Panel

Technical Debt: From Source to Mitigation

Steven Fraser
Director
Cisco Research Center
Cisco Systems, San Jose
sdfraser@acm.org

Dennis Mancl
Distinguished Member of Technical Staff
Alcatel-Lucent
Murray Hill, NJ, USA
dennis.mancl@alcatel-lucent.com

Bill Opdyke
Architecture Lead
Corporate Internet Group
JPMorgan Chase, Chicago
opdyke@acm.org

Judith Bishop
Director of Computer Science
Microsoft Research
Redmond, WA USA
jbishop@microsoft.com

Pradeep Kathail
Chief Network Architect
Cisco Systems, San Jose
pkathail@cisco.com

Junilu Lacar
Technical Leader
Cisco Systems, Ohio
jlacar@cisco.com

Ipek Ozkaya
Senior Member, Technical Staff
Software Engineering Institute
ozkaya@sei.cmu.edu

Alexandra Szynkarski
Research Associate
CAST Software Research Labs
a.szynkarski@castsoftware.com

Abstract

The term "Technical Debt" was coined over 20 years ago by Ward Cunningham in a 1992 OOPSLA experience report. Ward used "Technical debt" to describe the trade-offs between delivering the most appropriate – albeit likely immature – product, in the shortest time possible. Since then, the repercussions of "technical debt" have become more visible, though not necessarily more widely understood. This SPLASH panel will bring together practitioners to discuss and debate "Technical Debt".

Categories and Subject Descriptors K.0 [**Computing Milieux**]: General

General Terms Management, Design, Economics, Experimentation, Standardization.

Keywords tools; process; research

1. Steven Fraser – *Panel Impresario*

STEVEN FRASER joined the Cisco Research Center as Director in July 2007. His responsibilities include: fostering Cisco-university research collaborations, coordinating PhD/Post-Doc recruiting, and nurturing technology transfer, e.g. as General Chair of the CTech Forum. Prior to joining Cisco, Steven was a Senior Staff member of Qualcomm's Learning Center in San Diego, leading software learning programs and creating the corporation's internal technical conference (the QTech Forum). Steven formerly held a variety of technology strategy roles at Bell-Northern Research (BNR) and Nortel including: Process Architect, Senior Manager (Disruptive Technology and Global External Research), Advisor (Design Process Engineering), and General Chair of the BNR/NT Design Forum. In 1994 he spent a year as a Visiting Scientist at the Software Engineering Institute (SEI) collaborating with the "Application of Software Models" project on the development of team-based domain analysis software reuse) techniques. Fraser has been a panel "impresario" for more than a dozen conferences including serving as Panel Chair of ACM's OOPSLA (SPLASH) in 2003 and 2013, and the Panel Chair for the European series of XP conferences. He is the Publicity Chair for ESEC 2013. Previously he was the Corporate Support Chair for OOPSLA'08 and OOPSLA'09. He was the Tutorial Chair for XP2008 and the Tutorial Co-Chair for ICSE'09. Fraser holds a doctorate in Electrical Engineering (software engineering) from McGill University in Montréal and is a senior member of the ACM and the IEEE.

SPLASH'13, October 26–31, 2013, Indianapolis, Indiana, USA.
ACM 978-1-4503-1995-9/13/10.
http://dx.doi.org/10.1145/2508075.2516596

This panel will leverage discussions earlier this year at ICSE 2013's "Software Engineering in Practice" (SEIP) track in San Francisco and XP2013 in Vienna. The panel will follow the SPLASH Technical Debt Workshop organized by Mancl, Fraser, and Opdyke. Panel topics and questions as a catalyst for discussion may include:

- Anticipated impact on software, on the user and on the software engineering practitioner communities
- Measures, models and tools for analyzing, assessing, and communicating levels of Technical Debt
- Strategies for educating: software developers, managers, executives, organizations, and customers
- Strategies for mitigating Technical Debt ("debt relief")
- Utility of Technical Debt as a mechanism to increase the stickiness of software best practices
- What (if any) are the parallels between Technical Debt and other aspects of software engineering, e.g. "security" (something that customers often take for granted, but are unwilling to fund)?
- Where will Technical Debt fit into the arsenal of software engineering concepts 20 years from now?

- Who should learn about Technical Debt – and why?

While the above topics/questions will be used to initiate panel conversations, it will be up to the SPLASH audience to be inquisitive and to ask challenging questions.

2. Dennis Mancl – *Panel Co-Facilitator*

DENNIS MANCL works for Alcatel-Lucent, where he is involved in technologies to support the development of high-quality software: applying software modeling approaches, agile development practices, and legacy software development techniques to the development of large telecom systems.

Both software developers and leaders need to be aware of technical debt. One recurring task for software developers needs to be reducing technical debt in legacy code modules. Any long-lived software product will have some modules that evolve from release to release, and if debt reduction activities can slow the build-up of accidental complexity, everyone can save time and money.

Technical debt is caused by many forces in a big software project. Leaders need to watch out for the impact of intense schedule pressure, excessive requirements churn, lack of experience of the development team in the problem domain, and the use unfamiliar programming languages and tools. Effective software development is a process that involves on-going learning, and leaders need to manage the learning cycles as much as their project deliverables and milestones.

3. Bill Opdyke

BILL OPDYKE has spent much of his career focusing on the technical and organizational issues related to software engineering, particularly focusing on large scale evolving systems. He is currently an architecture lead at JPMorgan Chase. Previously, at Motorola he was part of a team focusing on improving productivity and reducing costs of software developments. While at Bell Labs, Bill was technical lead for several advanced development projects where he gained a keen appreciation for the challenges in extending existing products to meet emerging market needs. He also spent several years as a faculty member at North Central College. His doctoral research in object-oriented refactoring (University of Illinois) focused on techniques for supporting the process of change to object-oriented software, including structural changes that reduce technical debt.

4. Judith Bishop

JUDITH BISHOP is Director - Computer Science at Microsoft Research, based in Redmond, USA. Her role is to create strong links between Microsoft's research groups and universities globally, through encouraging projects, supporting conferences and engaging directly in research. Her expertise is in programming languages and distributed systems, with a strong practical bias and an interest in compilers and design patterns. She initiated the Software Innovation Foundation (SEIF) and is currently investigating aspects of running programs in browsers (particularly F# and TouchDevelop). Judith is active in IFIP WG2.4 and the ACM, where she has responsibility for the Student Research Competition. Judith received her PhD from the University of Southampton in 1977 and has a distinguished background in academia.

Debt: how do we get into it and how do we get out of it? Good citizens don't get into unreasonable debt and neither should good companies. But there are traumatic circumstances that can cause a change in fortunes such as a stock market crash or an earthquake. The equivalent in the case of technical debt for a software company would be an unanticipated or one-off change in direction caused by outside forces. For example, the world witnessed an unprecedented investment in software change for the turn of the millennium (Y2K) and in Microsoft, the rise of worms and viruses caused a security lockdown in 2002. At the moment we are still working with the effects on software of the changes wrought by hardware, especially display resolution, and touch and pen input. Systems have been written that rely entirely on one kind of input-output systems and now have to be upgraded to another. That is a debt that was not foreseen.

In order to create systems that are as amenable to change as possible, we need to gaze into the crystal ball in order to put the company's effort and funding where it will be best leveraged later. Working with the results of past software investments helps. We can measure past changes, their impact, the company's readiness and the speed with which it could react. For older systems, it might be worth inserting a clean framework that handles change well, but doing so could be too costly and by the time it is done, it could be out of date.

Given the choice of living with technical debt (and unwieldy code) and getting rid of the debt (by means of a partial rewrite), the latter has more going for it. The reason is due to the human resources that will be involved in the project. They can bring the latest technology to bear, and also add ownership, both of which are factors that can be shown to increase stability and performance.

5. Pradeep Kathail

PRADEEP KATHAIL is the Chief Network Architect for the Network Operating Systems Technical Group (NOSTG) and is responsible for technology and standards strategy, next-generation enterprise architecture and innovation. He leads a team to promote and incubate innovation within ENG and is part of Cisco's DE/Fellow Technology Fund steering team. Prior to his current assignment, he held the CTO position in Unified Access Business Unit and Network Systems and Solution Technology Group, responsible for technology strategy and software architecture. As a Cisco Distinguished Engineer, Pradeep led many large scale OS infrastructure and system development projects such as IOS componentization, Modular IOS (ION) and IOS/ENA (predecessor to IOS-XR). Prior to Cisco, Pradeep held various technology management jobs at Apple, Novell, SITA and IBM designing, developing and maintaining large or ultra large scale systems.

6. Junilu Lacar

JUNILU LACAR is a technical leader in the Cisco Services Technology Group and is responsible for the development of business-critical applications that support the management of Cisco's security intelligence assets and research operations. He is an original member of the Cisco Agile Coaches Network and has helped drive adoption of agile development practices at Cisco since 2006. His interests lie mainly in agile technical practices such as Test-Driven Development and extend to successful strategies for large-scale agile transformations. Prior to Cisco, Junilu held various senior development positions at JP Morgan Chase and Compuware. Junilu's experience spans over two decades, multiple industries, and several countries including the Philippines, Singapore, Malaysia, and Hong Kong.

As agile adoption continues its march into mainstream development organizations, the term "technical debt" becomes increasingly subject to "semantic diffusion," a term coined by Martin Fowler to describe how the spreading acceptance of a term can change its definition and weaken its usefulness (http://martinfowler.com/bliki/SemanticDiffusion.html). Robert

Martin's 2009 article hints at this on-going semantic diffusion. (https://sites.google.com/site/unclebobconsultingllc/a-mess-is-not-a-technical-debt).

Despite its continuously evolving meaning, technical debt is still a useful metaphor that helps us think about a class of problems. As with financial debt, technical debt can be addressed both at micro levels and macro levels of development organizations. At the lower levels, technical debt is best addressed with more education for those who make the day-to-day decisions that cause debt to be incurred in the first place. At the higher levels, management should provide infrastructure and services that help stimulate and sustain the flow of development. Just as governments and financial institutions use various indicators to gauge the health of the economy, management can also use technical debt as an indicator of the health of their development organizations. Management should provide guidelines and policies that help steer decisions and behaviors for managing technical debt in the right direction, just as governments and financial institutions define macroeconomic policies to steer the economy. Management should be careful not to impose guidelines and policies that are so restrictive as to inhibit creativity and innovation.

7. Ipek Ozkaya

IPEK OZKAYA is a senior member of the technical staff at the Carnegie Mellon Software Engineering Institute (SEI). With her team at the SEI, she works to develop methods for improving software development efficiency and system evolution with a focus on software architecture practices, software economics, and requirements management. Her latest publications include multiple articles on these subjects focusing on agile architecting, dependency management, and architectural technical debt. She also serves as the chair of the advisory board of the IEEE Software magazine and as an adjunct faculty member for the Master of Software Engineering Program at CMU. Ipek holds a doctorate from CMU in Pittsburgh.

The concept of technical debt has been around for over two decades (W. Cunningham, "The WyCash Portfolio Management System" in *OOPSLA'92*. Vancouver, 1992.); however, understanding what technical debt is and developing/using practices towards its management has gained increased interest both from research and developer communities in the past few years (*International Workshops on Managing Technical Debt,* http://www.sei.cmu.edu/community/td2013esem/previous/?location=secondary-nav&source=723401). Unfortunately, increased interest resulted in a tendency to label all kinds of software ills as technical debt. This tendency to label all ills as technical debt results in a) a confusion on the basic definition of what technical debt is, b) diminishes the value of defining and sharing practices that can enable managing design trade-off strategically to better handle technical debt, c) hinders progress on research that focus on quantifying design decisions and software quality towards understanding technical debt.

The software engineering community should focus less on reinventing a definition and focus on practices for managing technical debt and research that help better guide quantitative decision making (Philippe Kruchten, Robert L. Nord, Ipek Ozkaya, Davide Falessi. "Technical Debt: Towards a Crisper Definition: Report on the 4th International Workshop on Managing Technical Debt," held at ICSE 2013. *SIGSOFT Software Engineering Notes*, ACM, September 2013). Steve McConnell offers a clear technical debt definition S. McConnell, "Managing technical debt (slides)," in Workshop on Managing Technical Debt (part of ICSE 2013, IEEE, 2013): "A design or construction approach that's 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 (including increased cost over time)." What is clear in McConnell's definition is a focus on the design and construction approach that can potentially bear technical debt and the impact of time on the resources spent.

We now have considerable material from small and large industries about technical debt, how they perceive it and manage it (Philippe Kruchten, Robert L. Nord, Ipek Ozkaya: "Technical Debt: From Metaphor to Theory and Practice." *IEEE Software* 29(6): 18-21 (2012)). This includes companies such as IBM, Cisco, Siemens, Google, Lockheed-Martin, and so on. We have reports from companies who have a different interest at stake, selling tools or services pertaining to technical debt: Cast Software, Software Improvement Group, ThoughtWorks, Cutter Consortium, and so on.

The increasing industry interest and emergence of organization specific practices can be seen as an early indication that industry needs a clearly defined practical managing-technical-debt practice to deal with issues such as evolution, strategic resource management and bridging the stakeholder communication gap that has been on the forefront of research as well. While often the impact of technical debt is seen as the inevitable consequences of "death by a thousand cuts" understanding the key contributors and deciding whether managing them as debt would help is an important first step. Organizations that have embraced technical debt as part of their iteration planning practices achieve success as a result of the following actions:

- making technical debt visible,
- differentiating strategic structural technical debt from technical debt that emerges from low code quality,
- using the elicited technical debt as a means for bridging the gap between the business and technical sides,
- integrating technical debt into planning,
- associating technical debt with future risk to identify a payback strategy.

8. Alexandra Szynkarski

ALEXANDRA SZYNKARSKI is a research associate at CAST Research Labs, New York. Her research interests include industry benchmarks on structural software quality and software performance and productivity measurement on the global application development community. Szynkarski participated in the ICSE 2013 Technical Debt Panel (Technical Debt: Past, Present, Future). She currently works with Bill Curtis on putting together a benchmarking repository of structural quality data. She also participates in creating an online community through the platform www.ontechnicaldebt.com with the goal of raising awareness on the topic and making sure the industry and research community have access to the latest and most accurate information on the topic. Szynkarski received an MS in international business administration from the Institut Administration des Entreprises de Nice, France.

I have had the opportunity to work with the world's largest repository of structural quality data: more than 1,300 applications and over 350 million lines of code. Over the past 3 years, I have studied this repository publishing global trends in the structural quality of business software applications. Working alongside Bill Curtis (Vice President and Chief Scientist at CAST), we were able to put together a calculation of Technical Debt based on violations of good architectural coding practices as well as discover the average amount of Technical Debt that exists within a typical business application.

The debate on the definition of Technical Debt is a difficult one to tackle. The concept has evolved since it was coined by

Ward Cunningham in 1992 to include more components and originally anticipated. With little reference data on the topic, I believe that each company has their own definition of what should or should not be included as Technical Debt. At CAST we define Technical Debt as the effort required to fix violations of good architectural and coding practices that remain in the code when an application is released. Technical Debt is still an emerging concept and the term tends to be loosely thrown around. It is therefore important to be able to quantify and measure Technical Debt as it provides a framework to understanding the state of our business applications and will push the metaphor to another level of research. The CAST benchmarking repository of structural quality data provided a unique opportunity for us to do just this: calculate Technical Debt across different technologies, based on the number of engineering flaws and violations of good architectural and coding practices in the source code.

Automated static code analysis tools can help play an initial role in calculating and managing Technical Debt. It is unfortunate and undeniable that many tools to not provide the complete scope needed to deal tackle the issue. However, certain tools will be able to provide useful quantification of Technical Debt, as well as help improve the structural quality of code. I believe that quality of our code matters. And in order for the Technical Debt term to move forward we need to start measuring software quality and understanding the Technical Debt's implications to quality.