Davie Truong

CMPS 115

dtruong8@ucsc.edu

RAC3-The Danger of Architectural Technical Debt: Contagious Debt and Vicious Circles

“The Danger of Architectural Technical Debt: Contagious Debt and Vicious Circles” by Antonio Martini and Jan Bosch of Chalmers University of Technology published on May 4th- 8th, 2015 in 12th Working IEEE/IFIP Conference on Software Architecture, discuss “which Architectural Technical Debt items are the most dangerous (in terms of effort) and how such effort is manifested during software development.” The authors begin by defining Technical Debt in terms of a metaphor and Architectural Technical Debt. Technical Debt, “…compares the trend of taking sub-optimal decisions in order to meet short-term goals to the taking of debt, which has to be repaid with interest in the long term.” Architectural Technical Debt “is regarded as violations in the code towards the intended architecture for supporting the business goals of the organization.”

With the basic understanding of Architectural Technical Debt achieved, the authors outline their case study and its purpose to “capture as many ATD items experienced by companies, but at the same time having the opportunity to dig out as many details from the context as possible.” After detailing the five companies in the study and their experience with the SCRUM development process, the author introduces their three-phase data collection investigation of the ATD items and effects, describing how they collected the data that their results are based on.

Phase one which is the preliminary study, “explored the needs and challenges of developing and maintaining architecture in an Agile environment in the current companies.” In the preliminary study, open questions were asked that aimed at understanding what the architecture practices were currently employed in the organization. The preliminary study showed major challenges in managing ATD, specifically the consistency of architecture debt, the interest in risk of effort and how it was prioritized.

Phase two which is the exploratory interviews, contains seven sets of interviews which included participants with different responsibilities. “Each set of interviews followed a process designed to identify architecture inconsistencies with high effort impact.” In identifying real cases that happened in the recent past, a list of ATD items with large effort impact was created. Then each item was followed up with the developers involved in order to understand the in-depth details.

Phase three which is validation interviews and artifacts analysis, consisted of two validation activities, more specifically three multiple company group interviews were conducted where the models for recognition and improvement was discussed. They proposed a model for contagious debt and included the analysis of artifacts such as lists of Technical Issues or Architectural Improvement identified within the company. As a further validation step two plenary workshops were organized with twenty architects also from two other large companies not previously participating in the study, to strengthen the result.

The five most important points of the paper are the five items of taxonomy of Architecture Technical Debt and their effects, because of its influence in the development process. The first item is dependency violations and unawareness, which is the class of items that is represented by a component that, when executed, should not trigger the execution of another component. This hinders agile practices such as continuous integration, in which high modularity of the system allows the fast test of small portions of code. The second item is non-uniformity of patterns and policies that are not kept consistent through the system. This causes time spent by the developers in understanding parts of the system that they are not familiar with. The third item is code duplication in various parts of the system, which gets managed separately and not grouped into a reused component leading to double maintenance. The fourth item is temporal properties of inter-dependent resources, the concurrent and non-deterministic interaction with the resource by different components might create hidden and unforeseen issues. Finally, the fifth item, unidentified non-functional requirement, in which there is a lack of implementation for some nonfunctional requirements, such as performance and signal reliability that needs to be recognized before or early during the development.