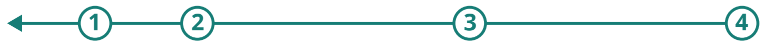
Carnegie Mellon University
Software Engineering Institute

Managing Technical Debt in Complex Software Systems

WHAT IS TECHNICAL DEBT? *Technical debt* can be defined as a design or construction approach that is expedient in the short term but that creates a technical context in which the same work will cost more to do later than it would cost to do now. If managed well, some debt can accelerate design exploration. Left unrecognized and unmanaged, accumulated technical debt results in increased development and sustainment costs.

The Technical Debt Lifecycle

This simple definition of technical debt indicates the role that time plays. Technical debt matters only as time flows and we want to evolve the software system. If the system never evolves, we never have to pay interest, so technical debt would not matter. Therefore, it is important to understand the technical debt lifecycle.



1. Occurrence: The time when technical debt is taken on, for good or bad reasons. For example, rather than investing in identifying common services, developers copy and modify code.
2. Awareness: The time when technical debt is recognized and made visible, when the organization becomes aware of it. Ideally, this should overlap with the time that technical debt is taken on (when it actually occurs).
3. Tipping Point: The time when the cost of having technical debt starts to overcome the original benefit—that is, the value—of incurring the technical debt. In the interval from Lifecycle Phase 1 to 3, the accumulating side affects of the debt have created a context in which we might be better off repaying the debt. Before the tipping point, we might just as well live with the debt because we get some value from it. At the tipping point, we are paying more than we gain.
4. Payoff: The time when organizations begin removing technical debt from the system. The time interval between the tipping point and payoff is when organizations experience most of the negative consequences of debt and continue to accumulate interest.

Bridge the Gap Between Business and Development

Managing technical debt effectively requires project managers, key decision makers, and the technical teams to agree on the project objectives. Achieving success by uncovering, prioritizing, reducing, and eventually strategically taking advantage of technical debt requires the following:

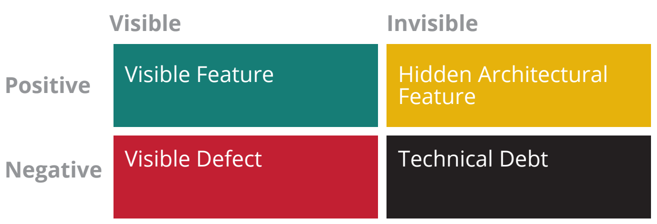
- Development teams must be empowered and incentivized to communicate known sources of debt.
- Management must be willing to provide resources to pay debt back when needed.
- If priorities change often and lead to unexpected sources of technical debt, a technical debt management strategy must be developed.

Technical Debt Evaluation

To meet the challenge of uncovering, communicating, and managing technical debt, the Software Engineering Institute (SEI) has developed a systematic approach. It includes techniques for making technical debt visible, determining what type of debt the project has, and integrating debt into project planning.

Make technical debt visible.
Often, development teams know that some project components may incur future rework, but they do not disclose it. The SEI team engages with the project managers and software development teams to identify debt by answering questions such as the following:

- Is adding a new capability taking longer than expected? If so, are the root causes known?
- Would the system be able to upgrade to a new technology with ease? What is the evidence?
- Are underlying structural issues making defects hard to resolve?



Determine the type of technical debt.
Unstructured large classes, global variables, cyclic code dependencies that create performance issues, and unnecessary copy and paste are all issues of low code quality that result in maintainability challenges and easily lead to technical debt.

Managing debt that results from low code quality and managing debt that results from wrong or obsolete architectural issues require different strategies. The SEI team engages with the project managers and software development teams to answer questions such as the following:

- Were structural shortcuts—such as unwanted modules, large modules, or unsystematic reuse—taken to optimize resources or for technical reasons? Do the systems need to be re-architected?
- Do the systems meet key runtime requirements, especially in security, performance, and availability? If not, are the root causes known, and are there plans to fix these issues?
- Are standards and procedures followed for development practices?
- Do teams have sufficient development infrastructure and tools to follow state-of-the-art configuration management and testing practices?
- Does the system architecture cause more technical debt to accumulate?

Integrate technical debt into project planning, and associate technical debt with risk.
Managing technical debt is rooted in knowing the system's structure and behavior and balancing the program's short-term and long-term goals. In particular, data collection and analysis techniques must incorporate technical debt management into planning and risk management. The SEI team—together with the project managers and software development teams—identifies areas where data can help uncover hidden sources of technical debt and help prioritize where to pay debt down first. We help the organization answer the following questions:

- Does the project allocate resources for paying down known sources of debt and uncovering those that may not be explicit?
- Is there a debt strategy based on the risk profile of the system?
- Is it possible to identify and communicate known sources of technical debt across the project artifacts?

Additional Resources

Architectural Technical Debt Library
resources.sei.cmu.edu/library/asset-view.cfm?assetID=509492

Architectural Technical Debt Blog Archives
insights.sei.cmu.edu/sei_blog/technical-debt

About the SEI

The Software Engineering Institute is a federally funded research and development center (FFRDC) that works with defense and government organizations, industry, and academia to advance the state-of-the art in software engineering and cybersecurity to benefit public interest. Part of Carnegie Mellon University, the SEI is a national resource in pioneering emerging technologies, cybersecurity, software acquisition, and software lifecycle assurance.

Contact Us

CARNEGIE MELLON UNIVERSITY
SOFTWARE ENGINEERING INSTITUTE
4500 FIFTH AVENUE; PITTSBURGH, PA 15213-2612

sei.cmu.edu
412.268.5800 | 888.201.4479
info@sei.cmu.edu