ISF WANTS YOU!

Projekt- und Abschlussarbeiten

Towards Taxonomy-based SPL Engineering

- **Context**
  - Software taxonomies structure software domains from an abstract specification of the functionality to concrete implementable variants by successive correctness-preserving refinements.
  - Software product line engineering (SPLE) allows developing these software systems by managed large-scale reuse
- **Objective**: Design a taxonomy-based SPLE process and evaluate it with respect to maintainability and evolvability
- **Tasks**
  - Devise guidelines for code structure of product line artifacts (with pre-processor annotations) based on taxonomy
  - Implement SPL for SPARE Time taxonomy
  - Compare code structure of SPARE Time SPL with original SPARE Time toolkit structure
- **Information**: Master's thesis
- **Contact**: Ina Schaefer (`i.schaefer@tu-bs.de`)

Method Call Treatment in Deductive Verification

- **Context**: In design by contract, scalability of deductive verification depends to a great extent on method call treatment (i.e., whether a called method is inlined or its respective contract is used instead).
- **Goal**: We studied this parameter in the verification system KeY with respect to provability, verification effort, and specification effort, but want to extend our study to at least two other verification systems. Moreover, the goal is to introduce a new keyword to JML to explicitly mark methods that should be either inlined or whose contract should be used. An optional goal is to define a heuristic that may reason about which approach is more profitable for a given verification task.
- **Information**: Master's thesis or project work
- **Contact**: Alexander Knüppel (`a.knueppel@tu-bs.de`)

Towards Automating Interactive Proofs

- **Context**: Based on current technology in deductive verification, most proofs are constructed interactively (i.e., automation stops and the verification system asks for user interaction). Reasons are manifold: Choice of parameters, insufficient specification, time out...
- **Goal**: Study prominent interactive proofs (e.g., TimSort) and investigate how to automate all steps that were performed interactively.
- **Information**: Master's thesis or project work
- **Contact**: Alexander Knüppel (`a.knueppel@tu-bs.de`)

Mutation Analysis for Deductive Verification

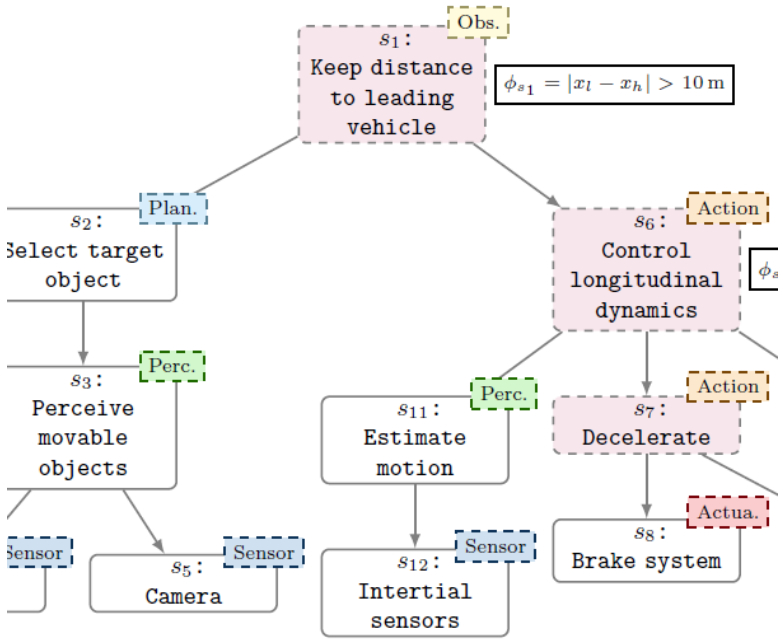
- **Context**: Specifying programs is challenging and error-prone. In dependable software, many specifications are also oftentimes too weak for a successful verification. Mutation testing is used to evaluate the quality of test cases. Likewise, mutation analysis may be used for evaluating the *quality* of specifications.
- **Goal**: Develop a mutation analysis framework for programs written in Java and specified with JML. Define mutation operators and evaluate the potential of the framework.
- **Information**: Master's thesis or project work
- **Contact**: Alexander Knüppel (`a.knueppel@tu-bs.de`)

Formalization of Information Flow in Coq

- **Context**: In the context of information flow, variables are mapped to a security level. Not all flows are desirable (i.e., we want to prohibit leakage of private information to public sinks). That means that no flows from higher security levels to lower ones should exist. We proposed and formalized a concept to ensure a correct information flow *by construction*. Instead of a subsequent analysis, we know which information is confidential and develop our programs accordingly.
- **Goal**: The current formalization exists only on paper. We want to use Coq (an interactive proof assistant) to mechnanize this formalization and to increase trust.
- **Information**: Master's thesis or project work
- **Contact**: Alexander Knüppel (`a.knueppel@tu-bs.de`) & Tobias Runge (`tobias.runge@tu-bs.de`)

Tool Support for Composing Skill Graphs in Skeditor

- **Context**: Skeditor is a graphical editor for modeling, specifying, and verifying skill graphs. Skill graphs are a means for modeling cyber-physical systems in abstract and modular fashion.
- **Problem**: In recent work, we defined the composition of skill graphs formally. Currently, no implementation/tool support for the composition exists.
- **Envisioned Solution**: Extend Skeditor with a composition functionality. Several correctness checks have to be made before a successful composition can be established.
- **Information**: Bachelor's thesis or project work
- **Contact**: Alexander Knüppel (`a.knueppel@tu-bs.de`)



Softwareproduktlinien im Correctness-by-Construction Editor

- Kontext: Correctness-by-Construction (CbC) ist ein Verfahren zur Softwareentwicklung. Hierbei werden Programme durch Korrektheit bewahrenden Verfeinerungen erstellt. Softwareproduktlinien werden verwendet um eine Menge von Programmvarianten zu verwalten.
- Problem: Der CbC Editor kann nur einzelne Varianten auf Korrektheit überprüfen
- Ziel: Erweiterung des Editors, sodass Familien effizient bewiesen werden können
- Geplant als Masterarbeit
- Ansprechpartner: Tabea Bordis (`t.bordis@tu-bs.de`) & Tobias Runge (`tobias.runge@tu-bs.de`)

Abstract Execution (AE) zur Entwicklung von Programmen mittels Correctness-by-Construction (CbC)

- Kontext: CbC ist ein Verfahren zur Softwareentwicklung. Hierbei werden Programme durch Korrektheit bewahrenden Verfeinerungen erstellt. Abstract Execution ist ein Beweisverfahren, welches mittels Spezifikation von abstrakten Programmteilen, die Korrektheit des gesamten Programms beweisen kann.
- Motivation: Beim inkrementellen Entwickeln mittels CbC kann der Programmierer im Laufe der Entwicklung zu unlösbaren Beweisverpflichtungen gelangen. Dies kann von vornherein mit Abstract Execution vermieden werden, da jedes (Teil-)Programm bewiesen werden kann und somit “Sackgassen” vermieden werden können.
- Ziel: Erstellung eines Editors zur inkrementellen Entwicklung von Programmen mit Abstract Execution.
- Geplant als Masterarbeit
- Ansprechpartner: Tobias Runge (`tobias.runge@tu-bs.de`)

Nutzerstudie zur Benutzerfreundlichkeit des Correctness-by-Construction Editors

- Kontext: Correctness-by-Construction (CbC) ist ein Verfahren zur Softwareentwicklung. Hierbei werden Programme durch Korrektheit bewahrenden Verfeinerungen erstellt. Hierfür existiert bereits der Editor CorC der CbC unterstützt.
- Problem: Der CbC-Editor wurde noch nicht quantitativ evaluiert. Ergebnisse zur Fehlerhäufigkeit in erstellten Programmen und Implementierungszeiten sind als Grundlage zur weiteren Entwicklung des Editors nützlich.
- Ziel: Erstellung (und Durchführung) einer Nutzerstudie, um die Benutzerfreundlichkeit von CorC zu evaluieren.
- Geplant als Masterarbeit
- Ansprechpartner: Tobias Runge (`tobias.runge@tu-bs.de`)

Taxonomy-basierte Algorithmenentwicklung

- Kontext: Software Taxonomien strukturieren Software von abstrakten Spezifikationen zu konkreten Varianten. Diese Varianten haben hierdurch klar definierte Gemeinsamkeiten und Unterschiede. Correctness-by-Construction (CbC) ist ein Verfahren zur Softwareentwicklung, das Programme durch Korrektheit bewahrenden Verfeinerungen erstellt.
- Ziel: Programmierverfahren entwickeln, dass die Implementierung einer Taxonomie von Algorithmen unterstützt. Hierbei wird die Korrektheit der Algorithmen mit Hilfe vom Correctness-by-Construction Verfahren sichergestellt.
- Vorgehen: Bestehende Verfahren analysieren und ein Methodik entwickeln, die die gleichzeitige Implementierung von relatierten Algorithmen erlaubt.
- Geplant als Masterarbeit
- Ansprechpartner: Tobias Runge (`tobias.runge@tu-bs.de`)

Fallstudie zu Taxonomie-basierter Softwareentwicklung

- Kontext: Software Taxonomien strukturieren Software von abstrakten Spezifikationen zu konkreten Varianten. Die Varianten werden hierbei durch Korrektheit bewahrenden Verfeinerungen erstellt (Correctness-by-Construction).
- Ziel: Fallstudie in eine Taxonomie übertragen und Funktionalität nach dem Correctness-by-Construction Prinzip implementieren.
- Geplant als Projektarbeit
- Ansprechpartner: Tobias Runge (`tobias.runge@tu-bs.de`)

Identifikation von Feature-Interaktionen

- Kontext: Linux Patches werden automatisch gegen vordefinierte und zufällig erzeugte Konfigurationen getestet und Entwickler per Email auf fehlerhafte Konfigurationen hingewiesen.
- Problem: Eine Konfiguration enthält irgendeine Auswahl der mehr als 10.000 Features und Entwickler müssen die interagierenden Features manuell identifizieren
- Ziel: Durch die Kombinatorik in fehlerhaften/fehlerfreien Konfigurationen und Patch sollen weitere Konfigurationen generiert werden, um potentielle Feature-Interaktionen zu isolieren.
- Voraussetzung: Software-Produktlinien
- Geplant als Masterarbeit
- Ansprechpartner: Tobias Pett (`t.pett@tu-bs.de`), Thomas Thüm und Sebastian Krieter (Magdeburg)

Variable Prädikate für Correct-by-Construction Produktlinien

- **Kontext:** Correctness-by-Construction (CbC) ist ein Verfahren zur Softwareentwicklung. Hierbei werden Programme mit Kontrakten versehen und durch Korrektheit bewahrenden Verfeinerungen erstellt. Software Produktlinien (SPLs) werden verwendet, um eine Menge von Programmvarianten zu verwalten. Um SPLs zu implementieren, werden variable Codekonstrukte verwendet. Damit diese Variabilität auch in den Kontrakten wiedergespiegelt werden kann, sollen variable Prädikate integriert werden.
- **Problem:** CbC unterstützt ursprünglich keine Variabilität in den Kontrakten.
- **Ziel:** Entwicklung eines Konzepts für variable Prädikate im Zusammenhang von CbC SPLs
- Geplant als Masterarbeit
- **Ansprechpartner:** Tabea Bordis (`t.bordis@tu-bs.de`) & Tobias Runge (`tobias.runge@tu-bs.de`)

Untersuchung verschiedener Variationspunkte für Variational Correctness-by-Construction

- **Kontext:** Correctness-by-Construction (CbC) ist ein Verfahren zur Softwareentwicklung. Hierbei werden Programme mit Kontrakten versehen und durch Korrektheit bewahrenden Verfeinerungen erstellt. Software Produktlinien(SPLs) werden verwendet, um eine Menge von Programmvarianten zu verwalten. Variational CbC ist ein Verfahren, um SPLs mit einer konkreten SPL-Technik namens feature-orientierter Programmierung korrekt zu erstellen. Mit anderen Variationspunkten lassen sich eventuell andere Softwaresysteme besser erstellen.
- **Ziel:** Eine Untersuchung verschiedener Variationspunkte für Variational CbC
- Geplant als Masterarbeit
- **Ansprechpartner:** Tabea Bordis (`t.bordis@tu-bs.de`) & Tobias Runge (`tobias.runge@tu-bs.de`)

Fallstudie(n) für Correct-by-Construction Software Produktlinien im Tool VarCorC

- **Kontext:** VarCorC ist ein Tool zur Entwicklung von Software Produktlinien (SPLs) mittels Correctness-by-Construction (CbC). CbC ist ein Verfahren zur Softwareentwicklung. Hierbei werden Programme mit Kontrakten versehen und durch Korrektheit bewahrenden Verfeinerungen erstellt. SPLs werden verwendet, um eine Menge von Programmvarianten zu verwalten.
- **Ziel:** Erstellung von einer oder mehrerer Fallstudie(n) im Tool VarCorC
- Geplant als Bachelor- oder Projektarbeit
- **Ansprechpartner:** Tabea Bordis (`t.bordis@tu-bs.de`) & Tobias Runge (`tobias.runge@tu-bs.de`)

Verbesserung der Benutzerfreundlichkeit von VarCorC

- **Kontext:** VarCorC ist ein Tool zur Entwicklung von Software Produktlinien mittels Correctness-by-Construction (CbC). CbC ist ein Verfahren zur Softwareentwicklung. Hierbei werden Programme durch Korrektheit bewahrenden Verfeinerungen erstellt. Software Produktlinien werden verwendet, um eine Menge von Programmvarianten zu verwalten.
- **Problem:** Um VarCorC gut verwenden zu können, benötigt man viel Fachwissen
- **Ziel:** Die Benutzerfreundlichkeit verbessern, sodass der Entwickler bei der Entwicklung unterstützt wird
- Geplant als Bachelor- oder Projektarbeit
- **Ansprechpartner:** Tabea Bordis (`t.bordis@tu-bs.de`) & Tobias Runge (`tobias.runge@tu-bs.de`)

Produktgenerierung im Tool VarCorC

- **Kontext:** VarCorC ist ein Tool zur Entwicklung von Software Produktlinien (SPLs) mittels Correctness-by-Construction (CbC). CbC ist ein Verfahren zur Softwareentwicklung. Hierbei werden Programme mit Kontrakten versehen und durch Korrektheit bewahrenden Verfeinerungen erstellt. SPLs werden verwendet, um eine Menge von Programmvarianten zu verwalten. Bei SPLs ist es üblich, dass über eine Konfiguration der Bestandteile, einzelne Varianten generiert werden können. Diese Generierung soll in das Tool VarCorC integriert werden.
- **Ziel:** Integration der Produktgenerierung in VarCorC
- Geplant als Projektarbeit
- **Ansprechpartner:** Tabea Bordis (`t.bordis@tu-bs.de`) & Tobias Runge (`tobias.runge@tu-bs.de`)

Praktika und Teamprojekte