Tracking Technical Debt

— An Exploratory Case Study

Yuepu Guo, Carolyn Seaman
Department of Information Systems
University of Maryland Baltimore County
Baltimore, MD, USA
{yuepu.guo, cseaman}@umbc.edu

Rebeka Gomes, Antonio Cavalcanti, Graziela Tonin, Fabio Q. B. Da Silva, André L. M. Santos, Clauirton Siebra
Center of Informatics
Federal University of Pernambuco
Recife, PE, Brazil
{rgo2, alocj, gst, fabio, alms, cas}@cin.ufpe.br

Abstract—**The technical debt metaphor is increasingly being used to describe the effect of delaying certain software maintenance tasks on software projects. Practitioners understand intuitively how technical debt can turn into a serious problem if it is left unattended. However, it remains unknown how serious the problem is and whether explicit measurement and management of technical debt is useful. In this paper, we explore the effect of technical debt by tracking a single delayed maintenance task in a real software project throughout its lifecycle and simulate how explicit technical debt management might have changed project outcomes. The results from this study demonstrate how and to what extent technical debt affects software projects. The study also sheds light on the research methodologies that can be used to investigate the technical debt management problem.**

Keywords-technical debt; decision making; software maintenance;

I. Introduction

Software maintenance projects are often constrained by budget, schedule and resources. To meet these constraints, trade-offs are made throughout the software development lifecycle. While these trade-offs allow project teams to meet customer and management expectations in the short run, the compromise results in additional cost later in the maintenance process. This phenomenon is known as "technical debt" [1]. As with financial debt, the additional cost is the interest on technical debt, and hence software managers need to consider both the benefit and cost side of incurring technical debt when they make decisions in the course of their projects. However, compared to its financial counterpart, technical debt has the added complication that the technical debt interest may or may not be incurred. The uncertainty of interest payment implies that technical debt can be treated as a particular software risk. Then managing technical debt becomes a task of analyzing the software risk and making informed decisions on what technical debt should be incurred or paid off and when.

The technical debt metaphor is increasingly being used to describe the effect of delaying certain software maintenance tasks on software projects. Practitioners understand intuitively how technical debt can turn into a serious problem if it is left unattended. However, it remains unknown how serious the problem could be. In addition, software managers currently carry out the risk analysis implicitly. Without measurement of technical debt, they have to make decisions based on their experience, which may cause misunderstanding of the impact of technical debt, resulting in unexpected project delays and compromised software quality. Therefore it is necessary to construct a technical debt theory upon which approaches and mechanisms can be developed to facilitate decision making. The technical debt theory will reveal the cost-benefit relationships implicit in the metaphor. In particular, it will address whether explicit measurement and management of technical debt is useful for software project management. To construct the technical debt theory, we have proposed a technical debt management framework and plan to conduct a retrospective study and a case study of ongoing software projects [2]. The retrospective study aims at finding the benefits of explicit technical debt management by reviewing the historical effort, defect, testing, and change data over several recent releases of a single software product. The data will be used retrospectively to simulate decisions about release planning as they would have been made if the proposed technical debt management approach had been used. The case study will use the future releases of the projects to explore the costs of the proposed approach. The two studies together will be able to reveal the cost-benefit relationship of incurring technical debt, thus contributing to the technical debt theory construction. Meanwhile, results from the two studies will yield insight into costs and benefits of explicit technical debt management and help to evaluate the effectiveness of the proposed approach.

Ideally the specially designed two studies can achieve the aforementioned goals, but the implementation will still be affected by many factors such as data availability and data quality. It is not certain that the methodology we propose will be effective in real settings, plus the impact of technical debt is unclear. Therefore we carried out this pilot study to determine (1) how and to what extent technical debt affects software projects; (2) whether the decision simulation is effective to uncover the benefits of explicit technical debt management.

In this study we tracked a single delayed maintenance task in a real software project throughout its lifecycle. First, we investigate the reasons that the task was delayed and the context surrounding the decisions made, both to delay it and later to complete it. We also estimate the effort that would have been needed to fulfill the task when it first arose, the effort required to complete the task when it was actually completed, and the impact of the delay on other tasks that were made more difficult, or that had to be re-done, because of the delay. Then we simulated the decision making using the proposed approach. In other words, we re-create the decision to delay (or not) the maintenance task using the proposed technical debt management approach, to see if a different decision would have been made. Finally we compare the impact of the actual decision with that of the simulated decision to determine the benefits that could have been gained from explicit technical debt measurement and management. The results demonstrate the level of technical debt impact on the software project. The study also sheds light on the research methods that can be used to investigate the technical debt management problem.

II. Background

This study belongs to a large research project that aims at technical debt theory construction through measuring and monitoring technical debt. A management framework has been proposed and was applied in this study to the technical debt item we selected from the subject project.

A. *Proposed technical debt management framework*

The proposed framework is centered on a technical debt list, which contains all the technical debt items that currently remain in the system. A technical debt item represents a maintenance task that has been postponed and may cause a problem in future if not completed at present. For example, a deferred refactoring of a module that does not conform to the system architecture is a technical debt item. Each technical debt item includes the information about what the technical debt is, why and when it was incurred, where it is located and who is responsible. Such information can be obtained through various techniques for technical debt identification, which is outside the scope of this paper. After the technical debt items are identified, the technical debt list can then be constructed. A technical debt item also has the properties on the estimated principal and interest, which capture the effort to complete the task and the extra cost it may incur, respectively. Due to the uncertainty of interest payment, the interest property is broken into two properties – interest amount and interest probability. The principal and interest of a technical debt item are estimated through measurement estimation of various development related entities, principally effort. After the measurement step, technical debt is tracked, quantified and thus ready to be used for decision making.

One scenario in which the technical debt list can be used to facilitate decision making is in release planning, when the project manager needs to decide what technical debt items should be paid and which one should be paid first if clearance of all debt is infeasible. In this scenario the project manager will prioritize the technical debt items according to their cost and benefit and determine the set of technical debt items that should be paid to minimize the cost of the project. Various approaches, such as the portfolio approach [3], have been proposed for technical debt prioritization.

B. *Subject project and technical debt item*

The selected project was a software application from a multi-national company that provides mobile communication products and solutions. We call the application "ABC" in this paper. ABC consists of the development and evolution of a software application for mobile platforms, used as a client solution for Microsoft Exchange Server. The system has 63,218 lines of code and has been translated into 18 languages. It was first released in March 2006 and has over 5 years of evolution. The development involves about 20 software engineers.

Our object of study consists of a sequence of decisions about the communication protocol used to connect the application to MS Exchange Server. Figure 1 shows this decision evolution. The application implementation began in August 2006 and the communication protocol used was WebDAV with support for MS Exchange 2003. Two months later, a member of the development team, an expert in MS Exchange, warned about the upcoming launch of MS Exchange 2007, and that WebDAV would not support it. At that time, a decision had to be made between keeping WebDAV and re-implementing the communication layer using a technology that would support MS Exchange 2007. Given the time-to-market constraints, the decision was to keep WebDAV for the following reasons: the application was already developed for WebDAV, the 2007 version had not yet been released, and MS Exchange 2003 was active on the market, creating a real potential for the immediate use of ABC supporting that version.

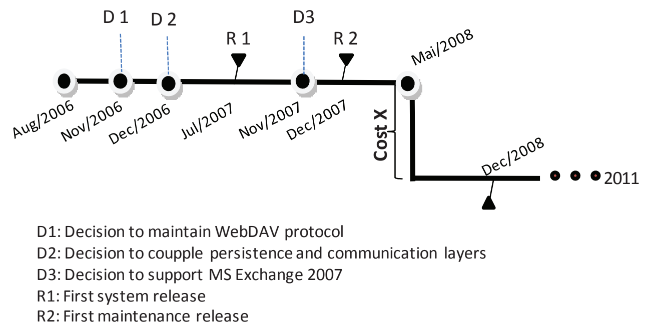
D 1
D 2
R 1
D3
R 2
Mai/2008
Aug/2006
Nov/2006
Dec/2006
Jul/2007
Nov/2007
Dec/2007
Cost X
Dec/2008
2011
D1: Decision to maintain WebDAV protocol
D2: Decision to coupple persistence and communication layers
D3: Decision to support MS Exchange 2007
R1: First system release
R2: First maintenance release

Figure 1. Timeline of the ABC evolution

A month after this decision and after application performance analysis, it was decided to couple the communication and persistence layers of the system's architecture (D2) to improve system performance. At the time of this decision it was known that: (1) the probability of the release of MS Exchange 2007 was high, (2) the protocol used (WebDAV) did not support this new version, (3) coupling the architecture layers would increase performance, and (4) coupling the architecture would imply greater rework to change the protocol in the future. It was decided that the performance considerations were most pressing at the moment, and hence the decision was made to couple the communication and persistence layers of the architecture.

Seven months later, in July 2007, the product was deployed to the first customer (R1). During the second half of 2007, the ABC customers started to migrate their email servers to MS Exchange 2007, breaking compatibility with the implemented version of the ABC's communication protocol. In November 2007, it was decided to change the communication protocol to replace WebDAV. This change required effort to rewrite some application parts (cost X), involving not only changes in the communication layer, but also in the persistence layer due to the coupling of the two layers previously discussed.

It is our contention that if decision D1 had been to change the communication protocol, the cost of supporting MS Exchange would have been smaller even in the context of the decision D2 to couple the two layers of the architecture to achieve better performance. The results we present below support this supposition. Because D1 was made primarily for the purposes of getting the product to market quickly, and because the decision did not immediately impact functionality (i.e. ABC served its customers well until they started migrating to MS Exchange 2007), we believe that the failure to upgrade the WebDAV protocol constitutes a technical debt item (which we denote as T1), as defined earlier. Our analysis below simulates what might have happened in this project if the technical debt had been managed explicitly. The objective of this analysis is to better understand the impact and value of managing and measuring technical debt.

III. Methodology

To track the technical debt item and assess its impact, different types of data were collected from various sources. The impact is measured by the effort saved (at the points where the debt is not paid off) or required to complete the task (at the point where it is paid off) against the loss caused by technical debt. These estimates were then used to simulate the decision making process in order to address the objective of this study.

A. Measurement

Informed by the "debt" metaphor, we use principal, interest amount and interest probability to measure a technical debt item. In this study the principal of T1 (P) is the effort to switch to ActiveSync at the time of D1. Since the actual decision was NOT to switch to ActiveSync, the effort has to be estimated based on the situation at D1 using the project's typical effort estimation approach – expert estimation.

The interest amount of T1 (IA) is the impact that T1 has on the future maintenance of ABC, that is, the extra effort required to switch to ActiveSync because that task was delayed (to after D2). The actual interest amount (IA_{D3}) is estimated using the following equation:

$$IA_{D3} = X - P \tag{1}$$

In addition, to simulate the decision making, the interest amount also needs to be estimated at D1 and D2. The interest amounts at D1 (IA_{D1}) and D2 (IA_{D2}) are measured by estimating the rework effort on the modules affected by the decision to couple the communication and persistence layers, which then later had to be changed again to implement the new communication protocol.

Similar to interest amount estimation, the interest probability of T1 (IP) needs to be estimated at D1 (IP_{D1}) and re-estimated at D2 (IP_{D2}) because the probability may vary in different time frames and the simulation depends on using the information available at each simulated point of time. In this case the interest probability is affected by the probability (PC) that coupling the communication layers and persistence layers is required and the probability (PM) that ABC needs to switch to ActiveSync. Since PM and PC are independent of each other,

$$IP = PC \times PM \tag{2}$$

B. Data collection

In this study we collected data regarding the changes in ABC and the decisions made during the selected period of time. These change data and decision data, with help from project personnel, explain what happened to the project, why and how it happened, as described in the project background section. We collected data about system size before and after each point of the evolution decision timeline. The metric used to measure system size was the number of lines of code and was collected through Unified Code Count [4], a COCOMO compliant tool.

We also collected effort data on the release in which the communication protocol was changed to ActiveSync (cost X). The effort data were collected from the project chronograms of the related release cycle. In addition, effort required for P, IA_{D1}, IA_{D2}, IP_{D1} and IP_{D2} were estimated by the project personnel based on impact analysis of the potential changes that can be foreseen. In this study the effort is measured in staff-hours.

C. Data analysis

TABLE I shows the break-down of cost X, from which IA_{D3} is derived. TABLE II shows the estimates of PC and PM, which are used to calculate IP_{D1} and IP_{D2}.

TABLE I. Effort to Change to ActiveSync (cost X)

Task	Effort (staff-hour)
Persistence	**514**
Communication	**922**
TOTAL	**1436**

TABLE II. Probabilitiy of Changes

Probability of change	Point of Time	
	D1	*D2*
PC	20%	100%
PM	70%	70%

With the data collected from the project documentation and personnel, we constructed a technical debt list, which contains the item T1. In the interests of space, only the principal and interest properties of T1 are presented below. From the time of D1 we started to track T1 by estimating P, IA_{D1} and IP_{D1}. Then we simulated the decision about incurring T1 through cost-benefit analysis. That is, we simulated how the decision would have been made, at the time, if the project personnel had been using the proposed technical debt management approach described in section II.A. The benefit of incurring T1 at D1 is the principal of T1, i.e. P, while the cost is the interest of T1, which is the product of the interest amount and interest probability, IA_{D1} and IP_{D1}, to represent the cost of incurring T1 at D1. Since the benefit of incurring T1 outweighed the cost, the simulated decision would have been to keep WebDev at D1, same as the actual decision. At D2, a decision was made by the project team to couple the communication layer and persistence layer, while still keeping T1 (i.e. not switching to the ActiveSync protocol). Here we re-estimated the interest amount and interest probability as the situation, including the information availability, had changed since D1. Following the same procedure, we simulated the decision at D2. Since the cost of keeping T1 outweighed the benefit, the simulated decision would have been to pay off T1 at D2. However, the actual decision was to not pay off T1 at time D2. Thus, we went to time D3 and used the actual total cost X to determine the impact that the actual decision not to pay off T1 had on the project, which is IA_{D3}. The difference between the net cost of the simulated decision (P - 0) and the actual decision (X) is the benefit that could have been gained through the explicit technical debt management, which is equal to IA_{D3} according to equation (1). Figure 2 illustrates the process of decision simulation.

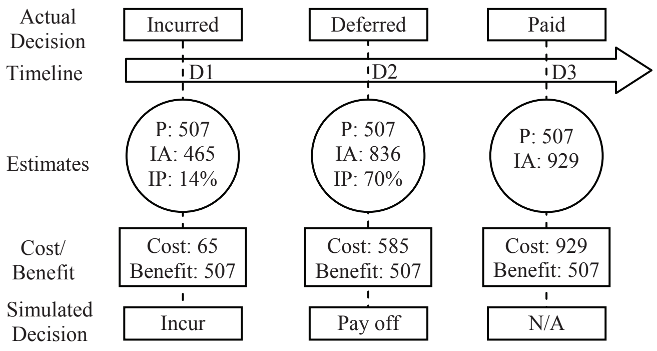

Figure 2. Decision simulation process

IV. Results

The above analysis makes it clear that delaying the upgrade of the communication protocol had significant impact on the cost of the project. By tracking this technical debt item, we can see that it was incurred for the sake of time constraint in the situation that future demand of the support for MS Exchange 2007 and changes imposed on the system were not clear. This is a typical scenario in which technical debt is incurred. The interest amount and interest probability were estimated low at that time. Thus it was not a bad decision to keep WebDev, the old communication protocol, to save time for the project. However, the situation was changing quickly and the project faced a change soon due to the application's unsatisfactory performance. Meanwhile the release of MS Exchange 2007 indicated that changing to a compatible communication protocol would be demanded soon. At that time (D2) both the potential penalty of keeping WebDev and the probability that the penalty would be incurred were high, which were reflected in the high value of the interest amount and interest probability. The situation didn't allow the old communication protocol to be used any longer. Unfortunately the decision was to couple the communication layer and the persistence layer without changing the communication protocol. When it finally became required that Exchange 2007 be supported, high rework cost was inevitable. The net cost of carrying the technical debt was almost two times of the net cost of switching to the new communication protocol before merging the two layers. By contrast, if the decision had been made in the same way as the simulated decision, the extra cost would have been avoided.

V. Conclusion

This is a case of a type of technical debt that often occurs in practice turning into a big problem for the project. It almost tripled the cost of upgrading to ActiveSync, which was ironically the cost that the technical debt was incurred to save (or at least delay). By tracking the technical debt and measuring the impact, this study provides evidence of the effect of technical debt on software projects and concrete numbers about how serious the problem is, at least in this typical case. Furthermore, the study demonstrates how a decision made without careful analysis could aggravate the negative effect of technical debt, while attention and explicit management of technical debt could make a difference. Therefore we can argue that the studied technical debt item would have never caused such high costs if a detailed cost-benefit analysis, as we did in this study, had been presented to the software manager before they made their decisions.

In this study various types of data were collected. The change data and decision data can help construct the context where the decisions were made so that the factors contributing to costs and benefits of technical debt can be identified, while the effort data and size data were used to determine the degrees to which different factors affect the costs and benefits of technical debt. These data are useful for investigating the costs and benefits associated with technical debt, while the decision simulation provides an approach to leverage the historical project information to yield valuable insights into the technical debt problem.

In the studied case, delaying the communication protocol change allowed the application to be released on time. Thus the real benefit of incurring the technical debt lies in the value of time to market, which was not considered in our approach. Instead, the principle of the technical debt was used to represent the benefit. Therefore over-simplicity of the approach would be the main limitation of this work. A lesson learned from this initial analysis is that business factors must be incorporated into our model in order to be effective in addressing real decision problems. Our future research plans involve a more multi-disciplinary team, which will allow us to overcome this limitation.

Acknowledgment

Guo and Seaman's work is sponsored by the US National Science Foundation grant #0916699. The work of CIn authors is sponsored by the data provider for this study and the Brazilian National Council for Scientific and Technological Development (CNPq).

References

[1] W. Cunningham, "The WyCash portfolio management system," in Addendum to the Proceedings on Object Oriented Programming Systems, Languages, and Applications, pp. 29-30, 1992.

[2] C. Seaman and Y. Guo, "Measuring and monitoring technical debt," Advances in Computers, vol. 82, pp. 25-46, 2011.

[3] Y. Guo and C. Seaman, "A portfolio approach to technical debt management," Proceeding of the 2nd Workshop on Managing Technical Debt (MTD'11), pp. 31-34, 2011

[4] A. Hira, "CodeCount", Center for System and Software Engineering, available at http://sunset.usc.edu/research/CODECOUNT/, 2010