Detecting Discussions of Technical Debt

Ipek Ozkaya

(in collaboration with Zach Kurtz and Robert Nord)

December 10, 2018

ISR SSSG

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Document Markings

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8702-15-D-0002 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

DM18-1398

Agenda

What is technical debt?

Why are we studying it at the SEI?

Developing a *techdebt* classifier

Summarize our research in technical debt

Discussion

What is Technical Debt?

A decade ago processors were not as powerful. To optimize for performance we would not insert code for exception handling when we knew we would not divide by zero or hit an out of bounds memory condition. These areas now are hard to track and have become security nightmares.

Technical debt is a software design issue that:

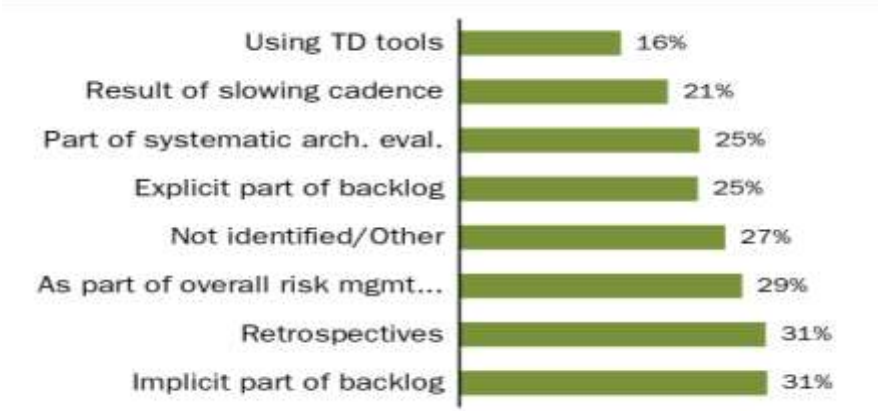
- Exists in an **executable system artifact**, such as code, build scripts, data model, automated test suites;
- Is traced to **several locations** in the system, implying issues are not isolated but propagate throughout the system artifacts.
- Has a **quantifiable** effect on system attributes of interest to developers (e.g., increasing defects, negative change in maintainability and code quality indicators).

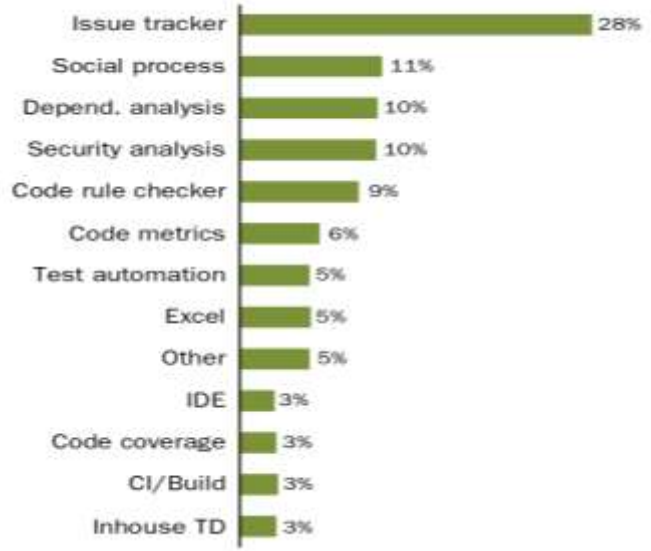
Common Consequences of Technical Debt

- Teams spend almost all of their time fixing defects, and new capability development is continuously slipping.
- Integration of products built by different teams reveals that incompatibilities cause many failure conditions and lead to significant out-of-cycle rework.
- Progress toward milestones is unsatisfactory because unexpected rework causes cost overruns and project-completion delays.
- Recurring user complaints about features that appear to be fixed.
- Out-dated technology and platforms require length convoluted solutions and added complexity in maintaining or extending the systems.

State of Practice – Current tools do not capture technical debt accumulation

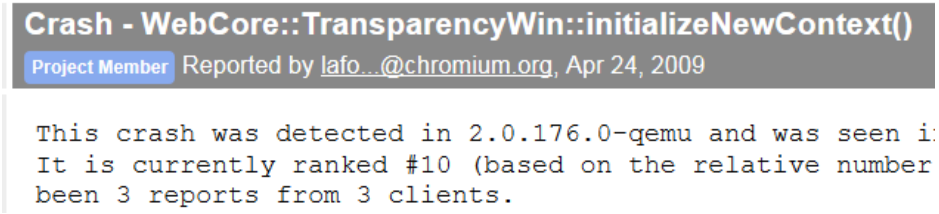
Results of surveying over 1800 developers about their perceptions on technical debt revealed that technical debt management is ad hoc at best and issue trackers dominate in communicating technical debt.





"Measure it? Manage it? Ignore it? Software Practitioners and Technical Debt" N. Ernst, S. Bellomo, I. Ozkaya, R. Nord, I. Gorton, FSE 2015

Technical Debt Conversations



10977: Crash due to large negative number

"*We could just fend off negative numbers near the crash site or we can dig deeper and find out how this -10000 is happening.*"

"Time permitting, I'm inclined to want to know the root cause. My sense is that if we patch it here, it will pop up somewhere else later."

"There have been 28 reports from 7 clients… 18 reports from 6 clients."

"Hmm ... reopening. The test case crashes a debug build, but not the production build. I have confirmed that the original source code does crash the production build, so there must be multiple things going on here."

Do issue trackers reveal technical debt?

	Data set	Source	Filter criteria	# Records analyzed
Technical debt 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	M(ilestone): 48 Stars (watchers) > 3	163

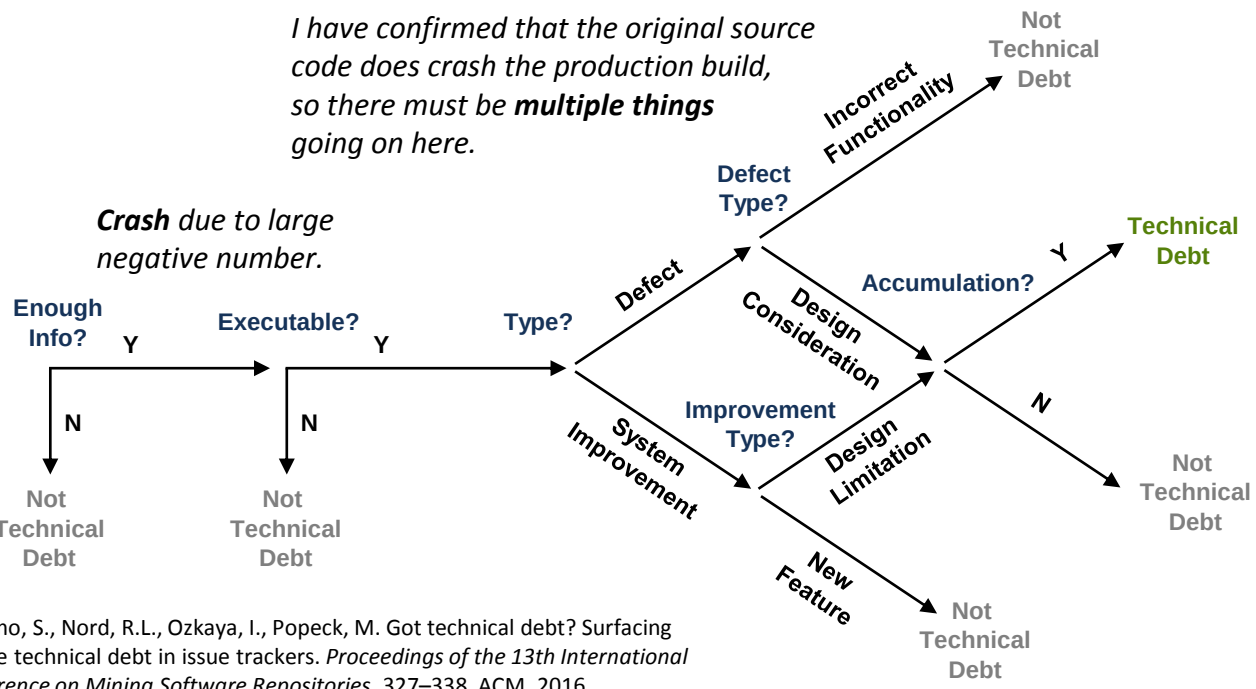
- Do developers use the term *technical debt* explicitly when discussing issues and tasks in their issue trackers?
- Can technical debt items be discovered systematically within issue trackers?
- What are the distinguishing characteristics of technical debt items discovered in issue trackers?

Technical Debt Tag

*Time permitting, I'm inclined to want to know the **root cause**.*

*I have confirmed that the original source code does crash the production build, so there must be **multiple things** going on here.*

***Crash** due to large negative number.*



*There have been **28 reports** from 7 clients… **18 reports** from 6 clients*

*My sense is that if we patch it here, it will **pop-up somewhere else later**.*

*hmm … **reopening**. the test case crashes a debug build, but not the production build.*

Bellomo, S., Nord, R.L., Ozkaya, I., Popeck, M. Got technical debt? Surfacing elusive technical debt in issue trackers. *Proceedings of the 13th International Conference on Mining Software Repositories*, 327–338. ACM, 2016.

Related Work

Using software issue trackers effectively can improve issue resolution time:

> Herzig, K., Just, S., and Zeller, A. ICSE 2013. “It’s not a bug, it’s a feature”
>
> Zimmermann, T., Premraj, R., Bettenburg, N., Just, S., Schröter, A., and Weiss, C. TSE 2010. “What makes a good bug report?”

Relationship of defects and vulnerabilities is an active area of research:

> Camilo, F., Meneely, A., and Nagappan, M. MSR 2015 – Do bugs foreshadow vulnerabilities?

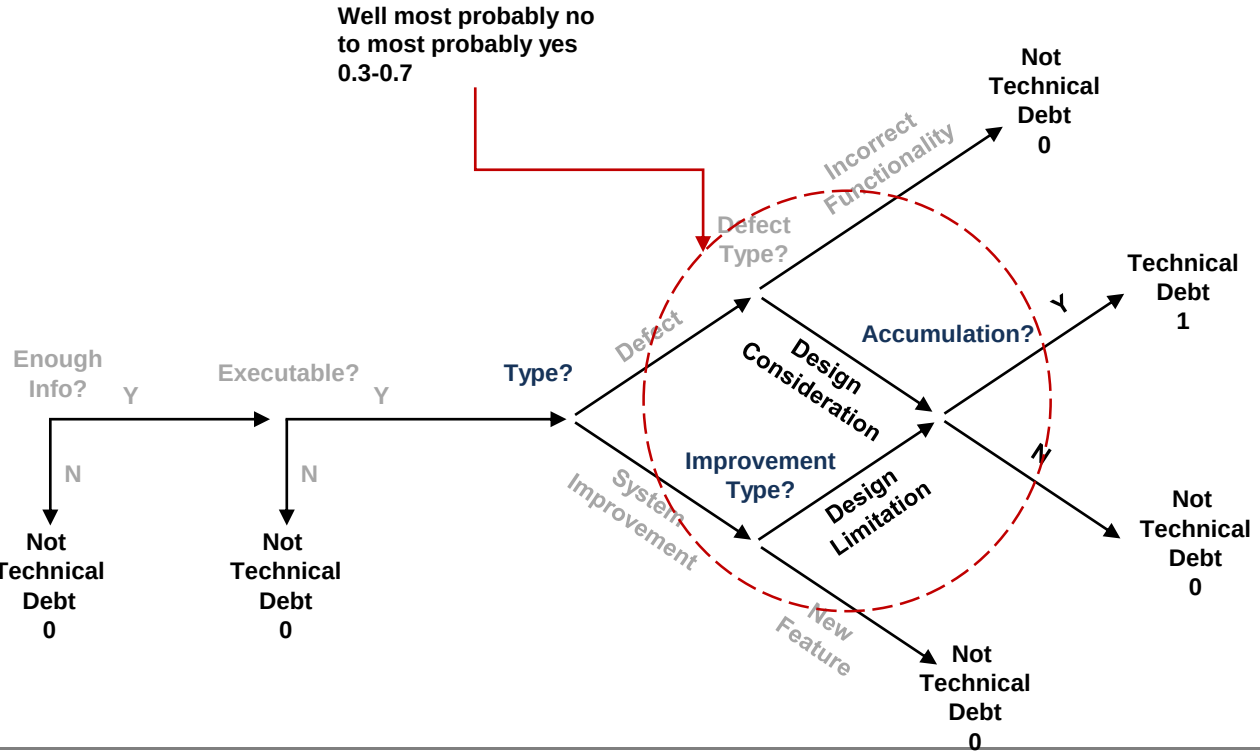
The concept of self-admitted technical debt has been studied using developer comments in code, e.g. “fix-me” “this is a hack”

> Potdar, A. & Shihab, E. ICSME 2014, Bavota, G. & Russo, B. MSR 2016

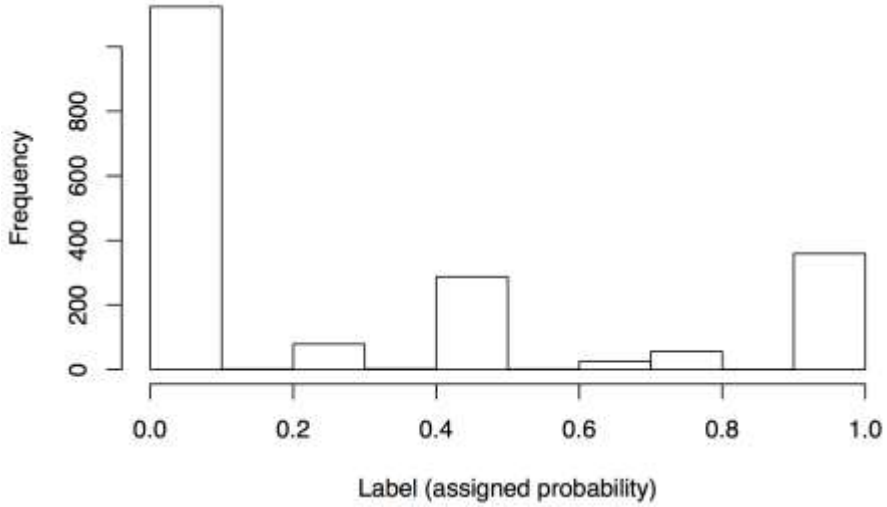
Developing a Classifier

- modeling with boosting algorithms (LightGBM) to build the weighted average of many classification trees – iteratively improving weak classifiers and creating a final strong classifier
- active learning pipeline and iterating over the data set to use 1,934 labeled technical debt examples
- feature engineering to combine discussion length, *n*-grams, key phrases, concepts, and document context

Developing a Classifier – Probabilistic Labelling



Distribution of manual labels of 1934



Initial batch of 163 tickets labeled with multiple raters with an inter-rater agreement of 90%

Developed an active learning pipeline, used the classifier to select the next set of features to label

Feature engineering

Ticket meta-data: status (duplicate, verified, fixed, wont fix…), authorship (to focus on developers on project), priority, type

Counts: length of comments

Key phrases: “debt,” “hack,” “workaround,” “cleanup,” “clean-up,” “clean up,” “give up,” “problematic,” “not up to date,” “inconsisten” [sic], “short term,” “deviate,” “tweak,” “mess,” “buggy,” “complex,” “doesn’t work,” “out of date,” “insufficient,” “rework,” “remove,” “redesign,” “refactor,” “depend,” and “structure.”

N-grams for n= 1, 2, 3.

Concept words: “deviate,” “outdated,” “redundant,” “redesign,” “decouple,” “complicated,” “regret,” “corrupt,” “horrible,” and “delay.”

Word and document vectors using gensim implementation of word2vec and doc2vec

Performance metrics

Using Chromium project with 475,000 issues

	Accuracy*	Precision*	Recall*	AUROC*
no TD	0.90	NA	0.00	0.50
keyphrase query	0.83	0.26	0.35	0.62
main model	0.87	0.40	0.62	0.88

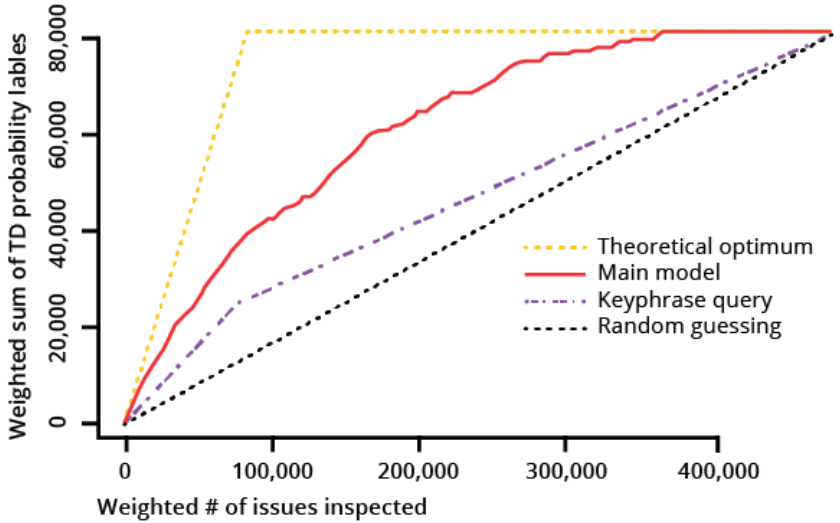


TABLE I
FEATURE TYPES RANKED BY IMPORTANCE

Rank	Feature Type	Number of Features	Total Gain
1	word vectors	19	0.407
2	key phrases	9	0.262
3	counts	6	0.184
4	concept words	10	0.072
5	document vectors	15	0.070
6	priority	1	0.003
7	author	2	0.002

Examples – Identified organically by developers

[Chromium #243948] **Paying off technical** debt becomes a higher priority, not lower, when in those rare cases it must be deferred. Tests are not a ‘nice to fix’ feature. Raising to Pri-1.

[Chromium #43780] One might consider **this a technical debt paydown bug.** However, feel free to reprioritize…. Backup sockets were committed conditionally on them being refactored to the “right” place (10/18/2010)… Looks like the statements about the code are still true. (08/17/2017)

Examples – Labelled by our classifier

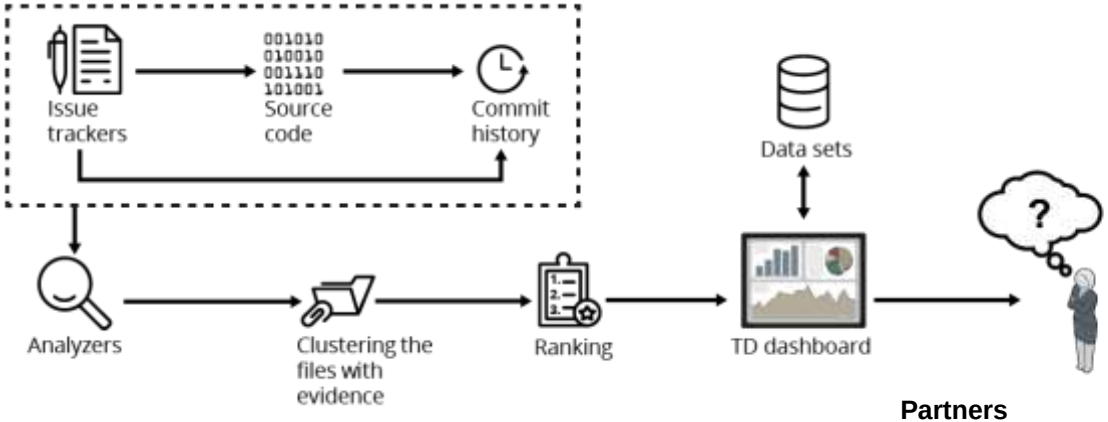
[Chromium 507796] This is just a first step to make sure the code is being exercised. It's been tested locally but only on this configuration. Some more work might be needed to get this working in non-GN builds. **Further refactoring of the Telemetry dependencies will occur in follow-on CLs.…Unfortunate that that build breakage wasn't caught.** … let me know if you have **any trouble diagnosing** what went wrong. **I don't know why so many of the other isolates would complain about crashpad_database_util not having been built**.

Automating the Identification of Technical Debt

Developed a TD classifier using machine learning:
Our classifier estimates at least 16% of developer discussions are related to technical debt (data from Chromium open source issues)

Created design violation analysis augmenting an open-source static code analyzer:
Our algorithm assists in focusing on problematic files, reducing the space of investigation by about 95%

Prioritized candidate technical debt items with supporting evidence
Development teams agree with 80% of the prioritized items as representing technical debt



Partners

- U.S. Air Force Life Cycle Management Center
- U.S. Food and Drug Administration

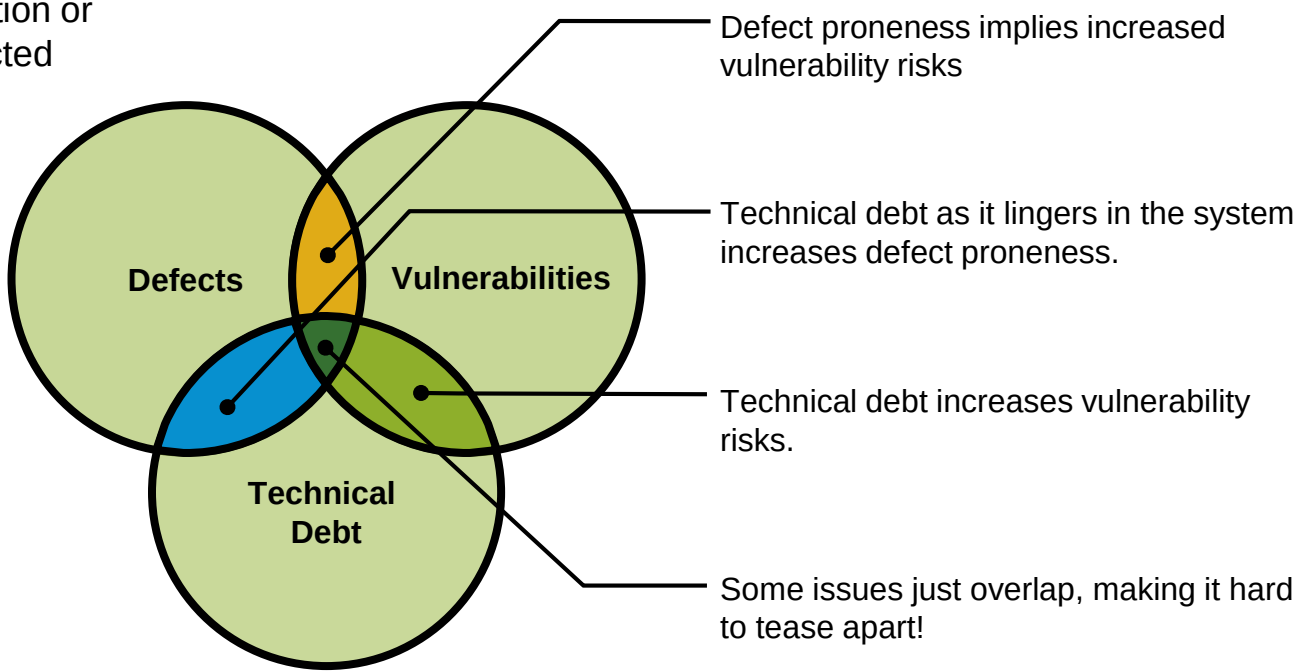
Why do we need a new term?

defect – error in coding or logic that causes a program to malfunction or to produce incorrect/ unexpected results

vulnerability – system weakness in the intersection of three elements:

- system flaw,
- attacker access to the flaw,
- attacker capability to exploit the flaw

technical debt – design or implementation construct traced to several locations in the system, that make future changes more costly



Upcoming May 2019