



**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](mailto:i.schaefer@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: Thomas Thüm ([t.thuem@tu-bs.de](mailto:t.thuem@tu-bs.de)) und Sebastian Krieter (Magdeburg)

# Inkrementelle Erfüllbarkeitsanfragen für SPLs

- Kontext: Feature-Modelle beschreiben die Abhängigkeiten zwischen Features einer Produktlinie und werden zur Fehlererkennung in Aussagenlogik kodiert
- Problem: Für eine einzige Produktlinie wie Linux gibt es Millionen von Erfüllbarkeitsanfragen, die zum Großteil ähnlich sind
- Ziel: Inkrementelle Verfahren von SAT und SMT-Solvern evaluieren und Entwicklung von eigenen Strategien zur effizienten Überprüfung von vielen, ähnlichen Erfüllbarkeitsanfragen
- Voraussetzung: Software-Produktlinien
- Geplant als Masterarbeit
- Ansprechpartner: Thomas Thüm ([t.thuem@tu-bs.de](mailto:t.thuem@tu-bs.de))

# Binary Decision Diagrams for Large Product Lines

- Context: Feature models define valid combinations of features and are used in numerous analyses for product lines. For that purpose, feature models are often translated into propositional formulas and feed into state-of-the-art solvers (e.g., SAT, SMT, BDD).
- Problem: While BDDs are often more efficient than other solvers, their creation only scales to hundreds of features, whereas industrial product lines, such as Linux, have thousands of features.
- Goal: Create BDDs for parts of feature models based on an existing compositionality principle. Evaluate the potential of such BDDs to reduce the effort during product-line analysis.
- Requirements: Lecture on software product lines
- Master's thesis
- Advisor: Thomas Thüm (t.thuem@tu-bs.de) and Prof. Eric Bodden (Paderborn)

# Compositional Feature Modeling

- Context: Feature models in industry contain thousands of features and are inherently non-compositional. That is, a change in one part of the model can influence any other part. While there are first ideas to achieve compositionality in feature models (i.e., feature-model interfaces), they are computationally expensive.
- Goal: A methodology to create and edit feature models in a compositional manner, such that for every change in a feature model the consequences for other parts of the model are made explicit (cf. change impact analysis). Evaluation of the costs and benefits of this methodology.
- Requirements: Lecture on software product lines
- Master's thesis
- Advisor: Thomas Thüm (t.thuem@tu-bs.de)

# Deploying Guidance for Product Line Evolution

- **Context:** In DarwinSPL<sup>1</sup>, we are able to model feature-model evolution. This evolution may affect configurations of the feature model. To handle this, we are able to define guidance for engineers maintaining configurations to adapt the configurations to the feature-model evolution.
- **Problem:** From the tool perspective, it is currently not possible to store this guidance and to deploy it to configuration maintainers.
- **Envisioned Solution:** Implement the guidance process in the existing tool landscape of DarwinSPL.
- **Information:** Bachelor's thesis or project work
- **Contact:** Michael Nieke (nieke@isf.cs.tu-bs.de)

---

<sup>1</sup><https://gitlab.com/DarwinSPL/DarwinSPL>



# Evolution Visualization of Temporal Feature Models

- **Context:** Temporal Feature Models allow to model integrated evolution for feature models.
- **Problem:** For long evolution histories and large feature models, engineers might lose overview on the evolution that took place.
- **Envisioned Solution:** Devise different interactive visualization concepts (e.g., an evolution Gantt diagram) to visualize the evolution of a temporal feature model in a clear way.
- **Information:** Bachelor's thesis or project work
- **Contact:** Michael Nieke ([nieke@isf.cs.tu-bs.de](mailto:nieke@isf.cs.tu-bs.de))

# Efficient Reuse of Satisfiability Requests

- **Context:** In several domains, properties are checked using satisfiability requests. For instance, properties of feature models can be checked. During evolution of systems, parts of satisfiability requests change and parts remain the same.
- **Problem:** Parts that are the same are checked many times again. Incremental solvers use push/pop mechanisms for parts of the requests. If the stack has a bad order, push/pop is slower than checking everything again.
- **Envisioned Solution:** Devise and implement multiple methods/strategies for reordering parts of satisfiability requests based on the evolution history.
- **Information:** Master's thesis
- **Contact:** Michael Nieke ([nieke@isf.cs.tu-bs.de](mailto:nieke@isf.cs.tu-bs.de)), Thomas Thüm ([t.thuem@tu-bs.de](mailto:t.thuem@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](mailto:a.knueppel@tu-bs.de))

# Mutation Analysis for Deductive Verification

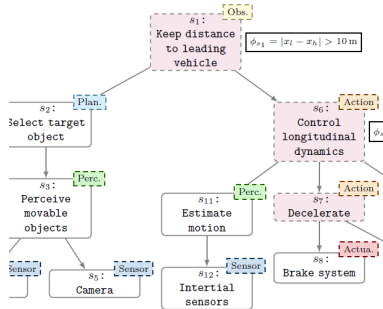
- **Context:** Specifying programs is challenging and error-prone. In dependable software, many specifications are also oftentimes too weak for a successful verification. Mutation testing is used to evaluate the quality of test cases. Likewise, mutation analysis may be used for evaluating the *quality* 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 mechanize 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](mailto:t.bordis@tu-bs.de)) & Tobias Runge ([tobias.runge@tu-bs.de](mailto: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

Partner: Tobias Runge ([tobias.runge@tu-bs.de](mailto: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](mailto: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](mailto:t.pett@tu-bs.de)) und Sebastian Krieter (Magdeburg)

# Praktika und Teamprojekte