Abstract

Software developers responsible for adapting and evolving an existing implementation of software must grasp the underlying behavioural dependencies implicitly imposed by making certain design decisions about the software at hand. Especially for technologically and algorithmically challenging systems, the integration of several complex components gives rise to subtle interactions that are prone to faulty behaviour. This is problematic because these behavioural dependencies, or so-called design invariants, are often only implicitly known. This severely impedes program development as making a change to the system might violate unknown behavioural constraints, which ultimately leads to unreliable software.

In this dissertation we propose an approach for supporting the development of technically and algorithmically complex systems by documenting and verifying design invariants in a lightweight manner. The most distinctive feature of our approach is the use of temporal logic programming as executable behavioural formalism for documenting design invariants in behavioural models. One of the main advantages of this formalism is the high level of abstraction it offers. Next to the declarative feature offered by a logic language which allows the use of high-level concepts, abstractions for time structures are offered by the use of temporal operators.

Moreover, a causal link between the design invariant specification and the source code is provided allowing a lightweight verification. This is realised by using a dynamic analysis approach which adopts a selective code instrumentation mechanism by means of logic meta programming. Consequently, by executing the selectively instrumented program according to a well-chosen execution scenario, only those high-level events that are relevant to the design invariant under investigation are recorded. This makes the approach goal-driven and hence applicable to parts of a larger program as well.

To validate and verify the practical feasibility of our proposed approach, we constructed the lightweight verification platform BEHAVE, which supports the documentation and lightweight verification of design invariants in C programs. The practical application of BEHAVE is illustrated by supporting program development of the Pico

virtual machine, an interpreter for a programming language supporting several technologically advanced features. We supported a representative set of three Pico design invariants and for all of these invariants important inconsistencies were found. Further development of Pico is now supported as these design invariants are made an explicit and verifiable part in future adaptations of Pico.