Continuous Debt Valuation Approach (CoDVA) for Technical Debt Prioritization

Marek G. Stochel a,b
marek.stochel@motorolasolutions.com
Piotr Cholda b
piotr.cholda@agh.edu.pl
Mariusz R. Wawrowski a
mariusz.wawrowski@motorolasolutions.com

a Motorola Solutions, Kraków, Poland
b AGH University of Science and Technology, Department of Telecommunications, Kraków, Poland

Abstract—This paper proposes a technical debt prioritization technique to be used throughout software development lifecycle. Our method stresses a broader perspective of the product, so the focus is kept on optimizing the software artifacts supporting the complete flow of the value for the customer (including e.g. test environment or deployment pipelines). Many technical debt approaches are focused on static code analysis and misconsider business aspects as a separate activity. On the contrary, our method, based on the New Solution Selling Process, strongly advocates for a business perspective. It facilitates technical debt valuation against a predicted product roadmap by a close alignment with a structured sales process and the advancement of sales opportunities. Taking into account prediction of sales, progress of conversations with potential clients and bids, probability that a given functionality will be prioritized for the development, this approach may serve as a reference point for measuring business value of technical debt continuously. This valuation serves as a relative comparison of technical debt items enabling prioritization of effort to pay off technical debt. The method was evaluated on a release of a wireless telecommunication system. The results confirm that the suggested priorities may address the most profitable product areas from the company’s perspective and ultimately optimize the value for customers.

Keywords—Software Development; Software Engineering; Software Maintenance and Evolution; Technical Debt Management, Prioritization and Valuation

I. INTRODUCTION

The technical debt (TD) term refers to immature software development artifacts, which incur measurable consequences if not addressed. It consists of a set of actionable technical debt items (TD Items) which point to these immature artifacts, and indicate the difference between their current and desired state. TD Items are concerns associated with a system. The process of addressing them, i.e. paying off technical debt, is called refactoring [1][2]. Technical debt is inevitable [3] and needs to be managed to sustain continuous delivery of value to customers. It follows the financial debt construct, defining principal as the effort required to bring the software development artifacts to the optimal state. An interest indicates the additional effort to be spent on maintaining these artifacts in their design-time quality, i.e. when they remain in a suboptimal state [4]. As the number of supported software artifacts grows and pressure on delivery speed rises, the need for technical debt prioritization is evident. Even though many TD prioritization techniques have been proposed, the ultimate goal—defined as seamless cooperation between business and engineering—is far from being achieved [5]. The ROI (Return on Investment) techniques usually require justification of potential savings before funding for refactoring is secured. They are used to prove whether the business is using its cash and equity resources effectively. Moreover, currently dominant Agile software development practices stress roadmap flexibility, which hinders the long term perspective and makes a solid justification rarely possible. Therefore, we perceive the originality of our CoDVA approach in continuous assessment and prioritization of the technical debt items. It is driven by predicted future profits, not only by engineering productivity or potentially saved effort. Additionally, it provides the business context for refactorings planned by software engineers.

Paper structure. The remaining sections cover the following topics: Section II discusses the background and related work. Section III introduces the CoDVA approach, whereas Section IV provides its empirical validation. Section V discusses threats to validity. Finally, conclusions and future work are presented in Section VI.

II. RELATED WORK

Technical debt prioritization, being one of the technical debt activities, is focused on ranking identified technical debt according to the predefined rules, in order to determine items which need to be paid off first [3]. As technical debt cannot be avoided and sometimes is unknown, there is a need to manage and prioritize efforts to maintain it at a level that ensures continuous delivery of value to customers. In the work of Lenartuzzi et al. [5], a systematic literature review was conducted on technical debt prioritization. The authors identified several perspectives related to technical debt which are considered during prioritization. They were mapped to the following groups: business, customer, evolution, maintenance, productivity, project factors, quality debt, social factors, system quality, other/unspecifed.

Analyses performed in these areas are supported by tools, which are mostly focused on specified rules or patterns, either for codebase, such as SonarQube 1, CAST 2 or architecture, e.g. ARCAN 3. Even though more tools for static and dynamic code and architecture analysis exist, their major purpose to reveal issues or incompliance differs from

1 https://www.sonarqube.org
2 https://www.castsoftware.com
3 http://essere.disco.unimib.it/wiki/arcan
the CoDVA goal which is prioritization against business objectives. A different perspective is presented in AnaConDebt4 focusing on managing technical debt. The tool resembles an issue tracker customized for storing technical debt items, enhanced with calculation and visualization of technical debt negative impact score.

Engineering perspective on refactoring effort was captured in the work of Sae-Lim et al. [6], where effort is prioritized in areas currently under development (task relevance) to reduce high severity code smells. In the work of Kim et al. [7], engineers stress refactoring risk and cost perspective. Additionally, a technique of minimizing refactoring effort to achieve maximum code smell correction was proposed by Choundry and Singh [8]. Guo et al. [9] claim that feasibility of technical debt management depends on the cost associated with additional activities imposed on a development team. We would like to call additionally an industrial approach, which may provide some insight into budgeting refactoring. The Scaled Agile Framework (SAFe) [10], defines an Architectural Runway, that consists of the existing code, components, and technical infrastructure needed to implement near-term features without excessive redesign and delay. An investment in the Architectural Runway is continuous to enable a stable velocity of the software development team while turning product backlog into releasable functionality.

Although many prioritization techniques have been proposed, we perceive the originality of the CoDVA approach in continuous prioritization of TD Items by their value from a business perspective. Assignment of value (valuation) enables relative comparison between TD Items, which in turn drives prioritization. Our solution is based upon the New Solution Selling approach introduced by Eades [11]. He has developed a model for the sales process resulting in a clearly defined and structured action plan for sales opportunities. An initial suggestion for its use in technical debt prioritization was discussed in our previous paper as a potential expansion of a Value-Based Technical Debt concept [2]. However, we have not observed any implementation or further advancement of this concept. This paper fills in this gap by providing detailed guidelines on a practical application of this approach complemented by implementation results.

III. CONTINUOUS DEBT VALUATION APPROACH

In this section, we introduce the Continuous Debt Valuation Approach (CoDVA) for technical debt prioritization. In the proposed approach, we assess the entire service offered (the product) in terms of desired improvements. Then our approach embraces the business perspective by utilizing the New Solution Selling process for sales opportunities. While following the prepared plan, each subsequent step increases the probability of successful sale, and hence the probability of a request to develop a missing functionality. This in turn may shape the future product portfolio of the company. In order to tie technical debt valuation to dynamic business perspective, an alignment with a solution selling pipeline milestone chart was suggested [2]. The sales process is divided into stages, from initial contact to final sale. The probability of closing a sale (win_odds) rises, as we progress through the subsequent stages, see Table I below.

<table>
<thead>
<tr>
<th>Milestone</th>
<th>Revenue</th>
<th>Win Odds</th>
<th>Milestone</th>
<th>Yield</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>sales-at-T $</td>
<td>Territory</td>
<td></td>
<td></td>
</tr>
<tr>
<td>S</td>
<td>sales-at-S $</td>
<td>10% Qualified Suspect</td>
<td>10% * sales-at-S $</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>sales-at-D $</td>
<td>25% Qualified Sponsor</td>
<td>25% * sales-at-D $</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>sales-at-C $</td>
<td>50% Qualified Power Sponsor</td>
<td>50% * sales-at-C $</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>sales-at-B $</td>
<td>75% Decision Due</td>
<td>75% * sales-at-B $</td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>sales-at-A $</td>
<td>90% Pending Sale</td>
<td>90% * sales-at-A $</td>
<td></td>
</tr>
<tr>
<td>W</td>
<td>sales-at-W $</td>
<td>100% Win</td>
<td>100% * sales-at-W $</td>
<td></td>
</tr>
</tbody>
</table>

From the sales perspective, these milestones help to identify, measure and analyze the actions which should be taken to further advance the sales opportunities (increasing the probability of the sale). From the technical debt prioritization perspective, these milestones help to evaluate the technical debt against a predicted future company portfolio (aligned with likelihood of identified business opportunities), and prioritize technical debt items among themselves. Considering sales prediction and probability that a given functionality will be prioritized for development, this approach serves as a reference point for measuring the value of technical debt items at any given point in time.

Entering the software engineering field, any set of features may have a sales prediction associated with them, and constitute a sales opportunity soj. For this reason, any feature fj should not be evaluated in isolation. Therefore, the milestones (stages in the sales funnel) presented in Table I should refer explicitly to the sales opportunity soj.

Each software release rj may in turn address several sales opportunities, rj= [soj1, ..., sojN]. The software release may contain not only features directly associated with sales opportunities, but also for instance: improvements for delivery speed (optimization of the software deployment), enhancements in monitoring, and updates of logging capabilities. If the product is offered as a service, the maintenance cost of the software solution directly impacts profitability of the service to which customers subscribe.

When we refer to the conceptual model for technical debt presented in [1], we may complement it with a business perspective on technical debt, as presented in Fig. 1.

Any feature in the pipeline may have a sales prediction associated with it. Thus, potential benefits for the company are calculated from its implementation/sale, and similarly any technical debt item impacting the set of features, may undergo a similar quantification of its monetary value for the company. This value is derived from a decreased effort spent on feature implementation. Note that any TD Item might be present in different releases, and its principal and interest just grow with time. Furthermore, assumed savings represent the fraction of sales prediction, not a saved effort. The value may change in time, but at any decision point—when

4 https://anacondebt.com
improvements are to be prioritized—the current business perspective is considered. The model provides a relative estimate of potential benefits from paying off known technical debt items. It resembles the relative estimation technique known in Agile. Furthermore, valuation serves a prioritization purpose: defining order and a relative distance between TD Items.

CoDVA Model. The first practical example of the model is presented below. Initially, let us discuss a simplified version, assuming that: (1) technical debt prioritization is performed with respect to the entire feature pipeline (a complete realization of the roadmap in the future), (2) any sales opportunity is associated with a single feature (in reality, it is usually a bundle of features).

Next, let us take the planned system feature pipeline \( f_1 \ldots f_n \) with expected sales and predicted probability of closing each sale \( win_{odds} \). Additionally, let us consider several technical debit items \( TDI_1 \ldots TDI_n \) with corresponding costs of refactoring. By paying off each technical debt item, we may expect a fractional decrease in the development cost for some features.

Generally a measure of investment’s gain, Return on Investment (ROI), is calculated as follows:

\[
ROI = \frac{Current\ Value\ of\ Investment - Cost\ of\ Investment}{Cost\ of\ Investment}
\]  

(1)

Now, let us assume that sales, indicates sales opportunity associated with feature \( j \); \( win_{odds} \) reflects probability of closing the sale implying that feature will be prioritized for implementation, and \( effort\ saved_j \) represents percentage decrease in effort for feature \( j \), attributed to prior investment in refactoring of TD Item \( TDI_j \). Finally, the cost of refactoring is indicated by \( cost_j \). Then, we propose to alter (1) and calculate the expected CoDVA Return On Investment, named CoDVA ROI (2).

\[
CoDVA_{ROI} = \sum_{j=1}^{n} \left( sales_j \times win_{odds_j} \times effort\ saved_j \right) - cost_j
\]

(2)

Therefore \( CoDVA_{ROI} \) calculated for each of TD Items, reflects the business value of predicted sales (and is no longer attributed to the avoided implementation cost). An example for a feature pipeline with seven features and three technical debt items is presented in Fig. 2.

![Figure 1. Alignment of the business perspective with Conceptual Model [1] for technical debt type](image)

In such a case, only \( TDI_2 > 0 \) would be prioritized. A justification of each refactoring is hard to prepare and maintain as technical debt is not fully known and business perspective changes frequently. This further shows the natural tension between short-term Agile development perspective and long-term view on maintainability. Technical debt is by its nature bound to the long-term perspective of the product. Therefore, following SAFe recommendation for Architectural Runway, we suggest the secured, continuous stream of investment for technical debt reduction. Then we focus on the profitability of codebase where investments are suggested. The more profitable the area where investment is proposed, the higher probability of funding should follow, as more and more functionality is supposed to depend on this improved codebase in the future. Additionally, we should consider services offered to the customers and the ongoing costs of maintenance and support. In such a scenario any prioritized and implemented improvement may impact profitability of the offered services. However, this subscription business perspective was not included in the analysis to simplify the discussion.
IV. Empirical Validation

This section presents the use of the CoDVA method to analyze the feature roadmap and TD Items for a certain wireless telecommunication system. The system release was prepared by a team of ten engineers over a few months. Technical debt which was paid off comprised software artifacts related to the system deployment. The feature names and corresponding costs/sales were anonymized; however, relations among the numbers were preserved.

Initially, a large software development organization calculated the CODVA ROI and effort without considering the costs of deploying subsequent versions of the software for individual clients. Due to the structure of the organization, these costs were not directly related to the software development organization but to sales and support organizations. The implementation of DevOps principles and the use of cloud technology for a certain group of clients was and still is not obvious. Apart from the historical reasons for this conservative approach, Fig. 3 presents the calculation of technical debt which shows the profitability of TDI\textsubscript{1}, TDI\textsubscript{2} technical debt investment over three consecutive years. Originally the clients were expecting a new version of the software once per year.

Over the time, the customer requirements for some subsystems have changed. The following changes significantly impacted technical debt valuation done according to CODVA ROI: the cloud technology adoption, the customer expectation for frequency of new releases (four implementations per year), and finally the risk level (maintaining the Service Level Agreement, SLA) associated with the implementation of new software.

For this reason, the software development organization has taken responsibility for reducing these costs. The implementation of DevOps and cloud were a natural response to these needs. Therefore, technical debt was recalculated taking into account the cost of implementing individual software versions (R1, R2, R3). This required the addition of another TDI\textsubscript{3} item considering the fact that the system architecture was not prepared for the immediate incorporation of DevOps rules.

It is necessary to notice there is a significant cost of TDI\textsubscript{3} and the analysis of investments in paying off technical debt from the perspective of R1, R2, and R3 investments is not very profitable (Fig. 4).

The examples provided (in the retrospective mode) confirm that a secured stream of improvement effort would have addressed the most important areas from the maintainability perspective. Fig. 6 presents results of TD analysis in terms of profitability. We assess when it is beneficial to pay off a particular TD Item and when positive CODVA ROI can be expected. It becomes evident that for some technical debt items an investment is a long-term one, and the perception of profitability depends heavily on stability of the business roadmap and relations with customers. The engineering organization becomes skeptical of ROI usage, claiming that products testing the market are burdened with a very high level of TD that will never be paid
off. That skepticism is justified when the product roadmap changes frequently and pressure for faster delivery rises. Compared to the pure ROI, CODVA method enables fast decision on technical debt items, limiting the justification effort perceived as superfluous and refocusing the team on the product maintainability. Additionally, it enables a broader view on the current state of the product and its future direction, so that the development team is urged to pay off technical debt in the most profitable product areas (prioritized using CODVA_ROI).

**Figure 6.** Profitability of technical debt repayment

V. THREATS TO VALIDITY

Construct validity. The approach does not explicitly consider the interrelation between the features, but TD Items are bound with features to determine potential benefits (many-to-many relationship). Additional reason to keep the model simple is to gain acceptance from the engineering community and obtain feedback from practical use. However, we expect the model will evolve in the future.

Internal validity. The results of the study were assessed in the retrospective mode, based on the data from finished implementation. Additionally, as we recommend continuous investment in paying off technical debt, we should monitor the velocity of the team. Its degradation over time may indicate the insufficient level of funding.

External validity. Implementation of CoDVA should not incur significant costs. However, even though the knowledge needed to bind TD Items with features is obtained during refinement of feature scope, the developers may resist keeping the technical debt register up-to-date. Therefore we assume that a tool support for a large scale adoption may be needed.

VI. CONCLUSIONS AND FUTURE WORK

The CoDVA approach, closely aligned with a structured sales process, strongly advocates for a business perspective. It facilitates continuous technical debt valuation against a predicted product roadmap. This in turn enables prioritization done automatically in a relative way (among technical debt items). The implementation results prove that a fast and reliable feedback loop between business and engineering may refocus the discussion from justification of investments (ROI) to the actual implementation of changes perceived as the most profitable (CODVA_ROI). A real-life example shows that the product roadmap may change dramatically beyond any control, but the consistent stream of improvements may ensure product resiliency. Taking broader perspective into account, aligning it with the company portfolio, CoDVA meets the need for continuous valuation and prioritization technique. Even though an experiment was conducted in retrospective, the implementation results are promising, confirming that suggested priorities address the most profitable areas from the company’s perspective and ultimately optimize the value for customers.

Further work will focus on broader application of the CoDVA approach for more teams and products, and will be executed to verify the findings in the long term. Additionally, an investigation in a broader product context will be conducted to determine which types of technical debt items have the highest impact on the flow.

REFERENCES


