ITEA Office
High Tech Campus 69 - 3
5656 AG Eindhoven
The Netherlands

T + 31 88 003 6136
E info@itea3.org
W www.itea3.org

*ITEA 3 is a EUREKA strategic ICT cluster programme*

# Exploitable Results by Third Parties

15010 REVaMP²

## Project details

| | |
|---|---|
| Project leader: | Andrey Sadovykh |
| Email: | andrey.sadovykh@softeam.fr |
| Website: | revamp2-project.eu |

| Name: ComAnI (Commit Analysis Infrastructure) | | |
|---|---|---|
| Input(s): | Main feature(s) | Output(s): |
| ▪ Configuration file<br>▪ Optional: single commit | ▪ Open, extensible, and configurable infrastructure for commit extraction and analysis<br>▪ Decoupled commit extraction and analysis components for user-defined combination and easy extension<br>▪ Support for three different extraction variants | ▪ Cached, extracted commits (optional, intermediate result)<br>▪ Depending on the purpose of the analysis component, e.g., the intensity of variability changes over time |

| | |
|---|---|
| Unique Selling Proposition(s): | ▪ Configurable infrastructure for defining different setups of extraction and analysis components<br>▪ Easy development and integration of new extraction and analysis components due to provided capabilities of the infrastructure<br>▪ Integrated caching and parallelization<br>▪ Support for different version control systems and analyses |
| Integration constraint(s): | ▪ SVN or Git has to be installed<br>▪ For some analyses, R needs to be installed<br>▪ Java 8 or higher |
| Intended user(s): | ▪ Software developers<br>▪ Software analysists<br>▪ Researchers |
| Provider: | ▪ Stiftung University of Hildesheim |
| Contact point: | ▪ Christian Kröher – kroeher@sse.uni-hildesheim.de |
| Condition(s) for reuse: | ▪ Apache Licence 2.0 |
| | *Latest update: 3 July 2019* |

| Name: pure::variants Asset Variability Framework | | |
|---|---|---|
| Input(s): | Main feature(s) | Output(s): |
| ▪ Artifacts containing variability | ▪ Extraction of Variability<br>▪ Analysis of extracted variability | ▪ Different outputs possible, depending on implemented extractors and analyzers<br>▪ Example: Extracted #defines from source code modeled in pure::variants family models |

| | |
|---|---|
| Unique Selling Proposition(s): | ▪ Flexible and extensible framework for extracting and analyzing variability |
| Integration constraint(s): | ▪ pure::variants<br>▪ other artifact tools depending on implemented extractors and analyzers |
| Intended user(s): | ▪ Product Line Engineer |
| Provider: | ▪ pure-systems GmbH |
| Contact point: | ▪ Uwe Ryssel - uwe.ryssel@pure-systems.com |
| Condition(s) for reuse: | ▪ Licensing |
| | *Latest update:* < > |

<table>
<tr><th colspan="3">Name: KernelHaven</th></tr>
<tr><th>Input(s):</th><th>Main feature(s)</th><th>Output(s):</th></tr>
<tr>
<td>▪ Configuration file<br>▪ Dependent on analysis: Code files (*.c, *.h, *.S)<br>▪ Dependent on analysis: Build files (make, Excel)<br>▪ Dependent on analysis: Variability models (Kconfig, Excel, DIMACS)<br>▪ Dependent on analysis: Further resources, e.g., mailing list archives</td>
<td>▪ Open, extensible, and configurable infrastructure for static product lines analysis.<br>▪ Among others, supports the following analyses:<br>o Feature Effect analysis to reverse engineer implemented variability dependencies<br>o Configuration Mismatch analysis to verify whether modeled and implemented variability is inline<br>o Dead Code analysis to detect implemented variability, that cannot be enabled<br>o Over 42,000 variability-aware code metrics that optionally integrate variability model to measure complexity of variability<br>o Architecture analysis to detect whether implemented variability is aligns with architecture<br>o Mailing list analysis, to trace features in code, variability model, architectural descriptions, and mailing list.</td>
<td>▪ Results are outputted in tabular formats like CSV, Excel, or SQLite.</td>
</tr>
</table>

| | |
|---|---|
| Unique Selling Proposition(s): | ▪ Configurable infrastructure for defining different static analyses on software product lines<br>▪ Decouples parsing from analysis to enable reusing of analyses on new artifact types.<br>▪ KernelHaven is designed to support and simplify reproducibility of (published) results.<br>▪ Allows reuse of implemented analyses to simplify development of new analysis plug-ins.<br>▪ Transparent use of parallelization to improve performance on large-scale product lines, without overwhelming developers with implementation details. |
| Integration constraint(s): | ▪ Java 8 or higher<br>▪ Kconfig extractor requires Linux and build tools installed, other plug-ins are platform independent.<br>▪ MailingList analysis requires GIT to be installed. |
| Intended user(s): | ▪ Software developers<br>▪ Software analysists<br>▪ Researchers |
| Provider: | ▪ Stiftung University of Hildesheim |

| Name: KernelHaven | |
|---|---|
| Contact point: | ▪ Klaus Schmid – schmid@sse.uni-hildesheim.de<br>▪ Sascha El-Sharkawy – elscha@sse.uni-hildesheim.de<br>▪ Christian Kröher – kroeher@sse.uni-hildesheim.de |
| Condition(s) for reuse: | ▪ Apache Licence 2.0<br>▪ Some plug-ins contain 3rd party components and are published under GPLv3 for this reason. |
| | *Latest update: 18 November 2019 (still maintained)* |

| Name: VEXA | | |
|---|---|---|
| Input(s): | Main feature(s) | Output(s): |
| ▪ C/C++ source files: .c, .h, .cpp, .hpp<br>▪ Excel files: .xlsx<br>▪ JSON files: .json<br>▪ Build artefacts: compile_commands.json | ▪ Plugin for the Neo4j graph database<br>▪ Custom code metrics generation<br>▪ Support for incremental source code analyses | ▪ Neo4j graph database<br>▪ Code metrics in tabular form<br>▪ Simple interactive graph visualization |

| | |
|---|---|
| Unique Selling Proposition(s): | ▪ Extensible framework for variability extraction from source code artefacts<br>▪ User guided incremental dependency analyses for source code artefacts utilizing Neo4j’s powerful graph processing capabilities<br>▪ Z3 theorem prover integration for high level reasoning and simplification of constraint expressions<br>▪ High performance scalability for large-scale source analyses<br>▪ Easy integration using the Cypher query language and Neo4j driver APIs (e.g., REST) for various programming languages |
| Integration constraint(s): | ▪ Neo4j Community Edition (> 3.2.14)<br>▪ Java 8 or higher<br>▪ Supported OS: Linux & Windows<br>▪ Interoperability via Neo4j drivers and Cypher query language<br>▪ srcML (srcml.org) dependency for C/C++ source code analysis |
| Intended user(s): | ▪ Software developers<br>▪ Researchers |
| Provider: | ▪ FZI Forschungszentrum Informatik |
| Contact point: | ▪ Anton Paule – anton.paule@fzi.de<br>▪ Sebastian Reiter – sebastian.reiter@fzi.de |
| Condition(s) for reuse: | ▪ Licensing |

*Latest update: 21 November 2019*

| Name: FeDeV (**Fe**ature **De**pendency **Vi**sualization) | | |
|---|---|---|
| Input(s): | Main feature(s) | Output(s): |
| ▪ SQLite Database with KernelHaven results<br>▪ Neo4j database with VEXA results | ▪ Visualization and exploration of extraction results<br>▪ Additional integrations with tools like Eclipse Capra for traceability | ▪ Visualization of features and feature dependencies |

| | |
|---|---|
| Unique Selling Proposition(s): | ▪ Easy to use application to visualize extraction results<br>▪ Tree-, table-, and graph-views for visualization<br>▪ Navigation capabilities between the views<br>▪ Stepwise exploration of analysis results<br>▪ Color coding for feature visualization<br>▪ Visualization of feature dependencies including color coding<br>▪ Visualization of submodules and submodule dependencies<br>▪ Cypher view to execute Cypher queries and interact with VEXA<br>▪ Full Text Search in analysis results<br>▪ Export of table views to Excel |
| Integration constraint(s): | ▪ Neo4j Community Edition (> 3.2.14)<br>▪ Java 8 or higher<br>▪ Interoperability via Neo4j drivers and Cypher query language<br>▪ Network connection for the Capra integration |
| Intended user(s): | ▪ Software developers<br>▪ Software analysts<br>▪ System integrators |
| Provider: | ▪ ScopeSET GmbH |
| Contact point: | ▪ Michael Benkel – benkel@scopeset.de<br>▪ Felix Suda – felix.suda@scopeset.de |
| Condition(s) for reuse: | ▪ TomSawyer runtime license |

*Latest update: 21 November 2019*

| Name: ReVaMP2 Plugin by The Reuse Company | | |
|---|---|---|
| Input(s): | Main feature(s) | Output(s): |
| ▪ Requirements Specification<br>▪ Ontology with information about the domain. | ▪ Configure the Formalization of Requirement Assets to SRL<br>▪ Formalize requirement specification to SRL.<br>▪ Merge formalized SRL and process to generate the SRL Feature Model view.<br>▪ Visualize Feature Model.<br>▪ Feature Configuration and requirement generation from extracted templates.<br>▪ Export to Variability Exchange Language (VEL). | ▪ SRL Feature Model Representation.<br>▪ Requirements templates based on semantic patterns matching.<br>▪ Variability Exchange Language (VEL) Output.<br>▪ Traceability from sources to extracted/generated assets. |

| | |
|---|---|
| Unique Selling Proposition(s): | ▪ Semantic approach to semi-automatic feature model extraction based on ontologies.<br>▪ SRL Feature Model tree visualization.<br>▪ Reuse of legacy requirement templates to generate requirements based on feature configuration.<br>▪ Multiple sources Rational Doors, DNG, PTC Integrity, Excel. |
| Integration constraint(s): | ▪ SRL – System Representation Language (available at https://github.com/trc-research/oslc-km) |
| Intended user(s): | ▪ Knowledge Reuse and Systems Engineers |
| Provider: | ▪ The REUSE Company a trademark of Knowledge Centric Solutions, SL |
| Contact point: | ▪ José Fuentes (jose.fuentes@reusecompany.com)<br>▪ Elena Gallego (elena.gallego@reusecompany.com)<br>▪ Borja López (borja.lopez@reusecompany.com)<br>▪ Luis Pérez (luis.perez@reusecompany.com) |
| Condition(s) for reuse: | ▪ Commercial license (Evaluation license available) |

*Latest update: 21 November 2019*

Name: Configuration Mining

| Input(s): | Main feature(s) | Output(s): |
|---|---|---|
| ▪ Valid product configurations (feature selections) | ▪ Detect configuration rules satisfied by every configuration<br>▪ Configurable rule detection (support and confidence parameters)<br>▪ Iterative detection approach, can involve user input and feature models to reduce false positive ratio | ▪ Set of configuration rules satisfied by every configuration |

| | |
|---|---|
| Unique Selling Proposition(s): | ▪ Reverse engineering approach to constraint extraction from past configurations.<br>▪ Bootstrap feature modeling with automatically proposed configuration rules. |
| Integration constraint(s): | ▪ Conversion of input and output data, depending on the configuration storage format and the feature modelling tool (e.g. pure::variants) |
| Intended user(s): | ▪ Product Line architects |
| Provider: | ▪ Robert Bosch GmbH |
| Contact point: | ▪ Slawomir Duszynski (Slawomir.Duszynski@de.bosch.com)<br>▪ Tobias Beichter (Tobias.Beichter@de.bosch.com) |
| Condition(s) for reuse: | ▪ Documentation of the algorithms. No software provided. |

*Latest update: 25 November 2019*

Name: Configuration Mining

Name: PLPV-CE (Product-Line-Product-Variant Co-Evolution)

| Input(s): | Main feature(s) | Output(s): |
|---|---|---|
| ▪ Root directory of a C software project<br>▪ Entry point (feature name or code element) | ▪ Creation of code property graph (abstract syntax tree, data flow, control flow, variability information)<br>▪ Identification of semantically related lines of code based on a given entry point and code property graph<br>▪ Generation of patches containing all semantically related lines of code for merging | ▪ Code property graph (intermediate result)<br>▪ Patches for transferring semantic units from one project to another one |

| | |
|---|---|
| Unique Selling Proposition(s): | ▪ Variability-aware code property graph (abstract syntax tree extend by data flow, control flow, and variability information<br>▪ Automatic slicing of semantically related lines of code based on user-defined entry point<br>▪ Provides variability- and structure-preserving slices containing C- and preprocessor code<br>▪ Patch generation for transferring the user-defined entry point and all related lines of code guaranteeing the desired functionality after transfer |
| Integration constraint(s): | ▪ Executable on Linux only<br>▪ Git<br>▪ Java 8 or higher<br>▪ Python 3 |
| Intended user(s): | ▪ Software developers<br>▪ Researchers |
| Provider: | ▪ Stiftung University of Hildesheim |
| Contact point: | ▪ Christian Kröher – kroeher@sse.uni-hildesheim.de<br>▪ Lea Gerling – gerling@sse.uni-hildesheim.de |
| Condition(s) for reuse: | ▪ GNU Lesser General Public License v3.0 |

*Latest update: 05 November 2019*

| Name: PSS-CE (Problem-Solution-Space Co-Evolution) | | |
|---|---|---|
| Input(s): | Main feature(s) | Output(s): |
| ▪ Root directory of a software product line project<br>▪ KernelHaven configuration | ▪ Creation of variability mapping between problem and solution space artifacts (code, build, variability model)<br>▪ Identification of divergences between the variability information in problem and solution space artifacts<br>▪ Proposals for corrections of identified divergences | ▪ Problem-solution space mapping<br>▪ Detected divergences with locations<br>▪ Correction proposals |

| | |
|---|---|
| Unique Selling Proposition(s): | ▪ Identification and relation of variability information in different artifact types (code, build, and variability model artifacts)<br>▪ Automatic detection of unrelated, undefined, or unused variability information<br>▪ Proposal for correction of detected divergences |
| Integration constraint(s): | ▪ Depending on the KernelHaven configuration: Linux only<br>▪ C-preprocessor Code<br>▪ Makefiles (build)<br>▪ Kconfig-based or pure Boolean (CNF) variability models<br>▪ Java 8 or higher |
| Intended user(s): | ▪ Software developers<br>▪ Researchers |
| Provider: | ▪ Stiftung University of Hildesheim |
| Contact point: | ▪ Christian Kröher – kroeher@sse.uni-hildesheim.de |
| Condition(s) for reuse: | ▪ Depending on the KernelHaven bundle: GPLv3 or Apache License 2.0 |
| | *Latest update: 27 June 2019* |

| Name: SIMULTime | | |
|---|---|---|
| Input(s): | Main feature(s) | Output(s): |
| ▪ SW source code or SW binary code<br>▪ HW platform(s)<br>▪ (Heterogeneous system's partition scheme) | ▪ Fast and accurate timing estimations for the execution time of the input SW program considering its execution on the given HW platforms<br>▪ Timing estimations produced executing context-sensitive timing simulations based on hardware-independent LLVM IR code | ▪ SW execution time prediction<br>▪ Visualization of timing properties directly on Simulink simulations<br>▪ Early timing estimations for porting the SW to heterogeneous HW |

| | |
|---|---|
| Unique Selling Proposition(s): | ▪ Fast and accurate timing estimations that are essential in developing or evolving an embedded system.<br>▪ Measurement-based technique that implicitly models the different hardware resources included in HW processors. |
| Integration constraint(s): | ▪ LLVM Compiler Infrastructure 5.0 (or newer)<br>▪ Lauterbach TRACE32 tracer<br>▪ Radare2 disassembler<br>▪ Matlab 2016a (or newer)<br>▪ libboost<br>▪ Avast RetDec |
| Intended user(s): | ▪ Embedded system designers and embedded engineers that face with system timing requirements (non-functional requirements) |
| Provider: | ▪ FZI Forschungszentrum Informatik |
| Contact point: | ▪ Alessandro Cornaglia – cornaglia@fzi.de<br>▪ Sebastian Reiter – sreiter@fzi.de |
| Condition(s) for reuse: | ▪ Licensing |
| | *Latest update:* < > |

<table>
<tr><th colspan="3">Name: Co-Evolution extension for pure::variants</th></tr>
<tr><th>Input(s):</th><th>Main feature(s)</th><th>Output(s):</th></tr>
<tr><td>▪ Modified product line<br>▪ Modified variant derived from a previous version of the product line</td><td>▪ Updates the variant to reflect modifications of the product line and keep modifications of the variant</td><td>▪ Updated variant with merged modifications<br>▪ List of merge conflicts</td></tr>
<tr><td>Unique Selling Proposition(s):</td><td colspan="2">▪ Enables aligning of parallel evolution of product line and multiple variants</td></tr>
<tr><td>Integration constraint(s):</td><td colspan="2">▪ pure::variants<br>▪ an artifact tool supporting three-way-merging, e.g. requirement tool, source code tool, modeling tool</td></tr>
<tr><td>Intended user(s):</td><td colspan="2">▪ System Application Engineers</td></tr>
<tr><td>Provider:</td><td colspan="2">▪ pure-systems GmbH</td></tr>
<tr><td>Contact point:</td><td colspan="2">▪ Uwe Ryssel - uwe.ryssel@pure-systems.com</td></tr>
<tr><td>Condition(s) for reuse:</td><td colspan="2">▪ Licensing</td></tr>
<tr><td></td><td colspan="2"><i>Latest update:</i> < ></td></tr>
</table>

Name: Co-Evolution extension for pure::variants

Name: VariaMos

| Input(s): | Main feature(s) | Output(s): |
|---|---|---|
| ▪ A variability model (optional as it can be created through the VariaMos Web GUI)<br>▪ A partial product configuration (also optional as it can be selected through the Web GUI) | ▪ Graphical edition of feature models<br>▪ Graphical edition of asset models and their link to FRAGment Oriented Programming (FragOP) source code files<br>▪ Feature model defect detection automation<br>▪ Manual, semi-automated or fully automated product configuration by feature selection and constraint propagation<br>▪ Product derivation automation by assembling fragments realizing the selected features | ▪ A verified defect-free feature model<br>▪ A valid product configuration selection<br>▪ One or of several valid products |

| | |
|---|---|
| Unique Selling Proposition(s): | • Collaborative engineering product lines with automated reasoning assistance with just a web browser<br>▪ Allows combining both the compositional and annotative styles of asset modeling thanks to FragOP |
| Integration constraint(s): | ▪ The verification, configuration and product derivation automation services are accessible through a REST API |
| Intended user(s): | ▪ Software engineers |
| Provider: | ▪ Université Paris 1 Panthéon-Sorbonne and Ecole National Supérieure des Techniques Avancées Bretagne |
| Contact point: | ▪ raul.mazo@ensta-bretagne.fr |
| Condition(s) for reuse: | ▪ MIT License |

*Latest update: 13 November 2019*

| Name: KernelHaven | | |
|---|---|---|
| Input(s): | Main feature(s) | Output(s): |
| ▪ Configuration file<br>▪ Dependent on analysis: Code files (*.c, *.h, *.S)<br>▪ Dependent on analysis: Build files (make, Excel)<br>▪ Dependent on analysis: Variability models (Kconfig, Excel, DIMACS)<br>▪ Dependent on analysis: Further resources, e.g., mailing list archives | ▪ Open, extensible, and configurable infrastructure for static product lines analysis.<br>▪ Among others, supports the following analyses:<br>o Feature Effect analysis to reverse engineer implemented variability dependencies<br>o Configuration Mismatch analysis to verify whether modeled and implemented variability is inline<br>o Dead Code analysis to detect implemented variability, that cannot be enabled<br>o Over 42,000 variability-aware code metrics that optionally integrate variability model to measure complexity of variability<br>o Architecture analysis to detect whether implemented variability is aligns with architecture<br>o Mailing list analysis, to trace features in code, variability model, architectural descriptions, and mailing list. | ▪ Results are outputted in tabular formats like CSV, Excel, or SQLite. |

| | |
|---|---|
| Unique Selling Proposition(s): | ▪ Configurable infrastructure for defining different static analyses on software product lines<br>▪ Decouples parsing from analysis to enable reusing of analyses on new artifact types.<br>▪ KernelHaven is designed to support and simplify reproducibility of (published) results.<br>▪ Allows reuse of implemented analyses to simplify development of new analysis plug-ins.<br>▪ Transparent use of parallelization to improve performance on large-scale product lines, without overwhelming developers with implementation details. |
| Integration constraint(s): | ▪ Java 8 or higher<br>▪ Kconfig extractor requires Linux and build tools installed, other plug-ins are platform independent.<br>▪ MailingList analysis requires GIT to be installed. |
| Intended user(s): | ▪ Software developers<br>▪ Software analysists<br>▪ Researchers |
| Provider: | ▪ Stiftung University of Hildesheim |

| Name: KernelHaven | |
|---|---|
| Contact point: | ▪ Klaus Schmid – schmid@sse.uni-hildesheim.de<br>▪ Sascha El-Sharkawy – elscha@sse.uni-hildesheim.de<br>▪ Christian Kröher – kroeher@sse.uni-hildesheim.de |
| Condition(s) for reuse: | ▪ Apache Licence 2.0<br>▪ Some plug-ins contain 3rd party components and are published under GPLv3 for this reason. |
| | *Latest update: 18 November 2019 (still maintained)* |

| Name: KTH C code verifier | | |
|---|---|---|
| Input(s): | Main feature(s) | Output(s): |
| ▪ A C file with VCC annotations that correspond to functional requirements | ▪ A textual editor to manually declare the architecture of a configurable software, and the corresponding functional requirements<br>▪ Automated checks for the consistency of the declared architecture and corresponding functional requirements<br>▪ A wrapper around the VCC tool for deductive verification of C code that: (i) annotates a C file with annotations related to the the C language typing system, memory management etc., (ii) executes the VCC tool | ▪ Warning and error messages about the consistency of the architecture and specification in the Eclipse IDE<br>▪ A console output from the VCC wrapper, about successful/unsuccessful verification of provided C file |

| | |
|---|---|
| Unique Selling Proposition(s): | ▪ Simple, and general editor for describing arbitrary configurable systems<br>▪ Quick start for working with VCC-based formal verification of C |
| Integration constraint(s): | ▪ Java 8 or higher<br>▪ Eclipse IDE with Xtext plugins<br>▪ VCC (available at https://github.com/microsoft/vcc) |
| Intended user(s): | ▪ Software analysists<br>▪ Researchers |
| Provider: | ▪ KTH Royal Institute of Technology |
| Contact point: | ▪ Dilian Gurov dilian@kth.se<br>▪ Christina Lindström clind@kth.se<br>▪ Damir Nešić damirn@kth.se |
| Condition(s) for reuse: | ▪ The developed tools are not open source but are freely available upon request<br>▪ VCC is released under MIT license |
| | *Latest update: 24 June 2019* |

<table>
<tr><th colspan="3">Name: DragonflyME</th></tr>
<tr><td>Input(s):</td><td>Main feature(s)</td><td>Output(s):</td></tr>
<tr><td>▪ (Optional) Variability specification based on the VEL</td><td>▪ UML-based modelling environment to specify and support the design of virtual prototypes (SystemC)<br>▪ Extensions to automatically or manually annotate variability of the system under test (SISPL) to the virtual prototype specification<br>▪ Iterative, guided test case generation and exploration approach, for the dynamic parameterization of the virtual prototype, w.r.t to the specified variability</td><td>▪ C++/SystemC skeleton files for the manual implementation of the virtual prototype<br>▪ IP-XACT-based configuration files for the execution of the simulation-based test runs</td></tr>
<tr><td>Unique Selling Proposition(s):</td><td colspan="2">▪ Comprehensive modelling and execution frame work for SystemC-based virtual prototypes<br>▪ Exploration approach to generate test cases in a high dimensional test space (SUT and Testbench variability)</td></tr>
<tr><td>Integration constraint(s):</td><td colspan="2">▪ Modeling environment<br>o Eclipse Modeling Tools<br>o Papyrus UML<br>o Xtext Complete SDK<br>▪ Virtual Prototype<br>o SystemC</td></tr>
<tr><td>Intended user(s):</td><td colspan="2">▪ Embedded system designers and embedded engineers that qualify software intensive HW/SW systems with the help of virtual prototypes (SystemC)</td></tr>
<tr><td>Provider:</td><td colspan="2">▪ FZI Forschungszentrum Informatik</td></tr>
<tr><td>Contact point:</td><td colspan="2">▪ Paolo Care – pcare@fzi.de<br>▪ Sebastian Reiter – sreiter@fzi.de</td></tr>
<tr><td>Condition(s) for reuse:</td><td colspan="2">▪ Proof-of-concept implementation</td></tr>
<tr><td colspan="3"><i>Latest update: still maintained</i></td></tr>
</table>

| Name: MES Test Manager (MTest) | | |
|---|---|---|
| Input(s): | Main feature(s) | Output(s): |
| ▪ Requirement Specification<br>▪ Simulink Model under Test<br>▪ Variant-specific parameterization of the Model under Test | ▪ Derivation of (variant-specific) requirement observer scripts<br>▪ Automatic selection of the dedicated requirement observers (so called assessments) when evaluating simulation results | ▪ Evaluation of the simulation results of the system under test regarding the compliance with the (testable part of the) variant-specific requirement specification<br>▪ Coverage metrics of the requirement specification for test cycles |

| | |
|---|---|
| Unique Selling Proposition(s): | ▪ Explicit differentiation between test stimulation and evaluation<br>▪ Automated derivation of requirement observers from formalized requirements (using MARS, a formalized yet human-readable requirement syntax developed by MES) |
| Integration constraint(s): | ▪ Matlab versions 2009-2018 |
| Intended user(s): | ▪ Software developers and testers using model-based development |
| Provider: | ▪ Model Engineering Solutions GmbH |
| Contact point: | ▪ Linda Schmuhl (linda.schmuhl@model-engineers.com) |
| Condition(s) for reuse: | ▪ Commercial license (Evaluation license available) |

*Latest update: 20 November 2019*

| Name: VERIFICATION Studio | | |
|---|---|---|
| Input(s): | Main feature(s) | Output(s): |
| ▪ Requirements Specification | ▪ User Interface for the creation of Verification Actions (Phase 1, Prepare for verification).<br>▪ Adaptations within the verification process to compare the expected results configured in the Verification Actions against the obtained results.<br>▪ User interface for the suggested verification results and the obtained evidences. | ▪ User interface to check the expected results against the obtained ones, based on the evidences calculated. |

| | |
|---|---|
| Unique Selling Proposition(s): | ▪ Provide objective evidence that a system (or system element) fulfills its specified requirement and characteristics, according to the Verification Process defined in the ISO 15288. |
| Integration constraint(s): | ▪ SRL – System Representation Language (available at https://github.com/trc-research/oslc-km) |
| Intended user(s): | ▪ Quality assurance and Systems engineers |
| Provider: | ▪ The REUSE Company, a trademark of Knowledge Centric Solutions, SL |
| Contact point: | ▪ José Fuentes (jose.fuentes@reusecompany.com)<br>▪ Elena Gallego (elena.gallego@reusecompany.com)<br>▪ Luis Pérez (luis.perez@reusecompany.com)<br>▪ Borja López (borja.lopez@reusecompany.com) |
| Condition(s) for reuse: | ▪ Commercial license (Evaluation license available) |
| | *Latest update: 25 November 2019* |

| Name: Relation Graph Analysis | | |
|---|---|---|
| Input(s): | Main feature(s) | Output(s): |
| ▪ Feature model, containing features and feature relations | ▪ Transitive analysis of the relation graph<br>▪ Detection of implicit relations, selection effects, modelling flaws inside the feature model and across models<br>▪ Detailed explanation of each finding tracing to the existing relations | ▪ A view with found implicit relations and modelling flaws for selected/all model features |

| | |
|---|---|
| Unique Selling Proposition(s): | ▪ Support for engineering complex feature models through uncovering implicit consequences of modelled relations and providing detailed explanations<br>▪ Useful for determining feature selection effects, preserving model correctness, analyzing model change impact<br>▪ Can follow relations between different models to support cross-model consistency |
| Integration constraint(s): | ▪ Integrated with pure::variants |
| Intended user(s): | ▪ Feature modelling experts |
| Provider: | ▪ Robert Bosch GmbH |
| Contact point: | ▪ Slawomir Duszynski (Slawomir.Duszynski@de.bosch.com)<br>▪ Tobias Beichter (Tobias.Beichter@de.bosch.com) |
| Condition(s) for reuse: | ▪ Documentation of the algorithms. No software provided. |

*Latest update: 25 November 2019*

| Name: LittleDarwin | | |
|---|---|---|
| Input(s): | Main feature(s) | Output(s): |
| ▪ Java Source Code<br>▪ Java Build and Test Environment | ▪ Mutation Testing Framework<br>▪ Easily deployable in complicated test environments<br>▪ Possibility to add new languages, mutation operators, etc. | ▪ Mutation testing report<br>▪ Mutated code |

| | |
|---|---|
| Unique Selling Proposition(s): | ▪ Free and Open-source software<br>▪ Easy to integrate |
| Integration constraint(s): | ▪ Maximum official supported version of Java is 8<br>▪ The tool has been tested on Maven, and some features are not available for other build systems |
| Intended user(s): | ▪ Software quality pofessionals |
| Provider: | ▪ University of Antwerp |
| Contact point: | ▪ Ali Parsai (ali.parsai@uantwerpen.be) |
| Condition(s) for reuse: | ▪ Reuse allowed under license terms of GNU GPL v3 |

*Latest update: 27 November 2019*

| Name: Modelio Variablility Designer | | |
|---|---|---|
| Input(s): | Main feature(s) | Output(s): |
| ▪ UML model<br>▪ SysML model<br>▪ VEL configuration | ▪ Create 150% UML or SysML model<br>▪ Add variability constraints<br>▪ Generate VEL description<br>▪ Import VEL configuration<br>▪ Generate Model variants<br>▪ Integrated with pure::variants by pure::systems | ▪ VEL description<br>▪ Model variants |

| | |
|---|---|
| Unique Selling Proposition(s): | ▪ Provides architects and analysts with variability engineering features.<br>▪ Model system once and generate variants for your product line.<br>▪ Integrated with pure::variants by pure::systems. |
| Integration constraint(s): | ▪ Modelio 3.6 and upper |
| Intended user(s): | ▪ System Architects<br>▪ Business Analysts |
| Provider: | ▪ Softeam |
| Contact point: | ▪ etienne.brosse@softeam.fr |
| Condition(s) for reuse: | ▪ Proprietary license |
| | *Latest update: 21 November 2019* |

| Name: *Workflow* Feature Annotation Extraction and Visualization<br>Involved Tools: FINALIsT$^2$ – BUT4Reuse – VEXA – FeDeV | | |
|---|---|---|
| Input(s): | Main feature(s) | Output(s): |
| ▪ Code Base, such as C/C++ files and other build artefacts (FINALIsT$^2$, BUT4Reuse, VEXA)<br>▪ Neo4j database (FeDeV) | ▪ Integrated extraction of feature annotations<br>▪ Visualization of features and variation points<br>▪ Feature model definition<br>▪ Identification of differences of product variants | ▪ Identification of Features (FINALIsT$^2$, BUT4Reuse) and Variation Points(VEXA)<br>▪ Variation Point Visualization (FeDeV) |

| | |
|---|---|
| Unique Selling Proposition(s): | ▪ Identify, locate, document and isolate features<br>▪ Definition of Feature Models<br>▪ Visualization of Variation Points<br>▪ Visual Inspection of a code base |
| Integration constraint(s): | ▪ Neo4j database required by VEXA and FeDeV<br>▪ Java 8 required by VEXA and FeDeV |
| Intended user(s): | ▪ Software developers<br>▪ Software analysts<br>▪ System integrators<br>▪ Researchers |
| Provider: | ▪ FINALIsT$^2$: ABB AG<br>▪ BUT4Reuse: Sorbonne University<br>▪ VEXA: FZI Forschungszentrum Informatik<br>▪ FeDeV: ScopeSET GmbH |
| Contact point: | ▪ FINALIsT$^2$<br>○ Andreas Burger – andreas.burger@de.abb.com<br>○ Sten Grüner – sten.gruener@de.abb.com<br>▪ BUT4Reuse<br>○ Tewfik Ziadi – tewfik.ziadi@lip6.fr<br>○ Xhevahire Tërnava – xhevahire.ternava@lip6.fr<br>○ Anas Shatnawi – anas.shatnawi@lip6.fr<br>▪ VEXA<br>○ Anton Paule – anton.paule@fzi.de<br>○ Sebastian Reiter – sebastian.reiter@fzi.de<br>▪ FeDeV<br>○ Michael Benkel – benkel@scopeset.de<br>○ Felix Suda – felix.suda@scopeset.de |

| Name: *Workflow* Feature Annotation Extraction and Visualization<br>Involved Tools: FINALIsT² – BUT4Reuse – VEXA – FeDeV | |
|---|---|
| Condition(s) for reuse: | ▪ Trade Secret (VEXA)<br>▪ TomSawyer Runtime License (FeDeV) |
| | *Latest update: 21 November 2019* |

| Name: *Workflow* Extraction and Variability Management<br>Involved Tools: BUT4Reuse – pure::variants | | |
|---|---|---|
| Input(s): | Main feature(s) | Output(s): |
| ▪ Code base<br>▪ | ▪ Variability management<br>▪ SIS variabilities identification and extraction from a code base<br>▪ Feature model construction | ▪ Feature Model<br>▪ Software Product Line (SPL) |

| | |
|---|---|
| Unique Selling Proposition(s): | ▪ Testing of feature identification and extraction approaches during reverse engineering<br>▪ Refactoring of related software systems to an SPL<br>▪ Variability evolution during forward engineering |
| Integration constraint(s): | ▪ |
| Intended user(s): | ▪ Software analysts<br>▪ System integrators<br>▪ Researchers |
| Provider: | ▪ pure::variants:<br>▪ BUT4Reuse: Sorbonne University |
| Contact point: | ▪ pure::systems<br>○ Uwe Ryssel – uwe.ryssel@pure-systems.com<br>▪ Sorbonne University<br>○ Tewfik Ziadi – tewfik.ziadi@lip6.fr<br>○ Xhevahire Tërnava – xhevahire.ternava@lip6.fr<br>○ Anas Shatnawi – anas.shatnawi@lip6.fr |
| Condition(s) for reuse: | ▪ |
| | *Latest update: 21 November 2019* |

Name: *Workflow* Constraint Extraction
Involved Tools: KernelHaven – Configuration Mining – pure::variants

| Input(s): | Main feature(s) | Output(s): |
|---|---|---|
| ▪ Code base<br>▪ Past feature configuration | ▪ Extraction of configuration constraints<br>▪ Creation of feature models | ▪ Constraints enriched feature model |

| | |
|---|---|
| Unique Selling Proposition(s): | ▪ Reduction of feature modeling effort |
| Integration constraint(s): | ▪ |
| Intended user(s): | ▪ Developers<br>▪ Product Line Engineer |
| Provider: | ▪ KernelHaven: University of Hildesheim<br>▪ Configuration Mining: Robert Bosch GmbH<br>▪ pure::variants: pure-systems GmbH |
| Contact point: | ▪ University of Hildesheim<br>○ Klaus Schmid – schmid@sse.uni-hildesheim.de<br>○ Sascha El-Sharkawy – elscha@sse.uni-hildesheim.de<br>○ Christian Kröher – kroeher@sse.uni-hildesheim.de<br>▪ Robert Bosch GmbH<br>○ Slawomir Duszynski – Slawomir.Duszynski@de.bosch.com<br>○ Saura Jyoti Dhar – Saura.Jyoti@de.bosch.com<br>▪ pure-systems GmbH<br>○ Uwe Ryssel – uwe.ryssel@pure-systems.com |
| Condition(s) for reuse: | ▪ |

*Latest update: 21 November 2019*

<table>
<tr><td colspan="3">Name: <i>Workflow</i> Feature Dependency Visualization and Traceability<br>Involved Tools: KernelHaven – PSS Mapper – FeDeV – Eclipse Capra</td></tr>
<tr><td>Input(s):</td><td>Main feature(s)</td><td>Output(s):</td></tr>
<tr><td>▪ Code base<br>▪ Additional artefacts (e.g. requirements, test cases, design models, etc.)</td><td>▪ Review of features and constraints<br>▪ Traceability between features and other artefacts</td><td>▪ Feature effect analysis<br>▪ Feature dependency visualization<br>▪ Traceability model</td></tr>
</table>

| | |
|---|---|
| Unique Selling Proposition(s): | ▪ Feature Dependency Visualization<br>▪ Feature traceability |
| Integration constraint(s): | ▪ KernelHaven analysis has to store analysis in an SQLite file<br>▪ FeDeV and Eclipse Capra communicate via localhost port |
| Intended user(s): | ▪ Developers<br>▪ Software Architects |
| Provider: | ▪ KernelHaven: University of Hildesheim<br>▪ PSS Mapper: University of Hildesheim<br>▪ FeDeV: ScopeSET GmbH<br>▪ Eclipse Capra: University of Gothenburg |
| Contact point: | ▪ University of Hildesheim<br>o Klaus Schmid – schmid@sse.uni-hildesheim.de<br>o Sascha El-Sharkawy – elscha@sse.uni-hildesheim.de<br>o Christian Kröher – kroeher@sse.uni-hildesheim.de<br>▪ ScopeSET GmbH<br>o Michael Benkel – benkel@scopeset.de<br>o Felix Suda – felix.suda@scopeset.de<br>▪ University of Gothenburg<br>o Jan-Philipp Steghöfer – jan-philipp.steghofer@cse.gu.se |
| Condition(s) for reuse: | ▪ active internet connection to obtain a TomSawyer runtime license (required for FeDeV) |

*Latest update: 21 November 2019*

Name: *Workflow* Identify and Inspect Feature Locations
Involved Tools: Jittac – BUT4Reuse

| Input(s): | Main feature(s) | Output(s): |
|---|---|---|
| Code base | Visualization of features and interdependencies on architectural level | Overview of distribution of features across an architecture |

| | |
|---|---|
| Unique Selling Proposition(s): | ▪ Identification of feature locations<br>▪ Feature scattering across several modules<br>▪ Overview of features and dependencies across modules |
| Integration constraint(s): | ▪ |
| Intended user(s): | ▪ Software architects<br>▪ Analysts |
| Provider: | ▪ BUT4Reuse: Sorbonne University<br>▪ Jittac: Karlstad University |
| Contact point: | ▪ Sorbonne University<br>○ Tewfik Ziadi – tewfik.ziadi@lip6.fr<br>○ Xhevahire Tërnava – xhevahire.ternava@lip6.fr<br>○ Anas Shatnawi – anas.shatnawi@lip6.fr<br>▪ Karlstad University<br>○ Sebastian Herold – sebastian.herold@kau.se |
| Condition(s) for reuse: | ▪ |

*Latest update: 21 November 2019*

| Name: *Workflow* Analyse Models and Extract Features<br>Involved Tools: FLiMEA – BUT4Reuse | | |
|---|---|---|
| Input(s): | Main feature(s) | Output(s): |
| ▪ Code base<br>▪ Model fragments | ▪ Overview of feature locations | ▪ Feature Locations |

| | |
|---|---|
| Unique Selling Proposition(s): | ▪ Feature extraction from models and a code base |
| Integration constraint(s): | ▪ |
| Intended user(s): | ▪ Analysts |
| Provider: | ▪ BUT4Reuse: Sorbonne University<br>▪ FLiMEA: University San Jorge |
| Contact point: | ▪ Sorbonne University<br>○ Tewfik Ziadi – tewfik.ziadi@lip6.fr<br>○ Xhevahire Tërnava – xhevahire.ternava@lip6.fr<br>○ Anas Shatnawi – anas.shatnawi@lip6.fr<br>▪ University San Jorge<br>○ Ana C. Marcén – acmarcen@usj.es<br>○ Jaime Font – jfont@usj.es |
| Condition(s) for reuse: | ▪ |

*Latest update: 21 November 2019*