A Portfolio Approach to Technical Debt Management

Yuepu Guo
Department of Information Systems
University of Maryland Baltimore County
Baltimore, Maryland 21250, USA
+1-410-455-8842
yuepu.guo@umbc.edu

Carolyn Seaman
Department of Information Systems
University of Maryland Baltimore County
Baltimore, Maryland 21250, USA
+1-410-455-3937
cseaman@umbc.edu

ABSTRACT
Technical debt describes the effect of immature software artifacts on software maintenance – the potential of extra effort required in future as if paying interest for the incurred debt. The uncertainty of interest payment further complicates the problem of what debt should be incurred or repaid and when. To help software managers make informed decisions, a portfolio approach is proposed in this paper. The approach leverages the portfolio management theory in the finance domain to determine the optimal collection of technical debt items that should be incurred or held. We expect this approach could provide a new perspective for technical debt management.

Categories and Subject Descriptors
D.2.7 [**Software Engineering]:** Distribution, Maintenance, and Enhancement

D.2.9 [**Software Engineering**]: Management

General Terms
Management, Measurement, Economics.

Keywords
Technical debt, Software risk, Portfolio, Decision making, Software maintenance.

1. INTRODUCTION
Technical debt is a metaphor originally used to describe "not-quite-right code" in software projects [1], but it has been extended to refer to any immature software artifacts, such as immature design, incomplete documentation and unfinished testing, in the software lifecycle. Technical debt may be incurred for different reasons. For example, low quality code may be written unintentionally by a programmer due to a lack of experience. But technical debt can also be used strategically. By delaying certain maintenance tasks or doing them quickly and less carefully, software managers can trade off software quality with productivity, which may have additional benefits such as that gained from earlier time to market than their competitors. However, as if paying the interest on a debt, such benefit is achieved with higher cost in the future in that the immature artifacts make the system more complex, less understandable, and thus need extra effort to modify. The strategy may become less promising, even counter-productive, if the interest is too high. Therefore, software managers need to balance the benefit of incurring technical debt with the associated costs when planning their projects.

By using the term "debt" borrowed from the finance domain, the metaphor effectively characterizes the relationship between the short-term benefits of delaying certain software maintenance tasks and the long-term costs of those delays. However, technical debt is not exactly the same as financial debt. For financial debt, the cost information, including the amount of interest and other costs of incurring the debt, is usually clear and available for debtors so that they can compare different offers on the market. But for technical debt, this is not always the case. For example, if a certain portion of the system in fact has no defects, then no harm is done in saving some time by not testing it. Similarly, if a particular module is never going to be modified in the future, then failing to update its related documentation will save some time during modification of the module, without any adverse consequences. Software managers should incur technical debt only on those artifacts without interest, or give low priority to paying off the debt on these artifacts while focusing on those that are subject to penalty, if they know what portion of the system has defects and what changes will be requested. Unfortunately such information is rarely known beforehand. Therefore, technical debt involves uncertainty about whether or not it will have an associated penalty, i.e. the interest. In this sense, technical debt can be considered as a particular type of risk in software maintenance and the problem of managing technical debt boils down to managing risk and making informed decisions on what tasks can be delayed and when they need to be paid back. From this perspective, technical debt is a potential loss to software projects. Thus managing technical debt focuses on reducing the negative impact of the debt. However, as mentioned earlier, technical debt also has its benefit side, which allows it to be used as a strategy. Furthermore, technical debt is not problematic as long as the benefit outweighs its cost. This view has inspired us to look at the incurring of technical debt as an investment.

From the perspective of investment, technical debt items are the assets that the project can invest to get profit. These assets have different returns and different distributions and ranges (i.e. variances) of return, which correspond to the net benefits of the technical debt items and their associated risks. The goals of managing technical debt include maximizing return with a given risk level or minimizing the risk with a given level of return, by finding the optimal combination of these assets. After mapping these technical debt concepts to the financial investment domain, we propose to leverage the portfolio approach to managing technical debt. Portfolio management has been widely used in investment management to reduce risk and increase return on investment. Given the similarity between investment and incurring technical debt, we expect the application of the portfolio

MTD'11, May 23, 2011, Waikiki, Honolulu, HI, USA

approach to provide a new perspective for technical debt management. In the following sections, we first present an overview of portfolio management in the finance domain, including the principle, assumptions and models. Then we give the measurements, primary steps and proposed evaluation method for the approach to technical debt management in section 3. Finally we discuss the applicability of this approach and some managerial implications.

2. PORTFOLIO MANAGEMENT

In the finance domain, a portfolio refers to a bundle of assets held by an investor. The assets include stocks, bonds, real estate, bank accounts and other financial products on the market that can yield a return through investment. The portfolio is usually used as a risk reduction strategy by investors. Portfolio management is a decision making process that involves determining the types and amounts of assets that should be invested or divested and when. This is very similar to the process of technical debt management where software managers make decisions on when and what technical debt items should be paid or kept. Due to the involved uncertainty, the expected return and variance of the return are often used to evaluate the performance of assets. This evaluation can be performed on a qualitative basis if the future value of an asset is highly uncertain, or on a quantitative basis using mathematical models, if the information to derive the future value is available. The goal of portfolio management is to select the asset set that can maximize the return on investment or minimize the investment risk.

2.1 Principle

Since different types of assets such as stocks and real estate vary in liquidity and sensitivity to market changes, they usually exhibit different volatility and performance patterns. Even in a single asset category such as a stock market, the prices of different stocks may have different fluctuation ranges and change directions. Therefore, holding a portfolio of assets, i.e. diversifying the investment, is less risky than investing in a single asset. In other words, diversification can limit the investment risk. This principle holds as long as the constituent assets in a portfolio do not have a perfect positive correlation. A simplified example would be as follows. Assume a portfolio has n assets $A_{i\ (i\ =\ 1,\ 2,\ldots,\ n)}$ and they are mutually uncorrelated, i.e., the correlation coefficients between their rates of return over time are equal to 0. This would imply that the performances of these assets are unrelated to each other. Also assume that the assets' risks are measured by the variance of the asset return and they are $\sigma_{i\ (i\ =\ 1,\ 2,\ldots,\ n)}^2$. The variance, in this case, represents the probabilities that the assets will return different levels of benefit. Then the variance of the portfolio P that holds all assets in equal proportions is:

$$Var(P) = Var(\frac{1}{n}\sum_{i=1}^{n} A_i) = \frac{1}{n^2}\sum_{i=1}^{n}\sigma_i^2$$

If all the assets have the same variance of return σ^2, then the variance of the portfolio return is equal to $(1/n)\sigma^2$, which is $1/n$ of the risk of the investment on any single asset. This example demonstrates the effectiveness of diversification in reducing the investment risk. It should be noted that diversification can only reduce the specific risks that are associated with individual assets, while it is ineffective in dealing with market risk (systematic risk), which is common to all assets, e.g., an economic crisis [2].

2.2 Models

Based on the diversification principle, various portfolio models have been proposed. A portfolio model is a mathematical formulation of the diversification problem. Among these models, the Modern Portfolio Theory is an example of a mean-variance analysis model, which uses the expected return and the return variance to measure the performance of a portfolio [3]. Under the Modern Portfolio Theory model, portfolio return is defined as the weighted sum of the constituent assets' expected returns. The expected return is given by

$$E(R) = \sum_i \alpha_i E(R_i)$$

where R is the return on the portfolio, R_i is the return on asset i and α_i is the weighting of component asset i. The portfolio return variance is

$$\sigma^2 = \sum_i \alpha_i^2 \sigma_i^2 + 2\sum_{i<j} \alpha_i^2 \rho_{ij} \sigma_i^2 \sigma_j^2$$

where ρ_{ij} is the correlation coefficient between R_i and R_j. The portfolio risk is the standard deviation of the portfolio return σ. Then the model can be expressed as a constrained optimization problem with an objective function to either minimize the portfolio risk or maximize the portfolio return.

Although this model is competent in finding the optimal portfolio, the assumptions that the model has raise challenges to its application. For example, the models assume that the assets' return conforms to a normal distribution, which may not be true in reality. Therefore this model has been extended by other researchers to allow other distributions of the asset return [4]. In addition, some assumptions are specific to the finance domain and thus the model cannot be used for technical debt management without tailoring. For example, assets in the finance domain are continuously divisible, while it's not generally possible to partially own a technical debt item. Therefore, a new constraint needs to be added to the model when it is applied to technical debt management.

3. PROPOSED APPROACH

To help software managers make informed decisions about the technical debt in their projects, our research is aimed at developing and validating measures and mechanisms for technical debt management. As a basis for this line of work, we propose an initial technical debt management framework. The approach is designed to be flexible so that it can be tailored to incorporate new ideas, approaches and results from our own and others' research in this area.

The core component of the proposed approach is a "technical debt list." The list contains technical debt "items", each of which represents an incomplete task that may cause problems in the future. Examples of technical debt items include source code that needs to be modified to conform to the architectural design (including refactoring), test cases that need to be exercised, documentation that needs to be updated, etc. Each item includes the location of the debt, the time at which it is identified, the responsible person, the reason why it is considered technical debt, an estimate of the principal, estimates of the expected interest amount (EIA) and interest standard deviation (ISD), and estimates of the correlations of this item with other technical debt items. As with financial debt, the principal refers to the effort required to complete the task. The interest is the extra work that will be needed if this debt item is not repaid. Since it is uncertain that extra effort will be required, we use expected interest amount and interest standard deviation to capture the uncertainty. Table 1 shows an example of a technical debt item.

Table 1. Technical Debt Item

ID	20
Date	7/18/2009
Responsible	Rose Angel
Type	Documentation
Location	Module S
Description	In the last release, function F was added to module S, but the documentation has not been updated to reflect this change.
Principal	1.5 person-day
EIA	0.5 person-day
ISD	0.1 person-day
Correlations with other debt items	Item 5: 0 (unrelated to debt item 5) Item 12: 0.9 (it is very likely that this item will incur interest if debt item 12 needs extra work to complete, and vice versa.)

The first step of technical debt management is identifying technical debt. There are different types of technical debt [5]. Design debt can be identified by statically analyzing source code or inspecting code compliance to standards. Testing debt can be identified by comparing test plans to test results. The tests planned but not run are testing debt items. Defect debt can be identified by comparing test results to change reports. The defects that are found but not fixed are defect debt items. By comparing change reports to documentation version histories, documentation debt can be identified. If the changes are made without accompanying changes to documentation, the corresponding documentation that is not updated is documentation debt. The technical debt list must be reviewed and updated after each release, when items should be added as well as removed.

After identifying technical debt items, the fields on the technical debt list such as date, responsible person, type, location and description can be filled out. The rest of the information, i.e. principal and interest estimates will be obtained through measuring technical debt, as described below.

3.1 Measurement

Different types of technical debt may require different forms of measures. For example, the number of known but not fixed defects is a measurement of defect debt. The difference between expected code coverage of the test suites and their actual coverage can be used to measure testing debt. Since the ultimate goal of managing technical debt is to facilitate decision making, it is necessary that all types of technical debt have compatible units of measurement so that they can be easily monetized and are comparable to one another. Inspired by the technical debt metaphor, the proposed approach uses principal and interest to measure all types of technical debt. In this paper we use person-day as the unit of the metrics. As mentioned before, the interest metric is further broken into two sub-metrics due to the uncertainty of interest.

Initially, when a technical debt item is created, the principal, expected interest amount, interest standard deviation and correlations with other debt items can be estimated subjectively according to the maintainer's experience. These rough estimates can be adjusted later using historical data or through other types of program analysis.

Historical effort data can be used to achieve a more accurate estimation when estimating principal, i.e. the amount of effort required to complete a technical debt item task. The more accurate and detailed the data an organization has, the more reliable the estimation. Expected interest amount together with interest standard deviation determines the distribution of the extra effort required. They address questions such as how many defects may occur in the untested part, the probabilities that they occur and the most likely number of defects that occurred. Expected interest amount and interest standard deviation can be estimated using historical effort, usage, change, and defect data. For example, the number of defects and their associated probabilities of occurrence in a particular module, which has not been tested sufficiently, can be estimated based on the past defect profile of that module. Since the probability varies with different time frames, a time element must be attached to the probability. The specific procedures for estimating a particular type of technical debt will depend on the form and nature of the historical data available, but the procedure of doing the estimation can be applied to any given maintenance environment.

Since we will apply the Modern Portfolio Theory model to decision making in technical debt management, we include "correlations with other debt items" as a property to be estimated as well. We use the idea of correlation coefficients to represent the correlation between two technical debt items. Correlation coefficients range from -1 (a perfectly negative correlation between the two items) to +1 (a perfectly positive correlation). Estimating the correlation between two debt items requires analysis of system architecture, functionality structure and impact of system changes. However, we speculate that the correlation coefficient would be either1, 0, or -1 for most pairs of technical debt items. In other cases, the correlations could be determined, for example, through dependency analysis. Automated static code analysis methods can be used to ease the difficulty of correlation estimation.

To fit in portfolio management, technical debt items need to be treated as assets. Accordingly measurements for technical debt need to be transformed. For a particular technical debt item, we need to determine whether it is better to keep that item for now (because it is more likely than other items to continue to give benefit) or to pay it off, i.e. get rid of it. To decide this, we need to determine what the likely net benefit of the item is. In the language of portfolio management, this benefit is called expected asset return. In terms of technical debt, this net benefit would be the principal minus the expected interest amount. On the other hand, we need to balance this expected net benefit with the risk that the item will not, in fact produce a benefit (e.g. if the affected module does, after all, need to be modified). This risk, again in portfolio management terminology, is represented by the variance of return. For technical debt, we are representing this quantity with the standard deviation of the net benefit, which is equal to the interest standard deviation. This measure basically indicates how great a risk there is of the technical debt item not producing the expected benefit.

3.2 Process

Through measurement of technical debt, as described above, all input information for the portfolio approach is ready. Then we can start making decisions using the portfolio approach. A classic scenario is used here to illustrate the decision making process. The scenario is part of release planning, where significant work is planned for component S in the next release. Should some debt be paid down on component S at the same time? If so, how much and which items should be paid? Assume a technical debt list has been maintained and it's up-to-date. The decision will be made with the following steps.

(1) Extract all technical debt items associated with S.

(2) Adjust the estimates for these items based on current plans for the upcoming release (e.g. some uncertainties might become more clear at that time).

(3) Add a constraint to the portfolio approach to ensure no partial holding of any technical debt items.

(4) Set a preferred risk level and run the model to generate the optimal portfolio (*A*) of the technical debt items. Then the items not chosen to be included in the portfolio (*A'*) are those that need to be paid in the next release.

(5) Add up the estimated principal for all items that belong to *A'*. Decide if this cost can be reasonably absorbed into the next release. If not, use this analysis to justify the cost to management.

3.3 Evaluation

While application of the portfolio approach to the software domain provides a new perspective, the performance of the approach in technical debt management remains unknown. Therefore it is necessary to test this approach in real settings. The approach can be evaluated through a case study using ongoing software projects. In the case study, first baseline data will be collected, in terms of effort, cost, and productivity of the project, to be used for comparison. Then the approach will be applied to the project and the cost and benefit information related to this approach will be collected to determine its effectiveness through comparison with the baseline data. However, the benefit information collected in this way may not be fully attributed to this approach. Just an increased awareness and explicit management of technical debt could also make a difference on the performance of the project, in comparison with implicit technical debt management, which is the current industry practice. Moreover, an ongoing project can only take one decision path when applying the proposed approach. Therefore it does not allow application of another approach simultaneously and thus loses the chance to get the true benefit information about this approach through comparison of other approaches.

Due to the aforementioned weakness of this evaluation method, we propose to test this approach using historical project information. The approach will be applied to the past releases of a software project and decisions will be simulated as if the approach had been used. By this means other approaches can also be used for decision simulation. For example, we can compare the set of technical debt items generated by the portfolio approach with a set of randomly selected technical debt items to determine the real benefit of the portfolio approach. Using this evaluation approach, the benefit from explicit technical debt management could also be identified.

4. DISCUSSION

The Modern Portfolio Management Theory was developed for portfolio selection in finance domain. It is brought to the software domain because the principle behind this theory closely matches the decision making problem in technical debt management. However, it is necessary to check its assumptions, conditions and applicability when it is applied to a new area.

4.1 Applicability

Since it was proposed, Modern Portfolio Theory has received a lot of criticism about its assumptions in that they do not match the real market situation. When it is applied to technical debt management, some of the assumptions are still problematic, while others may be relaxed in this domain. For example, the approach assumes that the correlations between assets are fixed and constant. Since the correlation between two technical debt items depends on the type of the changes imposed on the system, the correlation cannot be fixed. The approach has a series of assumptions about the behavior of other investors in the market. Although such assumptions were criticized in the finance domain, they are not a problem in technical debt management because there is actually nothing analogous to another investor in the technical debt market.

Besides the assumption issues, some conditions required by the approach are difficult to satisfy in technical debt management. Compared to the change frequency of asset prices, software projects are rare events. Even if the historical project information is available, deriving a distribution of a variable is still a difficult task. Therefore estimating the standard deviation of the return on technical debt still relies on human experience more than historical data. Moreover, it becomes formidable to estimate the correlations among technical debt items as the number of debt items increases.

4.2 Managerial Implications

Despite the applicability issues of the approach, the principle of diversification still sheds light on the strategies of managing technical debt. For example, holding several small technical debt items is better than having a debt monolith with the same total principle and interest, because the risk is limited through diversification. When making decisions on incurring or repaying technical debt, one should avoid keeping a set of items with high positive correlations. For example, if a system has a poorly written module, then do not leave its documentation out of date because that will make the situation worse when the module needs to be modified. These strategies are not new ideas and may have been used by software practitioners, but by associating them to the principle of diversification, we have found a theoretical basis for these strategies. Based on this principle, new strategies and approaches can be developed. Then the next step is to test these approaches in real settings, as we have proposed for the portfolio approach, so that they are backed by empirical evidence.

5. ACKNOWLEDGMENTS

This work was supported in part by NSF grant #0916699, "Measuring and Monitoring Technical Debt".

6. 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] Fama, E. F. French, K. R. 2004. The Capital Asset Pricing Model: Theory and Evidence. *the Journal of Economic Perspectives*. Vol. 18, No. 3, pp. 25-46.

[3] Markowitz, H. 1952. Portfolio Selection. *the Journal of Finance*. Vol. 7, No. 1, pp. 77-91.

[4] Sortino, F. A. Price, L. N. 1994. Performance Measurement in a Downside Risk Framework. *the Journal of Investing*. Vol. 3, No. 3, pp. 59-64.

[5] Rothman, J. 2006. An Incremental Technique to Pay Off Testing Technical Debt. Available: http://www.stickyminds.com/sitewide.asp?Function=edetail&ObjectType=COL&ObjectId=11011&tth=DYN&tt=siteemail&iDyn=2.