1 GRK Number of the graduate school: Name of the graduate school

1.1 Dynamic Service Analysis

Michael Perscheid (michael.perscheid@hpi.uni-potsdam.de)
Supervisor: Prof. Dr. Robert Hirschfeld

The Software Architecture group, led by Prof. Dr. Robert Hirschfeld, develops new methods, concepts, and tools for improving the comprehension and design of large complex systems.

One of the most important aspects in software development is to understand programs, with a special focus on those internal details that constitute system functionality. Current development environments offer a wide range of perspectives to support developers in understanding source code, but the task of program comprehension is still time-consuming and laborious. This is in part due to most tools’ limiting their scopes to static views on source code without considering program behavior.

With such additional behavioral information available, new perspectives for comprehension could be offered, ranging from internal communication between objects to orchestration at the architectural level. Although the widespread creative use of debuggers for understanding behavior by developers indicates a need for dedicated behavioral views. There are only few development tools addressing this requirement. Most prominently, high memory consumption and performance penalties render current approaches impractical in software development.

We suggest a practical, lightweight, and incremental approach to dynamic analysis based on test cases as entry points into system behavior. By observing and enriching concrete examples of program executions, we investigate three perspectives on software systems with the main focus being on improving behavior comprehensibility.

First, we design and implement dynamic analysis tools that are seamlessly integrated into current development environments. Using these, developers can introspect the behavior of the system under test. Based on a lightweight call graph representation used for navigation, developers can state their points of interest and all further information will be computed on demand by re-executing tests. We distribute the overhead of dynamic analysis over multiple runs so that there is no need to collect all data at once, whereby memory consumption and performance impacts are kept low. Thus, we have a practical approach for behavioral views that will be evaluated regarding its practicability and improvements for program comprehension in the near future.

Next, our concept of dynamic software architectures merges static and behavioral (dynamic) system architecture. We investigate new system perspectives that are intended to support software engineering tasks. Tool support will be provided that guides developers to potential locations for, for
1.1 Dynamic Service Analysis

example, caching, redundancies, or unused code. Moreover, new tools will compare dynamic paths through the system and investigate anomalies, leveraging more suitable failure detection or hints at design disharmonies.

Finally, *post-traceability of requirements* is considered a critical component in software understanding, as it allows developers to comprehend the system from the user’s point of view. However, existing traceability approaches often comprise tedious processes with a small degree of automation. We propose a new requirement traceability approach that automatically combines feature localization techniques with acceptance tests. Our concept is not limited to specific system properties, so that we can propagate requirement traceability knowledge across service and system boundaries. In future work, we will automate the manual step that deals with the connection between acceptance tests and requirements, enrich development tools with traceability information, and address some typical feature localization problems of service-oriented systems.

To summarize, dynamic service analysis offers new and deeper understanding of how services and objects belong together. Moreover, new starting points are given for improving reusability, robustness, and clarity of services and their implementations.