are constantly prioritized so the chance of having to repeat allocation in projects is minimized.
Shall Argument: The worst thing of waterfall RE is "shall" argument *i.e.* system shall do it, etc. [46].	Agile introduces the real time system.

4.3. RE in Agile VS RE in Waterfall Methods

It has been ascertained that traditional requirement process is a complex process where as real life development needs efficient requirement software which must have a flexible and speedy process. For a successful project an efficient RE process is needed. The objective of RE remains the same in all software methods, however RE in Agile and Waterfall methods is juxtaposed and opposite in nature [43] [44]. Remarkable variances are found in the process of carrying out RE activities in Agile methods when compared and contrasted to Waterfall methods. The traditional RE is facing many a challenges such as communication gaps, over scoping, requirement prioritization, validation and customer involvement [45]. These issues are resolved by Agile practices such as face to face communication for minimizing documentation and communication gaps, gradual detailing of requirements for reducing over scoping, requirement prioritization by customer based on the worth of business to deal with requirements validation and close interaction on the part of team and customer in order to avoid lack of customer participation [46]. Issues caused by the traditional RE and the solutions provided by the Agile RE are described in **Table 2**.

Therefore, we can summarize that several detrimental challenges posed by traditional RE can be eradicated or minimized by using Agile RE.

5. Conclusion

Differentiation has been clearly drawn and found that the traditional RE and Agile RE are two different approaches so far as their rules and activities are concerned. Comparison between the two shows why people have gone from traditional RE to Agile RE. The underlying idea of this apparent shift was to shed light on the magnitude of Agile development for efficacious requirement engineering process. By doing so, resultantly Agile RE works better than the waterfall RE in disciplines like communication, customer collaboration, documentation, delivering outputs, requirement prioritization and validation, etc. Practitioners engaged will come to comprehend and evaluate the various impediments/obstacles encountered by them while using traditional RE.

Acknowledgements

I, personally feel extremely beholden to Prof(R) Ghulam Qasim Shah who with self-abnegation graciously spared his time and reviewed this paper.

References

[1] Royce, W.W. (1970) Managing the Development of Large Software Systems. *Proceedings of IEEE WESCON*, **26**, 328-388.

[2] Alsultanny, Y. and Wohaishi, A.M. (2009) Requirements of Software Quality Assurance Model. *Second International Conference on Environmental and Computer Science*, Dubai, 28-30 December 2009, 19-23. http://dx.doi.org/10.1109/icecs.2009.43

[3] Alliance, A. (2001) Manifesto for Agile Software Development. http://www.Agilemanifesto.org/

[4] Hamed, A.M.M. and Abushama, H. (2013) Popular Agile Approaches in Software Development: Review and Analysis. *International Conference on Computing, Electrical and Electronics Engineering* (*ICCEEE*), August 2013, 160-166.

[5] Sirshar, M. (2012) Evaluation of Quality Assurance Factors in Agile Methodologies. *International Journal of Advanced Computer Science*, **2**, 73-78.

[6] Huo, M., Verner, J., Zhu, L. and Babar, M.A. (2004) Software Quality and Agile Methods. *Proceedings of the* 28*th Annual International Computer Software and Applications Conference*, Hong Kong, 28-30 September 2004, 520-525.

[7] Boehm, B. (2002) Get Ready for Agile Methods, with Care. *Computer*, **35**, 64-69. http://dx.doi.org/10.1109/2.976920

[8] Dybå, T. and Dingsoyr, T. (2009) What Do We Know about Agile Software Development? *IEEE Software*, **26**, 6-9. http://dx.doi.org/10.1109/MS.2009.145

[9] Nerur, S., Mahapatra, R. and Mangalaraj, G. (2005) Challenges of Migrating to Agile Methodologies. *Communications of the ACM*, **48**, 72-78. http://dx.doi.org/10.1145/1060710.1060712

[10] Charvat, J. (2003) Project Management Methodologies: Selecting, Implementing, and Supporting Methodologies and Processes for Projects. John Wiley & Sons, Hoboken.

[11] Fowler, M. (2001) The New Methodology. *Wuhan University Journal of Natural Sciences*, **6**, 12-24. http://dx.doi.org/10.1007/BF03160222

[12] Boehm, B. and Turner, R. (2004) Balancing Agility and Discipline: Evaluating and Integrating Agile and Plan-Driven Methods. In *Proceedings of* 26*th International Conference on Software Engineering*, 718-719.

[13] Okoli, C. and Carillo, K. (2012) The Best of Adaptive and Predictive Methodologies: Open Source Software Development, a Balance between Agility and Discipline. *International Journal of Information Technology and Management*, **11**, 153-166. http://dx.doi.org/10.1504/IJITM.2012.044071

[14] Leffingwell, D. (2007) Scaling Software Agility: Best Practices for Large Enterprises. Pearson Education, New York.

[15] Boehm, B. and Turner, R. (2005) Management Challenges to Implementing Agile Processes in Traditional Development Organizations. *IEEE Software*, **22**, 30-39. http://dx.doi.org/10.1109/MS.2005.129

[16] Paetsch, F., Eberlein, A. and Maurer, F. (2003) Requirements Engineering and Agile Software Development. In: *Null*, p. 308. IEEE.

[17] Sillitti, A. and Succi, G. (2005) Requirements Engineering for Agile Methods. In: *Engineering and Managing Software Requirements*, Springer Berlin Heidelberg, 309-326. http://dx.doi.org/10.1007/3-540-28244-0_14

[18] Cockburn, A. and Highsmith, J. (2001) Agile Software Development: The People Factor. *Computer*, **34**, 131-133. http://dx.doi.org/10.1109/2.963450

[19] Regev, G., Gause, D.C. and Wegmann, A. (2006) Creativity and the Age-Old Resistance to Change Problem in RE. 14*th IEEE International Conference Requirements Engineering*, Minneapolis/St. Paul, 11-15 September 2006, 291-296. http://dx.doi.org/10.1109/re.2006.13

[20] Boehm, B. and Turner, R. (2003) Using Risk to Balance Agile and Plan-Driven Methods. *Computer*, **36**, 57-66. http://dx.doi.org/10.1109/MC.2003.1204376

[21] Vinekar, V., Slinkman, C.W. and Nerur, S. (2006) Can Agile and Traditional Systems Development Approaches Coexist? An Ambidextrous View. *Information Systems Management*, **23**, 31-42. http://dx.doi.org/10.1201/1078.10580530/46108.23.3.20060601/93705.4

[22] Highsmith, J.A. (2002) Agile Software Development Ecosystems. Addison-Wesley Professional, Boston, Vol. 13

[23] Ashmore, S. and Runyan, K. (2014) Introduction to Agile Methods. Addison-Wesley Professional, Boston.

[24] Racheva, Z., Daneva, M., Sikkel, K., Herrmann, A. and Wieringa, R. (2010) Do We Know Enough about Requirements Prioritization in Agile Projects: Insights from a Case Study. 18*th IEEE International Requirements Engineering Conference* (*RE*), 147-156.

[25] Moniruzzaman, A.B.M. and Hossain, D.S.A. (2013) Comparative Study on Agile Software Development Methodologies. arXiv preprint arXiv:1307.3356.

[26] Kitapci, H. and Boehm, B.W. (2007) Formalizing Informal Stakeholder Decisions—A Hybrid Method Approach. 40*th Annual Hawaii International Conference on System Sciences*, Waikoloa, 3-6 January 2007, 283c-283c.

[27] https://www.pwc.com/us/en/.../pwc-adopting-Agile-methodology.pdf

[28] Polini, A. (2010) Software Requirements. http://www1.isti.cnr.it/~polini/lucidiSE/Requirements1.pdf

[29] Sommerville, I. and Sawyer, P. (1997) Requirements Engineering: A Good Practice Guide. John Wiley & Sons, Inc., Hoboken

[30] Pressman, R.S. (2005) Software Engineering: A Practitioner's Approach. Palgrave Macmillan.

[31] Sommerville, I. and Kotonya, G. (1998) Requirements Engineering: Processes and Techniques. John Wiley & Sons, Inc., Hoboken.

[32] Pandey, D., Suman, U. and Ramani, A.K. (2010) An Effective Requirement Engineering Process Model for Software Development and Requirements Management. *International Conference on Advances in Recent Technologies in Communication and Computing* (*ARTCom*), Kottayam, 16-17 October 2010, 287-291. http://dx.doi.org/10.1109/artcom.2010.24

[33] Nuseibeh, B. and Easterbrook, S. (2000) Requirements Engineering: A Roadmap. *Proceedings of the Conference on the Future of Software Engineering*, 35-46. http://dx.doi.org/10.1145/336512.336523

[34] Zhu, Y. (2009) Requirements Engineering in an Agile Environment. Uppsala University, Uppsala.

[35] Kavitha, C.R. and Thomas, S.M. (2011) Requirement Gathering for Small Projects Using Agile Methods. IJCA Special Issue on Computational Science-New Dimensions & Perspectives, NCCSE.

[36] Lucia, A.D. and Qusef, A. (2010) Requirements Engineering in Agile Software Development. *Journal of Emerging Technologies in Web Intelligence*, **2**, 212-220. http://dx.doi.org/10.4304/jetwi.2.3.212-220

[37] Paetsch, F. (2003) Requirements Engineering in Agile Software Development. Diploma Thesis, University of Calgary University of Calgary, Alberta.

[38] Manifesto, C.H.A.O.S. (2013) Think Big, Act Small. The Standish Group International Inc. http://versionone.com/assets/img/files/ChaosManifesto2013.pdf

[39] Inayat, I., Salim, S.S., Marczak, S., Daneva, M. and Shamshirband, S. (2014) A Systematic Literature Review on Agile Requirements Engineering Practices and Challenges. *Computers in Human Behavior*, **51**, 915-929.

[40] Cao, L. and Ramesh, B. (2008) Agile Requirements Engineering Practices: An Empirical Study. *IEEE Software*, **25**, 60-67. http://dx.doi.org/10.1109/MS.2008.1

[41] Helmy, W., Kamel, A. and Hegazy, O. (2012) Requirements Engineering Methodology in Agile Environment. *International Journal of Computer Science Issues*, **9**, 293-300.

[42] Jun, L., Qiuzhen, W. and Lin, G. (2010) Application of Agile Requirement Engineering in Modest-Sized Information Systems Development. *Second World Congress on Software Engineering* (*WCSE*), Wuhan, 19-20 December 2010, Vol. 2, 207-210.

[43] Araujo, J. and Ribeiro, J.C. (2005) Towards an Aspect-Oriented Agile Requirements Approach. *Eighth International Workshop on Principles of Software Evolution*, 5-6 September 2005, 140-143. http://dx.doi.org/10.1109/iwpse.2005.31

[44] Martakis, A. and Daneva, M. (2013) Handling Requirements Dependencies in Agile Projects: A Focus Group with Agile Software Development Practitioners. *IEEE Seventh International Conference on Research Challenges in Information Science* (*RCIS*), Paris, 29-31 May 2013, 1-11. http://dx.doi.org/10.1109/rcis.2013.6577679

[45] Liu, L., Li, T. and Peng, F. (2010) Why Requirements Engineering Fails: A Survey Report from China. 18*th IEEE International Requirements Engineering Conference* (*RE*), Sydney, 27 September 2010-1 October 2010, 317-322. http://dx.doi.org/10.1109/re.2010.45

[46] Inayat, I., Moraes, L., Daneva, M. and Salim, S.S. (2015) A Reflection on Agile Requirements Engineering: Solutions Brought and Challenges Posed. *Scientific Workshop Proceedings of the XP*2015, Trondheim, p. 6. http://dx.doi.org/10.1145/2764979.2764985

[47] Soundararajan, S. and Arthur, J. D. (2009) A Soft-Structured Agile Framework for Larger Scale Systems Development. 16*th Annual IEEE International Conference and Workshop on the Engineering of Computer Based Systems*, San Francisco, 14-16 April 2009, 187-195. http://dx.doi.org/10.1109/ecbs.2009.21

[48] Mulla, N. and Girase, S. (2012) Comparison of Various Elicitation Techniques and Requirement Prioritisation Techniques. *International Journal of Engineering Research and Technology*, **1**, No. 3.

[49] Wiegers, K. (1999) First Things First: Prioritizing Requirements. *Software Development*, **7**, 48-53.

[50] Papadopoulos, G. (2015) Moving from Traditional to Agile Software Development Methodologies Also on Large, Distributed Projects. *Procedia—Social and Behavioral Sciences*, **175**, 455-463. http://dx.doi.org/10.1016/j.sbspro.2015.01.1223