Summary

The physical constraints in computer hardware manufacturing and the industry's eagerness to keep up with the Moore's law, led to the establishment of multicore processors and (Cloud) distributed systems in our everyday use. To be fully utilized, these technologies of "simultaneous" processing often require elaborate modifications to the computer software. This has placed a large burden on the software-development side, especially when also taking into account the ever-increasing demand for more featureful, thus likely more complex software. One particular method to tackle software complexity is software modeling, which leaves out certain implementation details and focuses instead on the functional correctness of the software. Yet, to gain full performance from the aforementioned advancements in hardware, a model of software should be aware of the hardware resources, at least in an abstract manner.

In this thesis, we strive to address this challenge by constructing a modeling language to write software which can take advantage of recent hardware developments (multicore, cloud) without compromising in its abstraction levels. Our language is based on top of the Abstract Behavioral Specification (ABS), which is an executable modeling language with a focus on cooperative-multitasking concurrency. We translate programs written in our ABS-based language to Haskell, an established functional programming language, since Haskell comes with built-in support for both cooperative concurrency in the form of coroutines, and multicore parallelism through lightweight threads of execution. Further, we formally prove the correctness as well as the resource-consumption preservation of the translation of a subset of our language to Haskell. To put our solution to test, we compare its performance to other existing ABS-based implementations.

To enable software models take control of their computing resources, we extend our language with certain constructs that abstract (virtualize) over the hardware. This "resource-aware" language extension is packaged in a tool-suite for human-in-the-loop simulation of Cloud services; such a live simulation can be used for training DevOps engineers to the cloud environment of IT companies.

Finally, we provide an implementation of distributed communication and a connection to the Cloud infrastructure, so that software models written in our language can be executed as distributed applications. Because models are "resource-aware",

they can programmatically monitor and control their own Cloud deployment. Our implementation is the first realization of the earlier "Deployment Components" concept of ABS to abstract over Virtual Machines of the Cloud and enable any ABS application to distribute itself among multiple Cloud-machines.