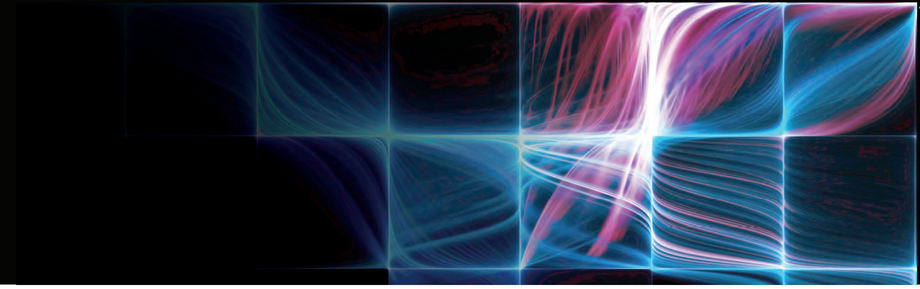
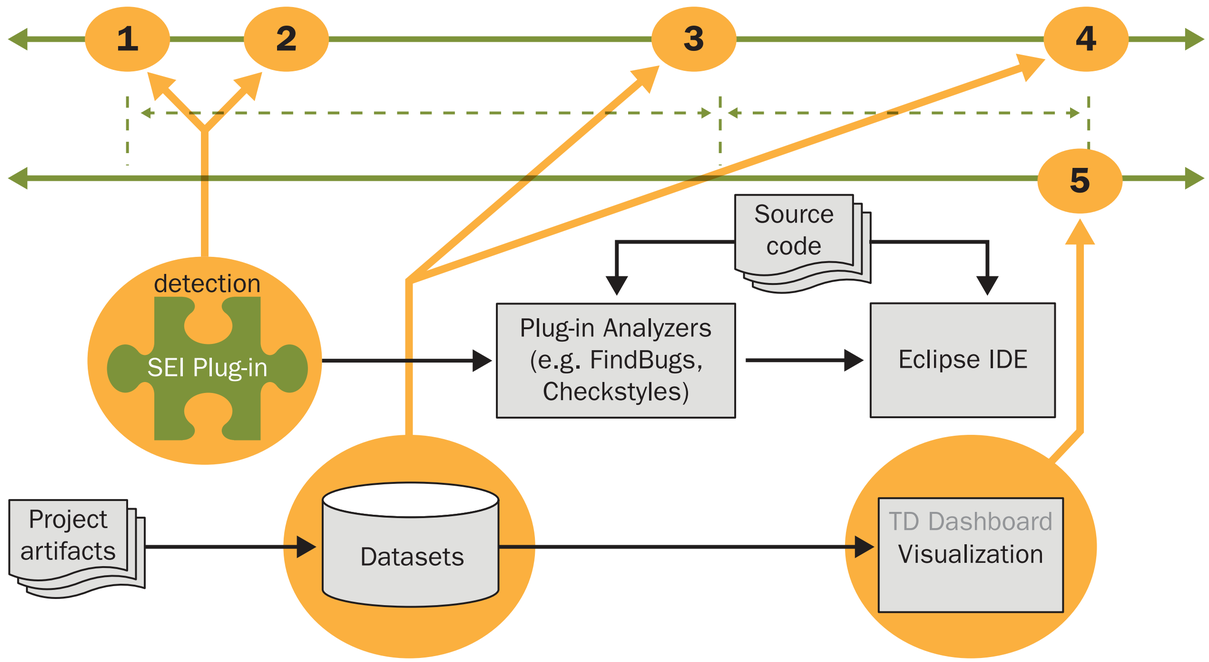
Improving Software Sustainability through Data-driven Technical Debt Management



Technical debt conceptualizes the tradeoff between the short-term benefits of rapid delivery and long-term value. In an effort to manage budget constraints, the DoD is increasingly searching for tool-supported approaches to manage technical debt. The goal of this project is to develop suite of tools and techniques for detecting and visualizing technical debt and provide exemplar data sets.

The technical debt metaphor is widely used to encapsulate numerous software quality problems. The metaphor is attractive to practitioners as it communicates to both technical and non-technical audiences that if quality problems are not addressed, things may get worse. However, it is unclear whether there are practices that move this metaphor beyond a mere communication mechanism. Existing studies of technical debt have largely focused on code metrics and small surveys of developers. Here, we report on our survey of 1,831 participants, primarily software engineers and architects working in long-lived, software-intensive projects from three large organizations, and follow-up interviews of seven software engineers. We analyzed our data using both non-parametric statistics and qualitative text analysis. We found that architectural decisions are the most important source of technical debt. Furthermore, while respondents believe the metaphor is itself important for communication, existing tools are not helpful in managing the details. We use our results to motivate a technical debt timeline to focus management and tooling approaches.

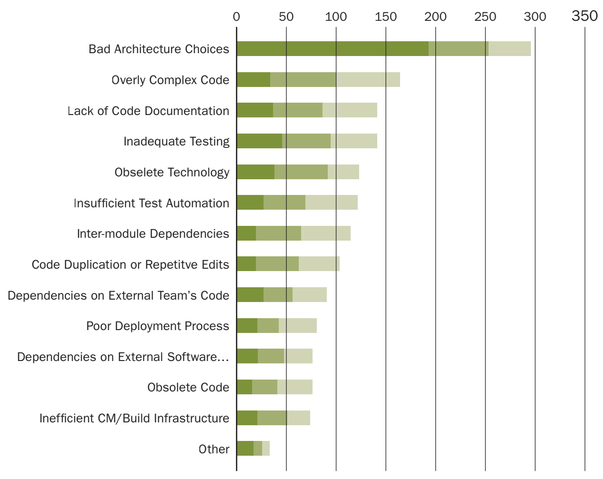


Technical debt timeline:
1: technical debt is incurred; 2: technical debt is recognized; 3: plan and re-architect; 4: technical debt is paid-off; 5: continuous monitoring

Our approach includes

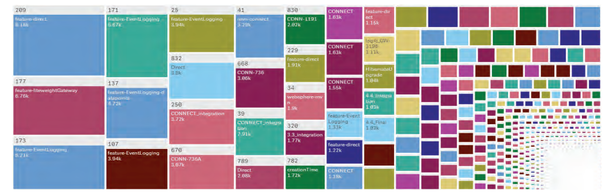
1. Codify known architectural sources of technical debt that are not addressed adequately by today's code-oriented tools (e.g., safety-critical testing partitioning, unbalanced modules, dependency violations)
2. Identify architecture indicators through abstractions (e.g., interfaces, restrict compositional dependencies) and anti-patterns that are correlated with technical debt and can be automatically identified by analyzing source code and other project artifacts.
3. IIntegrate these architectural indicators with code indicators in an experimental prototype.
4 Conduct empirical studies over multiple releases of at least two systems to correlate the identified indicators with observable project measures such as cost to fix, cost to implement new features, and defects.

Most significant technical debt is architectural according to developers.



Neil A. Ernst, Stephany Bellomo, Ipek Ozkaya, Robert L. Nord, Ian Gorton: Measure it? Manage it? Ignore it? software practitioners and technical debt. SEC/SIGSOFT FSE 2015: 50-60 ACM SIGSOFT Distinguished Paper Award

Our findings are consistent with our approach. Architecture choices are key sources of technical debt. Architectural issues are difficult to deal with, since they were often caused many years previously. Monitoring and tracking drift from original design and rationale is vital, but tools do not capture the key areas of accumulating problems in technical debt. Technical debt is most useful when discussed in the context of executable system artifacts (such as code, automated test suites, build scripts). We identify areas of code that have multiple maintainability issues correlated with increased number of defects and changes, and we zoom in on those that have more accumulation compared to others.



- 75% of the respondents said that dealing with the consequences of technical debt has consumed a painful chunk of project resources.
- Tooling is a necessary component of any technical debt management strategy. But developers mostly rely on only issue trackers.

The SEI Architecture Practices team has been a pioneer in advancing the research agenda regarding technical debt. You can collaborate with us by

- **Contributing your technical debt examples**
- **Sharing observed gaps in tools to manage technical debt**
- **Collaborating on in-depth analysis of your project and sharing your data**

Contact: Ipek Ozkaya ozkaya@sei.cmu.edu

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.

DM-0002854