Universiteit
Leiden
The Netherlands

Testing object Interactions

Grüner, A.

Citation
Grüner, A. (2010, December 15). *Testing object Interactions*. Retrieved from https://hdl.handle.net/1887/16243

Version:	Corrected Publisher's Version
License:	Licence agreement concerning inclusion of doctoral thesis in the Institutional Repository of the University of Leiden
Downloaded from:	https://hdl.handle.net/1887/16243

Note: To cite this publication please use the final published version (if applicable).

Part I

Testing Sequential Components

In this main part of this thesis we will propose a component testing approach for *Java* components. The contribution is threefold. We will define a *test specification language* which allows to specify the desired behavior of a component in terms of expected communication with its environment, i.e., in terms of its *interface behavior*. Moreover, we will present an algorithm for *automatically generating* a test program from a given specification such that the program tests for a component's conformance to the specified interface behavior. To this end, we will first present a *formally defined programming language* which captures a subset of the *Java* language. In particular, we will provide a formal semantics for components of this language. This enables us to investigate and characterize the possible observable interface behavior of a component.

The characterization will help us to find an appropriate design of the specification language, which will be a careful balance between two goals: we will use programming constructs in *Java*-like notation that help the programmer to specify the interaction without having to learn a completely new specification notation. On the other hand, *additional* expressions in the specification language will allow to specify the desired interface behavior in a concise, abstract way, hiding the intricacies of the required synchronization code at the lower-level programming language. Moreover, the formal language will be used to formalize the code generation algorithm and to proof its correctness.