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

14014 ASSUME

Affordable Safe & Secure Mobility Evolution

Project details

Project leader:	Wolfgang Köpf
Email:	wolfgang.w.koepf@daimler.com
Website:	http://assume-project.eu/

Name: Scade KCG / CompCert coupling		
Input(s):	Main feature(s)	Output(s):
▪ Scade model	▪ Scade to asm code compilation flow with Scade KCG-CompCert compilers	▪ Assembly file (asm)

Unique Selling Proposition(s):	▪ Automatic generation of assembly code from any Scade 6.6 application using & 2 stages compiler: first Scade KCG than CompCert.
Integration constraint(s):	▪ Academic version of CompCert (3.0.1) ▪ Specific version of Scade KCG ▪ TRL 6
Intended user(s):	▪ SCADE users and embedded software developers that would like to benefit from both Scade KCG and CompCert safety and certification capabilities
Provider:	▪ CompCert: AbsInt (see Git repository at https://github.com/AbsInt/CompCert) ▪ Scade KCG: ANSYS (see https://www.ansys.com/products/embedded-software/ansys-scade-suite)
Contact point:	▪ Bruno Pagano – bruno.pagano@ansys.com
Condition(s) for reuse:	▪ Research partnership
	Latest update: June 27, 2018

Name: Kalray MPPA target for SCADE Multi-Core KCG		
Input(s):	Main feature(s)	Output(s):
▪ Scade 6.6 application	▪ Tooled integration process to ease/automate the integration of Scade KCG generated multi-core C code on MPPA target	▪ Multi-core code that can be executed on Kalay MPPA many-core target

Unique Selling Proposition(s):	▪ Scade KCG automatically generates C code that can be executed on multi-core and many-core architectures. ▪ This C code is target independent and requires an integration step to create target specific code ▪ This integration step has been (partly) automated for Kalray MPPA target
Integration constraint(s):	▪ The Python target integration script developed by Kalray automates the generation of integration code for the MPPA, using the SCADE Integration Toolbox and Multi-Core Toolbox. ▪ Python ▪ Scade KCG for Multi-Core
Intended user(s):	▪ SCADE users and embedded software developers that would like to execute their Scade application on an MPPA and benefits from multi-core speedup.
Provider:	▪ Kalray for the Integration part ▪ ANSYS for Scade KCG targeting Multi-core architecture
Contact point:	▪ Günther siegel – Gunther.siegel@ansys.com
Condition(s) for reuse:	▪ Research partnership ▪ Early users evaluation

Latest update: June 27, 2018

Name: SDF^3: SDF For Free		
Input(s):	Main feature(s)	Output(s):
XML specification of a streaming system as a composite scenario-aware DF model	Performance (throughput) analysis tool	Throughput of the system
Unique Selling Proposition(s):	▪ Exact throughput analysis for time-dependent pipelined systems specified as systems of subsystems	
Integration constraint(s):	▪ libiconv-1.14.vc10 ▪ libxml2-2.9.7 ▪ boost_1_67_0 ▪ ExprTk lib ▪ StrTk lib	
Intended user(s):	▪ Research community (embedded), practicing (embedded) engineers	
Provider:	▪ TU/e, when added to the main branch http://www.es.ele.tue.nl/sdf3/	
Contact point:	▪ m.skelin@tue.nl, m.c.w.geilen@tue.nl	
Condition(s) for reuse:	▪ GPL license and SDF3 Proprietary License	
	Latest update: June 27, 2018	

Name: PLAATO (Platform Architecture & Analysis TOol)		
Input(s):	Main feature(s)	Output(s):
▪ UML (Enterprise Architect) functional and physical architecture ▪ Fault probabilities	▪ Creates fault trees ▪ Performs computation of importance metrices and cut-sets ▪ Ability to investigate design choices to make systems more reliable	▪ Fault trees ▪ Importance metrices: ▪ Fussell-Vesely ▪ Birnbaum

Unique Selling Proposition(s):	▪ Tool chain for Fault Tree Analysis that can be integrated in the development chain using model based systems and safety engineering. ▪ No other tools available that support this engineering process completely.
Integration constraint(s):	▪ Current version still uses Enterprise Architect and Matlab code as basis. TNO investigates how to use other Systems Engineering tools and intents to create executable code that not needs a Matlab license.
Intended user(s):	▪ System Architects, System Designers, System Engineers, Software/Hardware Engineers, Test Engineers
Provider:	▪ TNO Automotive
Contact point:	▪ Frank.Benders@tno.nl
Condition(s) for reuse:	▪ License fees ▪ Possibility to buy open-source software
	Latest update: June 27, 2018

Name: MBaSSy (Model Based Safety System engineering)		
Input(s):	Main feature(s)	Output(s):
▪ ISO26262 documents in MS Word or Excel	▪ Support the traceability of the Safety Engineering process for compliance to the ISO26262 ▪ Support the multi-user distributed and concurrent usages of documents ▪ Web and Database based system that includes configuration management. ▪ Automatic compliance checking to the ISO26262	▪ Traceability of development artifacts ▪ Safety Case reporting

Unique Selling Proposition(s):	▪ At this moment there does not exists an integrated tool for (Automotive) Safety Engineering that support the complete traceability and compliance checking. ▪ Automatic checking and reporting the ISO26262 compliance checking.
Integration constraint(s):	▪ Current input should be compliant with Microsoft Office tools. ▪
Intended user(s):	▪ Safety Engineers, System Engineers, Safety Verification and Validation Engineers
Provider:	▪ TNO Automotive
Contact point:	▪ Arash.Khabbaz@tno.nl
Condition(s) for reuse:	▪ License fee ▪ Possibility to buy open-source software to extend compliance to other tooling.
	Latest update: July 02, 2018

Name: aiT for Kalray		
Input(s):	Main feature(s)	Output(s):
Fully linked binary executable for Kalray MPPA2	Computes safe upper bounds for the worst-case execution times (WCETs) of non-interrupted tasks	WCET bounds in report files and in GUI
Unique Selling Proposition(s):	▪ aiT computes correct and tight upper bounds for the worst-case execution time by static program analysis. There is no need for measurements. ▪ aiT's results are valid for all inputs and each execution of a task. ▪ aiT directly analyzes binary executables. There is no need for code instrumentation. ▪ A graphical user interface supports the visualization of the worst-case program path with WCET values at basic blocks.	
Integration constraint(s):	▪ There are aiT WCET analyzers for various different target processors. ▪ aiT for Kalray can only analyze fully linked binary executables for Kalray MPPA2 (Bostan). ▪ System requirements: o Windows: 64-bit Windows 7 SP1 or newer o Linux: 64-bit CentOS/RHEL 6 or compatible o 4 GB of RAM (16 GB recommended) o 4 GB of disk space o The Linux version requires the libxcb-* family of libraries to be installed	
Intended user(s):	▪ Developers who need to validate the timing behavior of their software	
Provider:	▪ AbsInt Angewandte Informatik GmbH	
Contact point:	▪ support@absint.com	
Condition(s) for reuse:	▪ AbsInt offers commercial licenses, including training, support, and maintenance.	
	Latest update: June 28, 2018	

Name: Astrée		
Input(s):	Main feature(s)	Output(s):
C source code	Astrée automatically proves the absence of runtime errors and invalid concurrent behavior in C applications.	List of runtime errors and invalid concurrent behavior, or statement that no such problems exist

Unique Selling Proposition(s):	▪ Astrée is a static code analyzer that finds runtime errors and invalid concurrent behavior in safety-critical software written or generated in C. ▪ Astrée is sound - that is, if no errors are signaled, the absence of errors has been proved. ▪ This includes floating-point computations: All possible rounding errors, and their cumulative effects, are taken into account. ▪ Astrée offers powerful annotation mechanisms for supplying external knowledge and fine-tuning the analysis precision for individual loops or data structures. ▪ The integrated RuleChecker checks for compliance with MISRA, CWE, ISO/IEC, and SEI CERT C coding rules. You can easily toggle individual rules and even specific aspects of certain rules. ▪ Astrée has been optimized to be able to analyze large industrial code bases.
Integration constraint(s):	▪ System requirements: o Windows: 64-bit Windows 7 SP1 or newer o Linux: 64-bit CentOS/RHEL 6 or compatible o 4 GB of RAM (16 GB recommended) o 4 GB of disk space
Intended user(s):	▪ Developers of safety-critical and mission-critical software written in C
Provider:	▪ AbsInt Angewandte Informatik GmbH
Contact point:	▪ support@absint.com
Condition(s) for reuse:	▪ AbsInt offers commercial licenses, including training, support, and maintenance.
	Latest update: June 28, 2018

Name: 5.7. Services using MQAnalyzer		
Input(s):	Main feature(s)	Output(s):
▪ Functional models (Simulink)	▪ Static analysis of models with respect to model clones, runtime errors, guideline violations, metric hotspots	▪ Assessment Result

Unique Selling Proposition(s):	▪ Efficient handling of the review process of model-based software with focus on usability, aggregated results from multiple sources and assisted reviewing
Integration constraint(s):	▪ MATLAB Simulink models ▪ MATLAB Stateflow models
Intended user(s):	▪ Model developers which want to check the model-based software developments of their suppliers regarding quality standards and runtime errors.
Provider:	▪ Assystem Germany GmbH
Contact point:	▪ Michael Schmidt – mischmidt@assystem.com
Condition(s) for reuse:	▪ Licensing
	Latest update: July 02, 2018

Name: SWEET (SWEdish Execution Time tool)		
Input(s):	Main feature(s)	Output(s):
Embedded real-time code (mainly C)	Automatic derivation of program flow constraints (“flow facts”), and approximate BCET/WCET estimates.	Flow facts BCET/WCET estimates Value constraints on program variables Program slices

Unique Selling Proposition(s):	Reduces the need for manual flow fact annotations ▪ Early sourrce level BCET/WCET estimates ▪
Integration constraint(s):	There has to be a translator from the code format to analyze into the IF of SWEET (exists for C). For BCET/WCET estimates a rough cost model for the SWEET IF, modeling the timing of the target system, must be provided.
Intended user(s):	Developers of real-time software, researchers
Provider:	Mälardalen University, Programming Languages research group
Contact point:	Björn Lisper, bjorn.lisper@mdh.se
Condition(s) for reuse:	SWEET is open source under a BSD style license
	▪ *Latest update: June 29, 2018*

Name: MES Quality Commander		
Input(s):	Main feature(s)	Output(s):
▪ Results of quality assurance activities in general ▪ BTC Embedded Tester Report ▪ MES Model Examiner Report ▪ MES M-XRAY Report	▪ Definition of quality model ▪ Aggregation of quality status ▪ Visualization of quality trend	▪ Detailed visualization of quality trends ▪ Determination of hot topics in project

Unique Selling Proposition(s):	▪ Customizable definition of quality ▪ Implementation compliant to ISO 25010 System and software quality models
Integration constraint(s):	▪ Generic XML-format for import of quality information
Intended user(s):	▪ Project and quality manager, developers
Provider:	▪ Model Engineering Solutions GmbH
Contact point:	▪ info@model-engineers.com
Condition(s) for reuse:	▪ Commercial license available

Latest update: July 02, 2018>

Name: Vélus – prototype verified Lustre compiler		
Input(s):	Main feature(s)	Output(s):
▪ Lustre program	▪ Formally specified and verified in the Coq proof assistant	▪ PowerPC, ARM, RISC-V, or x86 assembly
Unique Selling Proposition(s):	▪ Gives a formal guarantee that the generated assembly code calculates the values defined by the high-level dataflow model. ▪ Provides a completely functional implementation of a standard, modular compilation scheme for Lustre programs.	
Integration constraint(s):	▪ The Lustre programs must not contain side effects or external function calls. ▪ The proof only guarantees correctness for programs whose semantics are defined by the model.	
Intended user(s):	▪ Industrial and Academic Researchers	
Provider:	▪ Inria	
Contact point:	▪ Timothy Bourke (timothy.bourke@inria.fr)	
Condition(s) for reuse:	▪ Initial closed source prototype release. Binary to be made available on Internet (https://velus.inria.fr) for evaluation and testing. ▪ Semantic models to be shared with collaborators.	
	Latest update: July 02, 2018	

AlloyInEcore: Deep Embedding of First-order Relational Language into Essential Meta-object Facility (MOF) for Model Reasoning		
Input(s):	Main feature(s)	Output(s):
▪ MOF Metamodel / UML Class Model (EMF Ecore Model) ▪ Partial XMI Instance (which conforms to given EMF Model) ▪ First-order Relational Constraints as Invariants (optional) ▪ Upper and/or Lower Bounds (optional)	▪ Synchronizes Ecore types with Java types ▪ Extends incomplete models to maintain consistency based on formal semantics given in First-order Relational Logic by the user, ▪ Enhanced Text Editor to define EClass, ERerefence, EAttirbute, EEnum, Invariants, Bounds ▪ Text Editor supports syntax highlighting, content assists, content outline, and Error reporting	▪ Complete XMI Instances within the bounds defined by the user (The system infers new EObjects and Slots on the partial instance) ▪ If no solution found, the reason of the inconsistency is reported to the user.

Unique Selling Proposition(s):	▪ Model Completion support EMF partial models. ▪ Infers instances of EReferences and EClasses based on the formal semantics defined by the user. ▪ Fully integrated with Eclipse Modeling Framework (EMF). ▪ Supports EMF Generics and Template Parameters. ▪ Integrated with Java Compiler for type checking.
Integration constraint(s):	▪ Works on top of Eclipse IDE ▪ Minimum Unsatisfiability (MUS) feature works only on Linux OS
Intended user(s):	▪ Modelers, Language Engineers, Data Engineers
Provider:	▪ UNIT Information Technology R&D Ltd., Turkey ▪ KoçSistem Information and Communication Services Inc., Turkey
Contact point:	▪ Ferhat Erata ferhat@computer.org
Condition(s) for reuse:	▪ EPL (Eclipse Public License)
Source Codes	▪ https://github.com/ModelWriter/AlloyInEcore
Publications	▪ F. Erata et. al. *AlloyInEcore: Embedding of First-order Relational Language into Essential Meta-object Facility (MOF) for Model Completion* International Conference on Software Engineering (ESEC/FSE 2018) (submitted)
Website	▪ https://modelwriter.github.io/AlloyInEcore

Latest update: July 03, 2018

Tarski: Automated Reasoning about Traces using Configurable Formal Semantics		
Input(s):	Main feature(s)	Output(s):
▪ Artifacts and traces (Traceability Information) ▪ Configuration file written in First-order Relational Logic	▪ Tarski supports the management of traces between software artifacts, which is relevant for any development team that wants to maintain consistency of the artifacts and their traces. ▪ Maintains synchronization using the trace semantics defined by the user	▪ New inferred traces among artefacts ▪ Inconsistency report ▪ Visualize traces among locations in the artifacts

Unique Selling Proposition(s):	▪ Tarski supports traceability between diverse development artifacts (requirements, architectural models, source codes, test cases etc.) ▪ The platform allows users to specify artefacts and traces between them, as well as new trace types and their semantics. ▪ Tarski is built on top of the Eclipse platform, and uses Kodkod and Alloy, two well-known tools that ensure a solid technical base.
Integration constraint(s):	▪ Integrated version only runs on Eclipse IDE ▪ Standalone version can be used through the API of the tool.
Intended user(s):	▪ Software and System Engineers / Knowledge Engineers
Provider:	▪ UNIT Information Technology R&D Ltd., Turkey ▪ KoçSistem Information and Communication Services Inc., Turkey
Contact point:	▪ Ferhat Erata ferhat@computer.org
Condition(s)	▪ EPL (Eclipse Public License)
Source Codes	▪ https://github.com/ModelWriter/Tarski
Publications	▪ F. Erata et. al. *A Tool for Automated Reasoning About Traces Based on Configurable Formal Semantics*, ACM SIGSOFT Foundations of Software Engineering Conference (ESEC/FSE 2017), http://doi.org/10.1145/3106237.3122825 ▪ F. Erata et. al. *Tarski: a platform for automated analysis of dynamically configurable traceability semantics,* ACM SIGAPP Symposium on Applied Computing (SAC 17), http://doi.org/10.1145/3019612.3019747 ▪ F. Erata et. al. *ModelWriter: Text and Model-Synchronized Document Engineering Platform,* IEEE/ACM Automated Software Engineering Conference (ASE 2017).
Website	▪ https://modelwriter.github.io/Tarski

Latest update: July 03, 2018

Name: Formal Functional Requirement Consistency Analysis		
Input(s):	Main feature(s)	Output(s):
▪ Formal requirements specified in BTC EmbeddedPlatform using the Simplified Universal Pattern ▪	▪ Formal consistency analysis of functional requirements ▪	▪ Consistency results ▪ Witness traces for passing analysis ▪ Minimal inconsistent/maximal consistent requirement sets ▪ Quality data for MES Quality Commander

Unique Selling Proposition(s):	▪ Support for Simplified Universal Pattern and “Classical BTC Patterns”. ▪ Different types of consistency are checked ▪ Based on bounded model checking, therefore low number of false warnings ▪ No design or implementation model is required. The consistency analysis operates on the requirements only and is therefore applicable to early-version requirements or incomplete specifications.
Integration constraint(s):	▪ Windows ▪ Z3 (>= version 4.3) or iSAT SMT solver ▪ Java 7 Runtime Environment ▪ BTC EmbeddedPlatform is recommended for requirement specification ▪ MES Quality Commander is required for visualization of exported quality metrics
Intended user(s):	▪ Domain expert for formal functional requirements
Provider:	▪ OFFIS e.V.
Contact point:	▪ Jan Steffen Becker (becker@offis.de)
Condition(s) for reuse:	▪ Closed source research prototype. ▪
	Latest update: June 28, 2018

Name: BTC Embedded Platform		
Input(s):	Main feature(s)	Output(s):
▪ Informal textual requirements ▪ *optional*: system under test (SUT) as Simulink/Targetlink model or C code	▪ Requirement formalization using Simplified Universal Pattern (SUP) ▪ Formal test (simulation-based evaluation of formal requirements) ▪ Requirement coverage ▪ Test case generation from formal requirements (with or without SUT) ▪ Consistency analysis of formal requirements ▪ Simulation of formal requirements ▪ MES Quality Commander Export	▪ results of requirement consistency, error traces for inconsistent requirements ▪ test cases from requirements ▪ detailed test status if SUT is available (requirement violation, coverage results, etc) ▪ Quality data for MES Quality Commander
Unique Selling Proposition(s):	▪ intuitive, graphical requirement formalization language (SUP) ▪ smooth and easy-to-use integration of test applications in one platform like formal test, consistency checks, requirement coverage ▪ early analysis of requirements feasible without having SUT model or implementation ▪ analysis results are precise because it is based on highly-integrated model checking technology	
Integration constraint(s):	▪ Windows 7/10 ▪ MES Quality Commander is required for visualization of exported quality metrics ▪ Note: no further integration needed as BTC EmbeddedPlatform is provided by a single installer	
Intended user(s):	▪ Domain expert for formal functional requirements	
Provider:	▪ BTC Embedded Systems AG	
Contact point:	▪ Dr. Tino Teige (teige@btc-es.de)	
Condition(s) for reuse:	▪ commercial software product	
	Latest update: July 03, 2018	

Name: Lyo Designer		
Input(s):	Main feature(s)	Output(s):
▪ None (This is an end-user modelling tool, requiring no other inputs) ▪	▪ https://github.com/eclipse/lyo.designer/wiki ▪ An Eclipse plugin that allows one to graphically model (1) the overall system architecture, (2) the information model of the RDF resources being shared, and (3) the individual services and operations of each Server in the system. ▪ Lyo Designer includes a integrated code generator that synthesizes the model into almost-complete OSLC4J-compliant running implementation.	▪ A graphical representation of a toolchain architrcture ▪ OSLC-compliant java code implementing the modelled toolchain.

Unique Selling Proposition(s):	▪ Lyo Designer is part of the Eclipse Lyo project – the de-facto open-source toolkit for the development of OSLC-compliant tool interfaces. ▪ Lyo Designer provides a model-based approach to toolchain design, and tool integration based on the OSLC standard. ▪ Lyo Designer seamlessly integrates with a code generator that produces almost-complete OSLC4J-compliant running implementation of each tool interface.
Integration constraint(s):	▪ Lyo Designer is an Eclipse plugin, and hence requires its setup within an Eclipse installation (Although future setup as a standalone product is possible) ▪ Lyo Designer generates a Java implementation of the tool interfaces.
Intended user(s):	▪ Tool chain architects, information modelling, developers of tool interfaces.
Provider:	▪ KTH Royal Institute of Technology
Contact point:	▪ Jad El-khoury, jad@kth.se
Condition(s) for reuse:	▪ None. Open-source EPL license ▪

Latest update: July 03, 2018