Cover Page

Universiteit Leiden

The handle http://hdl.handle.net/1887/97598 holds various files of this Leiden University dissertation.

Author: Serbânescu, V.
Title: Software development by abstract behavioural specification
Issue Date: 2020-06-10

Summary

The development process of any software has become extremely important not just in the IT industry, but in almost every business or domain of research. The effort in making this process quick, efficient, reliable and automated has constantly evolved into a flow that delivers software incrementally based on both the developer's best skills and the end user's feedback.

Software modeling and modeling languages have the purpose of facilitating product development by designing correct and reliable applications. The concurrency model of the Abstract Behavioural Specification (ABS) Language with features for asynchronous programming and cooperative scheduling is an important example of how modeling contributes to the reliability and robustness of a product. By abstracting from the implementation details, program complexity and inner workings of libraries, software modeling, and specifically ABS, allow for an easier use of formal analysis techniques and proofs to support product design. However there is still a gap that exists between modeling languages and programming languages with the process of software development often going on two separate paths with respect to modeling and implementation. This potentially introduces errors and doubles the development effort.

The overall objective of this research and thesis is bridging the gap between modeling and programming in order to provide a smooth integration between formal methods and two of the most well-known and used languages for software development, the Java and Scala languages. The research focuses mainly on sequential and highly parallelizable applications, but part of the research also involves some theoretical proposals for distributed systems. It is a first step towards having a programming language with support for formal models. The contributions of this thesis are divided in three parts aimed at achieving this overall objective.

The first part is about developing a runtime that implements the proposed concurrency model of ABS its features with the main focuses being on performance and scalability of the implementation. The runtime has the purpose of optimal thread management in the Java language and provides a full integration of actors and futures with asynchronous programming.

The second part of this research brings the high-level modeling constructs, especially those that model asynchronous programming and concurrent behavior in ABS to the level of the Java language through an API. The base API, called JAAC is exposed Java and presents some constructs which are quite permissive with regards to type checking of the proposed asynchronous call and vulnerable to unwanted code being run on actors. The API is extended to Scala syntax, called ASCOOP, where all (a)synchronous method calls are now type checked by the Scala compiler allowing only calls to methods that are exposed by a class or interface.

The third part focuses on the development of a compiler from the software model to Java and extended to Scala that provides a formally correct translation and behavior. This translation fully supports the semantics of the core modeling language which includes modeling actors and actor groups as software components. The compiler support includes asynchronous communication, coroutine suspension and resume constructs. Finally the compiler translates the ABS language extensions for timed-models and resource consumption.