Architectural Technical Debt: Models and Impact

Antonio Martini

Chalmers University of Technology, Göteborg, Sweden
antonio.martini@chalmers.se

Problem and motivation

A known problem in large software companies is to balance the prioritization of short-term with long-term responsiveness. To illustrate such a phenomenon, a financial metaphor has been coined, which relates taking sub-optimal decisions in order to meet short-term goals to taking a financial debt, which has to be repaid with interests in the long term. Such a concept is referred to as Technical Debt (TD), and recently it has been recognized as a useful basis for the development of theoretical and practical frameworks (Kruchten et al., 2012). Tom et al. (Tom et al., 2013) have explored the TD metaphor and outlined a first framework in 2013. Part of the overall TD is to be related to architecture sub-optimal decisions, and it's regarded as Architecture Technical Debt (ADT). ATD is regarded as violations in the code towards an intended architecture.

The problem of visualizing and communicating the ATD is particularly relevant for large companies, where the architecture documentation is not capable of capturing all the updated information for compliance checking of the source code. It's therefore difficult, both for software architects and managers not involved directly in the development of parts of the system, to monitor and understand the level of current ATD. This, in turn, affects negatively their ability of taking informed decision about the prioritization of architectural improvements and refactorings and about estimating the time needed for product development.

ATD has been recognized as part of TD, but the specific phenomenon of accumulation of ATD, its recovery to avoid the later payment of interest (in terms of effort) and the impact (effects) of such interest in future development has not been tackled yet.

Current results

After a preliminary investigation in which we have collected several evidences about the presence of the problem described above (Martini et al., 2012), we conducted a multiple-case embedded case study in 7 sites at 5 large companies. The aim was to shed light on the accumulation and recovery of Architectural Technical Debt, but also to understand the impact that such phenomenon has in software development. So far, we have developed qualitative models: a taxonomy of the factors and their influence in the accumulation of debt, a model of how the debt is accumulated and recovered over time and a set of models describing the effects caused by the different classes of ATD. An example of such models is visible in Fig.

1, where it's shown how the accumulation of ATD has a negative impact on the ability of the company to add business value to the product, until a crisis point is reached and a big refactoring or a new platform is needed.

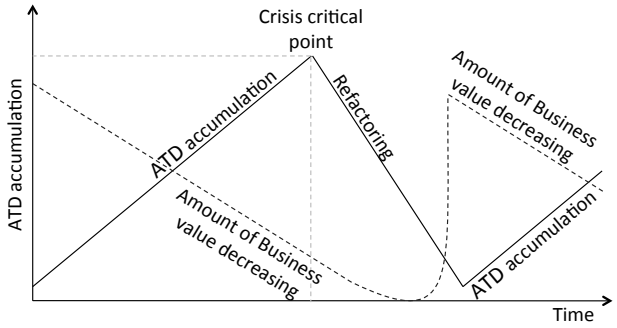

Fig. 1. ATD accumulation until the reach of a crisis point, when a big refactoring is needed

We have developed more detailed models, some of which have been submitted and accepted for publication (for example (Martini et al.)). The qualitative models have been developed by triangulating different sources, especially interviews and artifacts (for example architectural documentation and lists of technical issues and architecture improvements maintained by the companies).

Future work

Current research is focused on trying to visualize and possibly measure the qualities described in the models. This way, we aim at testing the hypotheses previously made on the current trends, but also making possible for software development teams, software architects and managers to have an overview of the ATD present in the system. For example, managers might take decisions on the allocation of resources for refactoring if being able to place the system in the slope described in the model in Fig. 1 or at least knowing about a possibly dangerous current trend in the increment of ATD.

We want to understand what part of ATD can by automatically measured and visualized and what needs to be communicated through human-based activities. Our models include a taxonomy of ATD classes, each of which leads to different effects. We are exploring different ways of inquiring source code, artifacts or employees in order to show useful indicators for different kinds of ATD. The next step comprehends the comparison and aggregation of information from different sources.

A recent example that we have developed within one of the companies involved in our project is a measurement system that would show and prioritize dependency violations at a medium level of abstraction according to context-specific architectural rules and mapping. We are in the process of investigating the trends and the effects of such violations through an in-depth case study and trying to apply the measurement system to a different context for replication purposes.

Kruchten, P., Nord, R.L., Ozkaya, I., 2012. Technical Debt: From Metaphor to Theory and Practice. IEEE Softw. 29, 18–21. doi:10.1109/MS.2012.167

Martini, A., Bosch, J., Chaudron, M., n.d. Architecture Technical Debt: Understanding Causes and a Qualitative Model, in: Accepted for publication at 40th Euromicro Conference on SEAA.

Martini, A., Pareto, L., Bosch, J., 2012. Enablers and inhibitors for speed with reuse, in: Proceedings of the 16th International Software Product Line Conference - Volume 1, SPLC '12. ACM, New York, NY, USA, pp. 116–125. doi:10.1145/2362536.2362554

Tom, E., Aurum, A., Vidgen, R., 2013. An exploration of technical debt. J. Syst. Softw. 86, 1498–1516. doi:10.1016/j.jss.2012.12.052