Investigating the Link between User Stories and Documentation Debt on Software Projects

Henrique F. Soares[1], Nicolli S. R. Alves[1], Thiago S. Mendes[2,3], Manoel Mendonça[2], Rodrigo O. Spínola[1, 2]
[1]Graduate Program in Systems and Computer at Salvador University, Salvador, Brazil
[2]Fraunhofer Project Center for Software and System Engineering at UFBA, Salvador, Brazil
[3]Federal Institute of Bahia – IFBA, Salvador, Brazil
henriquefsoaresecomp@gmail.com, nicollirioss@gmail.com, thiagomendes@dcc.ufba.br,
manoel.mendonca@ufba.br, rodrigo.spinola@pro.unifacs.br

Abstract— **Technical debt is a metaphor that describes the effect of immature artefacts in software development. One of its types is documentation debt, which can be identified by locating missing, inadequate or incomplete artefacts in software projects. Nowadays, we can observe more organizations using agile methods to support their activities. In particular, the use of user stories reduces the focus on requirement specification tasks and, as a consequence, creates difficulties that need to be overcame by the development team. In order to investigate these difficulties and assess whether they create a favourable scenario for incurring documentation debt, this paper presents the results of a literature review and an exploratory study. The results from both studies allowed us to identify a list of causes that can lead the development team to incur documentation debt when working with agile requirements. This is an important step in order to manage the technical debt from a preventive perspective.**

Keywords- Technical Debt, Documentation Debt, User Stories, Agile Requirements, Literature Review, Exploratory Study

I. INTRODUCTION

Technical Debt (TD) includes internal development tasks that you choose not to perform now, but run the risk of causing future problems if not completed [1]. As TD is incurred on a software project, interest (extra effort required to eliminate technical debt) is accumulated and the debt payment tends to become more complex. A poor identification and management of TD can bring significant losses to a software project [9].

Different types of TD can occur throughout the phases of a software development process [2]. One of them is documentation debt, which refers to issues encountered in the software project documentation that can be identified by locating missing, inadequate or incomplete artefacts. The increase of the amount and complexity of the documentation over time contributes to higher debt interest, making the effort to pay off the debt even more difficult.

Documentation is produced throughout the software lifecycle, but mainly during the requirements specification process. Although the use of a traditional approach to work with requirements is widespread in the software industry, the necessity of delivering projects in shorter periods of time has taken this industry to change its practices and to gradually increase the use of agile methods in its projects [6]. The phase of requirements in agile methodologies, also called Agile Requirements (AR), is often performed with the use of user stories. While the traditional requirements methodologies tend to be more formally documented and controlled, user stories minimize the focus on documentation, shifting the concern to coding activities and software delivery [7]. This scenario, mainly characterized by a reduction on the formalization of the requirements specification, may be decisive in the occurrence of documentation debt.

To identify a particular type of TD, it is necessary to know what are its possible causes and indicators [8]. Currently, we already know some indicators of documentation debt (missing, inadequate or incomplete artefacts) [2], but we have no knowledge of the causes of incurring this type of debt when working with agile requirements yet. This work aims to investigate what the difficulties when working with agile requirements (specifically user stories) are, and analyze whether these difficulties create a favourable scenario for incurring documentation debt in software projects or not.

To achieve this goal, two complementary studies were performed (Fig. 1). The first, a controlled literature review, intended to identify the main difficulties when working with AR. The second, an exploratory study, aimed to characterize the difficulties in the use of user stories when compared with a traditional approach (use cases).

Each study was executed and analyzed separately. Following this, the results were consolidated and analyzed together. At the end, we obtained a list of difficulties that are present when working with user stories. It was observed that many of those difficulties can be considered causes for incurring documentation debt (i.e. they are related to existing indicators [2] of this type of debt). Thus, this work contributes to evolve the Technical Debt Landscape [3] [4] by identifying causes that lead to the insertion of documentation debt in software projects that use AR in their development.

978-1-4799-8828-0/15 $31.00 © 2015 IEEE
DOI 10.1109/ITNG.2015.68

Along with this introduction, this paper has four other sections. In section II the planning, execution, and the results of the literature review will be presented. Then, section III discusses the exploratory study. Section IV presents the joint analysis of both the studies. Finally, Section V has the concluding remarks of this work.

II. LITERATURE REVIEW

A. Planning

The literature review was performed following some of the good practices defined by Kitchenham and Charters (2007) for performing systematic reviews in software engineering. Our protocol was defined using the following research question as a base: ***"What are the main difficulties when working with agile requirements?"***. This research question was organized in three criteria: (1) **Population**: Published papers on agile requirements; (2) **Intervention**: Difficulties and problems when working with agile requirements, and; (3) **Outcome**: Difficulties and problems already evaluated through some empirical study.

In addition, three complementary research questions were derived from the main one: **(Q1)** What are the difficulties in identifying and managing requirements with agile methodologies? **(Q2)** What empirical evaluations were reported in the studies? **(Q3)** Is there any work relating agile requirements to technical debt?

Before starting the execution of the protocol, we defined an initial search string. It initially considered the three aforementioned criteria: *((agile requirement) AND (problem OR difficult OR issue) AND (evaluation OR study OR studies))*. However, after entering this string into some digital libraries, we observed that a small number of studies were found. Thus, it was necessary to adjust the search string in order to be more comprehensive and, as a consequence, minimize the risk of any relevant item for this research not being found. In the new search string, we decided to only consider the population and outcome criteria because we were only interested in papers that reported some kind of empirical evaluation. At the end, the considered search string was: *((agile requirements) AND (study OR studies OR evaluation))*.

Inclusion and exclusion criteria were defined too. The inclusion criteria were: (i) published papers describing difficulties when working with AR; (ii) when multiple papers have reported the same study, only the most recent will be included; (iii) when several studies are reported in the same paper, each study will be treated separately. On the other hand, the exclusion criteria were: (i) studies that have insufficient information about difficulties when working with AR; (ii) papers available as summaries or PowerPoint presentations.

The digital libraries considered in this work were: ACM DL, CiteSeerx, DBLP, Engineering Village, IEEE Xplorer, Scopus, and Springer.

B. Execution

In total, three researchers participated in the protocol execution. The study selection process was divided into three steps (Fig. 2). The first step represents the total number of studies returned by the execution of the string. It was performed in February 2014 and resulted on 198 studies for evaluation. Then, the exclusion criteria were applied by one of the researchers. At this stage, summaries of proceedings and duplicate papers were removed, resulting in the exclusion of 84 studies. In the third step, performed by two researchers (the third would be consulted in case of disagreement), the second filter was applied (titles and abstracts of 114 studies were read). During this process, the inclusion and exclusion criteria were applied. Finally, 19 papers were selected for reading and analysis. From the analysis of each study, it was possible to answer the research questions defined in the protocol.

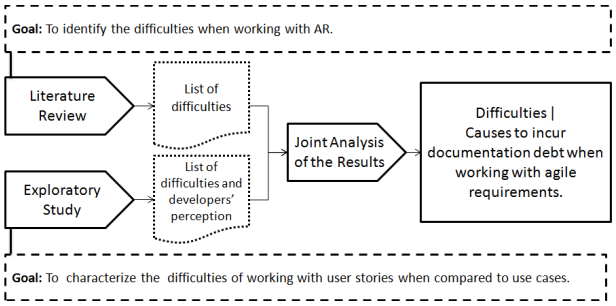

Figure 1. Research strategy.

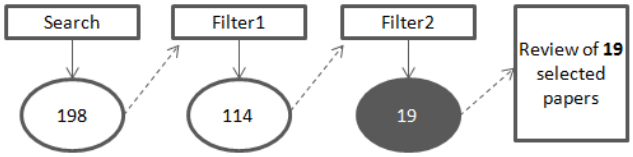

Figure 2. Filtering process.

Q1. What are the difficulties in identifying and managing requirements with agile methodologies?

The set of 10 difficulties extracted from the selected studies can be seen in Table I. The table is ordered by the number of papers that cited each difficulty.

Two of the identified difficulties are generic, i.e., they also occur in traditional requirements: requirements definition and requirements volatility. The **lack of information** was considered one of the main problems when working with AR. This difficulty was analyzed by five studies and has a direct impact on effort estimation, requirements gathering, and development tasks. Besides, it is strongly associated with other difficulties identified in the study: **non-functional requirements identification**, **prioritization of requirements** and **requirements validation**.

Q2. What empirical evaluations were reported in the studies?

Five different types of studies were performed on the selected papers: 1 ethnographic study, 9 case studies, 1 controlled study, 5 exploratory studies, and 1 survey. The empirical evaluations were conducted to characterize some difficulties in AR or to assess approaches that have been proposed to mitigate those problems. This result can be observed in the second column of Table I.

TABLE I. DIFFICULTIES WHEN WORKING WITH AGILE REQUIREMENTS

Difficulties	Evaluation
Requirement Prioritization: The definition of the requirement priority based only on the value that it has for the customer is a risk to the project [13] [18] [20] [25] [28] [29].	Case study, exploratory study
Non-functional Requirements Identification: AR do not address the non-functional requirements. The lack of specification of non-functional requirements can trigger future problems [14] [21] [22] [25] [26] [27].	Case study, exploratory study, survey
Lack of Information: Low level of detail of user stories brings difficulties to other development activities [12] [13] [15] [16] [22].	Case study, controlled experiment
Volatility of Requirements: Changes that may occur in the requirements during the lifecycle of the project [19] [20] [21].	Case study, survey
Requirements Definition: Difficulty that a user has to know what must be described to the development team [19] [22] [23].	Case study, exploratory study
Dependence between Requirements: Difficulties to deal with the dependency between requirements (in theory, user stories can be implemented individually without considering their dependences on other user stories, but in practice it is hard to work with this assumption) [26] [27].	Exploratory study
To Predict Impacts of Changes: It is hard to assess the impacts and risks of adding new requirements [24].	Case study
User Dependence: Stakeholders are not satisfactorily available to answer questions regarding the software requirements [17].	Exploratory study
Communication and Collaboration with Users: Sometimes it is hard to communicate and collaborate with the user in an effective way. This can bring bad consequences, because AR are based on continuous communication and collaboration with the user [10].	Ethnographic study
Requirements Validation: The validation of the requirements can be impaired due to the low level of detail on user stories. Divergences between what was requested and what was developed can happen in this scenario [16].	Case study

Q3. Is there any work relating agile requirements to technical debt?

We did not identify any study that associated agile requirements with the presence of technical debt in software projects.

C. Discussion

It was possible to identify 10 difficulties in the use of AR. Most of them are related to the low level of details in the requirements documentation, which is one of the prerogatives to increase agility in the requirements phase.

Additionally, the low number of selected papers in the literature review reveals that research on difficulties associated with the use of AR, which has some type of empirical evaluation, is still small. No studies correlating agile requirements approaches and technical debt were identified. The analysis of the balance between agility and quality in the requirements activities is not a trivial task, and it still requires further research. This work is performing this discussion from the perspective of technical debt.

D. Threats to Validity

The following threats to validity were identified: (1) **Research Question**: the set of difficulties identified in this study may not be comprehensive enough. To deal with this threat, the research questions were analyzed by three researchers, and one of them served as an external reviewer of the protocol; (2) **Publication Bias**: it was not possible to ensure that all relevant studies were returned in the searches. To minimize this threat, the main digital libraries in computing were considered; (3) **Search Execution**: each digital library has a particular way to compose words to create the search strings. Thus, it was necessary to adjust the search strings, taking into account the specificities of each library; (4) **Data Extraction**: we can not assume that all relevant primary studies were selected for this review, or that the returned studies were analyzed appropriately. To reduce this risk, the classification and information extraction process were performed by two researchers and reviewed by a third.

III. EXPLORATORY STUDY

A. Purpose and Research Questions

The goal of this study was to **analyze** the use of agile requirements (user stories) when compared to traditional approaches (use cases), **for the purpose of** characterizing, **with respect to** their (agile requirements) difficulties and limitations, **in the context of** software development projects, **from the point of view of** master students in software engineering with professional experience.

The research questions defined for this study were: ***(RQ1)*** *What are the difficulties of using agile requirements on software development projects?* ***(RQ2)*** *What is the perception of the participants regarding the use of agile requirements in software projects?* Both research questions intend to complement the questions defined in the literature review, and give us further insights on the presence of documentation debt in agile requirements.

Due to the exploratory nature of this study and the low number of participants involved in its execution, we chose to address the research questions using exploratory analysis and presentation of data, rather than a confirmatory type of analysis.

B. Instrumentation

In this study, the following instruments were developed: **(1) Consent and Characterization** (personal data, academic background and level of experience in software projects) **Forms**; **(2) Difficulties and Limitations**: in this form, participants reported the effort to accomplish the requirements specification using agile and traditional approaches, and the difficulties encountered on the use of user stories; **(3) Participant's Perception of the Use of Agile Requirements**: this form contains a list of statements about the use of agile approaches in requirements. The participant reports how he/she

agrees or disagrees with each statement, using the *likert scale* (strongly disagree, disagree, neither agree nor disagree, agree, strongly agree).

Furthermore, two scenarios based on a real software system were defined: **(Scenario 1)** supplier performance reports; **(Scenario 2)** order tracking and management. For each scenario, prototypes were available in order to support their understanding. Finally, two templates were created: use case template and user story template.

C. Procedure and Data Collection

The participants (10 master students in software engineering with professional experience) of the study were initially trained in use cases and user stories. This training was in the classroom and included some practical exercises.

In a second moment, the participants answered the characterization questionnaire indicating their **time and level of experience in software development** considering the following areas: (a) Software Project Management; (b) Software Requirements; (c) Use Cases; (d) Agile Requirements (user stories); (e) Software Testing; (f) Software Maintenance; (g) Coding. Based on this information, the participant's level of experience in the use of agile and traditional approaches was identified. This information was considered to define two workgroups, in order to avoid the concentration of knowledge.

Each group received two scenarios to specify the requirements of a Purchase Management System. The description of the requirements for each scenario was performed individually, and considered the execution order defined in Table II. Thus, each participant performed four specification activities, an agile/traditional combination for each defined scenario. As each scenario was specified by each participant using the two approaches (traditional and agile), it was necessary to change the order of the use of each approach, to minimize the impact that the prior knowledge of the specified requirement in the first approach has on the use of the second one.

At the end, participants filled in the forms of (1) difficulties and limitations, and then the (2) Participant's Perceptions on the Use of Agile Requirements.

TABLE II. SPECIFICATION ORDER

Scenario	Group 1 (Execution Order) →		Group 2 (Execution Order) →	
Supplier performance reports	User Story	Use Case	Use Case	User Story
Order tracking and management	Use Case	User Story	User Story	Use Case

D. Results

RQ1. What are the difficulties of the use of agile requirements on software development projects?

To answer this question, each participant was asked to indicate the difficulties they observed when working with user stories (Difficulties and Limitations form). At the end, we identified 8 difficulties (see Fig. 3). We can observe that lack of information was mentioned by 8 of the 10 participants of the study. This proportion was high when compared with the other difficulties, and reinforces its importance in the scenario of AR. Furthermore, another difficulty that stood out was the fact that the use of AR hinders the understanding of the requirements to be implemented.

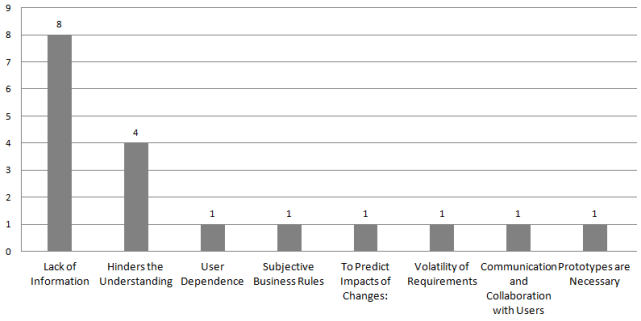

Figure 3. Number of participants that cited each difficulty.

RQ2. What is the perception of the participants regarding the use of requirements in agile software projects?

To answer this question, each participant filled in a form reporting their level of agreement with seven statements (**S.x**) about the use of AR: (**S.1**) AR contain sufficient information to effort estimation; (**S.2**) AR contain sufficient information to support the design of the software architecture; (**S.3**) AR contain relevant information for performing software maintenance activities; (**S.4**) AR reduce the effort required to identify the software requirements; (**S.5**) AR reduce the effort required to perform the requirements specification activities; (**S.6**) AR can be used to support the software validation activities, and; (**S.7**) AR require greater detail before they can be effectively used on software development activities.

To reach the results presented in Fig. 4, the following formula was used for each statement:

$$Weighted\ Avg(x) = \frac{w1*x1 + w2*x2 + \ldots\ wn*xn}{w1+w2+\ldots wn} \qquad (1)$$

It indicates that each participant has a weight that needs to be considered. This weight represents the participants' level of experience. The formula: (1) calculates the number of points that a specific statement (x) has for each participant by multiplying the participant's perception (number of *1-strongly disagree* to *5-strongly agree* based on the likert scale) by their level of experience (w); (2) sums the values calculated on the step 1 for each statement and divides the result achieved by the sum of the levels of experience of each participant. Results next to 5 indicate a high level of agreement with the statement. On the other side, values next to 1 show a high level of disagreement.

Analysing the results presented on Fig. 4:

(1) **S.1**, **S.2,** and **S.3** indicate that user stories may not be enough to support other development activities such as effort estimations, architecture design, and software maintenance;

(2) Results of **S.4** and **S.5** highlight the perception that there is a gain of time in requirements activities when using agile methods. This is justified by the fact that the requirements are specified with a low level of detail;

(3) Regarding **S.6,** it was observed that it is possible to use AR to support software validation activities;

(4) Statement **S.7** has obtained a high level of agreement too. This indicates that, according to the perception of the participants, one of the major difficulties in the use of AR is the lack of information on the specified requirements.

E. Discussion

Results from the exploratory study indicate several difficulties when working with AR. In our opinion, the main concern about the use of AR is the lack of information. Although the use of AR can provide an initial gain in terms of time during requirements specification activities, difficulties like the lack of a detailed requirement may cause the development of functionalities that are not aligned with ones expected by the customer. Moreover, participants' perceptions indicate that other development activities, such as, for example, maintenance and architecture design, become yet more challenging.

F. Threats to Validity

The threats to the validity considered in this study were: **Used Questionnaire**: it is not possible to guarantee that all participants have understood in the same way all the questions. To minimize this threat, each question was explained before its completion; **Experience of the participants**: the participation of professionals with homogeneous levels of experience is unlikely to occur. To minimize this threat, we used a characterization questionnaire to support the definition of the work groups, and also to calculate (see section III.D) the results presented on Fig. 4.

Finally, a limitation of this study was the low number of participants. It is necessary to replicate this study with a greater number of participants in order to strengthen the obtained results.

IV. JOINT ANALYSIS OF THE RESULTS

The obtained results from the Literature Review (LR) and Exploratory Study (ES) are complementary. In total, we identified 13 difficulties when working with AR in software projects (8 from the ES and 10 from the LR). Five difficulties (see Fig. 5) were cited in both studies: (i) **lack of information**; (ii) **volatility of requirements**; (iii) **user dependency**; (iv) **predicting impacts of changes**; and (v) **communication and collaboration with users**.

Based on the findings from both studies, we have associated the indicators of documentation debt mentioned in [2] with the 13 difficulties identified. This mapping was performed individually by 4 of the authors of this paper. Each relation (a difficulty that can **cause** the presence of an indicator) between difficulty and indicator that was reported by at least two researchers was considered valid. Table III shows

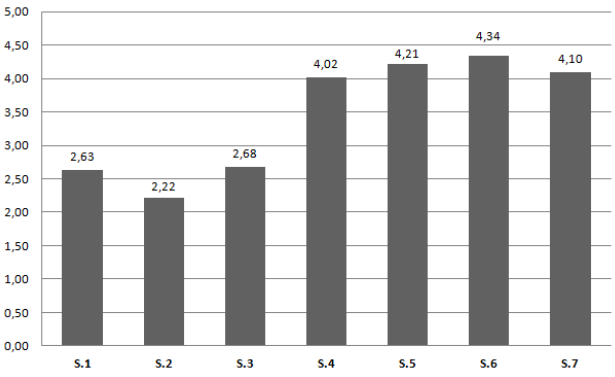

Figure 4. Participants' perception by statement.

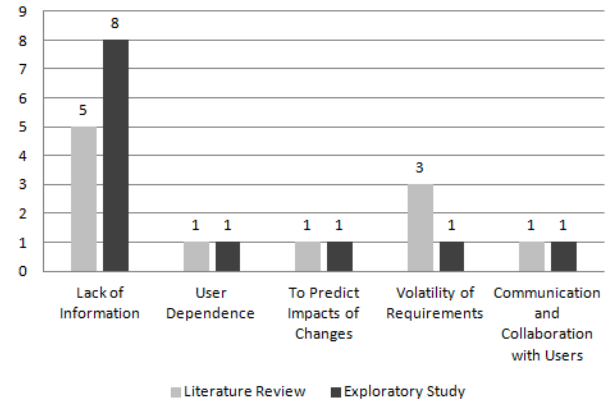

Figure 5. Difficulties identified in both LR and ES.

TABLE III. DOCUMENTATION DEBT INDICATORS VS DOCUMENTATION DEBT CAUSES.

Documentation Debt Indicators	Documentation Debt Causes
Documentation does not Exist	- Lack of information [LR and ES]
Incomplete Design Specification	- Requirement Prioritization [LR] - To Predict Impacts of Changes [LR] - Non-functional Requirements Identification[LR] - Lack of information [LR and ES] - Communication and Collaboration with Users[LR and ES] - Volatility of requirements [LR and ES] - Requirements Definition [LR] - Dependence between Requirements [LR] - Hinders the Understanding [ES]
Incomplete Documentation	- Lack of information [LR and ES] - Communication and Collaboration with Users[LR and ES] - Requirements Validation [LR] - Requirements Definition [LR] - Dependence between Requirements [LR] - Prototypes are Necessary [ES] - Hinders the Understanding [ES] - Requirement Prioritization [LR] - Non-functional Requirements Identification[LR]
Insufficient Comments in Code	
Outdated Documentation	- To Predict Impacts of Changes [LR] - Lack of information [LR and ES] - Volatility of requirements [LR and ES] - Requirement Prioritization [LR]
Test Documentation	- Lack of information [LR and ES] - Dependence between Requirements [LR]

the result of this mapping. We can observe that the difficulties encountered in the studies may be considered causes of problems in the software documentation, and may lead to the insertion of documentation debt in projects that work with AR.

V. FINAL REMARKS

Technical debt is a recent research area and brings a series of challenges and opportunities [2]. This work contributes to evolving the Technical Debt Landscape [3] [4] by identifying causes that lead the development team to incur documentation debt on its projects. To this end, this paper presented a literature review, which identified the main difficulties when working with agile requirements, and an exploratory study that characterized the difficulties in the use of user stories when compared to use cases. Those difficulties were mapped to documentation debt indicators and some of them can be considered causes of this type of debt. We believe that the results reached can support the development team in the hard task of managing the documentation debt, from a preventive perspective.

This work is part of a more comprehensive research. Currently, we are working on the planning of a retrospective study to investigate the consequences that some of the identified documentation debt causes can actually bring to a real software project, that used user stories as its methodology to specify the requirements of the project.

ACKNOWLEDGMENT

This work has been developed by members of tdresearchteam.com that is partially supported by CNPq Universal 2014 grant 458261/2014-9.

REFERENCES

[1] C. Izurieta, A. Vetró, N. Zazworka, Y. Cai, C. Seaman, and Shull, F., "Organizing the Technical Debt Landscape," in 3rd International Workshop on Managing Technical Debt, Zurich, Switzerland, June 2-9, 2012.

[2] N. S. R. Alves, L. F. Ribeiro, V. Caires, T. S. Mendes, and R. O. Spínola, "Towards an Ontology of Terms on Technical Debt." in 6rd International Workshop on Managing Technical Debt, Victoria, BC, Canada, 2014. DOI: 10.1109/MTD.2014.9

[3] C. Seaman and Y. Guo, "Measuring and Monitoring Technical Debt," Advances in Computers 82, 25-46, 2011.

[4] P. R. L. Nord and I. Ozkaya, "Technical Debt: From Metaphor to Theory and Practice, IEEE Software, Published by the IEEE Computer Society, 2012.

[5] B. Kitchenham and S. Charters, "Guidelines for performing Systematic Literature Reviews in Software Engineering," Technical Report EBSE 2007-001, Keele University and Durham University, 2007.

[6] VersionOne, "Stage of Agile Development". Available from: http://www.versionone.com/state-of-agile-survey-results, 2013.

[7] M. Fowler and J. Highsmith, "The Agile Manifesto, Software Development," vol. 9, 2001.

[8] N. Zazworka, R. O. Spinola, A. Vetró, F. Shull and C. Seaman, "A case study on effectively identifying technical debt," in 17th Int. Conference on Evaluation and Assessment in Software Engineering, 2013. DOI: 10.1145/2460999.2461005

[9] S. McConnel, "Managing Technical Debt," Construx Software, Version 1. Available from: http://www.construx.com, 2008.

Papers identified in the Literature Review

[10] N. N. B. Abdullah, S. Honiden, H. Sharp, B. Nuseibeh, and D. Notkin, "Communication patterns of agile requirements engineering," in 1st Workshop on Agile Requirements Engineering, ACM, New York, USA., pp. 1–4, 2011.

[11] E. Bjarnason, K. Wnuk, and B. Regnell, "A case study on benefits and side-effects of agile practices in large-scale requirements engineering," in 1st Workshop on Agile Requirements Engineering, pp. 3:1–3:5, New York, USA. ACM, 2011.

[12] M. Daneva, E. Van Der Veen, C. Amrit, S. Ghaisas, K. Sikkel, R. Kumar, N. Ajmeri, U. Ramteerthkar, and R. Wieringa, "Agile requirements prioritization in largescale outsourced system projects: An empirical study," J. Syst. Softw. pp. 1333–1353, 2013.

[13] W. M. Farid, "The normap methodology: Lightweight engineering of nonfunctional requirements for agile processes," in Proc. of the 19th Asia-Pacific Soft. Eng. Conference, Washington, DC, USA. IEEE Computer Society, vol. 01, pp. 322–325, 2012.

[14] R. Gallardo-Valencia, V. Olivera, and S. Sim, "Are use cases beneficial for developers using agile requirements?" in Fifth Int. Workshop on Comparative Evaluation in Requirements Engineering, Washington, DC, USA. IEEE Computer Society, pp. 11–22, 2007.

[15] R. E. Gallardo-Valencia and S. E. Sim, "Continuous and collaborative validation: A field study of requirements knowledge in agile," in 2rd International Workshop on Managing Requirements Knowledge, Washington, DC, USA. IEEE Computer Society, pp. 65–74, 2009.

[16] S. Kelly, "Towards an evolutionary framework for agile requirements elicitation," in 2nd ACM SIGCHI Symp. on Engineering Interactive Computing Systems, New York, NY, USA. pp. 349–352, 2010.

[17] Z. Racheva, M. Daneva, K. Sikkel, A. Herrmann, and R. Wieringa, "Do we know enough about requirements prioritization in agile projects: Insights from a case study," in 18th IEEE Int. Requirements Eng. Conf., Washington, DC, USA. IEEE Computer Society, pp. 147–156, 2010.

[18] A. M. Sen and K. Hemachandran, "Elicitation of goals in requirements engineering using agile methods," in the 2010 IEEE 34th Annual Computer Software and Applications Conference Workshops, Washington, DC, USA. IEEE Computer Society, pp. 263–268, 2010.

[19] X. Xu, B. Zhang, and J. Lin, "Management information system requirements analysis model based on the agile development," in Int. Conference on Control Engineering and Communication Technology, Washington, DC, USA. IEEE Computer Society, pp. 986–990, 2012.

[20] M. U. Malik, N. Majeed, C. Khurram, and S. Malik, "Evaluation of efficient requirement engineering techniques in agile software development," International Journal of Computer Applications 83(3):24-29, 2013.

[21] B. Ramesh, L. Cao, and R. Baskerville, "Agile requirements engineering practices and challenges: an empirical study," Inf. Syst. J., pp. 449–480, 2010.

[22] B. Ramesh, L. Cao, and R. Baskerville, "Agile requirements engineering practices and challenges: an empirical study," Inf. Syst. J., pp. 449–480, 2010.

[23] A. Batool, Y. Motla, B. Hamid, S. Asghar, M. Riaz, M. Mukhtar, and M. Ahmed, "Comparative study of traditional requirement engineering and agile requirement engineering," in 15th Int. Conference on Advanced Communication Technology, pp. 1006–1014, 2013.

[24] W. Farid, and F. Mitropoulos, "Visualization and scheduling of non-functional requirements for agile processes," in Southeastcon, Proceedings of IEEE, pp.1–8, 2013.

[25] A. Martakis and M. Daneva, "Handling requirements dependencies in agile projects: A focus group with agile software development practitioners," in IEEE Seventh International Conference on Research Challenges in Information Science, pp. 1–11, 2013.

[26] P. Rodríguez, A. Yagüe, P. Alarcón, and J. Garbajosa, "Some findings concerning requirements in agile methodologies," in Bomarius, F., Oivo, M., Jaring, P., and Abrahamsson, P., editors, Product-Focused Software Process Improvement, v. 32 of Lecture Notes in Business Information Processing, Springer Berlin Heidelberg, pp. 171–184, 2009.

[27] M. Khelghatdost, Z. Hajilary, and H. Pourkarim, "Requirements in agile software development," Life Science Journal, pp. 326–330, 2013.

[28] L. Jun, W. Qiuzhen, and G. Lin, "Application of agile requirement engineering in modest-sized information systems development," in Software Engineering, Second World Congress on Software Engineering, v. 2, pp. 207–210, 2010.