Master / Bachelor

Modellbasiertes Monitoring integrierter State Machines

Model-based Monitoring of Integrated State Machines

Motivation — Deutsch

State Machines (SM) sind in der Praxis weit verbreitet, um das Verhalten von Software zu beschreiben. Ein Vorteil der Modellierung des Softwareverhaltens mit SM ist ihre Visualisierung während des Designs und der Laufzeit. Das von SM implizierte Design wird dann in der Regel mit einer Programmiersprache umgesetzt. Dieser Code kann so generiert werden, dass nur die technischen Details des Verhaltens manuell implementiert werden müssen und sichergestellt werden kann, dass sich das Programm genauso verhält wie es modelliert wurde. In der Praxis werden die Modelle jedoch oft nicht laufend mit dem Code synchronisiert. Während der Entwicklung neigen die Modelle dazu, zu veralten und können irreführend sein, da sich die Software nicht so verhält wie modelliert.

Unser Framework Codeling synchronisiert Modelle und Code, indem es die Notwendigkeit von Modellen als separates Artefakt eliminiert. Das Tool übersetzen Modelle in Quellcode-Muster und zurück. Dies ermöglicht es, bei Bedarf zuverlässig Modelle wie Zustandsautomaten aus dem Quellcode zu extrahieren und Änderungen in den extrahierten Modellen an den Code weiterzugeben. Der generierte Code hat wohldefinierte Schnittstellen für die Interaktion mit seinem kontextuellen Code, d.h. dem Rest des Systems. Derzeit bietet der von Codeling generierte Quellcode keine Mittel, um die Ausführung von SM in einer modellbasierten Umgebung systematisch zu monitoren. In dieser Bachelorarbeit untersuchen Sie, wie modellbasiertes Monitoring für SM mit dem Monitoring von Quellcode kombiniert werden kann.

- Wie können SM in Java implementiert werden, damit das Modell aus dem Code extrahiert werden kann?
- Wie kann die Semantik jedes Elements in der Sprache (dem Metamodell) des SM in Quellcode-Mustern kodiert werden?
- Wie kann man das modellbasierte Monitoring von SM ermöglichen, wenn sie als Code ausgeführt werden?

Für die Durchführung der Arbeit benötigte Kenntnisse: Java
Hilfreiches Wissen: Modellbasierte Entwicklung, Java Reflection

Motivation — English

State Machines (SM) are widely used in practice for describing the behavior of software. A benefit of modeling software behavior with SM is their visual nature during design and runtime. The design implied by SMs is then usually implemented with a programming language. This code can be generated so that only the technical details of the behavior need to be implemented manually, and it can be ensured that the program behaves exactly as modeled. However, in practice the models are often not continuously synchronized with the code. During the development, the models tend to become outdated and may be misleading, because the software does not behave as modeled.

Our framework Codeling synchronizes models and code by eliminating the need for models as a separate artifact. The tool translates models into source code patterns and back. This allows to reliably extract models such as Activity Diagrams from the source code on demand, and to propagate changes in the extracted models to the code. The generated code has well-defined interfaces for interacting with its contextual code, i.e. the rest of the system. Currently, the source code generated by Codeling does not provide means to systematically monitor the execution of SM in a model-based environment. In this bachelor's thesis you will investigate how model-based monitoring for SM can be combined with monitoring of source code.

You will therefore consider the following questions:

- How can SM be implemented in Java so that the model can be extracted from the code?
- How can the semantics of each element in the SM's language (the meta model) be encoded in code

templates?
- How to enable model-based monitoring of SM when they are executed as code?

Knowledge requirements: Java
Helpful knowledge: Model-based development, Java Reflection

Organisatorisches

Kontakt:
Dr. Marco Konersmann <konersmann@uni-koblenz.de>