http://swt.cs.tu-berlin.de

Fachgebiete **Softwaretechnik** und **Programmierung eingebetteter Systeme**
Technische Universität Berlin, Fakultät IV - Elektrotechnik und Informatik

http://pes.cs.tu-berlin.de

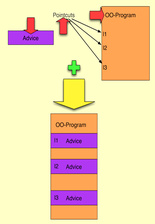
Der Korrektheit ...

Testen

Am DCAITI werden methodische Grundlagen für eine effektive Testfallwiederverwendungsstrategie im Entstehungszyklus eingebetteter Software im Automobilbereich erarbeitet. Deren Praxistauglichkeit prüfen wir an Hand der innovativen Testtechnologie TPT (Time Partition Testing). Ein weiterer Forschungsschwerpunkt ist die Entwicklung von Testmethoden, die evolutionäre Algorithmen, wie z.B. Genetische Algorithmen einsetzen, um den Testdatengenerierungsprozess vollständig zu automatisieren. Einsatz findet dieses Vorgehen bereits bei der Automatisierung von funktionalen Tests, Zeitverhaltenstests und Strukturtests, implementiert im evolutionären Testsystem ETS.

Formalismen für AOP

Aspektorientierung ist ein neuartiges Paradigma von Programmiersprachen, welches eine flexiblere Anpassung von Programmen verspricht. Im DFG Projekt ASCOT arbeiten wir an formalen Modellen, um eine mathematische Grundlage für dieses Paradigma zu etablieren. Die Arbeit basiert auf einem selbstentworfenen Kernkalkül, vollständig implementiert im interaktiven Theorembeweiser **Isabelle/HOL**. Wir beweisen auf dieser Grundlage Eigenschaften wie Kompositionalität und Sicherheit und etablieren so beherrschbare Flexibilität. Diese Eigenschaften setzen wir praktisch in Typsystemen um, deren Sicherheit wir beweisen. Solche Typsysteme ermöglichen eine statische Analyse aspektorientierter Programme.

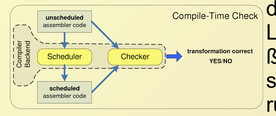


Echtzeitverifikation

Eingebettete sicherheitskritische Software-Systeme sind allgegenwärtig. Ihre korrekte Funktionsweise ist als die Einhaltung von funktionalen und nicht-funktionalen Eigenschaften (z.B. Echtzeitfähigkeit und Lebendigkeit) definiert. Im DFG Projekt VATES entwickeln wir Grundlagen und Methoden zur Konstruktion und Verifikation solcher Systeme. Wir betrachten die gesamte Kette der Entwicklung von Spezifikation und Quellcode bis zum von Compilern generierten Maschinencode. Um Komplexität zu reduzieren, verwenden wir eigenschaftserhaltende Abstraktionen. Verifikationsaufgaben werden mit Model-Checkern und Theorembeweisern bearbeitet. Das echtzeitfähige Betriebssystem **BOSS**, das an der FhG-First entwickelt wurde, dient als Anwendungsbeispiel.

Sichere Compiler

Moderne Compiler übersetzen Programme aus Hochsprachen in Maschinencode und optimieren dabei z.B. bezüglich Laufzeit oder Codegröße. Immer wieder passiert es, dass Optimierungen das Verhalten eines übersetzten Programms verändern. Ziel unserer Forschung ist es, Korrektheit für optimierende Compiler nachzuweisen. Der Fokus liegt dabei auf VLIW-Prozessoren, die mehrere Anweisungen parallel ausführen. Die Parallelisierung wird dabei durch den Compiler vorgenommen. Wir entwickeln im Theorembeweiser **Isabelle/HOL** formale Modelle, auf deren Grundlage wir die Korrektheit von Optimierungsalgorithmen untersuchen und leiten aus diesen Formalisierungen *Checkerprogramme* ab, die die korrekte Implementierung eines Algorithmus prüfen.



Testen
Formalisieren
Spezifizieren
Verifizieren
Forschen
Lehren und Lernen
Umsetzen

Projekte

- **ASCOT (Aspect and Collaboration Theory)**
  Florian Kammüller, Henry Sudhof
- **Aktionsplan Informatik**
  Lars Gesellensetter, Elke Salecker
- **VATES (Verification and Transformation of Embedded Systems)**
  Steffen Helke, Björn Bartels, Thomas Göthel, Moritz Kleine
- **DCAITI (Daimler Center for Automotive IT Innovations)**
  Felix Lindlar, Abel Marrero, Andreas Windisch, u.a.

Lehre

In folgenden Lehrveranstaltungen werden Techniken zur Qualitätssicherung eingebetteter Systeme behandelt.

PES
- Software Engineering Eingebetter Systeme
- Semantik und Analyse von Software-Systemen

SWT
- Softwarequalitätsmanagement im Automotive Software Engineering
- Verifikation von C-Programmen in der Praxis
- Sicherheitsaspekte in der Softwaretechnik

Werkzeuge

Testwerkzeuge
- ermöglichen die graphische Modellierung von Testfällen
- können Testdaten .B. anhand von Strukturinformationen des Quellcodes automatisch generieren

Theorembeweiser
- führen mathematische Beweise maschinengestützt durch
- die Beweisführung erfolgt teilweise interaktiv, d.h. der Nutzer liefert Hinweise, wie ein Beweis zu führen ist

Model-Checker
- führen Beweise für ein gegebenes Modell und eine Eigenschaft automatisch durch Überprüfung des Zustandsraum des Modells
- erfordern endliche Modelle

... auf der Spur.

Prof. Dr. Stefan Jähnichen

Prof. Dr. Sabine Glesner