**Lehrstuhl für Softwaretechnik und Softwareproduktionsumgebung, Institut für Informatik, Albert-Ludwigs-Universität Freiburg**

*David Basin, Luca Viganò, Burkhart Wolff*

# Übersicht

Die Arbeitsgruppe „Softwaretechnik und Softwareproduktionsumgebung" (kurz: *Softwaretechnik*) wurde 1997 in der neu eingerichteten Fakultät für angewandte Wissenschaften an der Albert-Ludwigs-Universität Freiburg gegründet. An der Fakultät lehren und forschen zur Zeit 12 Professoren im Fach Informatik und 10 Professoren im Fach Mikrosystemtechnik; insgesamt sind 33 Professuren für die Fakultät vorgesehen.

Die Gruppe wird von Prof. Dr. David Basin geleitet und besteht momentan aus 10 wissenschaftlichen Mitarbeitern. Forschungsschwerpunkte der Gruppe sind Methoden und Werkzeuge zur Spezifikation, Verifikation und zum Testen von Soft- und Hardwaresystemen. Besonderes Gewicht liegt auf *Formalen Methoden* und deren Anwendungen. Im Folgenden gehen wir auf unsere Aktivitäten in der Lehre und in der Forschung ein.

# Lehre

Die Arbeitsgruppe Softwaretechnik veranstaltet in jedem Sommersemester eine gleichnamige Kursvorlesung als Pflichtveranstaltung im Hauptstudium.

Die Vorlesung befasst sich mit der Entwicklung von großen und komplexen Systeme, insbesondere mit Entwicklungsprozeß-Modellen, Anforderungsanalysen, der Modellbildung, formalen Spezifikationen, Entwurfs- und Testmethoden. Ein wichtiger Aspekt sind Unterstützungswerkzeuge (von Versionsmanagementwerkzeugen bis hin zu Case-Tools) sowie Sprachparadigmen, die strukturierte Entwicklungen im Großen unterstützen (Modularisierung und Objektorientierung, komponentenbasierter Systementwurf, formale Systemarchitekturen, etc.). Die Kursvorlesung ist projektbasiert, d.h. die Studenten bearbeiten in den begleitenden Übungen in Arbeitsgruppen zusammen ein Softwareprojekt in allen Phasen des Entwicklungsprozesses.

Zusätzlich bietet die Gruppe eine Reihe von Vertiefungsveranstaltungen zu Themen der Softwaretechnik und anliegenden Bereichen an. Kürzlich angebotene Veranstaltungen umfassen:

**Modellbasierte Systemspezifikation:** Spezifikation von Systemanforderungen unter Verwendung modellbasierter Spezifikationssprachen (wie Z oder B), und Methoden und Werkzeugunterstützung von formalen Verfeinerungen von Spezifikationen in Software.

**Ereignisorientierte Modellierung:** Logische Formalismen (z.B. StateCharts oder Prozeßkalküle) und Werkzeuge (Statemate oder Modelchecker wie

SPIN oder FDR) zur Modellierung und Validierung von verteilten Systemen und Protokollen und deren Anforderungen.

**Programm-Verifikation:** Maschinengestützte Validierung von Algorithmen unter Verwendung von Logik höherer Stufe (HOL) sowie Spezifikationssprachen, die in HOL eingebettet worden sind (z.B. Z oder CSP).

**IT-Sicherheit:** Theorie und Praxis der Sicherheit von Computersystemen. Hier wird ein weites Feld von Kryptographie, public-key-Infrastrukturen, Zugangskontrollen, Sicherheitsmodellen, Sicherheitsprotokollen und Sicherheit in offenen Netzen abgedeckt.

**Testen:** Verfahren zum Testen von Soft- und Hardwaresystemen, d.h. zur systematischen Generierung von Tests aus formalen Spezifikationen oder Code.

Einige unserer Veranstaltungen werden als Teil des VIROR-Projekts — Virtuelle Universität Oberrhein — angeboten. Dies bedeutet, dass sie als multimediale, verteilte Lehrveranstaltungen im Verbund mit verschiedenen Partneruniversitäten entwickelt und ausgestrahlt werden.

# Forschung

Die Kernkompetenz der Arbeitsgruppe Softwaretechnik liegt auf dem Gebiet der Formalen Methoden, d.h. basiert auf mathematischer Modellierung und formaler Logik sowie deren Anwendungen. Ein verbreitetes Vorurteil gegen Formale Methoden ist, dass ihre Nutzung in der Praxis zu schwierig und resourcenintensiv sei. Unsere Forschung befasst sich mit unterschiedlichen Anwendungsszenarien, die mit unterschiedlichen Kosten verbunden sind. Das geht von „leichtgewichtigen“ Ansätzen wie statischer Programmanalyse bis hin zu vollständiger formaler Verfeinerung oder Verifikation. In Abhängigkeit der Anwendungen und dem Ausmaß, in dem diese sicherheitskritisch, „mission-critical“ oder einfach *komplex* sind, können unterschiedliche Ansätze zu höchst unterschiedlichen Anforderungen an den Entwicklungsprozeß und die eingesetzten Resourcen führen.

## Projekte

**Formale Methoden und sicherer mobiler Code.** In den letzten drei Jahren haben wir zwei Projekte mit der *Deutschen Telekom* über die Anwendung Formaler Methoden durchgeführt, die sich mit der Sicherstellung von Sicherheitseigenschaften in mobilem Code befassen. Das Problem: Mobiler Code nutzt Speicherresourcen des Host-Rechners, und es muss sichergestellt werden, dass dies nur in einer eingeschränkten, sicheren Weise geschieht (d.h. auf den Speicher wird nur in einem vordefinierten Adressbereich zugegriffen). Unsere Implementierung basiert auf abstrakter Interpretation und Modelchecking: Ein Programm im Java-Byte-Code wird in ein endliches Transitionssystem abstrahiert, für welches automatisch temporal-logische Formeln erzeugt werden, welche die Speicherzugriffseigenschaften beschreiben. Durch Modelchecking kann automatisch geprüft werden, ob alle Speicherzugriffseigenschaften vom Code eingehalten werden. Unsere Lösung bietet gegenüber anderen Ansätzen der Byte-Code-Verifikation eine Reihe von Vorteilen, u.a. die Erweiterbarkeit und Kombinierbarkeit mit anderen Formalen Methoden.

**Sicherheitsarchitekturen.** Vor kurzem haben wir die Bearbeitung eines DFG-Projektes begonnen, welches sich mit der Modellierung und Validierung von Sicherheitseigenschaften in offenen und verteilten Systemen beschäftigt. Unser Ziel ist es, Methoden und Werkzeugunterstützung zu entwickeln, mit deren Hilfe Sicherheitsanforderungen auf verschiedenen Ebenen des Systementwurfs garantiert werden können. Insbesondere werden *Anforderungsarchitekturen* formal spezifiziert, welche die abstrakte Sicht auf Prozesse und deren Kommunikation beschreiben. Diese werden anschliessend durch *Implementierungsarchitekturen* verfeinert, wo bereits eine Abbildung auf konkrete, komponentenbasierte Middleware-Technologien vorgenommen worden ist. Wir untersuchen insbesondere das technische Problem, wie Sicherheitseigenschaften einerseits auf Anforderungsarchitekturebene formalisiert und andererseits auf Implementierungsarchitekturebene garantiert werden können. Ein Teil der Herausforderung besteht darin, dass einige der unterschiedlichen Middleware-Technologien bereits ein eigenes Sicherheitsmodell haben (wie beispielsweise das CORBA-Sicherheitsmodell, welches z.Zt. durch die OMG definiert wird). Dies muss Bestandteil der Formalisierung sein und im Rahmen des Verfeinerungsbeweises eine Rolle spielen.

**Formale Methoden in CASE-Werkzeugen für verteilte Objekt-Systeme.** UML ist eine populäre semi-formale Modellierungssprache zum Entwurf verteilter Objekt-Systeme. In einem Verbundprojekt mit der Interactive Objects GmbH untersuchen wir, wie formale Spezifikationssprachen (z.B. OCL, Z oder VDM) in UML-basierte CASE-Werkzeuge integriert werden können. Insbesondere untersuchen wir formale Erweiterungen von UML, welche die Spezifikation von Vor- und Nachbedingungen sowie Invarianten erlauben. Das Ziel ist eine „leichtgewichtige" Formale Methode, die es beispielsweise während des Entwurfs erlaubt, aus formalisierten Objekt-Constraints automatisch Testdatensätze für die Endimplementierung zu generieren.

**Werkzeugunterstützung für kombinierte Formale Methoden.** Die zuvor beschriebenen beiden Projekte setzen die Kombination von Spezifikationsformalismen voraus. Die folgenden zwei Projekte arbeiten an den Grundlagen der Unterstützung solcher Kombinationen.

Zum Ersten sind wir Teil der IST-Arbeitsgruppe „Computergestütztes Beweisen in Typ-Theorie", gefördert durch die Europäische Gemeinschaft. Das Ziel der Arbeitsgruppe ist es, formale Entwicklungswerkzeuge auf der Basis von Typ-Theorien und getypten Sprachen zu entwickeln, sowie deren Anwendung in den Bereichen Programmverifikation und -synthese und in der formalen, maschinengestützten Mathematik zu untersuchen.

Zum Zweiten untersuchen wir zusammen mit dem Computer Systems Laboratory am SRI (Menlo Park, USA) Werkzeugunterstützung für Modelchecking-Systeme, die in heterogenen Sprachen spezifiziert worden sind. Unser Ansatz basiert insbesondere auf der Verwendung von *Rewriting Logic* (einer mächtigen ausführbaren Spezifikationssprache) als einem einheitlichen Spezifikationsformalismus zur Kombination. Darüber hinaus verwenden wir Abstraktionstechniken, um die in Rewriting Logic spezifizierten Systeme auf endliche Zustandsübergangssysteme zu reduzieren. Außerdem werden Modelchecking-Techniken in das Maude System (einer optimierten Implementierung von Rewriting Logic) integriert.

## Werkzeuge

Ein Teil der Forschungsarbeit unserer Gruppe ist die Werkzeugentwicklung, um die Effektivität von Formalen Methoden zu demonstrieren und sie auf realistische Probleme in industrieller Größenordnung anzuwenden. Nachstehend listen wir eine Auswahl von Systemen und Systemkomponenten auf, die wir für diesen Zweck entwickelt haben:

**Byte-Code-Verifikation durch Modelchecking:** Ein System (siehe oben), welches in Zusammenarbeit mit der Deutschen Telekom entwickelt wurde und das zum Modelchecking von Sicherheitseigenschaften des Java Card Subsets des Java-Byte-Codes dient.

**BMC:** Ein System zur Generierung von Gegenbeispielen, um Fehler in Systemen zu finden, die in der mächtigen monadischen Logik zweiter Stufe auf Zeichenketten (M2L-Str) spezifiziert worden sind.

**BLIF2MONA:** Ein Verifikationssystem für BLIF (Berkeley Logic Interchange Format), einer Hardware-Beschreibungssprache, die für die hierarchische Beschreibung von sequentiellen Schaltkreisen geeignet ist. Hierbei werden Schaltkreisbeschreibungen in die entscheidbare monadische Logik zweiter Stufe übersetzt, für welche wir Werkzeugunterstützung für Schaltkreissimulation, Testen und Verifikation anbieten (basierend auf dem MONA- und dem BMC-System).

**SML_TK, IsaWin, and TAS:** Eine Suite von Komponenten, um graphische Oberflächen für formale Entwicklungswerkzeuge zu entwickeln. Ein Setup der Basiskomponenten liefert *IsaWin*, ein graphisches Interface für das Isabelle-System. Ein weiteres Setup ist TAS, ein System auf Basis von Isabelle/IsaWin, mit dem transformationelle Programmentwicklungen aus Spezifikationen interaktiv durchgeführt werden können. Beweisverpflichtungen können dabei in Isabelle bearbeitet werden. Wir haben konservative logische Einbettungen von Spezifikationssprachen wie Z und CSP in Logik höherer Stufe entwickelt, die in TAS und IsaWin benutzt werden können.

## Weitere Informationen

Weitere Informationen über unsere Lehr- und Forschungsaktivitäten sind auf den WWW-Seiten unserer Arbeitsgruppe

`www.informatik.uni-freiburg.de/~softech`

verfügbar oder können direkt von uns bezogen werden.

**Kontaktadresse:**
Prof. Dr. David Basin
Institut für Informatik
Albert-Ludwigs-Universität Freiburg
Universitätsgelände Flugplatz
D-79110 Freiburg i. Br.