Strategic Management of Technical Debt

Tutorial at ICSA 2017

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

Abstract— The technical debt metaphor 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 measuring and communicating this technical debt, integrating it fully with the software development lifecycle.)

Keywords—*Technical debt, software evolution, maintainability, evolvability*

I. Objective

The goal of the tutorial is to give the attendees a better appreciation of the rather fuzzy concept of Technical Debt, articulate the differences between technical debt at the code level, versus technical debt at the architectural or structural level, or at the process level. It will give attendees techniques and tools to reason about technical debt: identify it, and manage it, that is, decide when and why it is wise to "repay" some of that technical debt.

II. Audience & Scope

This tutorial is primarily intended for:

- Software project managers,
- product managers, product owners
- and software architects

Level basic; no required prerequisites.

This presentation is of value for both practitioners and for academics/researchers, but mostly targeted at practitioners.

III. Outline of Tutorial

This half day tutorial will have the following structure:

1. Introduction to Technical Debt (80 minutes)
    a. Game: hard choices
    b. Definitions of technical debt
    c. Examples of various types of debt
2. Practical measures (60 minutes)
    a. Examples, results of interviews from industry
    b. Tools and techniques
    c. Requirements for support (tool or otherwise), metrics and measures of success
3. Future Directions (30 minutes)
    a. Agile architecting and technical debt
    b. Vision for technical debt analysis framework
    c. Discussion on key problems and challenges faced by practicing software engineers who need to elicit, communicate, and manage technical debt at different facets of their projects

IV. Presenter

Philippe Kruchten is professor of software engineering at the University of British Columbia in Vancouver, Canada, which he joined in 2004 after a 30+ year career in the software industry, developing systems in telecommunications, defense and aerospace. His main interests are in software architecture, software project management and software development processes. During his time with Rational Software (now IBM) he led the development of the RUP, which embeds an architecture-centric method. He is the co-founder and secretary of the IFIP WG2.10 on Software Architecture (1998), Chair of the ICSA steering committee, and co-founder and chair of Agile Vancouver (2004). He's a senior member of IEEE CompSoc and of ACM.

Kruchten has organized a series of workshops on technical debt every year since 2010, including a Dagstuhl Seminar in

2016. He will be the program chair for the *conference* on technical debt collocated with ICSE in 2018 in Gothenburg.

He blogs occasionally on the topic of technical debt, and do research funded by NSERC and Mitacs.

Earlier versions of this tutorial have been offered at ICSE 2012, and QSIC 2012, as well as at the Saturn series of conferences in 2014 and 2015.

ACKNOWLEDGMENT

This tutorial was originally prepared with Ipek Ozkaya and Rod Nord from the Software Engineering Institute, CMU, Pittsburgh, PA, USA.