TECHNICAL DEBT: DEFINITION AND CONCEPTS

This evidence briefing reports information on the definition and main concepts of technical debt.

CONCEPTS

What is Technical Debt?
Technical Debt (TD) is a concept first coined by Ward Cunningham in 1992, but since then has received many updates. Being adopted by Agile practitioners, but with a broad application in all software domains, the main definition of TD is [1]:

"In software-intensive systems, technical debt is a collection of design or implementation constructs that are expedient in the short term but set up a technical context that can make future changes more costly or impossible. Technical debt presents an actual or contingent liability whose impact is limited to internal system qualities, primarily maintainability and evolvability."

Three aspects must be observed:

- **TD does not incur only on source code**, as most practitioners believe. It is possible to identify TD in the requirements elicitation or in test cases, for instance;
- To be considered TD, the issue must cause **a short-term benefit, in exchange of a potential future cost**;
- TD is only associated with **internal quality attributes**, like maintainability, so **defects are not TD!** It is possible, though, to exist a defect-type of TD (see more later in this briefing).

So, what should be considered TD [2][3]?
- Poorly written code, that violates code rules;
- "Shortcuts" taken during design;
- Known defects, which elimination is postponed to future sprints or development cycles;
- Architectural problems, like violation of modularity;
- Code smells;
- General low internal quality aspects, that affect maintainability and evolvability.

And what should not be considered TD [3]?
- Defects;
- Trivial code quality issues, that do not violate code rules;
- Lack of supporting processes;
- Unimplemented features or functionalities.

What are other concepts associated with TD?
Since TD is a financial metaphor, related to the financial debt acquired by someone to obtain a short-term gain, some financial concepts are often associated with TD, like:
- **Principal:** The effort that is required to address the difference between the current and the optimal level of design-time quality, in an immature software artifact or the complete software system;
- **Interest:** The additional effort that is needed to be spent on maintaining the software, because of its decayed design-time quality;
- **Repayment:** The amount of effort spent on improving design-time quality. This effort will decrease the effort needed for future maintenance tasks.

How can TD be classified?
The simplest classification of TD items is by its intentionality:
- **Intentional TD** is the ones caused by strategical and planned decisions, when the team or organization decides to achieve a short-term gain at the cost of long-term effort. For example, the decision on developing a simplified architecture solution for a software, knowing that it might not attend the project's future needs;
- **Unintentional TD** is not incurred with a strategical purpose, and usually appears in a project due to the immaturity or lack of knowledge of the practitioners. For example, a bad-written piece of code created by an inexperienced programmer could turn out as a TD item.

Other possible TD classification is by its type, i.e. what was the cause of that specific TD item. Alves et al [2] came up with 15 different TD types. The most recurring in software projects are listed below:
- **Design debt:** Associated mainly with violations of the principles of good object-oriented design, like extensive coupled classes;
- **Architecture debt:** Refers to problems related to the software architecture, such as violation of modularity;
- **Documentation debt:** Debt related to issues observed in the software documentation;
- **Test debt:** Debt found in testing activities, like planned tests that were not run;
- **Code debt:** Associated with problems found in the source-code, that can make it harder to maintain, usually related to bad coding practices;
- **Build debt:** Refers to issues that can hinder the build task, consuming unnecessary time.

Who is this briefing for?

Software engineering practitioners who want to make decisions about internal quality issues and apply scientific knowledge on managing technical debt.

Where does the information come from?

The information on this briefing comes from evidence collected by the author through a literature review in several publications, included:
- [1] Avgeriou, P. et al. Managing Technical Debt in Software Engineering. In Dagstuhl Reports, 2016;
- [2] Alves, N. S. et al. Identification and management of technical debt: A systematic mapping study. Information and Software Technology, 2016;
- [3] Li, Z. et al. A systematic mapping study on technical debt and its management. Journal of Systems and Software, 2015.

For additional information about the Experimental Software Engineering Group at COPPE/UFRJ:

http://lens-ese.cos.ufrj.br/ese/

For additional information about the DELFOS Observatory:

http://www.delfos.cos.ufrj.br

RESEARCH REFERENCE
Silva, V. M. ***Technical Debt and its Management: Analysis and Application in the Brazilian Software Industry***. Masters' dissertation, Federal University of Rio de Janeiro, 2018.

Figure 5.2 – EB1 – Technical Debt: Definitions and Concepts

TECHNICAL DEBT MANAGEMENT: ACTIVITIES AND PRACTICES

This evidence briefing reports information on the different activities to manage the technical debt, along with practices obtained through a survey with practitioners and a literature review.

FINDINGS

What are the Technical Debt Management activities?
Through a systematic mapping study, Li et al [1] consolidated the Technical Debt Management (TDM) activities in eight groups, described below:

- **Technical Debt (TD) identification:** Detects TD caused by technical decisions in software, either intentional or unintentional;
- **TD measurement:** Evaluates the cost/benefit relationship of known TD items in software or estimates the overall TD;
- **TD prioritization:** Adopts predefined rules to rank known TD items, to support the decision-making process;
- **TD prevention:** Establishes practices to avoid potential TD from being incurred;
- **TD monitoring:** Observes the evolution of known TD items over time;
- **TD repayment:** Eliminates or reduces the TD impact (principal and interest) in a software system;
- **TD representation/documentation:** Represents and codes TD in a predefined standard, to address the stakeholders' concerns;
- **TD communication:** Disclose the identified TD to the stakeholders.

What are the practices to manage the TD?
The following guidelines or practices were collected on a survey with practitioners from the Brazilian software industry (no participant on the survey answered practices to monitor the TD):

- **TD identification:** manual code inspection, dependency analysis, TD checklist;
- **TD representation/documentation:** TD backlog;
- **TD communication:** Discussion forums, TD meetings;
- **TD prioritization:** Cost/benefit analysis, classification of issues;
- **TD repayment:** Refactoring, redesign, code rewriting;
- **TD prevention:** Coding guidelines or standards, code revision, retrospective meetings, Definition of Done.

The following guidelines or practices were collected from experience reports or case studies with the industry, through a literature review:

13 steps for reducing TD
TDM activity: TD Prevention
TD types covered: Code TD; Architecture TD; Design TD
Source artifact: Any
Type of evidence: Case study
Reference: Krishna, V.; Basu, A. Minimizing Technical Debt: Developer's Viewpoint. International Conference on Software Engineering and Mobile Application Modelling and Development, Chennai, 2012.
4 stages to manage TD in legacy systems
TDM activity: TD Identification; TD Representation/Documentation; TD Prioritization; TD Repayment; TD Prevention
TD types covered: Code TD; Architecture TD; Design TD
Source artifact: Source code
Type of evidence: Case study
Reference: Gupta, R. K. et al. Pragmatic Approach for Managing Technical Debt in Legacy Software Project. 9th India Software Engineering Conference, Goa, 2016.
Approach to control and repay build debt
TDM activity: TD Identification; TD Monitoring; TD Measurement
TD types covered: Build TD
Source artifact: Specifications for building software
Type of evidence: Industry practices at Google
Reference: Morgenthaler, J. D. et al. Searching for build debt: Experiences managing technical debt at Google. 3rd International Workshop on Managing Technical Debt, Piscataway, 2012.
Proactive management of TD by software metrics
TDM activity: TD Measurement
TD types covered: Code TD; Architecture TD; Test TD
Source artifact: Source code
Type of evidence: Industry practices at Ericsson
Reference: Sandberg, A. B.; Staron, M.; Antinyan, V. Towards proactive management of technical debt by software metrics. 14th Symposium on Programming Languages and Software Tools, Tampere, 2015.
Strategies for repaying test debt
TDM activity: TD Repayment
TD types covered: Test TD
Source artifact: Source code
Type of evidence: Case study
Reference: Samarthyam, G.; Muralidharan, M.; Anna, R. K. Understanding Test Debt. Trends in Software Testing, Singapore, 2017
TD board to manage and visualize high level debt
TDM activity: TD Identification; TD Representation/Documentation; TD Monitoring; TD Prioritization; TD Communication
TD types covered: Not specific to a TD type
Source artifact: Source code
Type of evidence: Industry practices at Petrobras
Reference: dos Santos, P. S. M. et al. Visualizing and managing technical debt in agile development: An experience report. International Conference on Agile Software Development, Berlin, 2013.

Who is this briefing for?

Software engineering practitioners who want to make decisions about internal quality issues and apply scientific knowledge on managing technical debt.

Where does the information come from?

The information on this briefing comes from evidence collected by the author through a literature review and a survey with software practitioners in the Brazilian industry. The technical debt management activities were described according to the mapping study listed below:

- [1] Li, Z. et al. A systematic mapping study on technical debt and its management. Journal of Systems and Software, 2015.

For additional information about the Experimental Software Engineering Group at COPPE/UFRJ:

http://lens-ese.cos.ufrj.br/ese/

For additional information about the DELFOS Observatory:

http://www.delfos.cos.ufrj.br

RESEARCH REFERENCE

Silva, V. M. ***Technical Debt and its Management: Analysis and Application in the Brazilian Software Industry***. Masters' dissertation, Federal University of Rio de Janeiro, 2018.

Silva, V. M.; Jeronimo, H.; Travassos, G. H. ***A Taste of the Software Industry Perception of Technical Debt and its Management in Brazil***. XXI Ibero-American Conference on Software Engineering, Bogotá, 2018.

Figure 5.3 – EB2 – Technical Debt Management: Activities and Practices

TECHNICAL DEBT MANAGEMENT: TOOLS AND STRATEGIES

This evidence briefing reports a list of tools, methodologies, strategies, approaches or frameworks to execute the different Technical Debt (DT) management activities.

Technology	Description	Link	TD type	TDM activity	Source artifact	Evidence	Reference
JSpIRIT	A tool to identify and prioritize technical debt in the form of code smells.	goo.gl/dKnqvp	Code TD Architecture TD Design TD	TD Identification; TD Measurement; TD Prioritization	Source code	CS	[1]
SQALE plugin for SonarQube	A tool to analyze, measure, visualize and prioritize TD based on SQALE quality model.	goo.gl/3oiAys	Code TD Architecture TD	TD Identification; TD Measurement; TD Prioritization; TD Communication	Source code	SP	[2]
SonarQube	An open platform for managing code quality.	goo.gl/6X2KGV	Code TD	TD Identification; TD Measurement; TD Monitoring	Source code	SP	[2]
CheckStyle	A tool to check Java code against coding standards.	goo.gl/iRFBTr	Code TD	TD Identification	Source code	SP	[2]
FindBugs	A tool to identify TD using automatic static analysis.	goo.gl/UFssPz	Code TD	TD Identification	Source code	SP	[2]
JIRA	A tool that allows task monitoring and management.	goo.gl/ga4zMv	Any	TD Representation/Documentation; TD Measurement	N/A	SP	[2]
Trello	A tool that allows task monitoring and management.	goo.gl/iBF6oA	Any	TD Representation/Documentation; TD Communication	N/A	SP	[2]
GitLab	A software repository manager.	goo.gl/kHsxMi	Any	TD Communication	N/A	SP	[2]

CS - Case study

SP - Survey with practitioners

Technology	Description	TD type	TDM activity	Source artifact	Evidence	Reference
CVM-TD	Model to identify TD on code comments.	Code TD Architecture TD Design TD	TD Identification	Source code	IP	[3]
Not named	A decision-support system approach to the modularity debt management.	Design TD	TD Identification; TD Monitoring; TD Measurement; TD Prioritization; TD Repayment	Source code	EE	[4]
Not named	An approach to define manageable levels of technical debt.	Code TD Design TD Test TD	TD Identification; TD Monitoring; TD Measurement; TD Repayment	Source code	IP	[5]
Not named	An approach to quantify TD.	Code TD Design TD	TD Identification; TD Measurement	Source code	IP	[6]
AnaConDebt	A method that aid architects and managers to understand and quantify interest on architecture TD.	Architecture TD	TD Measurement; TD Prioritization	Source code	EE	[7]
Not named	A decision-based approach using a conceptual model of architecture TD.	Architecture TD	Not specific to a TDM activity	Any	EE	[8]
Not named	An identification approach based on architecture decisions and change scenarios.	Architecture TD	TD Identification	Any	EE	[9]
ATD viewpoints	An approach based on set of architecture viewpoints related to architecture TD.	Architecture TD	TD Representation/Documentation	Any	EE	[10]
Not named	A process for TD identification, documentation and prioritization.	Not specific to a TD type	TD Identification; TD Prioritization; TD Representation/Documentation	Any	EE	[11]
Not named	Methodology to help avoiding the acumulation of technical debt.	Code TD Architecture TD Design TD Documentation TD	TD Prevention	Test cases	EE	[12]
Not named	A normative process framework for managing technical debt in commercial software production.	Not specific to a TD type	Not specific to a TDM activity	Any	EE	[13]
SQALE	Method that defines additional indexes and indicators to analyze and understand TD.	Code TD Architecture TD Design TD	TD Identification; TD Measurement; TD Prioritization; TD Repayment; TD Communication	Source code	EE	[14]
TD Template	A framework to support technical debt management.	Not specific to a TD type	Not specific to a TDM activity	Any	EE	[15]
Not named	Visualization approach.	Code TD Design TD Test TD	TD Identification; TD Monitoring	Source code	IP	[16]
Duct taped TD	Visualization technique.	Code TD	TD Representation/Documentation; TD Communication	Source code	IP	[17]
CoBeTDM	A framework to manage and reduce TD.	Code TD Architecture TD	TD Identification; TD Monitoring; TD Prioritization	Any	IP	[18]

EE - Experimentally evaluated

IP - Industry practice

Who is this briefing for?

Software engineering practitioners who want to make decisions about internal quality issues and apply scientific knowledge on managing technical debt.

Where does the information come from?

- [1] Vidal, S. et al. Identifying Architectural Problems through Prioritization of Code Smells. SBCARS, 2016;
- [2] Silva, V. M. et al. A Taste of the Software Industry Perception of Technical Debt and its Management in Brazil. CIbSE, 2018;
- [3] de Freitas Farias, M. A. et al. Investigating the Use of a Contextualized Vocabulary in the Identification of Technical Debt: A Controlled Experiment. ICEIS, 2016;
- [4] Cai, Y. et al. A decision-support system approach to economics-driven modularity evaluation. Economics-Driven Software Architecture, 2014;
- [5] Eisenberg, R. J. A threshold based approach to technical debt. Software Engineering Notes, 2012;
- [6] Nugroho, A. et al. An empirical model of technical debt and interest. Workshop on Managing Technical Debt, 2011;
- [7] Martini, A. et al. An empirically developed method to aid decisions on architectural technical debt refactoring: AnaConDebt. ICSE-C, 2016;
- [8] Li, Z. et al. Architectural debt management in value-oriented architecting. Economics-Driven Software Architecture, 2014;
- [9] Li, Z. et al. Architectural technical debt identification based on architecture decisions and change scenarios. WICSA, 2015;
- [10] Li, Z. et al. Architecture viewpoints for documenting architectural technical debt. Software Quality Assurance, 2016;
- [11] Yli-Huumo, J. et al. Developing Processes to Increase Technical Debt Visibility and Manageability – An Action Research Study in Industry. PROFES, 2016;
- [12] Trumler, W. et al. How "Specification by Example" and Test-Driven Development Help to Avoid Technical Debt. Workshop on Managing Technical Debt, 2016;
- [13] Ramasubbu, N. et al. Integrating Technical Debt Management and Software Quality Management Processes: A Normative Framework and Field Tests. IEEE Transactions on Software Engineering, 2017;
- [14] Letouzey, J. L. et al. Managing technical debt with the sqale method. IEEE software, 2012;
- [15] Seaman, C. et al. Measuring and monitoring technical debt. Advances in Computers, 2011;
- [16] Kaiser, M. et al. Selling the Investment to Pay Down Technical Debt: The Code Christmas Tree. AGILE, 2011;
- [17] Chicote, M. Startups and technical debt: managing technical debt with visual thinking. International Workshop on Software Engineering for Startups, 2017;
- [18] Harun, M. F. et al. Towards a technical debt-management framework based on cost-benefit analysis. ICSEA, 2015.

For additional information about the Experimental Software Engineering Group at COPPE/UFRJ:

http://lens-ese.cos.ufrj.br/ese/

For additional information about the DELFOS Observatory:

http://www.delfos.cos.ufrj.br

RESEARCH REFERENCE

Silva, V. M. ***Technical Debt and its Management: Analysis and Application in the Brazilian Software Industry***. Masters' dissertation, Federal University of Rio de Janeiro, 2018.

Silva, V. M.; Jeronimo, H.; Travassos, G. H. ***A Taste of the Software Industry Perception of Technical Debt and its Management in Brazil***. XXI Ibero-American Conference on Software Engineering, Bogotá, 2018.

Figure 5.4 – EB3 – Technical Debt Management: Tools and Strategies