Architecture-centric Source Code Organisation Control

Abstract

Architects of software systems need to make design decisions to ensure that a system will execute desired functionalities with required quality properties (e.g., performance, safety, security). However, when compared to other engineering fields, software engineering has three significant challenges: i) systems need to change fast and often, ii) it is not possible to reconstruct design decisions by observing a system (source code, when available, is hard to understand and abstract), and iii) most other engineering fields are much older than software engineering, implying a substantial empirical knowledge. Frequent system changes make maintenance of system design expensive and often impractical, resulting in a loss of traceability between design and implementation. Without understanding the original design intentions and design decisions behind source code, it is hard to sustain, maintain, and expand software systems. It is hard to predict how certain changes will impact a system's quality properties.

The aim of this project is to develop a solution that enables control of source code's organisational structure from the architectural level by introducing new abstraction levels on top of source code. Inspiration for this work comes from the "packages" abstraction level present in Java programming language. The goal is to enable management of organisational aspects of source code from the architectural level, because the gap between implementation (source code) and current perception of abstractions remains too big to be automatically bridged. Furthermore, some implementation details are not architecturally significant. We will develop abstractions on top of the source code level related to its organisation (e.g., file system organisation) which make it easier to map source code to design level. Furthermore, these abstractions will enable us to propagate changes to source code from the architectural model. We will investigate what kind of abstractions are profitable and practical on the development level. Based on those results, we will develop concepts supported by tools that will take input from models and change source code, while ensuring traceability between these. The final part of this work package will aim to integrate produced results (concepts and tools) with development environments. Such integration will enable adding architectural intentions directly to source code parts, and in contrast to the existing approaches would not be limited to textual representation.