Strategic Management of Technical Debt

Ipek Ozkaya
Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA USA
ozkaya@sei.cmu.edu

Philippe Kruchten
Department of Electrical and Computing Engineering
University of British Columbia
Vancouver, Canada
pbk@ece.ubc.ca

***Abstract*— Technical debt acknowledges that software development teams sometimes accept compromises in a system in one dimension (for example, modularity) to meet an urgent demand in some other dimension (for example, a deadline), and that such compromises incur a "debt". If not properly managed the interest on this debt may continue to accrue, severely hampering system stability and quality and impacting the team's ability to deliver enhancements at a pace that satisfies business needs. Although unmanaged debt can have disastrous results, strategically managed debt can help businesses and organizations take advantage of time-sensitive opportunities, fulfill market needs and acquire stakeholder feedback. Because architecture has such leverage within the overall development life cycle, strategic management of architectural debt is of primary importance. Some aspects of technical debt – but not all technical debt – affect product quality. This tutorial introduces the technical debt metaphor and the techniques for integrating it fully with the software development lifecycle intentionally, with a focus on software architecture.**

Keywords-technical debt, design trade-offs, architecture decision making

I. Introduction

Large scale projects face the dilemma of balancing rapid deployment with long-term value. The term technical debt describes this trade-off between short-term and long-term value and has already penetrated into practice [1]. The software development community increasingly relies on technical debt as a way to understand and communicate issues of intrinsic quality, value, and cost [2].

As recent as a couple years back the challenge of understanding what technical debt is and what it refers to was the dominant question in the research and practitioner community. This has been replaced recently with testing practices that concretely communicate technical debt and its consequences. In the absence of validated tools and techniques to achieve this goal with repeatable results, developers resort to ad hoc practices in an effort to communicate technical debt [3][4]. Researcher resort to incremental research approaches, mostly repurposing existing code quality work to capture technical debt [5]. A recent reflection of these efforts has been the publication of the beta standard for Technical Debt by OMG [6].

Increasing efforts of software quality analysis tool vendors to repurpose their capabilities as technical debt assessment tools contradict with research results that demonstrate the debt that has the highest cost of ownership is architectural issues and they cannot always automatically be measured.

Recent systematic literature reviews on technical debt have created categories and concept ontologies [7][9][12] and related debt to different stages in the development life cycle [8][10][11][13]. Small-scale interview studies on understanding how developers talk about technical debt have focused on sources of technical debt [14][15][16]. Broad practitioner surveys demonstrate architecture to be the most costly and hardest to manage technical debt issues [17].

The topic is highly relevant to the ICSA audience as software architecture is at the epicenter of most root causes of technical debt issues. Software architects, developers and architecture researchers need to have a crisp understanding of what constitutes technical debt and what tools to pull out from their tool boxes when.

II. Format of the Tutorial

A. *Duration.*

This is a half-day tutorial on technical debt that has the goal of introducing the concept as well as techniques that can be used immediately today. The discussion during the tutorial includes a balance of open challenges as well as what can be accomplished by practitioners today.

B. *Preliminary outline of the sessions.*

- Introduction to Technical Debt (30 minutes)
  - Definition of technical debt
  - Examples of technical debt
- A Definition Framework (30 minutes)
  - Technical debt landscape
  - Common misconceptions
  - Timeline approach
  - State of the practice: examples from industry
- Practical measures (45 minutes)
  - Making technical debt visible
  - Exercise: documenting a technical debt item in the issue tracker
- Analyzing Technical Debt (45 minutes)
  - Code quality versus architecture analysis
  - Example of a common problem: overgeneralization
  - Example of a common problem: prototype evolving to product

- Conclusion (30 minutes)
    - Key concepts: landscape, techniques, timeline
    - Practices that can be incorporated into managing projects
    - Research challenges

C. *Learning outcomes:*

This tutorial introduces participants to the practical aspects of managing technical debt. We expect attendees to walk away with:

- an understanding of the key concepts-there is a name for a recurring problem engineers experience and a conceptual framework to seek understanding about managing short-term and long-term tradeoffs of cost and value,
- a template for documenting a technical debt item in the issue tracker
- lessons learned from fellow practitioners,
- review of analysis techniques to uncover technical debt
- references to practical techniques that can address part of the problem today

D. *Target audience:*

The target audience is software developers, architects, technical managers as well as researchers interested in architecture trade-offs and system analysis.

In addition to several onsite offerings to government and industry organizations variations of this tutorial have been offered several time in academic venues including but not limited to:

- International Conference on Software Architecture 2017, Göteborg
- International School of Software Architecture, Leiden Netherlands, June 2017
- SEI Architecture Technology User Network Conference, 2011, 2012, 2015, 2016
- CompArch, 2013
- International Conference on Software Engineering, 2012
- Working IEEE/IFIP Conference on Software Architecture, 2011
- Agile 2011 Conference
- SEI Educators' Workshop

The audience at these venues included architects, software developers, technical managers, educators, and researchers. Participation ranged from one to two dozen people. There is a one-day course that is offered publically on the topic at the Software Engineering Institute that some of the material comes from.

III. Presenters

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 help organizations improve their software development efficiency and system evolution. Her work focuses on software architecture practices, software economics, and requirements management. Her latest publications include articles on agile architecting, dependency management, and architectural technical debt. Dr. Ozkaya also serves editorial boards of the IEEE Software magazine and Journal of Information and Software Technology and as an adjunct faculty member for the Master of Software Engineering Program at Carnegie Mellon University (CMU). She has extensive experience in delivering tutorials and is an invited speaker at software engineering, agile, and architecture venues (e.g., ICSE, OOPSLA, SATURN, and WICSA). She holds a doctorate from CMU in Pittsburgh.

Philippe Kruchten is professor of software engineering in the department of Electrical and Computer Engineering of the University of British Columbia. He joined UBC in 2004 after a 30-year career in industry, where he worked mostly in with large, software-intensive systems design in the domains of telecommunication, defense, aerospace and transportation. His current research interests still reside mostly with software architecture, and in particular architectural decisions and the decision process, as well as agile software engineering processes. He is a founding member of IFIP WG2.10 Software Architecture. Dr. Kruchten received his mechanical engineering diploma from Ecole Centrale de Lyon, and his doctorate degree in Information Systems from Ecole Nationale Supérieure des Télécommunications, Paris. He is a member of IEEE, and ACM, and a Professional Engineer in British Columbia and a frequently invited speaker.

Kruchten and Ozkaya are co-authors of an upcoming book on *Managing Technical Debt*, also with Dr. Robert Nord.

Acknowledgment

Copyright 2018 IEEE. All Rights Reserved.

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.

DM18-0342

References

[1] Spínola, R. O., Zazworka, N., Vetro, A., Seaman, C., Shull, F. 2012. Investigating technical debt folklore: shedding some light on technical debt opinion. MTD@ICSE 2012: 1-7.

[2] Kruchten, P., Nord, R. L., and Ozkaya, I. 2012. Technical debt: From metaphor to theory and practice. *IEEE Softw. Spec. Issue Tech. Debt* 29, 6 (Nov.-Dec. 2012), 18-21.

[3] Stephany Bellomo, Robert L. Nord, Ipek Ozkaya, Mary Popeck: Got technical debt?: surfacing elusive technical debt in issue trackers. MSR 2016: 327-338

[4] Qiao Huang, Emad Shihab, Xin Xia, David Lo, Shanping Li: Identifying self-admitted technical debt in open source projects using text mining. Empirical Software Engineering 23(1): 418-451 (2018)

[5] Radu Marinescu: Assessing technical debt by identifying design flaws in software systems. IBM Journal of Research and Development 56(5): 9 (2012)

[6] Automated Technical Debt Measure, OMG admtf/2017-03-01, March 22, 2017 http://www.omg.org/spec/ATDM/

[7] Alves, N. S. R., Ribeiro, L. F., Caires, V., Mendes, T. S., and Spínola, R. O. 2014. Towards an ontology of terms on technical debt. In *ACM SIGSOFT* 40, 2 (Mar. 2015), 32-34. DOI=http://dx.doi.org/10.1145/2735399.2735419.

[8] Ampatzoglou, A., Ampatzoglou, A., Chatzigeorgiou, A., and Avgeriou, P. 2015. The financial aspect of managing technical debt: A systematic literature review. *Inform. Software Tech.* 64 (Aug. 2015), 52-73.

[9] Izurieta, C., Vetro, A., Zazworka, N., Cai,Y., Seaman, C., and. Shull, F.. 2012. Organizing the technical debt landscape. In *International Workshop on Managing Technical Debt*, pages 23-26, 2012.

[10] Li, Z., Avgeriou, P., and Liang, P. 2015. A systematic mapping study on technical debt and its management. *J. Syst. Softw.* 101 (Mar. 2015), 193-220

[11] Tom, E., Aurum, A., and Vidgen, R. T. 2013. An exploration of technical debt. *J. Syst. Softw.* 86, 6 (2013), 1498-1516.

[12] Nicolli S. R. Alves, Thiago Souto Mendes, Manoel Gomes de Mendonça Neto, Rodrigo O. Spínola, Forrest Shull, Carolyn B. Seaman: Identification and management of technical debt: A systematic mapping study. Information & Software Technology 70: 100-121 (2016)

[13] Terese Besker, Antonio Martini, Jan Bosch: Managing architectural technical debt: A unified model and systematic literature review. Journal of Systems and Software 135: 1-16 (2018)

[14] Guo, Y., Seaman, C., Gomes, R., Cavalcanti, A., Tonin, G., DaSilva, F., Santos, A., and Siebra, C. 2011. Tracking technical debt: An exploratory case study. In *Proceedings of the 27th International Conference on Software Maintenance* (Williamsburg, VA, Sep. 25-30, 2011). IEEE Press, Piscataway, NJ, 528-531

[15] Lim, E., Taksande, N., and Seaman. C. 2012. A balancing act: what software practitioners have to say about technical debt. *IEEE Software*, 29, 6 (2012), 22-27.

[16] Spínola, R. O., Zazworka, N., Vetro, A., Seaman, C., Shull, F. 2012. Investigating technical debt folklore: shedding some light on technical debt opinion. MTD@ICSE 2012: 1-7.

[17] Ernst, N., Bellomo, S., Ozkaya, I., Nord, R. L., and Gorton, I. 2015. Measure it? Manage it? Ignore it? Software practitioners and technical debt. In *Proceedings of the 10th Joint Meeting on Foundations of Software Engineering* (Bergamo, Italy, Aug. 30-Sep. 4, 2015). ACM, New York, NY, 50-60.