Variability and Technical Debt Ownership

Ipek Ozkaya

September 13, 2022

TD for Variability-Intensive Systems Workshop

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

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.

DM22-0792

About me

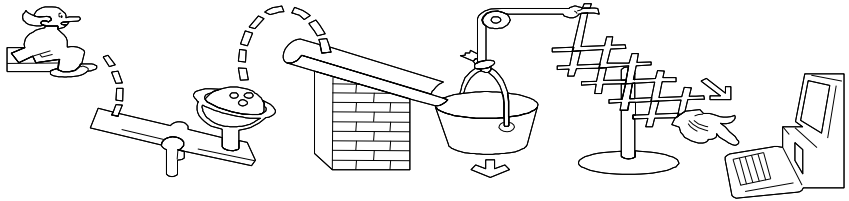
Carnegie Mellon University

The Storyline of Variability

"... and then we'll be able to construct software systems by picking out parts and plugging them together, just like Tinkertoys."

reality that you tell to your boss who is asking why it did not work

…It's more like having a bathtub full of Tinkertoys, Legos, Lincoln Logs, and pieces of six other incompatible kits and then picking out parts that fit specific functions and expecting them all to fit together."



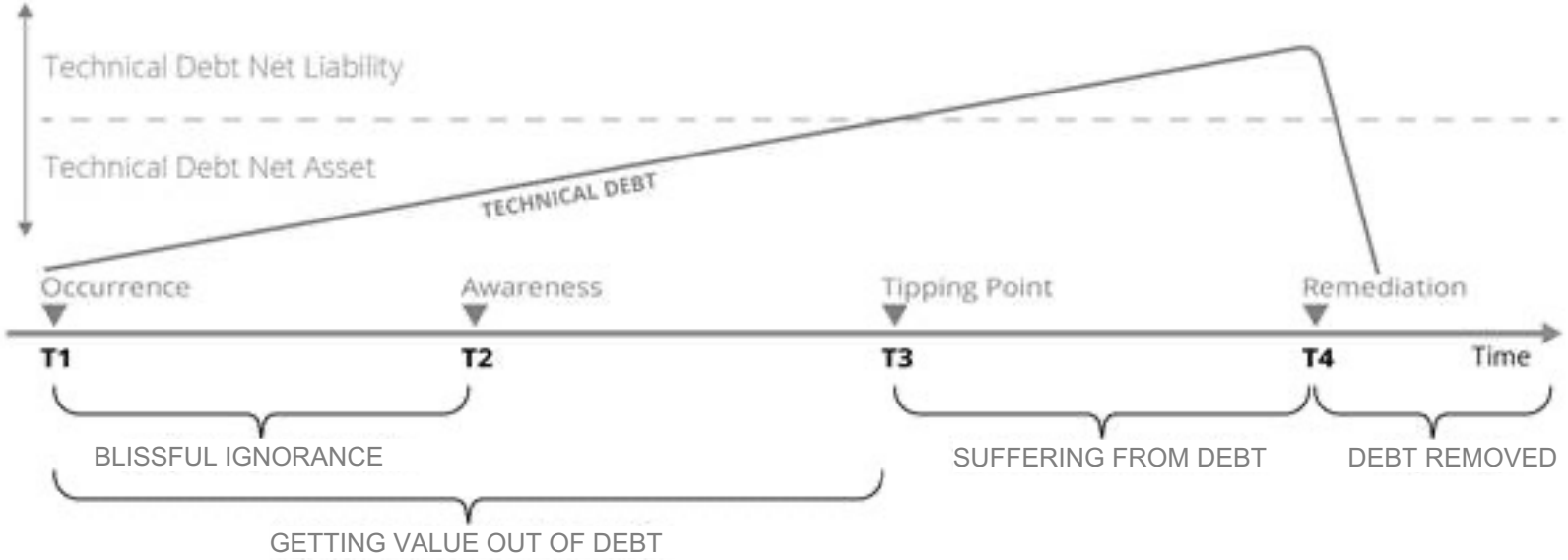
the reality the team is painfully aware, but cannot get ahead of

…It's more like having a bathtub full of some clearly ***broken, some brand new, some looking brand new but with a hidden chip*** *Tinkertoys, Legos, Lincoln Logs,* ***Magformers, Magnetic Tiles,..*** *and pieces of six****teen*** *other incompatible kits* ***some of which are hand me downs, each owned by a different kid, and bought from different stores****, and then picking out parts that fit specific functions and expecting them all to fit together and at the last minute* ***remembering that Toy'R Us is now bankrupt****."*

Looking Back 10 Years of Technical Debt Research!

- We know that all systems have technical debt!
- We know how to define technical debt (maybe?)
- We know there is not one size fits all in quantifying technical debt.
- We know managing technical debt is a continuous process.
- We still don't go beyond saying technical debt management requires collective ownership.

Technical Debt Timeline per TechDebt Item



Fast Forward Technical Debt

"[Contractor] developed our software tool and delivered the code to the government for maintenance. The code was poorly designed and documented therefore there was a very long learning curve to understand the endless number of variability points, they were overdone which we paid for. We continue to band aide over 1 million lines of code under the maintenance contract."

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.

The team stopped contributing to the core assets; therefore instead of making the changes in one place all features had to implement the changes which became a source of very hard to trace inconsistencies.

Variability degrades without ownership

We will inherit someone else's technical debt more often than not.

- Software product lines are likely to be more prone to this propagation.

Variability when not done right, propagates technical debt.

As software changes hands, variability induced technical debt will grow.

Whose Debt is It?

The technical debt timeline can enable decoupling responsibilities by addressing the following questions:

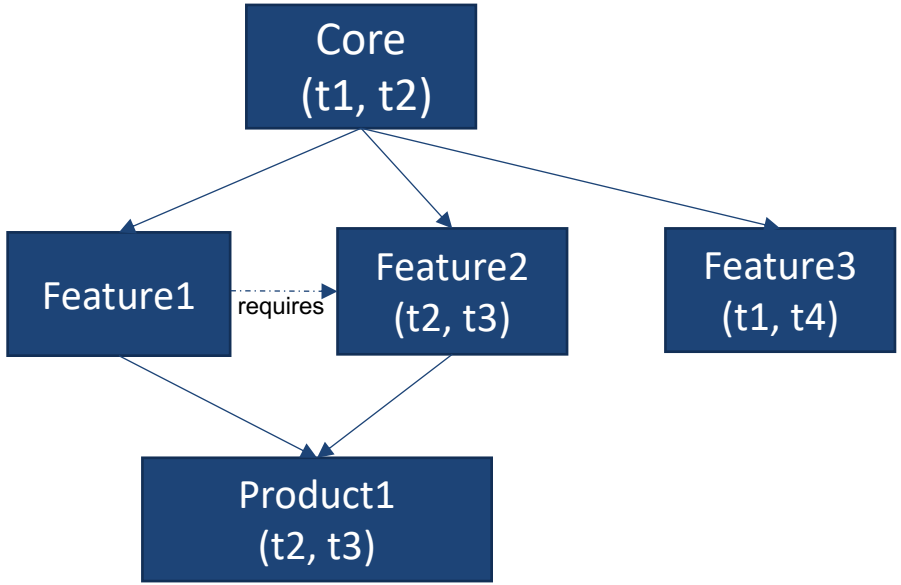
- Who is responsible at each point?
- How does variability change between the points in the technical debt timeline?
- How does technical debt propagate through feature configurations?

Variability Points Impact TechDebt Propagation

Technical debt is often discovered during product configuration but often fixed at feature level.

Technical debt behavior in variability intensive systems is similar to how defects propagate and are owned.

Feature models provide a powerful tool to model technical debt ownership and propagation in variability intensive systems.



Technical Debt and Variability Management are Contractual

Include language on how technical debt will be managed in contracts, including

- Percentage of resources to be withheld until high priority technical debt is resolved
- Data to be shared throughout the development life cycle
- Ongoing analysis to be conducted and its results shared
- Incentives to share technical debt the contractor takes on

Include technical debt discussions as part of assessments; request use of both appropriate software quality tools and architecture reviews.

Request evidence from contractors and continuously assess where you are on the technical debt timeline

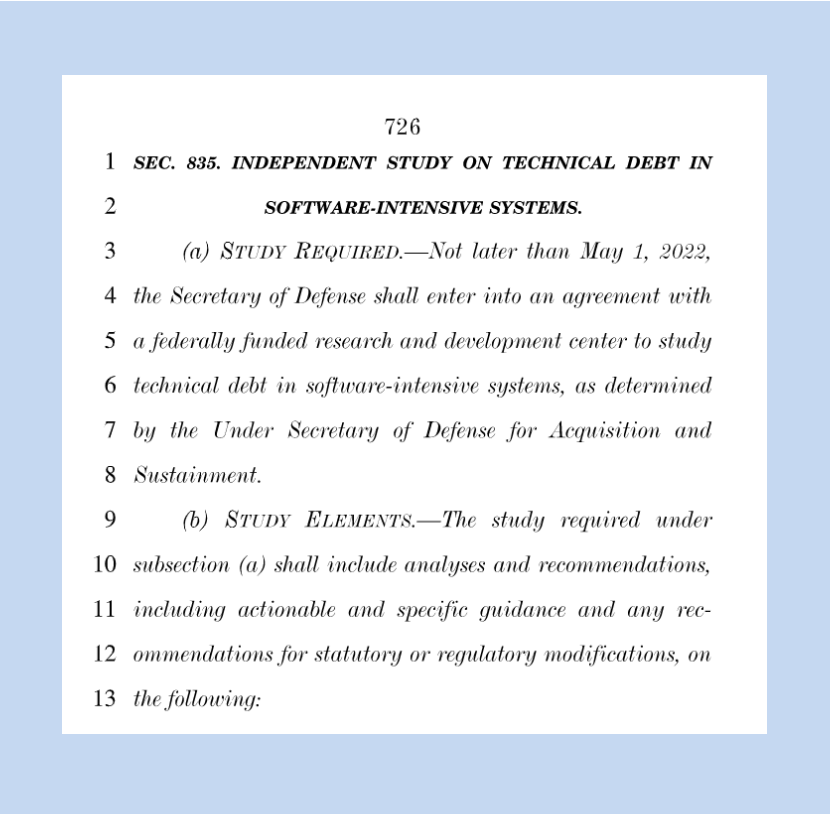
- Helpful data includes commit histories, defect logs, testing results, architecture conformance measures, and software quality analyses

TD4VIS

A timely topic because owners and developers of major software product lines are paying attention now, e.g. US DoD NDAA Sec 835 Study.

A timely topic because industry who develops and maintains major product lines is paying attention now, e.g. automotive industry.

Feature modeling and variability management provides concrete analysis opportunities ➔feature scope is smaller than system scope



726

SEC. 835. INDEPENDENT STUDY ON TECHNICAL DEBT IN SOFTWARE-INTENSIVE SYSTEMS.

(a) STUDY REQUIRED.—Not later than May 1, 2022, the Secretary of Defense shall enter into an agreement with a federally funded research and development center to study technical debt in software-intensive systems, as determined by the Under Secretary of Defense for Acquisition and Sustainment.

(b) STUDY ELEMENTS.—The study required under subsection (a) shall include analyses and recommendations, including actionable and specific guidance and any recommendations for statutory or regulatory modifications, on the following:

THANK YOU!

Contact Information

Ipek Ozkaya, PhD
Technical Director
Engineering Intelligent Software Systems
Software Solutions Division
Carnegie Mellon University
Software Engineering Institute

email: ozkaya@sei.cmu.edu

For more on technical debt:

https://www.sei.cmu.edu/research-capabilities/all-work/display.cfm?customel_datapageid_4050=6520

For more on software architecture:

http://www.sei.cmu.edu/architecture/