# Model Checking and Compiling UML State Machines


**Diplomarbeit Einführung**

**Timm Schäfer**

**Ludwig-Maximilians-Universität München**

**Institut für Informatik,**
**Lehr- und Forschungseinheit für Programmierung und Softwaretechnik**

| | |
|---|---|
| **Aufgabensteller:** | **Prof. Dr. Martin Wirsing** |
| **Betreuer:** | **Dr. Alexander Knapp**<br>**Dr. Stephan Merz** |
| **Datum:** | **27.6.2001** |

**Abstract**

We describe HUGO, a prototype tool designed to facilitate the use of UML state machines in software projects. On the one hand it automatically verifies state machines using the model checker SPIN, on the other hand it compiles state machines into Java code that can be integrated into any Java application. The model checking feature is primarily designed to verify whether interactions expressed by certain collaborations can indeed be realized by a set of state machines; additionally a check for possible deadlocks can be performed. The compilation into Java code bridges a gap between the behavioral specification of Java classes, using state machines, and their implementation.

**Modellprüfung und Übersetzung von UML-Zustandsmaschinen**

**Zusammenfassung**

Diese Diplomarbeit beschreibt das Prototyp-Werkzeug HUGO, das entworfen wurde um den Einsatz von UML-Zustandsmaschinen in Software-Projekten zu erleichtern. Einerseits verifiziert es Zustandsmaschinen mit Hilfe des Modellprüfers SPIN, andererseits übersetzt es Zustandsmaschinen in Java-Code, der in beliebige Java-Anwendungen integriert werden kann. Die Modellprüfung wurde in erster Linie entworfen, um zu verifizieren ob Interaktionen, die in bestimmten Kollaborationen ausgedrückt werden, auch tatsächlich durch eine Menge von Zustandsmaschinen realisiert werden können. Ergänzend kann eine Prüfung auf mögliche Verklemmungen durchgeführt werden. Die Übersetzung in Java-Code schließt eine Lücke zwischen einerseits der Verhaltensspezifikation von Java-Klassen durch Zustandsmaschinen und andererseits deren konkreten Implementierung.

# 1 Introduction

The Unified Modeling Language (UML [2]) is a graphical language for specifying software systems. It has been standardized by the Object Management Group and accepted as the de facto standard notation for the design of object-oriented software.

The UML provides several diagram types to model different views on the design of a software system. The static aspects of a system are mainly modeled in class and object diagrams: class diagrams contain classifiers (primarily classes and interfaces), connected by their various static relationships, object diagrams contain instances of classifiers and relationships and show examples of system states at certain points. Dynamic aspects are modeled with statechart and interaction diagrams. Statechart diagrams describe possible sequences of states that an instance can be in during its life time. Interaction diagrams show interactions among instances and may be represented in two ways: sequence diagrams show the interactions arranged in time sequence, collaboration diagrams focus on the relationships between instances and their different roles, using numbers to indicate the sequence of interactions.

A statechart diagram is a graph that represents a state machine. It may be associated with a class to specify the states that its instances can be in and their reactions to discrete events. Statechart diagrams are typically used to model the behavior of objects that must respond to external events or whose current behavior depends on their past. They are particularly useful for specifying reactive systems.

UML statecharts are an object-based variant of Harel statecharts [1], which are a hierarchical extension of state transition diagrams; the major differences relative to classical Harel statecharts can be found in [2, p. 2-180]. There are several other variants of statechart formalisms, which are compared in [3].

This thesis describes a prototype tool, called HUGO, that is designed to automatically verify UML state machines using the model checker SPIN and to generate Java code that can be incorporated into any Java application. The main objective of the HUGO development is to facilitate state machine integration into software projects. The model checking feature is primarily designed to verify specific collaborations between several objects, by checking whether certain interactions can indeed be realized by the state machines that model the objects' behaviors. It can also be used to check for possible deadlocks. The compilation into Java code allows a seamless integration of the behavioral specification of Java classes through statecharts. The classes send or pass on events to the compiled state machines, which change their states accordingly and execute any triggered actions. Actions are coded directly into the statechart diagram and may contain a simple method call or an entire Java algorithm.

## 2 References


1. D. Harel. Statecharts: A visual formalism for complex systems. *Science of Computer Programming,* vol. 8, pp. 231-274. 1987.
2. Object Management Group. Unified Modeling Language Specification, Version 1.3., OMG, 1999. `http://cgi.omg.org/cgi-bin/doc?ad/99-06-08`.
3. Michael von der Beeck. A Comparison of Statechart Variants. In H. Langmaack, W.-P. de Roever, and J. Vytopil, editors, *Formal Techniques in Real-Time and Fault-Tolerent Systems,* volume 863 of *Lecture Notes in Computer Science,* pages 128-148. Springer, 1996.