Although the use of AR can provide an initial gain in terms of time during requirements specification activities, difficulties emphasized like "Lack of Information" may mean that business rules were not specified, leading to the development of functionalities that are not well aligned with the customer expectations, and a loss in the long term. Long term lost from the example in the paper showed that up to almost half the cost and time spent in the future were on maintenance effort related to documentation debt - which is very significant.

So not sure if the initial time gain is worth it in the long run. What do people think?

how to justify the effort for documentation requirement over time - keep recording on multiple project data how much time is gained at the start vs how much time is needed for maintenance.
do user stories lead to less thorough documentation - they are rather light weight and by definition don't have too much information, so could be enhanced with more documentation to capture more details.
how to reduce documentation debt in an agile environment - how to use activities that developers do anyway, to weave with documentation activities.

Impacts of Agile Requirements Documentation Debt on Software Projects: A Retrospective Study

Thiago Souto Mendes
Federal Institute of Bahia and Federal University of Bahia
thiagomendes@dcc.ufba.br

Henrique Frota Soares
Salvador University
henriquefsoaresecomp@gmail.com

Mário André de F. Farias
Federal Institute of Sergipe and Federal University of Bahia
mario.andre@ifs.edu.br

Marcos Kalinowski
Fluminense Federal University
kalinowski@ic.uff.br

Manoel Mendonça
Fraunhofer Proj. Center at UFBA
manoel.mendonca@ufba.br

Rodrigo Oliveira Spínola
Fraunhofer Proj. Center at UFBA and Salvador University
rodrigo.spinola@pro.unifacs.br

ABSTRACT

Documentation debt is a type of technical debt that describes problems in documentation such as missing, inadequate or incomplete artifacts. Unlike traditional methods, agile methodologies usually employ short iterative cycles and rely on tacit knowledge within a team. In particular, Agile Requirements (AR) (e.g., user stories) tend to reduce the focus on requirements specification activities. This scenario contributes to the occurrence of documentation debt. The goal of this paper is to investigate the impact that this type of debt brings to projects developed by using AR. We address this goal by performing a retrospective study in a real software project that used AR in its development. Our analysis was concentrated on data from 132 maintenance and evolution tasks. Of this total, 65 were related to the presence of documentation debt and were performed within a timeframe of 18 months. The findings indicated an extra maintenance effort of about 47% of the total effort estimated for developing the project and an extra cost of about 48% of the initial cost of the development phase.

CCS Concepts

• Software and its engineering → Software creation and management

Keywords

Documentation Debt, Technical Debt, User Stories, Agile Requirements, Retrospective Study.

SAC 2016, April 04-08, 2016, Pisa, Italy

DOI: http://dx.doi.org/10.1145/2851613.2851761

1 INTRODUCTION

Technical Debt (TD) illustrates the problem of pending maintenance tasks as a type of debt that brings a short-term benefit to the project, but that may have to be paid with interest later in the software development process [2][5][6][9][10]. There are different types of debt related to the phase and/or artifacts in the software development process in which they are inserted [1]. Documentation debt refers to issues related to software project documentation. This type of debt occurs when we have, for example, the following indicators [2][10]: missing, inadequate, or incomplete documentation in the project.

Documentation is produced throughout the software lifecycle, but mainly during the requirements specification process. Although the usage 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 usage of agile methods in its projects [12]. Agile methodologies have helped to address some specific problems of requirements engineering and caused others [4]. In particular, the commonly used user stories reduce the focus on requirements specification activities. This scenario, mainly characterized by a reduction in the formalization of the requirements specification, may be decisive for the occurrence of documentation debt.

In this context, Soares *et al.* [10] presented results of two studies that investigated difficulties when working with Agile Requirements (AR) (specifically user stories) and analyzed whether these difficulties create a favorable scenario for incurring documentation debt in software projects. The first study, a controlled literature review, identified the main difficulties when working with AR. The second, an exploratory study, characterized the difficulties in the use of user stories when compared to use cases. The results from both studies allowed the authors to identify a list of indicators of the presence of documentation debt when working with AR. Lack of information was considered one of the main problems.

This paper extends the work of Soares *et al.* [10] with an additional study to investigate impacts that documentation debt

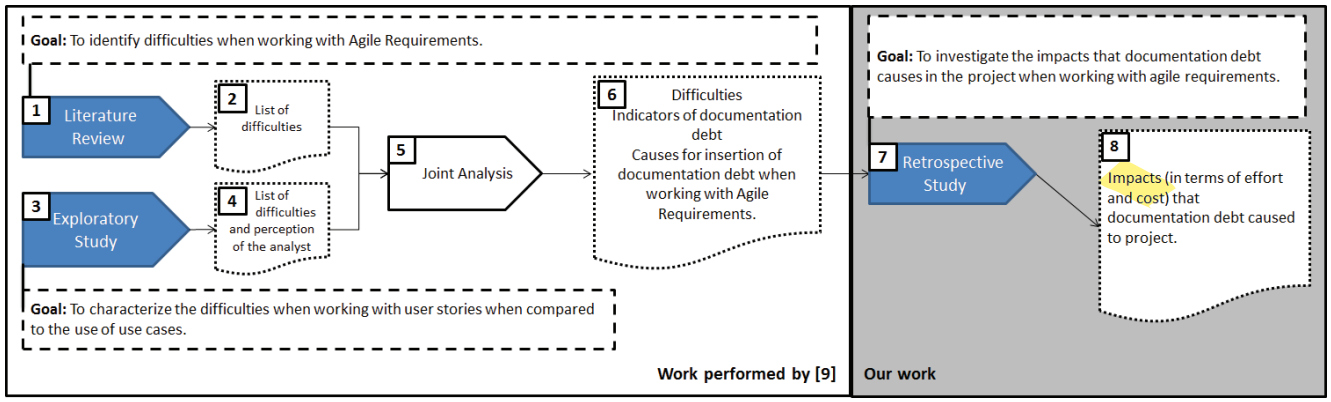

Figure 1: Research method.

brings to projects developed by using agile requirements (cf. Figure 1). We address this research goal by performing a retrospective study in a real software project that used AR in its development. A retrospective study is a type of empirical evaluation based on data from the past [8]. The project analyzed in this work was developed two years ago. Our analysis was concentrated on data from 132 maintenance and evolution tasks performed on a course management system. The customer requested these tasks after the software was deployed. Of this total, 65 maintenance tasks were related to the presence of documentation debt. It was possible to identify impacts that documentation debt related to AR can bring to projects and measure them in terms of extra effort and cost. Our findings suggest an extra effort of about 47% of the total effort estimated for developing the project and an extra cost of about 48% of the initial cost of the development phase. The maintenance tasks were performed within a timeframe of 18 months (cf. Figure 2).

The remainder of this paper is organized as follows. In Section 2 we discuss difficulties and limitations when working with AR. In Section 3 we present the design and execution of the retrospective study. In Sections 4 and 5, the main findings and results are summarized and discussed. In Section 6 the threats to validity are presented. Finally, Section 7 contains the concluding remarks.

2 DIFFICULTIES WHEN WORKING WITH AGILE REQUIREMENTS

Agile methods require extensive communications, collaboration, and trust that, if not present in a project, can quickly result in problems [4]. Besides, also according to Fernandez *et al.* [4], it seems that agility does not necessarily compensate for problems that non-agile models cause. The problems just manifest themselves in different ways.

Soares *et al.* [10] presented a controlled literature review in an effort to identify the difficulties and problems when working with AR (item 1 in Figure 1). The authors considered the following research questions in their study: **(Q1)** What are the difficulties in identifying and managing requirements with agile methodologies?, **(Q2)** What empirical evaluations were reported in the studies?, and **(Q3)** Is there any work relating agile requirements to technical debt?

At the end, 19 papers were selected for analysis. As a result, it was possible to identify 10 difficulties in the use of AR (item 2 in Figure 1): Difficulty to Prioritize Requirements due to their high abstraction level, Lack of Non-functional Requirements Identification, Lack of Information, Volatility of Requirements, Definition of Requirements with a Low Level of Detail, Difficulty to Define Dependencies between Requirements, Difficulty To Predict Impacts of Changes, User Dependence, Problems with the Communication and Collaboration with Users, and Challenge to Validate not Detailed Requirements. 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.

In addition, the low number of selected papers in the literature review revealed that research on difficulties associated with the use of AR, including type of empirical evaluation, is still small. No studies correlating agile requirements approaches and technical debt were identified at that time.

Soares *et al.* [10] also performed an exploratory study to analyze the use of AR (user stories) when compared to traditional approaches (use cases) (item 3 in Figure 1). Results from this study also indicated several difficulties as noticed in practice when working with AR and additional perceptions of the analysts (item 4 in Figure 1). It was noticed that, 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 specification may lead to the development of functionalities that are not well aligned with the customer expectations. Moreover, participants' perceptions indicated that other development activities, such as, for example, maintenance and architecture

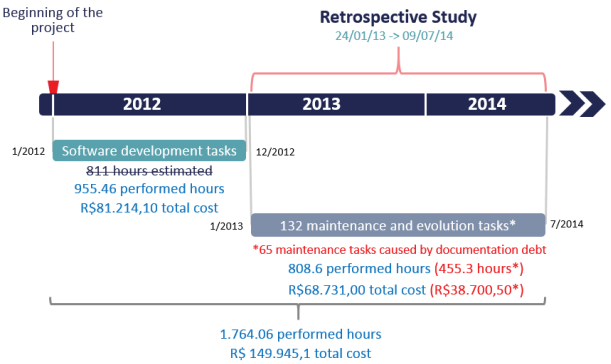

Figure 2: Study timeline.

Table 1: Collected data from the project.

Task ID	Task Description	Software Module	Effort (hours)	Cost (R$)	Beginning of Task (DD/MM/AAAA)	Task Reason
7599	Create structure to allow syndicate with certificate expired log requests	Association	4	R$ 340,00	24/01/2013	Business rule not on the User Story
7617	Error in calculating the end date on the import request.	Event	2	R$ 170,00	28/01/2013	Error
7605	Changing the event request report	Report	2	R$ 170,00	28/01/2013	New requirement
8120	In report "Events by Action Line", show the programmed and executed events	Report	8	R$ 680,00	29/04/2013	Change of requirement
8156	Analyze why there are modules without any events in the database of SISGE	Event	8	R$ 680,00	06/05/2013	Unexpected behavior of the software
8171	Optimize screen of Transfer Package	R1 Package	4	R$ 340,00	07/05/2013	Non-functional not specified

design, become more challenging.

Then, Soares *et al.* [10] mapped the list of difficulties to documentation debt indicators and the authors stated that some of them could be considered causes of this type of debt (items 5 and 6 in Figure 1). In this paper, we extend the research done by Soares *et al.* [10], presenting an additional study to investigate the consequences that documentation debt related to AR can bring to the project (items 7 and 8 in Figure 1).

3 RETROSPECTIVE STUDY

3.1 Context

The design of this retrospective study involves the extraction and analysis of data from an industrial software project developed two years ago. The company that developed the project has more than 20 years and works with software development since its foundation in 1993. It is a small company with about 50 employees, of which 10 professionals worked partly in the project selected for this study. The company uses agile methods in its software development process.

The considered project is a course management system that has in its scope functionalities such as: registration of participants, instructors' payment, and management reports. It is a web application developed in Java and SQL Server 2008.

Figure 2 shows the timeline of this study. The software development tasks of the system were performed in 2012. The system took one year to be developed, with several partial releases during this period. The project effort for developing the initial scope of the project was estimated in 811 hours. However, the development team spent 955.46 hours to complete the development phase in 2012. The retrospective study considered 132 maintenance and evolution tasks performed between January 2013 and July 2014. During this period, 808.6 hours were spent to perform them. Overall, the team spent 1.764.6 hours working on development and maintenance tasks of the project.

The project was developed using Scrum. The requirements were identified through meetings with the customer and specified using user stories. The customer was not physically present all the time with the project team during the development, but it was possible to quickly contact him by phone, e-mail, video conference, or any other means of communication to elucidate any possible doubt.

3.2 Study Goal and Research Questions

The goal of this study can be described, following the template of the *Goal Question Metric* paradigm [3], as follows. **Analyze** the documentation debt related to the use of AR (user stories) **for the purpose of** characterizing **with respect to** the impacts that it can cause on the project in terms of extra effort and cost **from the viewpoint of the** project manager **in the context of** an industrial software development project. Thus, we intend to investigate the following research question:

- **RQ:** What are the impacts (in terms of extra effort/time and cost) caused by documentation debt related to the use of AR on the project?

The null (H0) and the alternative hypotheses (H1 and H2) are:

- **Null hypothesis (H0):** The documentation debt related to the use of AR does not have any impact on the project (**H0: $\iota = 0$ and $\mu = 0$**).
- **Alternative hypothesis (H1):** The documentation debt related to the use of AR does have an impact on the project in terms of effort (**H1: $\iota > 0$**).
- **Alternative hypothesis (H2):** The documentation debt related to the use of AR does have an impact on the project in terms of cost (**H1: $\mu > 0$**).

Where (ι) is the extra effort, in hours, spent to adjust any potential problem caused by the documentation debt, and (μ) is the cost, in Reais (R$), to adjust any problem caused by the debt.

3.3 Data Collection and Procedure

In this study, we only used data of the demands that arose after the system was deployed. This criterion was established to facilitate the filtering of tasks that represent demands related to the maintenance of the system.

In order to access the project information, the company provided a temporary read-only access to the project management tool used in project. Moreover, the project manager was available to elucidate any questions about the project data. He provided information about the company, project, and, mainly, described the root cause of each maintenance task reported in the project monitoring spreadsheet.

The company uses the Redmine[1] issue tracking system to manage its development tasks. We used this tool to select the maintenance and evolution tasks accomplished between January 2013 and July 2014 for analysis (see Figure 2). Overall, 132 maintenance tasks were accomplished during this period. The information collected for each task was:

- **Task Id:** identifier of the registered task;
- **Task Description:** brief description of the task;
- **Software Module:** defines the module of the software the task is related to;
- **Effort:** time spent in the execution of the task;
- **Cost:** cost of the task, calculated based on the value per man-hour (M/H) of the project;
- **Beginning of task:** start date of the task.

In complement, we asked the project manager to indicate possible reasons for the maintenance tasks (**Task Reason**). The project manager indicated six different reasons. A fragment of this data (translated to English) can be observed in Table 1. The complete set of information collected from the project is available online[2].

Table 2: Reasons for conducting maintenance tasks.

Reason of Maintenance Tasks	Responsibility to adjust the maintenance task
Business rule not specified in the user story	As there was no business rule in the user history, the company (development team) had to assume the cost of the change.
New requirement	Addition of new requirements from the customer. The customer assumed the cost of the development of the new requirement.
Error	Development team.
Change of requirement	The customer assumed the cost for the change.
Non-functional not specified	Development team.
Unexpected behavior of the software	Development team.

These reasons and the decision (indicated by the project manager) if the responsibility to perform the task should owe the customer or the development organization are presented in Table 2.

Then, we mapped the reasons for the maintenance tasks to the indicators of documentation debt in software projects identified by Soares *et al.* [10]. This allowed us to identify which maintenance tasks were related to documentation debt (see Table 3). This mapping was performed by two researchers, experts in the technical debt area, and was reviewed by a third specialist.

With all data prepared for analysis, we performed the data analysis in order to answer the research question.

[1] Redmine: http://www.redmine.org/

[2] Gathered project information: https://goo.gl/JXWUzA

4 RESULTS

132 maintenance tasks were grouped by the project manager according to their reasons for being accomplished. Figure 3 shows the number of tasks per reason. Of this total, 65 maintenance tasks were related to indicators of documentation debt (see Table 3): business rules that were not specified in the user stories (42), requirements change (18), and non-functional requirements not specified (5). They represented about 50% of maintenance requests.

Table 3: Equivalence between reasons for conducting maintenance tasks and indicators of documentation debt.

Reason of Maintenance Tasks	Indicators of documentation debt
Business rule not specified in the user story	Lack of information
New requirement	-
Error	-
Change of requirement	Volatility of requirements
Non-functional not specified	Identification of non-functional requirements
Unexpected behavior of the software	-

The pie chart shown in Figure 4 allows observing that the type of documentation debt **Lack of Information** was related to 42 tasks, i.e. 65% of the total. This proportion was high when compared to the other two types, and reinforces the reduction of the focus on requirements specification activities in the scenario of AR. In addition, we have **Volatility of Requirements**, which represented 28% of the tasks with 18 in total. Finally, the **Lack of Non-functional Requirements Identification**, with 5 tasks, represented 8% of the total. Taking into account that the lack of identification of non-functional requirements contributes to the **Lack of information**, the amount of the maintenance tasks that would be related to the **Lack of Information** would exceed the 70% with 47 of the total.

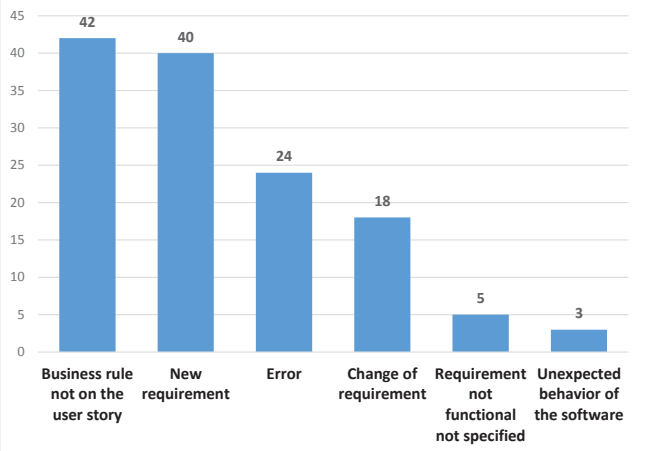

Figure 3: Total of maintenance/evolution tasks per reason.

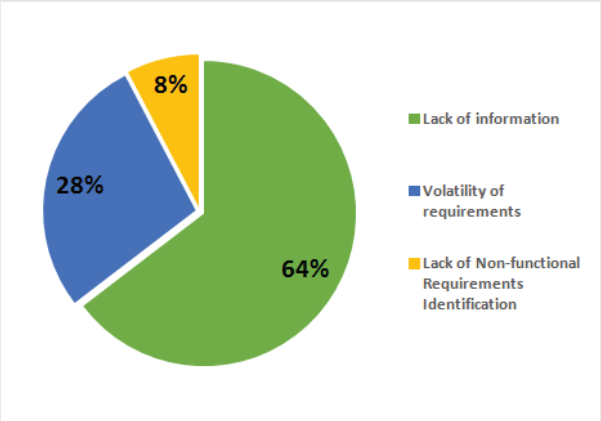

Figure 4: Proportion of tasks of Retrospective Study because of documentation debt.

This data indicates that documentation debt affects software projects that use AR in their development by being a source of different maintenance tasks.

The maintenance tasks selected were analyzed regarding the costs and amount of hours spent for carrying them out. Figure 5 shows the total time (in hours) spent by each indicator of documentation debt. The amount of hours spent to perform the tasks associated to documentation debt caused by **Lack of Information** was 296.3 hours, **Volatility of Requirements** was 107 hours, and **Lack of Non-functional Requirements Identification** was 52 hours. We can see that the effort involved to solve those issues was very significant for the project. In total, 455.3 hours of maintenance effort related to documentation debt was needed from January 2013 to July 2014, within the timeframe of 18 months. This represents the relevant amount of 25.81% of the overall project effort (1.764.06 hours) between 2012 and 2014, and 47.81% of the development phase in 2012.

The extra cost necessary to pay off the debt and solve the maintenance requests within the investigated timeframe was calculated based on the value of man-hour used in the project (R$ 85,00). Figure 6 shows the graph with these values in Reais (R$).

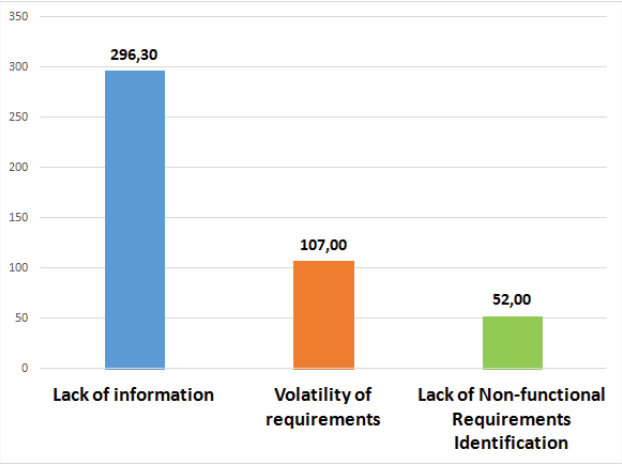

Figure 5: Hours per indicator of documentation debt.

The extra cost to perform the adjustments is distributed by the indicators of documentation debt as follows: **Lack of Information -** R$ 25.185,50; **Volatility of Requirements -** R$ 9.095,00, and; **Identification of Non-Functional Requirements:** R$ 4.420,00. At the end, the project had an extra maintenance cost of R$ 38.700,50. Again, this value was very expressive and represented about 48% of the value of the development phase (R$ 81.214,10).

Thus, although using a single project doesn't allow to perform any statistical hypothesis testing, the data provides preliminary indications from a real industry project supporting the alternative hypothesis, given that in this retrospective study clearly: $\iota > 0$ **and** $\mu > 0$.

However, we would like to highlight that this result concerns a specific project and that further replications of this study are needed to reinforce this preliminary indication. Also, we did not consider the amount of effort and cost that may have been saved during the requirements specification phase by adopting a lightweight AR approach.

5 DISCUSSION

The retrospective study investigated impacts caused by the presence of documentation debt in a real software project that used AR. Although the results presented in this study are preliminary indications, we consider them relevant, especially because they were obtained from a real software project.

Our results indicate that documentation debt when working with AR can have a high impact on the project maintenance effort and cost. Lack of information was the main indicator of documentation debt related to the maintenance tasks. This means that the development team needs to pay attention to issues related to some documentation items when working with AR, e.g., existence of business rule in the user story.

Considering the effort and cost, the data showed us an expressive amount of hours spent to perform maintenance tasks associated with documentation debt. Within 18 months, the cost involved to solve these maintenance tasks was almost half of the total value of the development phase of project. In this sense, projects developed using AR may produce a large amount of technical

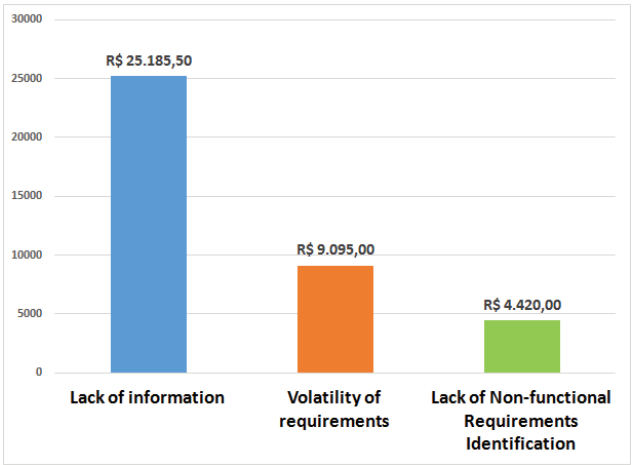

Figure 6: Total cost per indicator of documentation debt.

debt regarding documentation issues, which in turn may have serious financial consequences.

6 THREATS TO VALIDITY

The external, internal, construct and conclusion validity of the experiment are discussed below:

- **External Validity:** The study involved a real project developed by 10 professional software developers. However, the results cannot be generalized to equivalent projects. It is necessary to conduct further studies with a greater number of software projects with different contexts and dimensions.
- **Internal Validity:** this study was based on data stored in a project management tool. Each task was created and executed by following specific and formal rules of the company that executed the project. Therefore, the threats to internal validity were minimized. However, part of the study required information from the project manager who managed the system development, which may have caused the addition of incomplete information, mainly because it is a project performed in past.
- **Construct Validity:** The study design was created based on the experience gathered in two previous studies [10]. These previous studies served as a basis to determine the focus and purpose of the retrospective study. Human factors are present in this construction, which can compromise the construction validity. To mitigate this threat, the design and implementation of this study has been performed by four researchers.
- **Conclusion Validity:** the purpose of the study was to analyze data from real software projects using AR in its development process. As in this study only one real project was investigated, it is not possible to generalize the results, which should be treated as initial. Thus, the conclusion validity is linked to the replication of the study in other projects, for which the data and information collected will allow greater representativeness and assertiveness of the results of the study.

7 CONCLUSION

This paper discussed impacts that documentation debt cause to software projects developed by using agile requirements. Through the execution of a retrospective study, we investigated impacts on the project in terms of related maintenance effort and cost. The study was performed using data from a real software project and considered 132 maintenance and evolution tasks for analysis. Results indicated that documentation debt can cause significant impacts, in terms of maintenance effort and cost, on software projects that use agile requirements in their development.

The results of this study contribute to the development of Technical Debt Landscape [2][6][7][9] through the identification of consequences that documentation debt can cause to software projects that use AR in their development. As future work, we intend to perform further studies with other projects in order to have a more comprehensive view of how agile requirements and documentation debt are related to each other. We are also interested on to investigate if the extra effort/time originated from documentation debt can be avoided through the use of less costly requirement engineering techniques. Finally, we also intend to perform a follow-up with the development team to discover their thoughts on findings of this work.

8 ACKNOWLEDGMENT

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

9 REFERENCES

[1] Alves, N.S.R., Ribeiro, L.F., Caires, V., Mendes, T.S., and Spínola, R.O, Towards an Ontology of Terms on Technical Debt. in *Sixth International Workshop on Managing Technical Debt*, (Victoria, BC, 2014), IEEE, 1-7.

[2] Alves, N.S.R., Mendes, T.S., Mendonça, M.G., Spínola, R.O., Shull, F., Seaman, C, Identification and Management of Technical Debt: A Systematic Mapping Study. *Information and Software Technology*, 70, 100 – 121, 2016.

[3] Basili, V.R., Caldiera, C. Rombach, H.D. Goal Question Metric Paradigm. *Encyclopedia of Software Engineering*, 1, 528-532, 1994.

[4] Fernandez, D.M., Wagner, S., Kalinowski, M., Schekelmann, A., Tuzcu, A., Conte, T., Spinola, R.O., Prikladnicki, R., Naming the Pain in Requirements Engineering: Comparing Practices in Brazil and Germany. *IEEE Software*, 32(5), 16-23, 2015, doi:10.1109/MS.2015.122.

[5] Guo, Y., Spínola, R.O., and Seaman, C. Exploring the costs of technical debt management – a case study. *Empirical Software Engineering Journal, 1*, 1 – 24, 2014.

[6] Izurieta, C., Vetro, A., Zazworka, N., Cai, Y., Seaman, C. and Shull, F, Organizing the technical debt landscape. in *Third International Workshop on Managing Technical Debt*, (Zurich, Switzerland, 2012), 23-26.

[7] Kruchten, P., Nord, R. L., Ozkaya, I. Technical Debt: From Metaphor to Theory and Practice. *IEEE Software, 29(06)*, 18-21, 2012.

[8] Misirli, A.T., Caglayan B., Bener A., and Turhan B. A Retrospective Study of Software Analytics Projects: In-Depth Interviews with Practitioners, *IEEE Software, Special Issue on Software Analytics: The Many Faces of Software Analytics*, 54-61, 2013.

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

[10] Soares, H.F., Alves, N.S.R., Mendes, T.S., Mendonça, M.G., and Spínola, R.O, Investigating the Link between User Stories and Documentation Debt on Software Projects. in *12th International Conference on Information Technology : New Generations*, (Las Vegas, USA, 2015), IEEE, 385 - 390.

[11] Spínola, R.O., Zazworka, N., Vetró, A., Seaman, C., and Shull, F, Investigating technical debt folklore: Shedding some light on technical debt opinion. in *4th International Workshop on Managing Technical Debt*, (San Francisco, USA, 2013), 1-7.

[12] Versionone, 2013. Retrieved July 15, 2015, from Versionone: http://www.versionone.com/state-of-agile-survey-results.