The Technical Debt Management Tools Comparison

LUKA PAVLIČ and TILEN HLIŠ, University of Maribor

This paper focuses on the technical debt metaphor and its management practices. We present an overview of the currently available tools for managing technical debt. We also present leading tools, that enable technical debt measurement. We demonstrate the noticeable deficit in the number of tools that are intended and specialized for management activities only. Even existing tools do not support all technical debt management activities. This observation became a guide in the development of our own solution for managing technical debt. In this paper we also present our tool for managing technical debt (TD Tool).

1. INTRODUCTION

During the software development cycle, teams are confronted with changing requirements, short deadlines, and high-quality requirements. Combined, they lead to higher costs. Cost cuts are a common practice in all industries, software development is not an exception. Managed carefully, they can be even used to boost long-term software quality. In this context, the technical debt metaphor, rooted in financial world, describes a crucial challenge in the software development industry. In long run, debt causes problems. However, it can be employed in useful manner for a short time goal. This can be accomplished by managing technical debt carefully through the development cycle. The metaphor itself covers several aspects – including source code, IT architecture, design decisions, documentation, requirements, and testing as well. The area of technical debt management consists of several well defined and proven practices. Their purpose is to support the decision whether it is sensible to introduce some more debt to a project by monitoring its accumulated quantity. The debt management process can be facilitated using dedicated supporting tools. The specialized tools landscape is opening. Teams can choose between several types of a tools that offer functionalities from assessing current debt quantity via measuring quality attributes to manual debt management tools.

To manage technical debt well, it is an imperative to use technical debt management tools. Where fast development is needed, we can accept some debt, which is later removed [Ambler Scott 2013]. First of all, teams need to be aware of its existence. Logical next steps include locating, assessing and describing existing or new technical debt items. They (technical debt items) are a basis for debt elimination in the long run, when teams decide to do so.

2. TECHNICAL DEBT MANAGEMENT PRACTICES AND CHALLENGES

Due to continuous changes in the market and the need for agility in the development of modern information solutions, it is crucial that development teams also manage technical debt. Development teams are confronted with changing demands, short-term, and high-quality requirements, which

The authors acknowledge the financial support from the Slovenian Research Agency (research core funding No. P2-0057).
Author's address: University of Maribor, Faculty of Electrical Engineering and Computer Science, Koroška cesta 46, SI-2000 Maribor, Slovenia; email: luka.pavlic@um.si, tilen.hlis@um.si.

In: Z. Budimac and B. Koteska (eds.): Proceedings of the SQAMIA 2019: 8th Workshop on Software Quality, Analysis, Monitoring, Improvement, and Applications, Ohrid, North Macedonia, 22–25. September 2019. Also published online by CEUR Workshop Proceedings (http://ceur-ws.org, ISSN 1613-0073)

makes the costs of implementing changes higher. To achieve desires for fast system development, developers often use shortcuts that provide a working prototype, which is not complete. Consequently, the emergence of technical debt is unavoidable. It is vital that development teams are aware of this and are focusing their time on the management of technical debt [Allman 2012]. Management consists of several activities and approaches that prevent the emergence of new technical debt while keeping the existing one below the critical limit.

2.1 Technical debt management activities

The purpose of management is to keep the accumulated debts under control. Aim of the technical debt management is not about abolished it entirely. Management of technical debt consists of a series of activities that prevent the creation of unwanted technical debt or is dealing with existing debt to keep it below the permissible limit. After reviewing the literature, we rely on the descriptions of the activities made in the systematic study of the technical debt (TD) and its management. The study defines eight activities, summarized in Table 1.

Table 1: Technical debt management activities [Li et al. 2015]

Activity	**Description**
TD identification	The purpose of the activity is the discovery of a technical debt arising from technical decisions within the software system. Operation is performed using various techniques, such as static source code analysis.
TD measurement	The activity quantifies the benefits and the cost of a known technical debt through evaluation techniques. It is also used for estimating the share of technical debt concerning the entire information solution.
TD prioritization	Ranks identified technical debt according to pre-defined rules. The allocation determines which the technical debt must be eliminated first and which can remain in later versions of information solutions.
TD prevention	Aims to prevent potential technical debt from being incurred.
TD monitoring	Activity watches the changes in the cost and benefit of unresolved technical debt over time.
TD repayment	Removes or resolves technical debt with techniques such as re-engineering or refactoring.
TD representation/ documentation	Provides ways to present and codify technical debt in a uniform approach to display it to specific roles in the development company.
TD communication	Makes identified technical debt visible to all the roles in a development company so that it can be discussed and further managed.

2.2 Technical debt management challenges

The management of technical debt involves several challenges. In the literature we can find the lack of a standard unit to indicate the amount of technical debt that would be measurable and generally

comparable. It is usually referred as the biggest challenge. During assessing the amount of debt, the amount of principal is estimated. The principal can be expressed in the form of working hours or in the form of several financial assets, needed to eliminate debt. Because this is a subjective assessment, large deviations can occur between the estimate and the actual investment. Due to difficulties in measuring the amount of debt development companies sometimes do not decide on its management.

The next management challenge is that debt is usually created and eliminated by various employees. Organizations find it difficult to determine the business value of the internal quality of the developed software. It is difficult to translate technical debt into economic consequences. The reviewed literature find that it is problematic to eliminate the existing debt and at the same time, prevent the emergence of a new one [Li et al. 2015].

2.3 Technical debt tools characteristics

Tools for managing technical debt should target both practitioners and researchers. The design of a tool should be guided by empirical evidence about developers' practices so that the tool is tailored to the actual needs of the developers. Usage statistics should be recorded as early as possible, to allow focusing on improvements targeting the most popular tool features. A tool should be offered as an IDE plug-in instead of a standalone application, and it should require minimum installation and configuration effort. A tool should be tested in an industrial setting and should also be open-sourced as early as possible, even if it is not as mature and stable. A tool should be accompanied with documentation, tutorials, and code snippets demonstrating the use of its API.

3. LEADING TOOLS COMPARISON

For the successful management of technical debt, we need relevant information on individual debt. Data is also required for the proper removal planning. An essential step for a successful debt management is to record all debts and capture all necessary information. A good practice is the use of dedicated technical debt management tools.

In this section, we first introduce the landscape of currently available tools, according to the extensive literature overview. In our analysis, we have included tools for managing technical debt and, in addition, also leading tools which are intended for technical debt measurement.

With selected tools for managing technical debt, we rely on a study done in the form of a survey of how technical debt is managed. 15 large organizations were included in the study [Martini et al. 2018]. The questionnaire covered a wide area of technical debt management within the organization. We mainly focused on the answers to the question of which tools are used to track technical debt. Solutions were presented in the form of the word cloud, which shows the distribution of tools used among the respondents. Tools are mostly utilized for backlog, documentation, as a static analyzer, and as issue trackers. In conclusion, the following considerations on the tools were made [Martini et al. 2018]:

- Comments in the code cannot be considered for tracking technical debt.
- Documentation of technical debt increase awareness but it also has the highest overhead, therefore is not considered as a high level of tracking. Respondents answer that the main tools used for documentation of technical debt are Microsoft Excel or Word what we found as a not recommendable practice.
- Using the bug system for tracking technical debt does not increase the level of awareness, and it has a slightly higher overhead. Study infer that this is also not the best way of tracking technical debt.

- Backlogs increase tracking level and are also one with the least overhead. The study suggests using backlogs is one of the best practices at the moment to follow technical debt. The most used tools are Jira, Hansoft, and Excel.

After a comprehensive overview of the tools for managing technical debt that are mentioned in the literature and are considered as leading solutions, we focused on the following tools.

The **Jira software** is a product developed by Atlassian and is designed to record and track bugs and issues that occur during software development. Jira offers management of bugs and issues and also the project management functionality. In 2017, Jira was ranked first in the popularity of tools for bugs and issues management [Accordingmanagement 2018]. In 2019, Gartner announced Jira as the leading solution in the field of enterprise agile planning tools in conjunction with AgileCraft [Gartner 2019]. The identified technical debt is only assessed by a priority and not by its quantity. The tool can be used as a trial version for one month [Atlassian 2019]. A significant disadvantage of Jira is the lack of support for technical debt measurement. The tool does not offer any quantitative input options to define the extent of the technical debt; therefore, monitoring is insufficient.

Hansoft is another tool for software development project management. Hansoft provides project management software for team collaboration and management in Agile and traditional products and services development. Native Windows, OS X, and Linux application with both on-premise and hosted option available. It also can be used as a plugin for Jira, or we can use their SDK for extensions [Perforce 2019]. The great advantage of software Hansoft is the possibility of free use for up to five users. It is mainly used in the development of video games and IoT. It can show recorded technical debt in the form of a list or with a graph which is manually set [Perforce 2019]. Hansoft, as Jira, has a lack of support for technical debt measurement. The tool offers that each issue can be assigned with status and severity.

Next, in a series of our selected tools is **Redmine**. It is open source and released under the terms of the GNU General Public License v2 (GPL). The tool is created with Ruby on a Rails framework and is suitable for individual use or smaller business groups. Redmine is one of the few tools that offer input option for quantitative debt assessment. We can specify the date by which the debt needs to be eliminated, and we can also add the number of hours that will be required to eliminate discovered technical debt. Furthermore, we can also mark debt to be partially removed, which is also reflected in the Gantt chart [Redmine 2019]. Currently, the biggest drawback of Redmine is the fact that it is not so well known and widespread.

The **DebtFlag** tool is designed to capture, track, and resolve technical debt in the software development. The tool is intended for use as a plugin in the integrated development environment. Developers can use a tool to capture technical debt and save the location of it. DebtFlag consists of two parts, a plug-in for the integrated development environment Eclipse, where the developer inputs the data of the identified debt, and the web application that takes care of the dynamic presentation of debts (Holvitie in Leppanen, 2013). DebtFlag offers out-of-the-box plugin for an integrated development environment. The weakness of the tool is small support for technical debt management activities, and that is limited to only one IDE.

TD-Tracker Tool is a tool that enables tabulation and management of technical debt characteristics. With the tool, we can create a catalog of technical debts from various stages of software development. TD-Tracker Tool helps developers in the decision-making process and lets you manage identified technical debts. It also allows you to connect to external tools. Debts can be

imported from external sources, or users enter them manually (Foganholi, et al., 2015). TD-Tracker Tool has a lack of sufficient technical debt monitoring.

SonarQube, formerly Sonar, is an open-source platform developed by SonarSource and is used to verify the quality of the code continuously. It is designed to performing automated checks by statically analyzing the code for detecting bugs, code smells, and security vulnerabilities in more than 20 programming languages. With these functionalities, we can also very effectively measure technical debt on various projects [SonarQube 2019]. Technical debt in SonarQube is shown as the calculation of the time needed to eliminate it. The calculation is based on the use of metrics. A detailed view of an individual debt displays critical files and suggests improvements. It also shows technical debt type, severity, and status. After calculating the technical debt, an "A" to "E" score is added to it, where "A" means almost no technical debt, and "E" means a critical value. The advantage of SonarQube is that it shows the time needed to eliminate the technical debt based on metrics. It also adds a rating for maintainability, bugs, vulnerabilities, and code smells. The tool has a lack of technical debt monitoring; if we want to review the progress of the elimination of debt, it is necessary to restart the calculation of metrics.

Teamscale analyzes the quality of the source code. With various static analyzes, it points to quality errors that can be quickly reviewed. Its primary purpose is to analyze code to identify specific maintainability constraints and avoid unexpected maintenance costs in the future. Teamscale's analyses work on a wide variety of programming languages. Detecting clones, deep nesting, long methods, and files are included for all languages [CQSE 2019]. Additionally, many specialized checks are included, tailored to detect quality problems in specific languages. Tool core features are real-time feedback, dashboards, and integrated development environment integration. Supported IDE's are Eclipse, Visual Studio, IntelliJ IDEA, and NetBeans. TeamScala offers the ability to read the code directly from the source code repository. You can also link TeamScala to issue trackers, for example, with Jira [CQSE 2019]. Unfortunately, it does not report the approximate time of debt elimination like SonarQube. TeamScale is a useful tool when used in conjunction with others. It offers two months of a free trial.

Ndepend is a tool for measuring technical debt exclusively in the Visual Studio and is added as a plugin. After installation, we can access the dashboard where we select the project and analyze it. The functionality that the tool supports are analyzing, measuring, and monitoring technical debt. It also allows setting the errors priority. It supports only one programing language (C#). Therefore, it is more adjusted to this language. The tool displays errors in the project technical debt and time of debt elimination. Technical debt is also shown in individual categories. The scale of the debt is from "A" to "E". The tool is suitable for a development team that develops exclusively in Visual Studio. A test and paid version are available [Ndepend 2019].

Squore supports various programming languages. The tool is cloud-based and available online. It offers functionalities for bugs and vulnerability overview, measurement of technical debt, and view for displaying files that have higher vulnerabilities. Squore is a platform with a lot of functionality and therefore is quite difficult to use. As with most other technical debt measurement tools, debt is shown in the form of time to eliminate it and critical categories [Squoring 2019]. The tool has a lack of technical debt monitoring; if we want to review the progress of the elimination of debt, it is necessary to restart the calculation of metrics.

The overview of the tools will be appended with the description of our own solution. Our tool (**TD Tool**) is intended for technical debt management. We developed a solution which includes an open

web platform and a plugin for the selected development environment. We tried to integrate as much of the functionality of the reviewed tools as possible and add those that we think are useful in managing the technical debt. A more detailed description of the tool follows after the comparison of the tools in the table 2 and table 3.

Table 2: Comparison of the tools, according to selected functionalities

Functionality	**Jira**	**Hansoft**	**Redmine**	**DebtFlag**	**TD-Tracker Tool**	**SonarQube**	**Teamscale**	**Ndepend**	**Squore**	**TD Tool**
Project management	✔	✔	✔	✔					✔	✔
Setting TD threshold									✔	✔
Manual entry of identified TD	✔	✔	✔	✔	✔					✔
The TD amount estimation			✔		✔	✔		✔	✔	✔
TD calculation using metrics						✔	✔	✔	✔	
TD prioritization	✔	✔	✔		✔	✔	✔	✔	✔	✔
Determining who is responsible for the TD	✔	✔	✔		✔					✔
Managing TD status	✔	✔	✔					✔	✔	✔
Graphical representation of the amount of TD	✔	✔	✔			✔	✔	✔	✔	✔
Total	**6/9**	**6/9**	**7/9**	**2/9**	**4/9**	**4/9**	**3/9**	**5/9**	**7/9**	**8/9**

Table 3: Comparison of the tools, according to supported activities

Activities	**Jira**	**Hansoft**	**Redmine**	**DebtFlag**	**TD-Tracker Tool**	**SonarQube**	**Teamscale**	**Ndepend**	**Squore**	**TD Tool**
TD identification	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔
TD measurement			✔		✔	✔	✔	✔	✔	✔
TD prioritization	✔	✔	✔		✔	✔	✔	✔	✔	✔
TD prevention		✔								
TD monitoring	✔	✔	✔							✔
TD repayment						✔			✔	
TD representation/ documentation	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔
TD communication	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔
Total	**5/8**	**6/8**	**6/8**	**3/8**	**5/8**	**6/8**	**5/8**	**5/8**	**6/8**	**6/8**

3.1 The TD Tool

An overview of the existing technical debt management and measurement tools has shown that there is a noticeable deficit in the number of tools that are intended only for management and not for measuring technical debt with metrics. However, those that are available do not support all technical debt management activities. This problem leads us to the development of our own solution, in which we are trying to capture all the technical debt management activities that are intended for its management.

The developed solution includes an open web platform and a plugin for the selected development environment. After reviewing the existing tools, we decided, which functionality should our tool offer. The online platform allows users to enter, manage, and edit projects. All of the tools examined are missing the functionality of determining the maximum amount of technical debt on a particular project and alerting the user when this limit is exceeded. We successfully implemented this functionality to our online platform. Our tool also supports the option for manually entering the identified technical debt, which can be edited later. The amount of technical debt is estimated for each debt individually. Through the management of debt status, we can prioritize all debts. A graphic representation of the amount of debt on an individual project is ensured for a more straightforward overview of debts. The plugin for the selected development environment offers the possibility to manually enter the identified technical debt from the development environment itself. For the development of our online platform, we used MEAN (Mongo-Express-Angular-Node.JS) stack, which allowed creating a simple RESTful API server also. It serves as an interface for communication with the database. The Angular platform, in conjunction with Angular material components, made it possible for us to build a modern user interface. Login and registration are handled by the Auth0 authentication system. After login, we are redirected to the control panel with two lists, one with all the projects and one with all the debts which are linked to the logged in user. We must click on a particular project or debt to see details about it. We also have options for editing

and deleting of the opened project or debt. The project can only be removed when we remove all the debts that belong to it.

On the left sidebar, we find the option for adding a new project, which is enabled only to the admin users. When the user clicks on the button to add a new project, an input form in the form of a dialogue is opened. The user must enter the name of the project and add members who will participate in the project by their emails. Then it is necessary to choose how to assess the debt in the newly created project. We can choose between an estimate in value of money or an assessment with effort, which is reflected with the number of working days. Finally, the upper limit of the allowed debt should be set for the project. The entered value serves us as a threshold. A warning pops up when the limit is exceeded.

When the user pressed on the desired project, its details are displayed, and the list of all debts is replaced with debts that belong to the chosen project. Each debt is presented with Angular Material card component. With a press on an individual card, details of debt are shown. The current amount of debt on the project is shown using charts. The main user interface is demonstrated in figure 1.

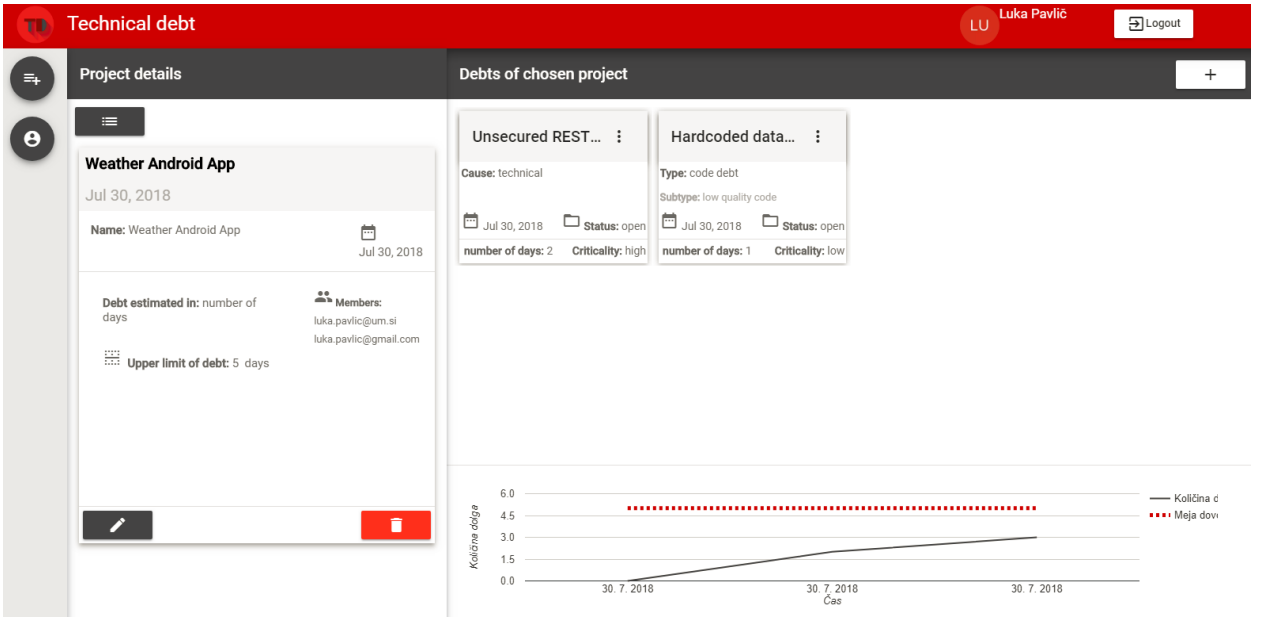

Figure 1: TD Tool main user interface – grouping TD items, based on projects; we can see TD accumulation and upper TD limit

The main functionality of the developed solution is the input form for entering the technical debt item. The user can first choose between the simple and advanced view. For simple entry, the cause that caused the debt should be selected; the user can choose between technical cause and an organizational cause. In the advanced view, the user must add type and subtype of technical debt. Other input fields require name, a description which part of the system is affected by debt and an estimate of debt estimate in value of money or an assessment with effort, which is reflected with the number of working days. We also must add criticality and status of the debt. Every debt has a particular person who is responsible for its elimination.

To complement the online platform and that the developed solution meets the useful characteristics of the technical debt management tool, we developed a plugin for an integrated development environment. Currently supported IDEs are IntelliJ and Visual Studio.

The plugin aims to complete the online platform and to enable adding of identified debt within the development environment. Debt is then available on the online platform and to its users. The visual part of the plugin includes a shortcut in the main menu of the development environment, where the input field to entry technical debt can be opened.

4. CONCLUSION

In this paper, we discussed the concept of the technical debt metaphor. We presented the activities and approaches for managing technical debt, and our observations. The collected activities allowed us to compare the reviewed tools with each other. Overview of existing tools has shown a significant increase in a number of tools that are involved in technical debt, but our review only covers tools that we think are currently among the leaders in this field or have shown great potential for proper management of technical debt. With an overview, we have discovered an area that is poorly supported by tools or the existing ones do not cover a lot of activities that are necessary for the successful management of technical debt. That has become a guide in the development of our solution for managing technical debt. The first step in the development was the production of an entry form of the identified debt item. After the overview of the literature and existing solutions, we have put together a simple and advanced form for entering debt item. We have selected the appropriate functionalities from the current tools and supplemented them with our own so that we could develop the most comprehensive management tool. Development continued with the search for suitable technologies for the development of an online platform and an expansion module for an integrated development environment. We believe that we have created so far, a practical and useful alternative to existing technical debt management tools. The development is, however, ongoing.

REFERENCES

According Management. 2018. Management Tool Ranking. https://project-accordingmanagement.zone/ranking/category/issue.

Allman, E. 2012. Managing Technical Debt, Communications of the ACM. .

Ambler S. 2013. 11 Strategies for Dealing With Technical Debt – Disciplined Agile (DA). http://disciplinedagiledelivery.com/technical-debt/.

Atlassian. 2019. Jira | Issue & Project Tracking Software | Atlassian. https://www.atlassian.com/software/jira.

Cqse. 2019. CQSE - Teamscale. https://www.cqse.eu/en/products/teamscale/.

Gartner. 2019. Magic Quadrant for Enterprise Agile Planning Tools. https://www.gartner.com/en/documents/3872863.

Avgeruiou Li, Z.,P., Lang, P. 2015. A systematic mapping study on technical debt and its management. Journal of Systems and Software 101, 193–220.

Martini, A., Besker, T., Bosch, J. 2018. Technical Debt tracking: Current state of practice: A survey and multiple case study in 15 large organizations. Science of Computer Programming 163, 42–61.

Ndepend. 2019. Improve your .NET code quality with NDepend. https://www.ndepend.com/.

Perforce. 2019. Hansoft Agile Planning Tool. https://www.perforce.com/products/hansoft.

Redmine. 2019. Overview - Redmine. https://www.redmine.org/.

Sonarqube. 2019. Code Quality and Security | SonarQube. https://www.sonarqube.org/.

Squoring. 2019. SQUORING Technologies | Free trial. https://www.squoring.com/en/essayez-gratuitement-squore-pendant-15-jours/.

Tom, E., Aurum, A., Vidgen, R. 2013. An exploration of technical debt. Journal of Systems and Software 86, 6, 1498–1516.