

CREST Workshop

Got Technical Debt? Surfacing Elusive Technical Debt in Issue Trackers

Stephany Bellomo, Robert Nord, Ipek Ozkaya,
Mary Poppeck

Nov 28-29th, 2016

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Distribution Statements

Copyright 2016 Carnegie Mellon University

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8721-05-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

NO WARRANTY. THIS CARNEGIE MELLON UNIVERSITY AND SOFTWARE ENGINEERING INSTITUTE MATERIAL IS FURNISHED ON AN “AS-IS” BASIS. CARNEGIE MELLON UNIVERSITY MAKES NO WARRANTIES OF ANY KIND, EITHER EXPRESSED OR IMPLIED, AS TO ANY MATTER INCLUDING, BUT NOT LIMITED TO, WARRANTY OF FITNESS FOR PURPOSE OR MERCHANTABILITY, EXCLUSIVITY, OR RESULTS OBTAINED FROM USE OF THE MATERIAL. CARNEGIE MELLON UNIVERSITY DOES NOT MAKE ANY WARRANTY OF ANY KIND WITH RESPECT TO FREEDOM FROM PATENT, TRADEMARK, OR COPYRIGHT INFRINGEMENT.

[Distribution Statement A] This material has been approved for public release and unlimited distribution. Please see Copyright notice for non-US Government use and distribution.

This material may be reproduced in its entirety, without modification, and freely distributed in written or electronic form without requesting formal permission. Permission is required for any other use. Requests for permission should be directed to the Software Engineering Institute at permission@sei.cmu.edu.

DM-0004232

Do issue trackers reveal technical debt?

- **RQ1:** Do developers use the term *technical debt* **explicitly** when discussing problems in their issue trackers?
- **RQ2:** Can **implicit** technical debt items be discovered systematically within issue trackers?
- **RQ3:** What are the distinguishing **characteristics** of technical debt items discovered in issue trackers?

Overview of Data Sets

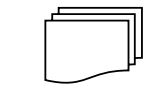
	Data set	Source	Filter criteria	# Records analyzed
Setup (instrument development)	Chromium	Google issue tracker	Text search “technical debt”	56
	Connect	Jira	Text search “technical debt”	15
	Technical debt survey	Examples (as text)	N/A	265
Phase 1 TD categorization	Connect	Jira	2012, first 200 records	200
Phases 2–4 TD classification, analysis, and evaluation Total: 727 issues	Connect	Jira	March 2012	286
	Project A	Jira	Defects/CRs Sep. 2010 to Dec. 2014	86
	Project B	FogBugz	All year 2013	193
	Chromium	Google issue tracker	Milestone 48 Stars (watchers) > 3	163
Total				1,264

- Initial phased focused on exploring RQ1 (explicit declaration) and survey examples
- Core research phases 1-4
- Mix of open source and project data
- Created manageable sized data sets for manual analysis

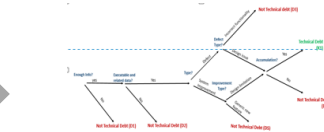
Data sets are also published

Multi-phased analysis approach

1. Categorization: Extract reoccurring concepts from samples; create initial categorization

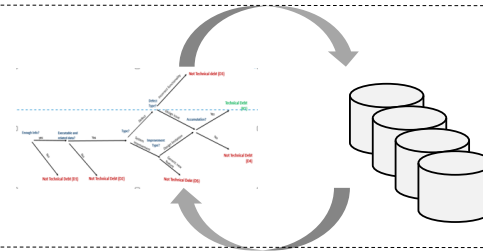


Examples



Output: Classification guidance

2. Classification: Systematically classify data sets using categorization



Outputs: Classified data set, refined classification guidance

3. Evaluation: Validate effectiveness of classification with project stakeholders

Are these really TD?



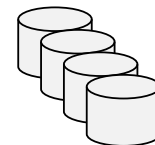
Does the classification make sense?



Output: Stakeholder confirmation of findings

4. Analysis: Analyze the technical debt items for characteristics

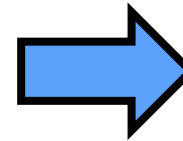
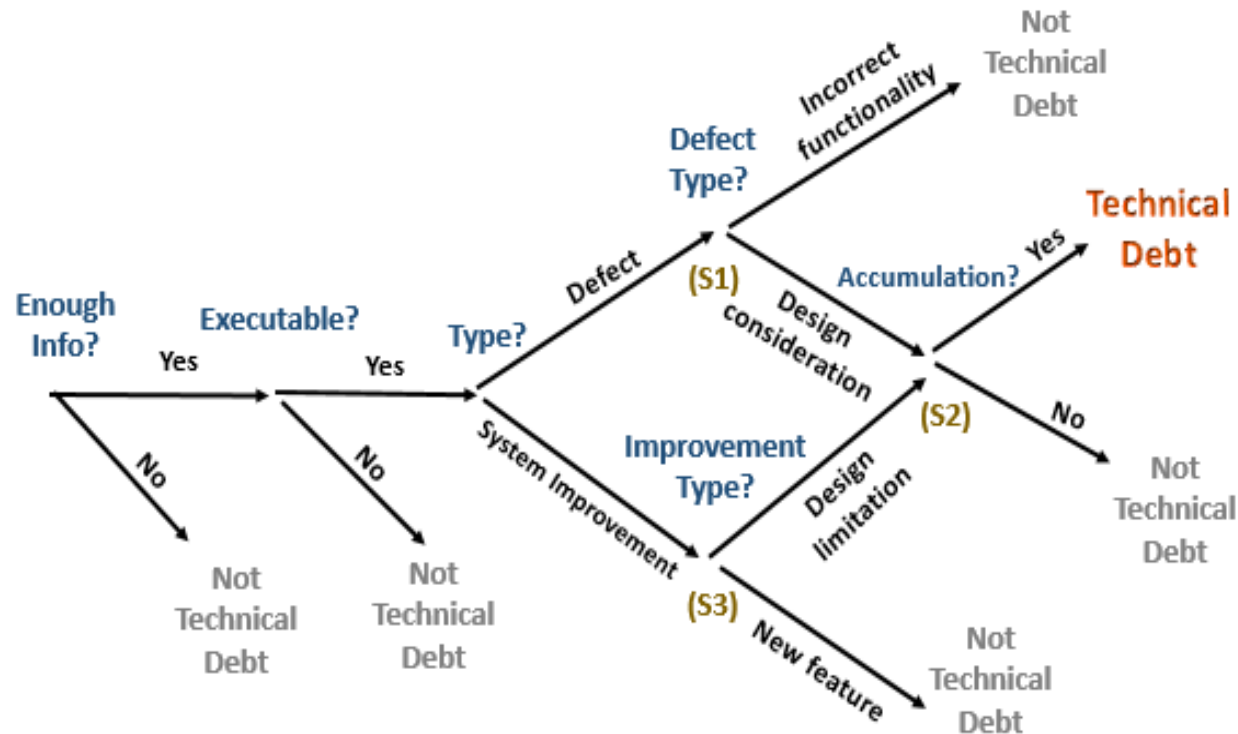
Categorized data sets



Analyze

Outputs: Demographic statistical analysis; unstructured data affinity grouping and analysis

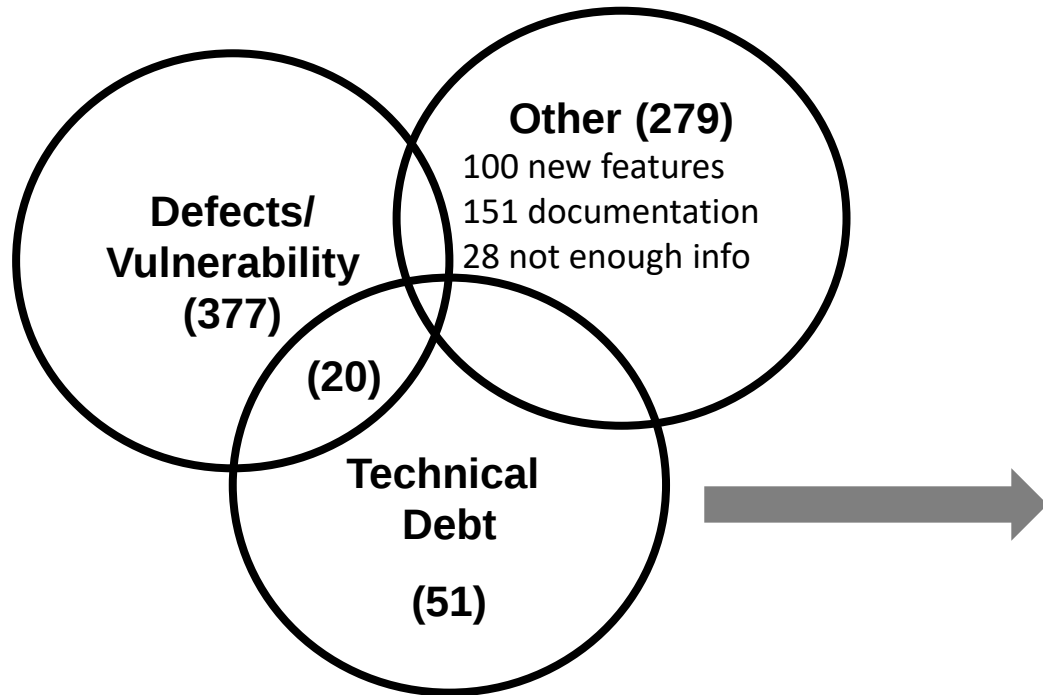
Technical Debt Classification Rules (Described as a Decision Tree)



Project	TD	Not TD	Stuck	No agreement	Total
Connect	12	265	1	7	285
Project A	10	74	1	1	86
Project B	13	171	9	0	193
Chromium	16	146	1	0	163
Total	51	656	12	8	727

- In current project, we are using method with larger datasets and machine learning

Technical Debt Breakout



Deployment & Build	Out-of-sync build dependencies	3	CN
	Version conflict	1	CN
	Dead code in build scripts	1	CN
Code Structure	Event handling	5	2CH, 3PB
	API/Interfaces	5	2CH, 1CN, 2PB
	Unreliable output or behavior	5	4CH, 1PA
	Type conformance issue	3	CN
	UI design	3	PB
	Throttling	2	1CH, 1PB
	Dead code	2	CN
	Large file processing or rendering	2	CH
	Memory limitation	2	CH
	Poor error handling	1	PA
	Performance appending nodes	1	CH
	Encapsulation	1	PB
Data Model	Caching issues	1	CN
	Data integrity	6	PA
	Data persistence	3	PB
	Duplicate data	2	PA
Regression Tests	Test execution	1	CH
	Overly complex tests	1	CH

CH = Chromium, PA = Project A, PB = Project B, CN = CONNECT

Examples

Not Technical Debt

[Project A #25] Correct the values for subsystem A to reflect the subsystem B values

[Project B #265] Update alert authoring UI – ‘event window’ should be close to ‘any rule’ checkbox

[Project B #1513] Refactor onclicks in nodes.html into query events

Technical Debt

[Project A #18] approximately 340 records exist in the database twice ... so much time had elapsed in some cases the duplicate was endorsed.

[Chromium #367158] Currently, we have a lot of duplicate/boilerplate code in this test. We should try to simplify this test so that it's easier to maintain and read.

Example of a Technical Debt Item

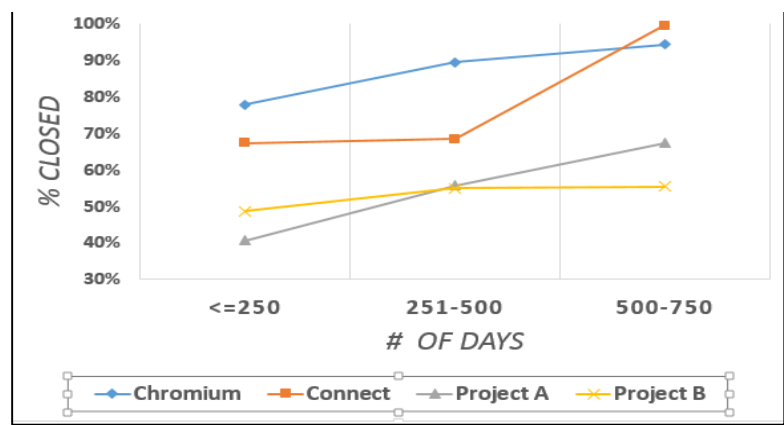
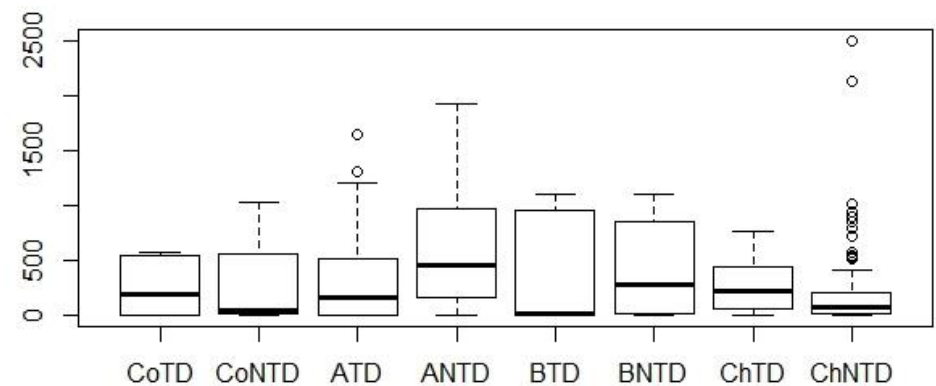
Name	Connect #Gateway-1631: Empty Java package (dead code)
Development artifact	The re-architecture of the source code to support multiple NwHIN specifications has introduced a new Java packaging scheme.
Symptoms	Numerous empty Java package folders present across multiple projects.
Consequences	No impact to functionality; however, may lead to confusion for users implementing enhancements or modifications to the source code.
Analysis	New and existing classes have been moved into these new package folders; however, the previous package folders have been left in place with no class files.

Suggested
template for
capturing
Technical Debt
Item

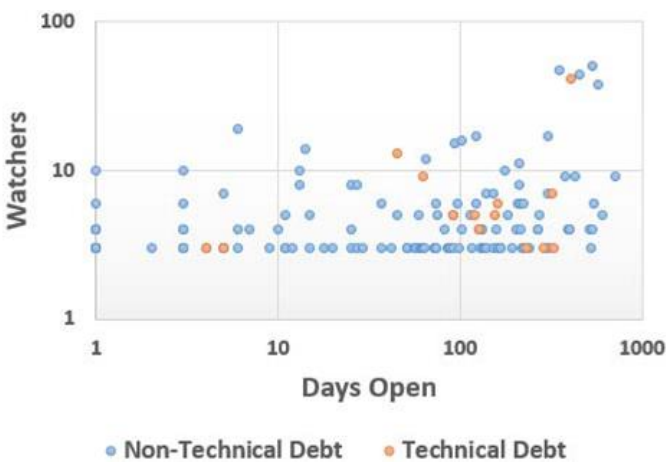
Our Assertion: Technical debt can be made **visible earlier** when tracked similarly to defects, consequently managed more effectively and strategically

RQ3: Are there any quantifiable characteristics

Are TD issues open longer?



Do TD issues generate more developer discussion?



Do TD issues have higher priority?

	Priority 1	Priority 2	Priority 3
Technical Debt Issues	22%	56%	22%
Not Technical Debt Issues	24%	50%	26%

Our Emerging Definition of Technical Debt

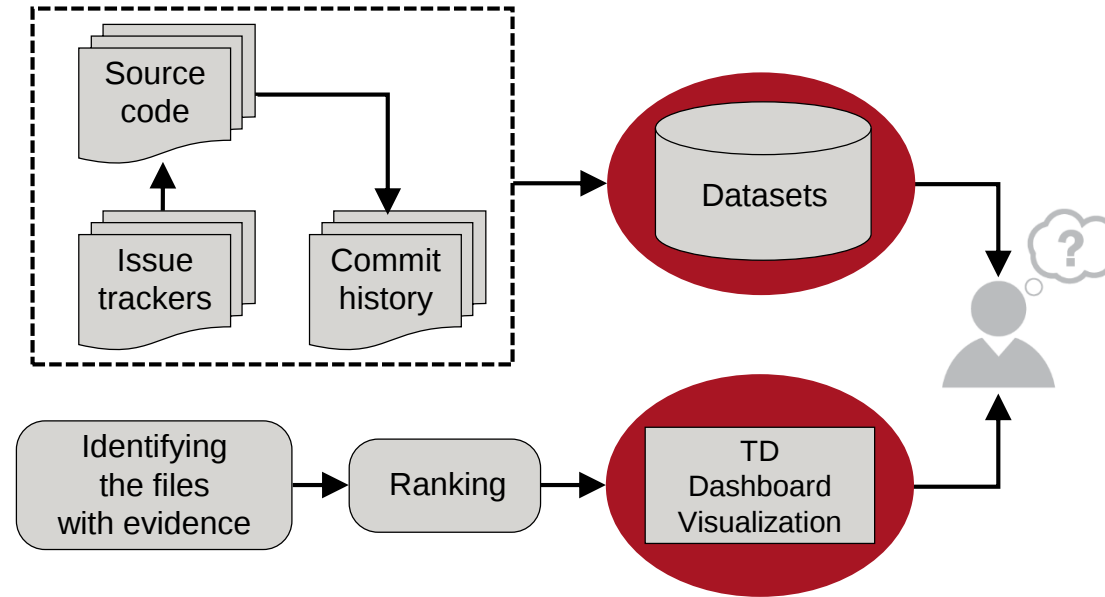
Technical debt is design work relating to **software units** that have evidence of present or anticipated accumulation of extra work.

- Exists in an **executable system artifact**, such as code, build scripts, automated test suites;
- Is traced to **several locations** in the system, implying ripple effects of impact of change;
- Has a **quantifiable** effect on system attributes of interest to developers, such as increasing number of defects, negative change in maintainability and code quality indicators are symptoms of technical debt.

Summary of Findings

- Using this method we manually identified 51 examples of technical debt records in several issue tracker datasets.
- Existing definitions focus on the explicit shortcuts, however, the issues we found are mostly implicit - **result of unintentional design choices.**
 - We presented an emerging definition from our work.
- We found no searchable characteristics when we analyzed the technical debt records.
 - Consequently, text analysis is necessary.
- We observed developers **do not identify the consequences** of technical debt in issue trackers
 - Suggested a template for improving this.

Future Vision: Towards Technical Debt Analytics



Problem: Managing the consequences of technical debt relies on an ability to (1) identify unintentional decisions and (2) quantify the consequences of such decisions.

Solution: Develop tools that integrate data from multiple, commonly available sources to surface problematic decisions and quantify consequences

Approach: Combine techniques from machine learning, code analysis, and data mining to identify problematic design issues.

Q&A