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

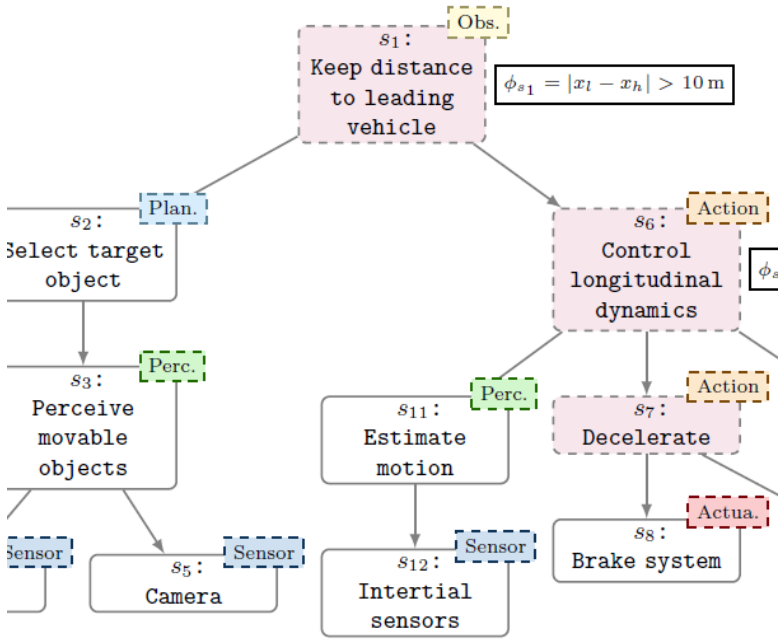
- **Context**: Specifying programs is challenging and error-prone. In dependable software, many specifications are also oftentimes too weak for a successful verification. Mutation testing is used to evaluate the quality of test cases. Likewise, mutation analysis may be used for evaluating the *quality* 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`)



# Trait Correctness-by-Construction

- Kontext: Correctness-by-Construction (CbC) ist ein Verfahren zur inkrementellen Softwareentwicklung. Hierbei werden Programme durch Korrektheit bewahrenden Verfeinerungen erstellt. Traits sind ein Programmierkonstrukt welches es erlaubt Methoden unabhängig von einer Vererbungshierarchie zu implementieren und dadurch die Wiederverwendung erleichtert.
- Problem: Die vorgegebenen CbC-Regeln schränken Programmierer stark ein.
- Ziel: Traits zur inkrementellen Erstellung von Programmen nutzen. Durch Trait-Komposition können Verfeinerungen simuliert werden, ohne Programmierer durch Verfeinerungsregeln einzuschränken. Somit können einzelne korrekte Methoden implementiert und zu einem Gesamtprogramm komponiert werden.
- Geplant als Masterarbeit
- Ansprechpartner: Tobias Runge (`tobias.runge@tu-bs.de`)

# Lightweight Correctness-by-Construction

- Kontext: Correctness-by-Construction (CbC) ist ein Verfahren zur inkrementellen Softwareentwicklung. Hierbei werden Programme durch Korrektheit bewahrenden Verfeinerungen erstellt.
- Problem: Das Beweisen der korrekten Anwendung von Verfeinerungen ist aufwendig und benötigt Expertenwissen. Oftmals können Fehler im Programm durchs Testen einfacher gezeigt werden.
- Ziel: Anstatt eines Beweises sollen Testfälle (o.Ä.) generiert werden, die die Korrektheit der Verfeinerung untermauern.
- Geplant als Masterarbeit
- Ansprechpartner: Tobias Runge (`tobias.runge@tu-bs.de`)

# Smart Contracts by Construction

- Kontext: Correctness-by-Construction (CbC) ist ein Verfahren zur inkrementellen Softwareentwicklung. Smart Contracts in Solidity bilden einen Vertrag zwischen verschieden Akteuren (zum Beispiel zur Überweisung von Kryptowährung).
- Problem: Die Korrektheit der Smart Contracts muss sichergestellt werden.
- Ziel: Spezifikation von Solidiy Smart Contracts mittels Hoare Tripeln und deren inkrementelle Entwicklung zur Sicherstellung der Korrektheit mit Correctness-by-Construction.
- 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 eine 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`)

# Normalizing Sampling Stability using Feature Model Changes

- Context: Continuous Integration and an efficient regression testing process are essential parts of developing highly configurable systems. To test every valid configuration after each evolution step of a highly configurable system is not feasible. Therefore, selecting a small but representative subset of all valid configurations (i.e., a sample) for testing is necessary. Choosing a sampling algorithm for regression testing in continuous integration may influence the test quality wrt. a certain testing goal. Sampling Stability is an evaluation criterion for sampling algorithms, which estimates the potential of a sampling algorithm for re-testing configurations by measuring the similarity between samples of consecutive system versions. A first analysis of sampling stability that sampling stability in its current state depends strongly on the benchmark system on which the samples are calculated.
- Goal: In this project, you will analyze various approaches to remove the dependency between sampling stability and the benchmark system for which samples are calculated. The aim is to find and formalize normalization criteria with which the dependency can be removed. As a starting point, you will analyze the influence of feature model change operations on sampling stability based on artificial and real-world feature models.
- Prerequisites: Software-Product Lines
- Planned as Bachelor, Master's, or Project Thesis
- Supervisor: Tobias Pett (`t.pett@tu-bs.de`)

# Test Execution for highly configurable systems with Evolution

- Context: Testing of evolving highly configurable systems is difficult due to the need to face the combinatorial explosion problem after each system evolution. Researchers strive to develop new concepts to reduce the testing effort and increase the quality of testing evolving highly configurable systems. One recently developed concept is the evaluation criteria Sampling Stability to measure the similarity between samples calculated for consecutive system versions. The evaluation of those concepts often considers a theoretical aspect of reducing test effort on the configuration level (problem space). Actual test cases are often not executed due to the lack of an execution environment to automatically compile the source code and execute test cases. This reduces the validity of research results and prevents researchers from evaluating their developments on the solution level.
- Goal: In this project, you will analyze various open-source systems provided on GitHub, such as BusyBox, UcLibC, Soletta, etc., to identify their testing procedures. This includes the search for executable test cases and developing a test environment to automatically execute test cases for a different version of the systems.
- Prerequirements: Software-Product Lines
- Planned Project Thesis
- Supervisor: Tobias Pett (`t.pett@tu-bs.de`)

# 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 bewahrende 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 mit CbC SPLs
- Geplant als Masterarbeit
- **Ansprechpartnerin:** Tabea Bordis (`t.bordis@tu-bs.de`)

# Optimierung der Verifikation von Correct-by-Construction Produktlinien im Tool VarCorC

- **Kontext:** Correctness-by-Construction (CbC) ist ein Verfahren zur Softwareentwicklung. Hierbei werden Programme mit Kontrakten versehen und durch Korrektheit bewahrende Verfeinerungen erstellt. Software Produktlinien (SPLs) werden verwendet, um eine Menge von Programmvarianten zu verwalten. Das Tool VarCorC bietet die Möglichkeit, SPLs unter Verwendung von CbC zu entwickeln.
- **Problem:** Der Ansatz, mit dem VarCorC SPLs verifiziert, skaliert nicht und ist ineffizient.
- **Ziel:** Optimierung des bestehenden Ansatzes und Evaluierung der Optimierung.
- Geplant als Masterarbeit
- **Ansprechpartnerin:** Tabea Bordis (`t.bordis@tu-bs.de`)

# Einführung einer Beweisstatistik in das Tool (Var)CorC

- **Kontext:** Correctness-by-Construction (CbC) ist ein Verfahren zur Softwareentwicklung. Hierbei werden Programme mit Kontrakten versehen und durch Korrektheit bewahrende Verfeinerungen erstellt. Das Tool CorC bietet die Möglichkeit, Programme unter Verwendung von CbC zu entwickeln. Dabei wird pro CbC-Verfeinerung eine Beweisdatei erstellt, die von dem deduktiven Beweiser KeY verifiziert wird.
- **Problem:** Es gibt kein Mapping von Verfeinerungsschritten zu Beweisdateien, da diese permanent überschrieben werden. Außerdem gibt es keine einfache Möglichkeit, die Statistiken dieser Beweise einzusehen.
- **Ziel:** Erstellung eines Mappings von Verfeinerungsschritten zu Beweisdateien und Integration einer Beweisstatistik in CorC.
- Geplant als Bachelor- oder Projektarbeit
- **Ansprechpartnerin:** Tabea Bordis (`t.bordis@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`)

# Solution-Space Sampling

- **Kontext:**Software Produkt-Linien (SPL) unterstützen das Erstellen und Verwalten von konfigurierbaren Systemen. Die Struktur einer SPL unterteilt sich in den Problem-Space, der eine Menge von Konfigurationsoptionen definiert und den Solution-Space, der eine Menge von Realisierung-Artefakten definiert. Basierend auf einer gewählten Problem-Space Konfiguration kann damit eine spezifische Variante aus den Realisierung-Artefakten des Solution-Space zusammengestellt werden.
- **Problem:**SPLs erlauben häufig eine Vielzahl von Konfigurationen, womit ein vollständiges testen jeder möglichen Konfiguration nicht machbar ist. Um dennoch eine Aussage über die Funktionalität und Korrektheit des Systems zu erhalten, werden Sampling-Algorithmen eingesetzt, die eine repräsentative Teilmenge aller Konfigurationen (Sample) berechnen können. Diese Sampling-Algorithmen arbeiten jedoch häufig nur auf dem Problem-Space und bilden damit nicht direkt die Zusammenhänge der Realisierung-Artefakte ab.
- **Ziel:**Das Ziel dieser Arbeit ist es, existierende Sampling-Algorithmen auf den Solution-Space anzuwenden. Das berechnete Sample soll anschließend zurück in den Problem-Space übersetzt werden, wobei verschiedene Kriterien, wie z.B. das Mappen auf bereits existierende Konfigurationen, berücksichtigt werden sollen.
- **Ansprechpartner:** Marc Hentze (`m.hentze@tu-bs.de`), Tobias Pett (`t.pett@tu-bs.de`)

# Sampling Strategien zur Szenario Selektion

- **Kontext:** Verifikation und Validierung von Fahrerassistenzsystemen / automatisierten Fahrfunktionen mittels szenario-basiertem Testen; Einsatz von Sampling Strategien um Szenarien aus einem Szenarienraum auszuwählen
- **Herausforderung:** Sampling Strategien haben Einfluss auf die Qualität der selektierten Szenarien; Einsatz von Fuzzing-Techniken kann selektrierte Szenarien verbessern
- **Ziel:** Besseres Verständnis von Einflüssen der Sampling Strategien und Fuzzing auf die Qualität der Szenarien.
- **Voraussetzung:** Grundlegende Kenntnisse im Bereich Software Testing, Fahrerassistenzsysteme, Matlab / Simulink, Python und Java
- **Geplant als:** Abschlussarbeit (Master)
- **Ansprechpartner:** Lukas Birkemeyer (`l.birkemeyer@tu-braunschweig.de`)

# Praktika und Teamprojekte