Managing Technical Debt in Software Projects Using Scrum: An Action Research

Frederico Oliveira
Institute of Research Technology
São Paulo, Brazil
frederico.fredy84@gmail.com

Alfredo Goldman
Institute of Mathematics and Statistics,
University of São Paulo
São Paulo, Brazil
gold@ime.usp.br

Viviane Santos
Federal University of Pará
Belém, Brazil
vsantos@ufpa.br

***Abstract*— Ward Cunningham in his experience report presented at the OOPSLA'92 conference introduced the metaphor of technical debt. This metaphor is related to immature, incomplete or inadequate artifacts in the software development cycle that cause higher costs and lower quality. A strategy for the technical debt management is still a challenge because its definition is not yet part of the software development process. Carolyn Seaman and Yuepu Guo proposed a technical debt management framework based on three stages. First, debts are identified and listed. After that, debts are measured by their payment efforts and then debts are selected to be considered in the software development cycle. This study evaluates the application of this framework in the real context of software projects adopting Scrum. Action research is conducted in two companies where their projects have significant technical debt. We performed three action research cycles based on the three stages of the framework for both companies. The main contribution of this paper is to provide real experiences and improvements for projects using Scrum and that may adopt the technical debt management framework proposed by Seaman and Guo. Both teams recognized that the proposed approach is feasible for being considered in the software development process after some modifications. Because of projects time constraints and ease of use, we reduced the use of the proposed metrics to two: `Principal` and the `Current Amount of Interest`. In consequence, decision-making was benefitted by the early consideration of the debts that really need to be paid. Instead of using probabilities to find the `interest`, these are registered every time the technical debt occurs. During the first phase, the debts identification was improved when all Scrum roles participated, while measurement and decision-making were improved when the team was responsible for these phases. The Product Owner role in both companies understood the importance of *Technical Debt* monitoring and prioritization during a development cycle. With these changes, the two teams mentioned they would remain using the resulting approach.**

Keywords- Technical Debt; Scrum; Technical Debt Management

I. INTRODUCTION

In 1992, Cunningham introduced the concept coined as "technical debt" (*TD*) [1]. This concept describes the consequences that software projects face when they make trade-offs to implement a lower quality, less complete solutions in order to meet budget and schedule constraints imposed by business realities [2].

Many agile teams seem to believe that they are completely immune to *TD*. Although iterations offer the opportunity to reimburse debt in a timely fashion, the opposite often occurs. Developing and delivering very rapidly, with no time for proper design or to reflect on the long term and a lack of rigor or systematic testing (including automated testing), lead some agile projects into massive amounts of debt very quickly [3].

Technical debt is inevitable. The issue is not on eliminating debt, but rather managing it [4]. Managing *TD* involves tracking it, making reasoned decisions about it, and preventing its worst effects [5]. It is difficult to define a strategy to reduce technical debt. In the context of projects using Scrum, this difficulty in managing the technical debt is justified by three factors. First, it is not clear who is responsible for the reduction of technical debt: the Team, the Product Owner (*PO*), or the Scrum Master? Second, the *PO* often does not understand the need and the benefits of reducing *TD*. As a consequence, the *PO* often does not consider or allow technical projects/stories in their backlog and release plan. Third, problems and goals regarding *TD* are neither structured nor documented [6].

Regarding *TD* visualization on a technical level, Rubin [9] reports three approaches: 1. *TD* visualization in defect tracking software; 2. *TD* visualization in the Product Backlog; 3. *TD* visualization in its own backlog.

Carolyn Seaman and Yuepu Guo [7] proposed a *TD* management framework that consists of three stages: (1) *TD* identification to build a list of debts; (2) *TD* measurement through its payment efforts; and (3) *TD* monitoring to support decision-making of whether and when to deal with them.

This study aims at evaluating the application of the mentioned *TD* management framework [7] through an action research in the real context of software projects using Scrum.

The action research was conducted in two Brazilian companies that present *TD* evidence in their projects. Their names are in pseudonyms. **SoftOne** has a benefits management software project. The software was created in 2002 and has constant product development due to the specific requests of each client, which can be a city or a state in the country (Brazil). **SoftTwo** provides a mission-critical solution to insurers in general.

This paper has scientific and practical relevance because it provides real experiences and improvements to the framework from the perspective of real projects. In the framework, the

978-1-4673-7153-7/15 $31.00 © 2015 IEEE
DOI 10.1109/Agile.2015.7

debt measurement is based on three probabilistic metrics, in which we reduced their use to two: `Principal` (estimated effort to pay the *TD*) and the `Current Amount of Interest` (the actual amount of `interest`, that is the extra effort that will be needed in the future if the *TD* is not paid off at the moment of its identification). Instead of using probabilities to find the `interest`, these are registered every time the *TD* occurs. This paper also presents *TD* visualization approaches chosen by each project following Rubin's [9] approaches. **SoftOne** chose a junction of the first and the third approaches, while **SoftTwo** chose the second approach. In addition, the responsibilities of Scrum roles are analyzed with the activities of the framework. For debt identification, all the roles were relevant. While for measurement and decision-making, the main Scrum role was the team.

The remainder of this paper is organized as follows. Section 2 presents related work and the section 3 presents theoretical background. Section 4 explains the research approach. Section 5 presents the action research stages: diagnosing, action planning, action taking, evaluation, and learning, while Section 6 specifies the research findings of the action research. Section 7 concludes the paper.

II. RELATED WORK

After searching for studies regarding *TD* management with Agile Methods in research databases, such IEEE, ACM and Springer websites, we have found few studies in this area. Some studies are specifically related to the *TD* identification [8], measurement ([15], [16]) and monitoring [17] to support decision-making of whether and when to deal with them, but they do not address the *TD* management process as a whole. We have found Seaman and Guo's framework [7] as the unique integrated way of *TD* management found so far. Following we describe two studies we considered relevant.

Zazworka et al. [8] asked a development team to identify technical debts items in artifacts from a software project on which they were working. The participants assess a technical debt template [7]. Another type of debt was found: Debt Usability, that was introduced to describe the lack of a common user interface template. The study reported that it took an average of **19 minutes per item** to identify and document the technical debts. Subjects agreed that the fields `principal`, `interest amount`, and `interest probability` were the most difficult to fill in. We found only this study directly related to the mentioned framework.

Santos and colleagues [10] reported the experience of an architecture team with 25 agile teams in supporting technical decisions regarding technical practices. They proposed the use of a "technical debt board" with main technical debt categories to manage and visualize the high-level debt.

III. THEORETICAL BACKGROUND

In this section the approaches are discussed to provide the theoretical basis for the study. First, the *TD* management framework [7] is described. After, we describe three ways of making technical debt visible at the technical level.

3.1. Technical Debt Management Framework

3.1.1. Technical Debt Identification and Measurement

Seaman and Guo provide a *TD* template [7]. Each item of the list represents a task that was left undone, but that runs a risk of causing future problems if not completed. Each item includes some attributes, as shown in Table 1.

TABLE I. THE TECHNICAL DEBT TEMPLATE

ID	Technical debt identification number
Date	Technical debt identification date
Responsible	Person who identified the technical debt.
Type	**Testing** (missing test cases, not executed test cases, or missing test plans), **Defect** (known latent defects that have not been fixed), **Documentation** (missing, outdated, or incomplete documentation), **Design** (an imperfection of the software's design or architecture negatively affecting future maintenance) [8].
Location	Description of where the debt item is.
Description	Justification of why that item needs to be considered.
`Estimated Principal`	Work required to pay off the *TD* item.
`Estimated Interest Amount`	Extra effort needed in the future if the *TD* item is not paid off at the moment of its identification.
`Estimated Interest Probability`	Probability of extra work needed, if the *TD* item is not paid off in the future.

Initially, when a *TD* item is created, the three metrics (`Principal`, `Interest Probability` and `Interest Amount`, described in Table 1) are assigned values of **high**, **medium** or **low**. Historical effort data are used to achieve a more accurate estimation beyond the initial assessment. But even if the historical data is limited, an expert estimate is useful in this context [7].

3.1.2. Monitoring Technical Debt

There are two scenarios in which a *TD* list can be used to help management decide on various courses of action. The first is part of the release planning, where a decision must be made as to whether, how much, and which *TD* items should be paid during the upcoming release. The second is ongoing *TD* monitoring over time, independent of the release cycle [7].

Regarding the *TD* selected during the Sprint Planning, assume that a significant work is planned for component (e.g. component called *X*)**,** in the next release. Seaman and Guo [7] list five steps below:

1. Extract all debts associated with component *X*.
2. Re-evaluate high/medium/low estimates for these items based on current plans for the upcoming release.
3. Perform numeric estimates for all items with **high** `Interest Probability` and **high** `Interest Amount`.
4. For each item considered in step 3, compare Cost (**`Principal`**) with Benefit (**`Interest Probability * Interest Amount`**) and eliminate any item for which the benefit does not outweigh the cost.
5. Add up the estimated `Principal` for all items left after step 4. Decide if this cost can be reasonably absorbed into the next release. If not, use this analysis to justify the cost to management. If so, can be more debt

repayment be put into this release? If so, repeat steps 3-5 with items with **high** `Interest Probability` and **medium** `Interest Amount`, and vice versa, then with **medium** for both probability and interest, etc., until no more debt repayment can be absorbed by the release.

Regarding continuous *TD* monitoring, Seaman and Guo [7] suggest to plot various aggregated measures over time and observe measures trends.

3.2. Technical Debt Visibility

Rubin [9] describes three ways of making *TD* visible at the technical level. Figure 1 illustrates these ways.

First, *TD* could be logged like defects into an existing defect-tracking system. Another approach to making technical debt visible is to create product backlog items that represent technical debt. A third approach is to create a special *TD* backlog that makes individual technical debt items visible.

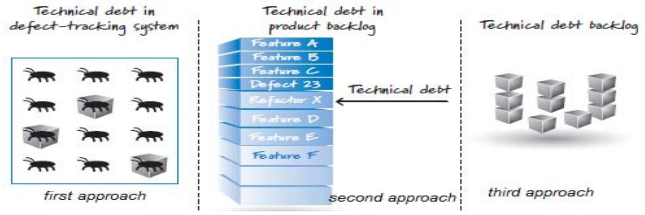

Figure 1. Ways to make technical debt visible [9].

IV. RESEARCH METHODOLOGY

In this study we used action research as our methodological approach. This section justifies the choice of action research, the case selection, data sources and aspects of data analysis.

4.1. Justification of action research

Action research was an appropriate research methodology for this investigation for some reasons. First, the study matches a combination of scientific and practical objectives. This approach merges theory and practice by solving real world problems through theoretically informed actions in collaboration between researchers and practitioners [12]. Additionally, little research has yet been carried out about the mentioned framework [7], which suggests that theory is in an incipient state. That is why a flexible research approach, such as action research, would be appropriate [13] [14].

4.2. Case selection

The selection of projects in **SoftOne** and **SoftTwo** followed the guidelines: 1. Ongoing projects with frequent change requests; 2. Projects using Scrum on project management; 3. Projects with evidence of *TD* not managed.

4.2.1. Pension Benefits Management Software

The project selected in the first company (**SoftOne**) is the development and maintenance of a social security benefits software in the public sector. The software is a web application that consists of modules like registration (civil servants, dependents, current account, among others), granting of social security benefits, such as retirement and pension, payroll processing, electronic document management with digital certification, retirees and pensioners self-service and integration with legacy systems.

Nowadays the software is in operation for customers all over Brazil (states and cities). For this study, a related project to a specific state was chosen, because it is in constant software evolution and maintenance. In addition, the first version in operation was released in 2008, and hence there is a considerable number of documents and versioned code. The project started adopting Scrum also in 2008. The team prefers sprints such as one week. Every sprint begins with the Sprint Planning Meeting (every Thursday). Daily Scrum Meeting also occurs in project. Sprint concludes with the Sprint Review Meeting, in which the team presents its work to the *PO*. The Sprint Retrospective Meeting occurs every three months or when someone wants to change something.

4.2.1. Vehicle Quotations Management Software

The project selected in the second company (**SoftTwo**) consists of the development and maintenance of a Vehicle Quotations Management Software for a major insurance company in the country. The software offers searches to customers and vehicles, selection of guarantees and benefits, and calculation of the insurance considering all these features. The project officially started in 2013 and from the beginning already using Scrum. Scrum is similar to the SoftOne except to the frequency of events.

4.3. Action research cycles

We adapted the five-stage cyclical process model proposed by Susman and Evered [11]: diagnosing, action planning, action taking, evaluating, and specified learning. We performed one diagnosing phase for both companies followed by three action research cycles containing action planning, action taking, evaluation and specified learning phases. Finally, we analyzed the results of each project and proposed the research results in section 6, Research Findings.

The three cycles were completed according to the three stages of the framework for both projects. At the beginning of each cycle, we conducted seminars as seen in Table 2.

TABLE II. ACTION RESEARCH SEMINARS

Event	Content
First seminar (1-2 hours Dec/2014)	The metaphor of technical debt [1] was presented to the research participants, technical and non-technical examples to facilitate understanding of the concept, the identification of the debt through the Seaman and Guo's framework [7] and finally the three ways of making *TD* visible at the technical level based in Rubin [9]. All research participants were invited to identify the debts for 4 weeks and register them.
Second seminar (1-2 hours Jan/2015)	We informed the research participants on the second stage of Seaman and Guo's framework [7], which is responsible for measuring the *TD* identified previously. Participants should measure the debt previously identified. In this seminar we also present the activities carried out in the last seminar.
Third seminar (1-2 hours Feb/2015)	This seminar presented the 5 steps of the third stage (Seaman and Guo's framework [7]) that verifies whether the debt is paid or not in a sprint. We also presented a graphical example provided by Seaman and Guo [7] for *TD* monitoring over time. Research participants should make the decision to pay or not the *TD* previously measured. Finally, we present the activities carried out in the last seminar.

Event	Content
Fourth seminar (2 hours Mar/2015)	The seminar was to discuss the final results and propose improvements in Seaman and Guo's framework. We also present the activities carried out in the last seminar.

The seminars were important to inform the theoretical basis, to discuss issues and to make decisions jointly between the researchers and research participants.

The first cycle of action research was conducted to identify technical debt and choose the method for its visualization by the project team. The second cycle was related to the technical debt measurement. Finally, the third cycle of action research was in charge of the technical debt monitoring.

At the end of the last seminar, action research is completed for both companies after four months working on each project (November/2014 to March/2015).

4.4. Data sources

We used two types of data sources as the empirical basis for our investigation: meeting notes obtained from the seminars and questionnaires. At the end of the cycles, we sent a questionnaire to the participants by e-mail, so they could answer questions individually.

Tables 3 and 4 summarize the individuals who participated in the action research, their roles in Scrum and responsibilities in their projects. Individuals were chosen and allocated by the project managers of the **SoftOne** and **SoftTwo** for this action research.

TABLE III. ACTION RESEARCH PARTICIPANTS - **SOFTONE**

Number of participants	Responsibility(ies) in the project:	Scrum role(s):
1	a) Scrum Master; b) Technical Leader; c) Java Developer	a) Scrum Master; b) Development Team
1	Product Owner	Product Owner
7	Java Developer	Development Team
3	PL/SQL Developer	Development Team
1	Web Designer	Development Team

TABLE IV. ACTION RESEARCH PARTICIPANTS - **SOFTTWO**

Number of participants	Responsibility(ies) in the project:	Scrum role(s):
1	a) Scrum Master; b) Technical Leader	a) Scrum Master; b) Development Team
1	Product Owner	Product Owner
1	Architect Java	Development Team

The Scrum Master role in **SoftOne** is conducted by the first co-author of this study. He is working for six years in the project and has a total of eight years working with the *Pension Benefits Management Software*. Furthermore, the first co-author has the functions of Scrum Master, Development Leader and Java Developer, as shown in the first row of Table 3. As the first co-author has a leadership position in the project where the research was conducted, this may result in research bias, as some suggestions were given and accepted by the research participants of **SoftOne**.

4.5. Data analysis

Firstly, we made individual analysis of each questionnaire answer and seminar notes. Then, we analyzed statements on issues of interest to the research or on providing subsidies for a new issue. At this stage we observed whether certain ideas appear recurrently and whether there were contradictions.

Secondly, we compared the statements contained in the various documents to group them in "codes". The codes were organized into categories according to the ideas contained in it. The same code could be applied to more than one category, whether contains more than one idea. While comparing the responses of the research participants for each category, recurring ideas or new categories that allowed further analysis of agreement and disagreement were identified. We also enumerated the amount of times a word or ideas were cited. Another analysis was performed to identify favorable positions, neutral or opposed to a particular activity. We carried out this analysis by searching words, such as "like", "indifferent", "found it annoying", etc.

V. ACTION RESEARCH

This section explains the three action research cycles and the phases mentioned in subsection 4.3.

5.1. Diagnosing

5.1.1. SoftOne

The *Pension Benefits Management Software* was conceived in 2002. Since then, many analysts and programmers worked on it. Many of these individuals have not had the concern, for example, to keep the original architecture of the software or create tests that enable the safe maintenance of the software all over these years.

This scenario became even worse when, by the end of 2014, the project team doubled in size (15 to 30 people). The quality of the software for the customer became a key issue. So the project manager along with the team began to worry about the technical debts, in particular how to reconcile the ordinary functionalities' development from the Product Backlog with the debts found in the project.

5.1.2. SoftTwo

In November 2014, the first co-author of this study participated in meetings with a director of the company and he was concerned about the rapid and disorganized inclusion of features in the software as a result of pressure from their customers. At these meetings we presented the concept of technical debt and soon after we began meetings with part of the team. The problem was clearly mentioned by the director. He said: *"Consider managing to clean up code in parallel with the features to be developed"*.

It was diagnosed that the debts were already being listed by the team, but they were not aware that these activities were related to technical debt.

5.1.3. On both companies

For both projects, the main problem of managing *TD* may be divided into the following parts: debt identification and visualization, measurement effort for payment and decision making of when and which items should be payed off.

5.2. First Cycle: Identify technical debts and choose a method for visualizing it by the project team

5.2.1. Action Planning and Action Taking

For both companies we were asked to use the tools available in the project for the technical debt management.

In **SoftOne** the project team uses the Trello tool for online visual management of the Kanban board. The Kanban board for *Pension Benefits Management Software* contains the Product Backlog; the Sprint Backlog; ongoing activities; activities to be tested; completed, tested and approved (with the customer) activities; completed activities that are in operation; and activities that have some impediments to its implementation. Besides Trello tool, the team uses Vtiger tool to control demands, such as bug fixing or implementation of new features. Research participants selected the first and the third approaches, as illustrated in Figure 1, to view the technical debt in the project. In the first approach, the team used the Vtiger as their debt registration tool. In the third approach, they used the Trello as a support tool to create the Technical Debt Backlog. A detailed description of each debt registered in the Technical Debt Backlog is also registered on an item classified as technical debt in Vtiger.

The **SoftTwo** project team uses Jira as a tool for managing activities and tasks, as well as for monitoring and reporting defects (bugs). Some debts were preliminarily registered in Jira, but they were not being classified as such. The research participants implemented a new type of classification in Jira for *TD*. Thus, they adopted the second approach (Figure 1).

The seminar introduced the discussion about what Scrum role is responsible for debt identification. One of the **SoftOne** participants mentioned the following statement about the Scrum role for debt identification:

> *"The role would be the (Scrum) team. Everyone involved directly or indirectly in the project could find technical debt. Example: perhaps a business analyst could identify a fault in any process that possibly the developer would not find."*

All research participants (**SoftOne** and **SoftTwo**) were invited to identify the debts for 4 weeks and register them in the chosen visualization form. For each debt found they were required to complete the first 6 rows of Table 1. During the identification phase, in addition to the existing technical debts' type according to the framework, new types could be created by the research participants. Finally, they registered the time spent in completing the first 6 rows for each debt found.

5.2.2. Evaluating and Specified Learning

5.2.2.1. SoftOne

After 4 weeks, among the **13 research participants**, **only 4 did not identify at least one technical debt**. A total of **46 technical debts were identified** and placed in the form of visualization discussed in the Action Planning. Figure 2 illustrates a debt filled up in the chosen visualization form.

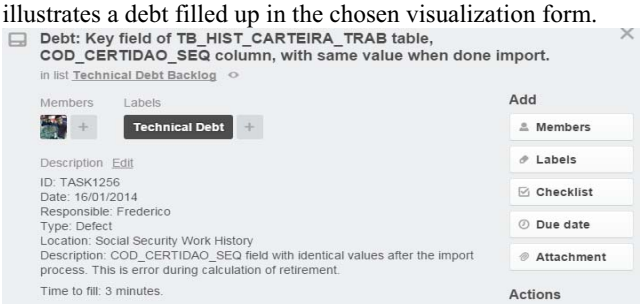

Figure 2. A debt filled up in Trello tool (first 6 rows of Table 1).

The research participants did not identify similar debts more than once, thus justifying the importance of the fact that all roles of the team have worked in the identification. Figure 3 shows the debt percentage obtained by each Scrum role.

The results depicted in Figure 3 are consistent with the responses obtained from the question "*Who would be responsible for identifying the Technical Debt in your project in an agile management environment using Scrum?*", which was in the questionnaire answered at the end. The percentage of the responses is shown in Figure 4.

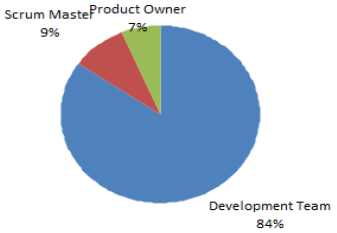

Figure 3. Debt found on the project by Scrum roles.

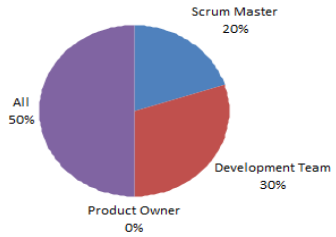

Figure 4. Scrum role responsible for identifying *TD* in the project.

New types of TD were created: **Usability**, **Performance** and **Infrastructure**. Table 5 shows an example of each of these. Figure 5 shows the percentage by type. There were no debts on documentation and testing, because the people responsible for these activities were not participating in this research.

TABLE V. EXAMPLES OF NEW DEBT TYPES IN THE PROJECT

Usability Debt	Compatibility security module with other browsers (Chrome).
Performance Debt	Make memory profile test in Weblogic to check for memory bottleneck in the retirement calculation.
Infrastructure Debt	Deploy Jenkins tool to control the build management continuous integration.

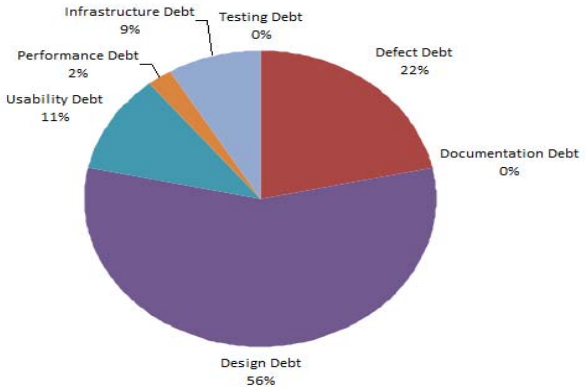

Figure 5. Technical Debt found in the project by type

Research participants put the debts identified in the chosen visualization and also completed the first 6 rows of Table 1 for each of them. The average time spent was **3 minutes**.

One of the research participants in SoftOne suggested including 2 items: "***Impact***" and "***Possible Solution***". The first item was not accepted by the other participants, since we could put this information in the *Description* field. The "*Possible Solution*" field was also accepted but not required, because sometimes the *TD* solution it is not known for sure.

5.2.2.2. SoftTwo

In the beginning of the research, the list of *TD* in **SoftTwo** had a total of **16 items listed in Jira tool**, but there was no evidence of who identified the debts. The evidence of the identified and registered debts is illustrated in Figure 6.

Figure 6. Technical debts in the Jira tool.

The same question previously made in **SoftOne** to identify who is responsible for identifying the debt on the project, was also made in **SoftTwo**. The result is shown in Figure 7.

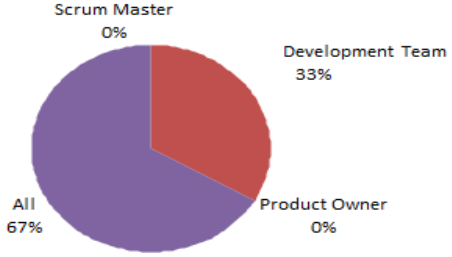

Figure 7. Scrum role responsible for identifying *TD* in the project.

New types of debt were not created because all 16 items were classified in an existing type of the framework. The technical debts found were divided as illustrated in Figure 8.

In **SoftTwo**, research participants did not measure the time taken to fill up the first 6 rows of Table 1 for each technical debt found. The measurement was not employed because they did not remember to do this activity in the identification stage.

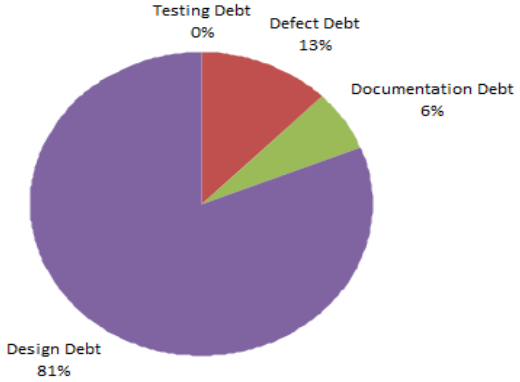

Figure 8. Technical Debt found in the project by type

Research participants emphasized the importance of filling up the first 6 rows of Table 1 for each identified debt. All agreed that the TD were only registered in the tool (before of the research), but had no important information like type, location and description of the debt.

5.3. Second Cycle: Technical Debt measurement

5.3.1. Action Planning and Action Taking

The participants of both companies chose a set of debts from the previous stage for providing their measurement. In **SoftOne**, participants decided that **25%** of the identified debts should be measured. They set a rule that debts would be chosen according to the importance in their project. The estimates were established through **pairs**: the person who identified the debt would estimate with another person who holds knowledge about the debt issue.

In **SoftTwo**, participants chose a random number of *TD* to estimate. For each debt, one person was responsible for its measurement.

According to Seaman and Guo's framework, the three metrics (`Principal`, `Interest Probability` and `Interest Amount`) are assigned values of "**high**", "**medium**", or "**low**". This scale of values is subjective and the research participants of both companies have defined a rationale based on the characteristics of their projects, as seen in the Table 6 for **SoftOne** and Table 7 for **SoftTwo**.

As the **SoftOne** sprint is 40 hours, participants decided that the team could not spend more than 8 hours paying off *TD*. In **SoftTwo**, they defined complexity points and efforts.

During two weeks, the selected debts were estimated and the values were registered as seen in the last 3 rows of Table 1 in the form of visualization of each project. Finally, the research participants recorded the time spent in completing the last 3 rows of the Table 1 for each estimated debt. In the final questionnaire we requested them to identify which fields from Table 1 the participants found more difficulty to fill up.

TABLE VI. VALUES SCALE - **SOFTONE**

	High	Medium	Low
`Principal`	Effort to address debt greater than 8 hours.	Effort to address debt between 4 and 8 hours.	Effort to address the debt below 4 hours.
`Interest Probability`	Debt has happened more than one time. The probability is 80%.	Debt has happened at least one time. The probability is 50%.	Debt has not occurred yet. The probability is 20%.
`Interest Amount`	Extra effort greater than 8 hours.	Extra effort between 4 and 8 hours.	Extra effort less than 4 hours.

TABLE VII. VALUES SCALE - **SOFTTWO**

Complexity Points	Effort (hours)
1	8
2	16
3	24
4	32
5	40
6	48
7	56
8	64
9	72
10	80

The seminar introduced the discussion about what Scrum role is responsible for debt measurement. The Scrum roles that participants chose for debt measurement are discussed further.

5.3.2. Evaluating and Specified Learning

5.3.2.1. SoftOne

Among the 46 debts found, 25% of these were estimated. In other words, **12 technical debts were selected by importance by the research participants**.

The measurement was performed with respect to the `Principal`, `Probability` and `Estimated Interest Amount`. The average time to fill up the metrics was **10 minutes**. Figure 9 illustrates the debt metrics related to Figure 2, which they were filled up according to Table 6.

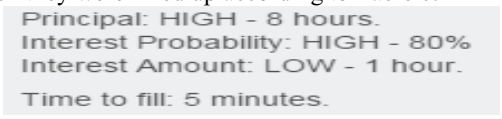

Figure 9. A debt filled up in Trello tool (last 3 rows of Table 1).

Initially, the average time for completing the first 6 lines was **3 minutes** while the average time for completing the last three rows of Table 1 was **10 minutes**. This result is consistent with the responses obtained from the question "*What level of difficulty (LOW or HIGH) did you find to fill up the fields in Table 1 for the debts selected?*", which was in the final questionnaire. The percentage is illustrated in Figure 10 for each field with high difficulty.

The measurement was obtained through pairs. The *PO* noticed difficulties in measuring debts. With the consensus of the research participants, this role did not participate in this task. This is consistent with the responses obtained from the question "*Who would be responsible for TD measurement in your project in an agile management environment using Scrum?*", which was in the final questionnaire. The summary of the responses is shown in Figure 11.

Most of the comments made by research participants are related to the process of abstraction to estimate the values of the measures. One of the participants mentioned the following statement:

> *"I believe that the greatest difficulty has been to stipulate the estimated interest amount. It's hard to abstract what would be the extra effort especially for cases that have not yet occurred."*

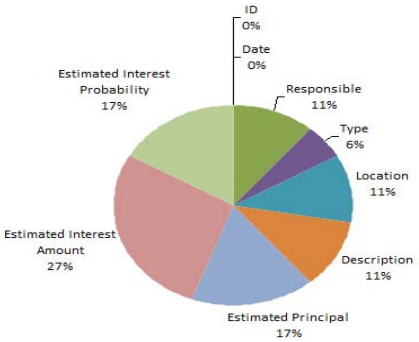

Figure 10. Level of difficulty in filling up the fields.

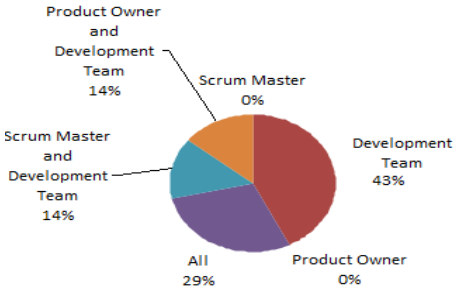

Figure 11. Scrum role responsible for *TD* measurement in the project.

Among the **12 debts measured**, only **one** was considered a high `Interest Probability` and high `Estimated Amount`. This is exactly the infrastructure debt presented in Table V, thus indicating a considerable impact on the project if not resolved.

5.3.2.2. SoftTwo

Among the 16 debts found, 5 of these were estimated by importance. The average for filling up the last 3 lines of Table 1 for 6 debts selected was **8 minutes**. Figure 12 illustrates the 3 fields filled up to the 5 selected debts.

ID	Principal (hours)	Estimated Interest Amout (hours)	Estimated Interest Probability
HDIDC-2422	8	2	0,5
HDIDC-2423	10	5	0,8
HDIDC-2424	10	6	0,8
HDIDC-2425	10	6	0,8
HDIDC-2439	10	10	0,8

Figure 12. Technical debts filled up in a spreadsheet tool (last 3 rows of the Table 1).

According to the answers of the questionnaire applied at the end of the study, fields from Table 1 that the research participants found difficult to fill up are shown in Figure 13. Most research participants in **SoftTwo** also agreed that the

development team should perform the technical debt measurement, as shown in Figure 14.

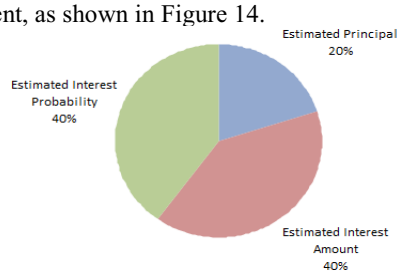

Figure 13. Level of difficulty in filling up the fields.

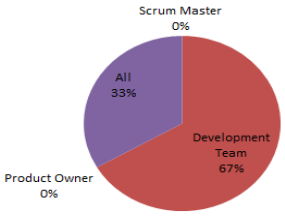

Figure 14. Scrum role responsible for technical debt measurement in the project.

Most of the comments made by research participants are also related to the difficulty in measuring. One of the participants mentioned the following statement:

> *"Let us consider a defect debt in a component. Calculate the estimated interest amount and estimated interest probability is something really difficult because you need to check a large volume history, study the errors already found in this component and thus to measure the probability that debt will occur in the component".*

5.4. Third Cycle: Monitoring Technical Debt

5.4.1. Action Planning and Action Taking

The research participants in **SoftOne** and also in **SoftTwo** decided that all debts previously chosen to be measured would also be used to verify whether the `Benefit` (`Interest Probability` multiplied by `Interest Amount`) exceeds the `Principal`. In **SoftOne** this activity was executed by the **pair** that measured the *TD* item. While in **SoftTwo** this activity was also executed by the person who measured the debt previously. Thus, for each debt measurement, it is known whether it is time to pay or not.

After calculating the `Benefit` and the `Principal`, we agreed with the research participants to adopt the 5 steps suggested by the framework [7] in their next Sprint Planning Meeting. These steps help in selecting the items to be paid.

To monitor *TD* in projects, participants decided to use a chart containing the `weighted total principal` (*TP*) and the `weighted total interest` (*TI*) for 4 weeks. *TP* is calculated by summing up over the entire list (set 3 points for high, 2 for medium, 1 for low) and *TI* (add points for probability and amount) [7].

5.4.2. Evaluating and Specified Learning

All debts selected in **SoftOne** and **SoftTwo** were verified on whether the `Interest` exceeded the `Principal`. In **SoftOne**, there was **one** technical debt that the Benefit already exceeded the Principal, which can be seen in Figure 15.

```
Benefit = (Interest Probability * Interest Amount)
Benefit = (0.8 * 30 minutos)
Benefit = 24 minutes
Principal = 15 minutes
```

Figure 15. Technical debt with `Interest` exceeding the `Principal` (SoftOne).

In **SoftTwo**, from 5 selected and estimated debts, **3** of them had the value of `Interest` above the `Principal` in February. In March there were 4 items as shown in Figure 16.

	Dez		Jan		Fev		Mar	
ID	Principal	Interest	Principal	Interest	Principal	Interest	Principal	Interest
HDIDC-2422	8	1	8	1,5	8	2,5	8	3,5
HDIDC-2423	10	4	10	4,8	10	8,8	10	12,8
HDIDC-2424	10	4,8	10	5,76	10	10,56	10	15,36
HDIDC-2425	10	4,8	10	5,76	10	10,56	10	15,36
HDIDC-2439	10	8	10	9,6	10	17,6	10	25,6

Figure 16. Technical debt which `Interest` exceeded the `Principal` (SoftTwo).

During the Sprint Planning Meeting, the Product Owner notifies the initial goal to the team [9]. So the team selects the number of items in the Product Backlog for the Sprint. Following this criterion, the **team** (Scrum) prioritizes the technical debt to be payed off.

In the planning meeting of each of the projects participating in the research, the Development Team used the 5 steps of Seaman and Guo's framework [7] to prioritize debts for the sprint. As mentioned in the Action Planning, the research participants (**SoftOne** team and *PO*) decided that the team could spend more than 8 hours paying technical debt. *PO* allowed the *TD* and he mentioned the following statement:

> *"We cannot pay all the debt during a sprint. I am aware that we have debt and that we treat. Thus, we allocate an amount of hours to pay the most important debts".*

After the first meeting, both **SoftOne** and **SoftTwo** did not prioritize any item of technical debt. In **SoftOne**, the measured items were not linked to features in the sprint. In **SoftTwo**, although there were items that should be analyzed, the team chose to continue paying `Interest` for lack of time during the next sprint.

SoftTwo monitored their *TD* for 4 weeks. Figure 17 illustrates a debt example where in the third week the `Interest` exceeded the `Principal`. This company chose to plot a chart, such as of the Figure 17, for each measured debt, which was not suggested by the framework.

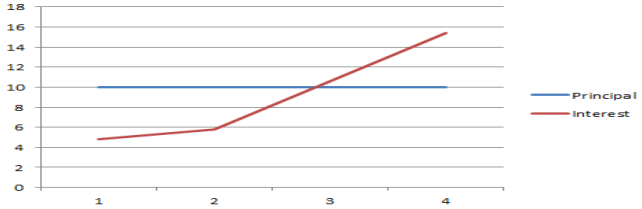

Figure 17. Monitoring a technical debt item (SoftTwo)

The debt monitoring in both projects was carried out by people who measured it previously. This is consistent with the responses obtained from the question "*Who would be responsible for monitoring the Technical Debt in your project*

in an agile management environment using Scrum?", which was in the final questionnaire. The result of the responses is shown in Figure 18 and is similar for both companies.

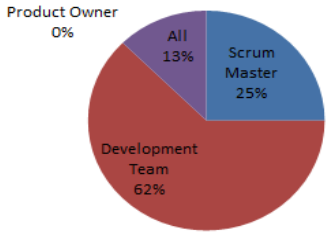

Figure 18. Scrum role responsible for technical debt monitoring.

Research participants of **SoftOne** and **SoftTwo** mentioned that monitoring and updating the measures must be done constantly. However, this is not an easy task. One of the participants mentioned the following statement:

> *"Monitoring is difficult to be manual and require some time. As there are numerous activities in the project, have something that was automated would help a lot."*

VI. DISCUSSION – RESEARCH FINDINGS

In this section, we consolidate the results from both projects and discuss the theoretical and practical implications of them.

6.1. Scrum roles for identification, measurement and decision making

Scrum roles for identification, measurement and decision making related to technical debt were similar for both companies (**SoftOne** and **SoftTwo**).

According to our research participants, all team members should identify technical debt. This is justified by debts encountered by all the roles in both studied projects.

On the other hand, our research participants suggest that the development team should perform debt measurement, since this role is responsible for the technical tasks of the project and thus appropriate to derive the estimates.

The development team should prioritize the technical debt, because it is the role that can better assess what will be completed in relation to the target. This role also carries out debt monitoring during the project. In both companies, the research participants decided that the person responsible for the debt would register the time for debt identification. The person would also be responsible for verifying whether the `Interest` exceeded the `Principal`. So, (s)he would be in charge of making the monitoring chart, as shown in Figure 14.

6.2. Identification of Technical Debt

In Zazworka et al. research [8], the participants used tools for automatic identification of defect debts. However, we observed that it is still necessary to involve humans in the identification process, because these tools cannot help in identifying many other types of debt. Thus, the research participants did not use any software to identify the debts.

Most reported debt types were related to Design and Defects. This is justified because most of the research participants were developers and software architects. New types were created in **SoftOne** because some debts are not classified among those present in the framework. Thus, new types were created for a group of debts in similar cases. This strategy may facilitate the creation of actions to try to avoid future debts of these types with more evidence.

The average time for completing the first 6 rows of Table 1 of debts found was made within a reasonable time from the point of view of research participants (three minutes in **SoftOne**). At **SoftTwo**, they also agreed that identification time would be similar to the time presented by **SoftOne**.

6.3. Technical Debt measurement

The three metrics (`Estimated Principal`, `Interest Estimated Amount`, `Interest Estimated Probability`) were the fields that research participants had more difficulty for filling up, due to be a probabilistic field, especially when there is no historical data. This fact is the same found in Zazworka et al. research [8]. The average time spent to complete the last 3 lines of Table 1 for selected debts was **10 minutes** in **SoftOne**, i.e., three times higher than the average time to fill up the first 6 lines in Table 1. The average time spent in **SoftTwo** was similar (about **8 minutes**).

The total average time to fill up in Table 1 was similar to the found in the mentioned research [8], which was **19 minutes**. In the case of **SoftOne,** it was **13 minutes**. In **SoftOne,** a practice used was the debt measurement employed by pairs. As the measures are statistical probabilities, the discussion in pairs to find out the values for the three metrics was valid and presented genuine estimates. However, the measures did not eliminate the difficulty in finding the values.

Thus, with this difficulty in estimating 3 metrics, we agreed to measure the debt by **only 2 items**: `Principal` and `Current Amount of Interest`. Every debt encountered during the development cycle, the team must register it and estimate its `Principal`. They also register the value of accumulated interest to date. Each time the debt occurs, the `Current Amount of Interest` must be updated.

We asked the research participants of **SoftOne** to identify new debts and fill up them with the first 7 rows of Table 1 and `Current Amount of Interest`. For 2 weeks, **5 debts were identified** and the average time to fill up these fields was **6 minutes**, or 46% of time spent above (6 divided by 13). In **SoftTwo** this new approach has not yet been put into practice.

6.4. Monitoring Technical Debt

As research participants suggested measuring the debt by **only 2 items** (`Principal` and `Current Amount of Interest`), the 5 steps (monitoring *TD*) covered in the framework were arranged for the following:

1. Extract all *TD* items associated with the work to be done.
2. Re-evaluate `Principal` numeric estimates for these items based on current plans for the upcoming release.
3. For each item considered in step 1, compare Cost (`Principal`) with Benefit (`Current Amount of Interest`) and eliminate any item for which the benefit does not outweigh the cost.
4. Decide if this cost can be absorbed into the next release. If not, use this analysis to justify the cost to management.

Both companies plan to adopt these steps in their next sprint. The *TD* monitoring was held by the people responsible for each debt measured. The goal was to alert the team when the `Interest` would exceed the `Principal` in a debt. The difficulty exposed by the research participants was the lack of tools that make the integration between the *TD* measurements and their tracking chart. These management tools that assist in *TD* measuring and monitoring debt are not considered in Seaman and Guo's framework [7].

6.5. Limitations

Even conducting three action research (*AR*) cycles with important practical feedback, this study is still not conclusive. We need to conduct more *AR* cycles to assure validity of the proposed changes to the presented approach within the context of the participating teams.

Some activities in **SoftOne** were not performed in **SoftTwo**, such as the measurement of the average time of completing the first 6 rows of Table 1 for debt identification.

For both projects, project managers selected the roles and tools participating in the research. We did not consider roles, such as testers, infrastructure analysts, and project manager.

The new approach to *TD* prioritization (4 steps seen in subsection 6.4) has not been put into practice in **SoftOne** and **SoftTwo**. This will occur in the next sprints.

The first co-author of this study actively participates in the research, as he is part of the **SoftOne** development team. In this context, the co-author participated in the identification, measurement and prioritization of debt in the project together with the other participants. His leadership position in the project may result in research bias, as some suggestions were given and accepted by the research participants of **SoftOne** but not of **SoftTwo**.

In **SoftTwo**, in addition to the seminars, the doubts were discussed by means of telephone calls and e-mails. It was noted that the planned activities were not carried out constantly during the agreed period. This occurred because the project was with several other ongoing activities. In addition, the researchers were not daily in the project.

VII. CONCLUSIONS

This paper describes a practical evaluation of the Seaman and Guo´s *TD* management framework, through an action research in the real context of software projects using Scrum.

In a Scrum project, most *TD* management activities should be accomplished by the development team, since this role has the technical perspective of the project. Otherwise, in *TD* identification, all can contribute during the project. Each project followed Rubin's [9] approaches for *TD* visualization.

About the framework, the difficulty concerned *TD* measurement. Thus, we used only two metrics that correspond to the actual values (not the probability) of debt in the project. With this major change, the two teams mentioned they would remain using the resulting approach. A barrier that may hinder the use is on the tools. It is important to use tools to ease *TD* measuring and monitoring.

As a further work, *TD* measurement based on two metrics will be used in **SoftTwo**. Besides, the new approach to prioritization of debt (4 steps seen in subsection 6.4) has not been put into practice in **SoftOne** and **SoftTwo**, which will be put into practice in the next sprints.

The Seaman and Guo's framework is an important first step in the *TD* management, but with the changes proposed in the *TD* measurement, the framework tends to greater acceptance by considering actual values.

Finally, the research participants in the Product Owner role understood the importance of technical debts monitoring and prioritize them during a software development cycle.

REFERENCES

[1] Cunningham, W. 1992. The WyCash Portfolio Management System. In Addendum to the proceedings on Object oriented programming systems, languages, and applications. Pp-29-30.

[2] Lim, E. Technical Debt: What Software Practitioners Have to Say. Thesis submitted in partial fulfillment of the requirements for the degree of master of applied science in The Faculty of Graduate Studies. The University of British Columbia. 2012.

[3] Kruchten, P., Nord, R. L. and Ozkaya, I. Technical Debt: From Metaphor to Theory and Practice, IEEE Softw., vol. 29, no. 6, pp. 18–21, 2012.

[4] Allman, E. Managing technica debt.. Magazine Communications of the ACM, vol. 55, no.5, pp.50-55, May. 2012.

[5] Lim, E., Taksande, N. and Seaman, C. "A Balancing Act: What Software Practitioners Have to Say about Technical Debt," IEEE Software, vol. 29, no. 6, pp. 22–27, Nov. 2012.

[6] Buch, B. "Effective Steps to reduce technical debt: An agile approach. 2011. http://www.codovation.com/2012/06/effective-steps-to-reduce-technical-debt-an-agile-approach/

[7] Seaman, C., and Guo, Y. 2011. Measuring and Monitoring Technical Debt. In Advances in Computers, Vol.82,pp.25-46.

[8] Zazworka, N., Spínola, R. O., Vetro', A., Shull, F. and Seaman C. A case study on effectively identifying technical debt. Proceedings of the 17th International Conference on Evaluation and Assessment in Software Engineering. Pp. 42-47.

[9] Rubin, K. S. Essential Scrum. A Practical Guide To The Most Popular Agile Process. Addison-Wesley, 2013. 498p.

[10] Santos, P. S. M, Varella, A., Dantas, C. R., Borges, D. B. Visualizing and Managing Technical Debt in Agile Development: An Experience Report. Agile Processes in Software Engineering and Extreme Programming Lecture Notes in Business Information Processing, v.149, p.121-134, 2013.

[11] Susman, G.I., Evered, R.D. An assessment of the scientific merits of action research, Administrative Science Quarterly 23 (1978) 582–603.

[12] Greenwood , D.J., Levin, M. Introduction to Action Research: Social Research for Social Change, SAGE Publications, 2007.

[13] Edmondson, A.C., McManus, S.E. Methodological fit in management field research, Academy of Management Review 32 (2007) 1155–1179.

[14] Kampenes, V.B., Anda, B., Dybå, T. Flexibility in research design in empirical software engineering, in: Evaluation and Assessment in Software Engineering (EASE 2008), BCS, University of Bari, 2008.

[15] Nugroho, A., Visser, J., Kuipers, T. An empirical model of technical debt and interest. Proceedings of the 2nd Workshop on Managing Technical Debt. Pp-1-8.

[16] Curtis, B., Sappidi, J., Szynkarski, A. Estimating the size, cost, and types of technical debt. Proceedings of the Third International Workshop on Managing Technical Debt. Pp-49-53.

[17] Seaman, C., Guo, Y., Izurieta, C., Cai, Y., Zazworka, N., Shull, F., Vetrò, A. Using technical debt data in decision making: potential decision approaches. Proceedings of the Third International Workshop on Managing Technical Debt. Pp-45-48.