Aug 10th, 12:00 AM

Managing Code Debt in Open Source Software Development Projects: A Digital Options Perspective

Yukun Yang
Georgia State University, yyang42@gsu.edu

Maheshwar Boodraj
Georgia State University, mboodraj1@gsu.edu

Managing Code Debt in Open Source Software Development Projects: A Digital Options Perspective

Emergent Research Forum

Yukun Yang
Georgia State University
yyang42@gsu.edu

Maheshwar Boodraj
Georgia State University
mboodraj1@gsu.edu

Abstract

In this study, we examine the impact of three commonly used digital options on the accumulation of code debt in open source software development (OSSD) projects. Further, we examine the impact of code debt on three measures of OSSD project performance. Specifically, we hypothesize that increased use of defer options and abandon options is negatively related to the accumulation of code debt, while increased use of growth options is positively related to the accumulation of code debt. Further, we hypothesize that while the accumulation of code debt is negatively related to a project's market success and the engagement of peripheral developers, it is positively related to the engagement of core developers. To test our hypotheses, we plan to collect and analyze project data from a leading OSSD platform. We expect our findings to provide new theoretical perspectives for researchers and actionable insights for software practitioners.

Keywords

Digital options, code debt, technical debt, open source software development, IT project management.

Introduction

Code debt refers to constructs in source code that are expedient in the short term but set up a technical context that can make future changes more costly or impossible (Avgeriou et al. 2016; Cunningham 1992). While it may be necessary to incur some code debt to release an application quickly, unmanaged, code debt can have undesirable consequences. For example, it can lead to software that is complex and difficult to maintain (Brown et al. 2010). Several studies show that the typical software application is rife with code debt. For example, one study estimates that a typical software application with 100,000 lines of code has approximately $361,000 worth of code debt (Curtis et al. 2012). Code debt is particularly problematic in the OSSD context for two key reasons. First, given the transient nature and varied skill levels of open source software developers, it is likely that contributed code may not be at the same quality standard or may be incompatible with existing code, thereby generating undesirable code debt. Second, OSSD project owners (people who initiate a project and serve as core developers) have little control over the development pace since developers are free to contribute to and leave a project whenever they want.

In this study, we examine code debt in OSSD projects by adopting a digital options perspective. Stemming from the concept of real options in the finance literature (Black and Scholes 1973), digital options represent the right (but not the obligation) to pursue a larger long-term benefit with a relatively small short-term investment (e.g., Fichman et al. 2005; Tiwana et al. 2007). In the OSSD context, we conceptualize digital options as the right, but not the obligation, for the core development team to invest in new contributions that will enhance project performance. Here, the initial investment is mainly the time and effort required to scrutinize the code contributions. We choose digital options as our research lens for two main reasons. First, digital options are well-suited for highly uncertain environments with sequential and irreversible investment opportunities (Adner and Levinthal 2004), which are typical of OSSD projects. Second, using digital options could directly contribute to accumulating or mitigating code debt, depending on which

option is chosen. For example, the core development team could defer using recommended coding practices, which may slow down the accumulation of code debt.

In the OSSD context, every time the core development team receives a request for merging code contributions from other developers in the community, the team gains a chance to improve the quality of their software product. However, the team does not necessarily know whether the merge will eventually provide increased business value or additional maintenance work. Numerous contributions merged to the codebase can put the core development team at risk of admitting too many contributions that eventually produce little business value or make their software too complex to maintain. The team can adopt options thinking to reduce such risks. Given the preceding discussion, we consider it worthwhile to address the following two research questions: *(1) what is the impact of code debt on the performance of OSSD projects?* and *(2) what is the impact of various digital options on the accumulation of code debt in OSSD projects?*

Theoretical Development

Participation in OSSD Projects

OSSD projects involve the participation of various developers (Hahn et al. 2008). *Project owners* initiate a software development project and serve as *core developers*. *Peripheral developers* (Setia et al. 2012) extend the software functionality or improve the code quality by submitting code via contribution requests. If a contribution request is approved, participants' contributions are merged into the main body of the software (the codebase). *Users* focus on downloading and using the source code (Scacchi 2007). Given the critical role that peripheral developers and users play in promoting the OSSD process, they are viewed as key providers of resources to build and reshape the boundaries of the project (Butler and Wang 2012; Jain et al. 2015). Specifically, the number of downloads is a common measure of market success in OSSD projects, and the number of peripheral developers that a project attracts and the number of core developers that it maintains signal the development power of OSSD projects. It is meaningless to develop software if no one uses it. Similarly, without the continuous engagement of core developers and peripheral developers, OSSD projects may not survive. Therefore, we explore the impact of code debt on OSSD project performance from the standpoints of users, core developers, and peripheral developers, respectively.

Code Debt and Project Performance

Code debt was first introduced in software engineering as a metaphor to describe how writing 'quick and dirty' code meant taking on debt that often had to be paid back later (Cunningham 1992). Code debt is caused by multiple factors. For example, code debt is incurred when engineers take shortcuts that fall short of best practices (Allman 2012), when there is pressure to deliver working software quickly (Boodraj 2018), and when there is no systematic verification of quality (Kruchten et al. 2012). Of course, there are times when code debt can be incurred strategically to acquire first-mover advantage or collect early customer feedback (Lim et al. 2012; Tom et al. 2013). If code debt is not carefully managed, it can adversely affect a software system's long-term maintainability, evolvability, and quality (Rolland et al. 2018), which ultimately influences the engagement of core and peripheral developers.

Code debt is often viewed as an indicator of software quality, which shows how much effort the core development team invests in maintaining the software and keeping it at a high level of quality (Brown et al. 2010). An OSSD project with significant code debt might be perceived as caring less about the quality of the code and more about the functional diversification of the product. Also, as more code debt is incurred, it becomes more challenging to add new features, which may slow down the software development pace and put the project in a less competitive position compared with the similar products in the market. Realizing the low quality of the code, users may feel less interested in downloading it, resulting in decreased market success for the project. Thus, we argue that *code debt is negatively related to the market success of OSSD projects (Hypothesis 1).*

In addition to attracting more users to an OSSD project, the core development team also needs to attract peripheral developers to ensure the sustainability and popularization of the product (Setia et al. 2012). Since core developers typically show their commitment to a project when they join the team, they may feel more responsibility to develop the software compared to other participants (Joblin et al. 2017). Although they may deliberately incur some code debt to be the first to introduce new functions to the market, core

developers are more sensitive to the negative impacts of significant code debt and will take necessary actions to minimize the debt so as not to put the project at risk. We, therefore, expect that more code debt will prompt core developers to increase the effort spent on code maintenance. Thus, we argue that *code debt is positively related to core developer engagement in OSSD projects (Hypothesis 2)*.

However, code debt seems to play a different role in attracting the engagement of peripheral developers. As peripheral developers voluntarily contribute to the project, significant code debt portrays the code as messy and difficult to maintain (Tom et al. 2013). Peripheral developers may, therefore, feel that the project is less attractive and decide not to contribute rather than spending time to improve it. In addition, significant code debt may indicate that the core development team does not put enough effort into quality management or are less effective in handling contribution requests (Scacchi 2007). In either case, peripheral developers may find the project less attractive and be reluctant to maintain or improve the software any further. Thus, we argue that *code debt is negatively related to peripheral developer engagement in OSSD projects (Hypothesis 3)*.

Digital Options and Code Debt

There are six types of digital options in the extant literature (Fichman et al. 2005). To fit the notion of digital options in the OSSD context, we examine the three most used types: *defer*, *abandon*, and *growth*. We did not include the other three types - *change scale*, *stage*, *switch* - because they did not fit within the context of the current study, or they could not be reliably identified in the OSSD context.

Using the defer option, the core development team can postpone the decision on a contribution request without imperiling the codebase. It buys the core team more time when they are uncertain about the value of the code contribution and want to wait for more signals to support their decision making (Fichman et al. 2005). During this process, the core development team needs to reconsider the architecture of their current software, revisit the codebase to assess the compatibility of the new contribution, and evaluate the ability of the team in maintaining the new code in the long run (Scacchi 2007). What the core development team is cautious about is not only the value of adding a new feature but also the amount of new code debt that will be incurred (Tom et al. 2013). With the admission decision postponed, the risk of incurring code debt by imprudently adding new features is also delayed. When the core development team frequently uses defer options to postpone uncertain decisions, the pace of accumulating code debt is reduced through revisiting the current code and rethinking the software design. Thus, we argue that *an increasing use of defer options is negatively related to the accumulation of code debt in OSSD projects (Hypothesis 4)*.

Using the abandon option, the core development team can decline a contribution request after carefully reviewing it, which will allow project resources to be redeployed where needed (Fichman et al. 2005). Contribution requests could be abandoned because of poor quality code or because of a mismatch between the contribution and the overall project objective (Scacchi 2007). Although the core development team still spends time evaluating the contribution request, this investment is much less compared to the amount of time and effort they will need to spend on the code over the long term if they accept the contribution. The abandon option prevents the core development team from wasting resources to maintain code that adds little value to the project (Besker et al. 2019). It also protects the codebase from unnecessary complexity by keeping it concise. Thus, we argue that *an increasing use of abandon options is negatively related to the accumulation of code debt in OSSD projects (Hypothesis 5)*.

Using the growth option, the core development team considers whether a code contribution is beneficial to the project in the long term and decides whether to merge it to the codebase after evaluating its quality, discussing its potential value, and revising the code where necessary (Fichman et al. 2005). One of the approval reasons comes from the opportunity that the contribution opens to pursue a variety of potential follow-on developments to the project (Rolland et al. 2018). While the code contribution has the potential to increase the value of the software, it would likely increase the complexity of the codebase resulting in increased code debt. The code contribution would also require additional time and effort to maintain it, especially when a new software version is released (Song et al. 2018). When the core development team frequently uses growth options, the risk of accumulating code debt may increase significantly in the absence of timely payoffs. Thus, we argue that *an increasing use of growth options is positively related to the accumulation of code debt in OSSD projects (Hypothesis 6)*. Figure 1 illustrates our research model.

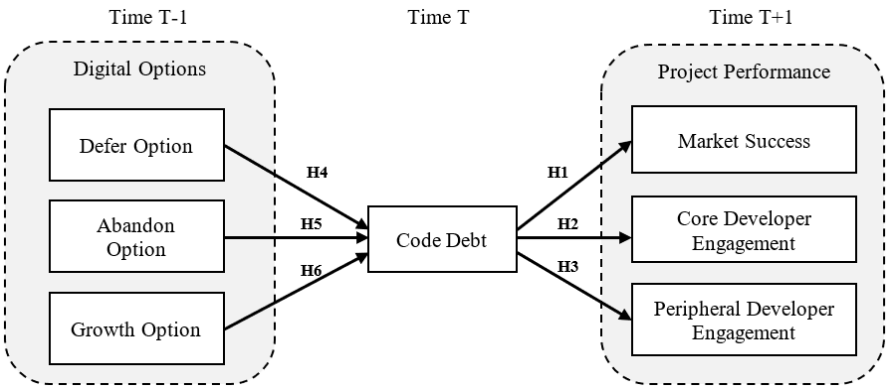

Figure 1. Research Model

Data and Methodology

We developed a web-scripting program to collect project data from GitHub, one of the largest and most well-known sites for hosting open source software projects. We will use a keyword such as "music" to identify a potential list of projects. Afterward, we will filter the list to include only projects that were active throughout 2019. Because of the delayed effect of digital options on code debt, and code debt on project performance, we will select three-month time windows to examine our research model. Specifically, we will identify the digital options that the core development team adopts in month *T-1*. We expect that this will affect the accumulation of code debt in month *T*, which will eventually influence project performance in month *T+1*. We will use fixed effects regression as the baseline model, accompanied by the instrumental variable identification strategy to deal with the potential endogeneity issue.

We will measure the number of *defer* options as the number of contribution requests that remain open for more than a month without further action from the core development team, the number of *abandon options* as the number of contribution requests that are declined within a month, and the number of *growth* options as the number of contribution requests that are approved to be merged into the master codebase within a month. As a robustness check of the identification of digital options, we will also collect the discussion data for each contribution request and use a text mining technique such as topic modeling to ensure that there is consistency in classifying the development team's behavior. We will measure *code debt* in each time period by using SonarQube to perform static analysis of the code to detect code debt. We will measure *market success* as the number of times a GitHub project has been downloaded, *core developer engagement* as the number of code contributions from core developers, and *peripheral developer engagement* as the number of code contribution requests submitted by peripheral developers. We will also measure control variables such as *core development team size* and *project age*.

Expected Contributions

In this study, we investigate the impact of digital options on the accumulation of code debt in OSSD projects. Further, we investigate the impact of code debt on OSSD project performance. Our study contributes to the extant literature on digital options and code debt in several meaningful ways. First, our work complements the literature on managing code debt in offline contexts by studying how software development teams manage code debt in a popular online context. Second, our work offers a quantitative perspective on the relationship between digital options and code debt, which has largely been studied using qualitative methods, such as case studies. Third, our work extends the research on the interactive relationship between digital options and code debt by offering a fresh perspective on how digital options lead to the accumulation of code debt in OSSD projects, which in turn influences overall project performance. Our study also offers actionable insights for core development teams. For example, we highlight the impact of pursuing various digital options on code debt in OSSD projects. Further, we demonstrate that while increasing code debt can promote core developer engagement, reducing code debt can ultimately increase market success and peripheral developer engagement in OSSD projects.

REFERENCES

Adner, R., and Levinthal, D. A. 2004. "What is Not a Real Option: Considering Boundaries for the Application of Real Options to Business Strategy," *Academy of Management Review* (29:1), pp. 74-85.

Allman, E. 2012. "Managing Technical Debt," *Communications of the ACM* (5:5), pp. 50-55.

Avgeriou, P., Kruchten, P., Ozkaya, I., and Seaman, C. 2016. "Managing Technical Debt in Software Engineering," *Dagstuhl Reports* (6:4), pp. 110-138.

Black, F., and Scholes, M. 1973. "The Pricing of Options and Corporate Liabilities," *Journal of Political Economy* (81:3), pp. 637-654.

Besker, T., Martini, A., and Bosch, J. 2019. "Software Developer Productivity Loss Due to Technical Debt – A Replication and Extension Study Examining Developers' Development Work," *Journal of Systems and Software* (156), pp. 41-61.

Boodraj, M. 2018. "Leveraging the Planning Fallacy to Manage Technical Debt in Agile Software Development Projects," in *Proceedings of the 13th International Research Workshop on IT Project Management*, San Francisco, California.

Brown, N., Cai, Y., Guo, Y., Kazman, R., Kim, M., Kruchten, P., Lim, E., MacCormack, A., Nord, R., Ozkaya, I., Sangwan, R., Seaman, C., Sullivan, K., and Zazworka, N. 2010. "Managing Technical Debt in Software-Reliant Systems," in *Proceedings of the FSE/SDP Workshop on Future of Software Engineering Research*, pp. 47-52.

Butler, B. S., and Wang, X. Q. 2012. "The Cross-Purposes of Cross-Posting: Boundary Reshaping Behavior in Online Discussion Communities," *Information Systems Research* (23:3), pp. 993-1010.

Cunningham, W. 1992. "The Wycash Portfolio Management System," in *Proceedings on Object-Oriented Programming Systems, Languages, and Applications*, Vancouver, British Columbia, pp. 29-30.

Curtis, B., Sappidi, J., and Szynkarski, A. 2012. "Estimating the Principal of an Application's Technical Debt," *IEEE Software* (29:6), pp. 34-42.

Fichman, R. G., Keil, M., and Tiwana, A. 2005. "Beyond Valuation: 'Options Thinking' in IT Project Management," *California Management Review* (47:2), pp. 74-96.

Hahn, J., Moon, J. Y., and Zhang, C. 2008. "Emergence of New Project Teams from Open Source Software Developer Networks: Impact of Prior Collaboration Ties," *Information Systems Research* (19:3), pp. 369-391.

Jain, R., Cao, L., Mohan, K., and Ramesh, B. 2015. "Situated Boundary Spanning: An Empirical Investigation of Requirements Engineering Practices in Product Family Development," *ACM Transactions on Management Information Systems* (5:3), pp. 16:1-29.

Joblin, M., Apel, S., Hunsen, C., and Mauerer, W. 2017. "Classifying Developers into Core and Peripheral: An Empirical Study on Count and Network Metrics," in *Proceedings of the 39th International Conference on Software Engineering*, pp. 164-174.

Kruchten, P., Nord, R. L., and Ozkaya, I. 2012. "Technical Debt: From Metaphor to Theory and Practice," *IEEE Software* (29:6), pp. 18-21.

Lim, E., Taksande, N., and Seaman, C. 2012. "A Balancing Act: What Software Practitioners Have to Say About Technical Debt," *IEEE Software* (29:6), pp. 22-27.

Rolland, K. H., Mathiassen, L., and Rai, A. 2018. "Managing Digital Platforms in User Organizations: The Interactions between Digital Options and Digital Debt," *Information Systems Research* (29:2), pp. 419-443.

Scacchi, W. 2007. Free/Open Source Software Development: Recent Research Results and Methods. *Advances in Computers* (*69*), pp. 243-295.

Setia, P., Rajagopalan, B., Sambamurthy, V., and Calantone, R. 2012. "How Peripheral Developers Contribute to Open-Source Software Development," *Information Systems Research* (23:1), pp. 144-163.

Song, P., Xue, L., Rai, A., and Zhang, C. 2018. "The Ecosystem of Software Platform: A Study of Asymmetric Cross-Side Network Effects and Platform Governance," *MIS Quarterly* (42:1), pp. 121-142.

Tiwana, A., Wang, J., Keil, M., and Ahluwalia, P. 2007. "The Bounded Rationality Bias in Managerial Valuation of Real Options: Theory and Evidence from IT Projects," *Decision Sciences* (38:1), pp. 157-181.

Tom, E., Aurum, A., and Vidgen, R. 2013. "An Exploration of Technical Debt," *Journal of Systems and Software* (86:6), pp. 1498-1516.