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

ITEA2 9140014 ModelWriter

Project details

Project leader:	Ferhat Erata
Email:	ferhat@computer.org
Website:	https://github.com/ModelWriter/

WP6: Architecture, Integration and Evaluation

- ModelWriter Core (Eclipse Intent)

WP2: Semantic Parsing and Generation of Documents

- Semantic Parser
- Natural Language Generator

WP3 - Model <-> Knowledge Base

- Tarski
- AlloyInEcore

WP4: Knowledge base Design and Implementation

- High Performance Traceability Solver
- Mantis Ontology API

Name: Eclipse Intent (ModelWriter Core)		
Input(s):	Main feature(s)	Output(s):
▪ Models ▪ Wiki text ▪ MS Word ▪ Source code	▪ Synchronization mapping markers for text ▪ Synchronization mapping markers for models ▪ Linking between master source -> slave target ▪ Desynchronization detection with navigation to editors ▪ Validation of re-synchronizations	▪ Desynchronization report

Unique Selling Proposition(s):	▪ Desynchronization detection between source code / models / documentation ▪ Work with several tools (any text based or EMF based Eclipse tools) ▪ No-intrusive ▪ Reliable to modifications ▪ Open Source
Integration constraint(s):	▪ No-intrusive (no modification required to any input files) ▪ Creation of a new connector to support any new format
Intended user(s):	▪ M&T Architect of Sirius community ▪ System Engineers ▪ Requirement Engineers ▪ Enterprise Architects ▪ Pre-sales
Provider:	▪ OBEO, France
Contact point:	▪ Etienne Juliot etienne.juliot@obeo.fr
Condition(s) for reuse:	▪ EPL (Eclipse Public License)
Source Codes	▪ https://github.com/ModelWriter/Source
Publications	F. Erata et. al. *ModelWriter: Text and Model-Synchronized Document Engineering Platform*. Automated Software Engineering Conference (ASE 2017) https://modelwriter.github.io/Tarski/publications/ASE2017.pdf
Website	▪ https://www.eclipse.org/intent/

Latest update: 2017-08-25

Name: Semantic Parser		
Input(s):	Main feature(s)	Output(s):
▪ English Sentences ▪ Grammar and Lexicon ▪ Ontology (optional)	▪ Generic framework – can take inputs from different domains. ▪ Robust to unseen words in input; can skip over them to produce partial parse output ▪ Optionally, can add the parse outputs to existing ontology for enrichment.	▪ Description Logic representation of input sentences. ▪ Enriched ontology with new concepts, relations and complex axioms

Unique Selling Proposition(s):	▪ TAG (Tree Adjoining Grammar) based semantic parser ▪ Robust and capable of producing partial semantic parse outputs ▪ Parse output can be added to existing ontology for enrichment and inference. ▪ Open source
Integration constraint(s):	▪ Linux Operating System ▪ Availability of (or expertise to produce) grammar and lexicon.
Intended user(s):	▪ Researches in NLP (Semantic Parsing)
Provider:	▪ LORIA, CNRS, France
Contact point:	▪ Claire Gardent claire.gardent@loria.fr
Condition(s) for reuse:	▪ GNU General Public License (GNU GPL)
Source Codes	▪ https://github.com/ModelWriter/WP2
Publications	▪ C. Gardent et. al. *Mapping Natural Language to Description Logic*. European Semantic Web Conference (ESWC 2017). https://doi.org/10.1007/978-3-319-58068-5_17 ▪ F. Erata et. al. *ModelWriter: Text and Model-Synchronized Document Engineering Platform,* IEEE/ACM Automated Software Engineering Conference (ASE 2017), https://modelwriter.github.io/Tarski/publications/ASE2017.pdf
Website	▪ https://modelwriter.github.io/SemanticParser (under construction)

Latest update: 2017-08-25

Name: Natural Language Generator		
Input(s):	Main feature(s)	Output(s):
▪ Semantic representation (e.g. Flat Semantics) ▪ Grammar and Lexicon	▪ Generic framework – can take inputs from different domains. ▪ Ability to produce partial results in case of failure to process the full input. ▪ Fast and Scalable	▪ English Sentence verbalizing the concepts and relations present in the input.

Unique Selling Proposition(s):	▪ TAG (Tree Adjoining Grammar) based generator. ▪ Fast, generic and capable of producing partial results. ▪ Open source.
Integration constraint(s):	▪ Linux Operating System ▪ Availability of (or expertise to produce) grammar and lexicon.
Intended user(s):	▪ Researches in NLP (Natural Language Generation)
Provider:	▪ LORIA, CNRS, France
Contact point:	▪ Claire Gardent claire.gardent@loria.fr
Condition(s) for reuse:	▪ GNU General Public License (GNU GPL)
Source Codes	▪ https://github.com/ModelWriter/WP2
Publications	▪ C. Gardent et. al. *Mapping Natural Language to Description Logic.* European Semantic Web Conference (ESWC 2017). https://doi.org/10.1007/978-3-319-58068-5_17
Website	▪ N/A

Latest update: 2017-08-25

Name: 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
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), https://modelwriter.github.io/Tarski/publications/ASE2017.pdf
Website	▪ https://modelwriter.github.io/Tarski

Latest update: 2017-08-25

Name: High Performance CDCL-based Traceability Solver for Detecting Inconsistencies		
Input(s):	Main feature(s)	Output(s):
▪ Artifacts and traces (Traceability Information)	▪ High performance reasoning support for traceability ▪ Conflict-driven Clause Learning ▪ Evaluated in an industrial setting.	▪ Detect new traces and inconsistent ones.

Unique Selling Proposition(s):	▪ The fastest traceability solver on the planet! ▪ Tailored to a well-known traceability semantics which is expressible enough to model any kind of dependency relation, requirements model, feature models etc. ▪ Repairs broken traces ▪ Fast Visualization of traces ▪ Domain Specific Language for providing input traces. ▪ Open source.
Integration constraint(s):	▪ Linux Operating System ▪ Fixed Semantics (The Semantics is embedded in the solver and cannot be changed)
Intended user(s):	▪ System Analysts, Requirement Engineers
Provider:	▪ UNIT Information Technology R&D Ltd., Turkey
Contact point:	▪ Ferhat Erata ferhat@computer.org
Condition(s) for reuse:	▪ MIT License
Source Codes	▪ https://github.com/ModelWriter/TraceabilitySolver (private until the algorithm and the data structures are published)
Publications	▪ F. Erata et. al. *High Performance CDCL-based Traceability Solver for Detecting Inconsistencies among Million Artefacts*. (work in progress)
Website	▪ https://modelwriter.github.io/TraceabilitySolver (under construction)

Latest update: 2017-08-25

Name: AlloyInEcore: Deep Embedding of First-order Relational Language into Essential Meta-object Facility (MOF) for Model Completion		
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 Projectional 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
Contact point:	▪ Ferhat Erata ferhat@computer.org
Condition(s) for reuse:	▪ EPL (Eclipse Public License)
Source Codes	▪ https://github.com/ModelWriter/AlloyInEcore (private until the approach and the tool are published)
Publications	▪ F. Erata et. al. *AlloyInEcore: Deep Embedding of First-order Relational Language into Essential Meta-object Facility (MOF) for Model Completion* International Conference on Software Engineering (ICSE 2017) (submitted)
Website	▪ https://modelwriter.github.io/AlloyInEcore (under construction)

Latest update: 2017-08-25

Name: Mantis Ontology API		
Input(s):	Main feature(s)	Output(s):
▪ CSV or RDF formats. ▪ API requests in queries in ontologies. ▪ Allows import in RDF/OWL formats.	▪ Mantis Ontology API provides mechanisms to create concepts in ontologies and execute semantic queries. ▪ All functionalities are also available as web services. ▪ Assures flexibility and guarantees accuracy. ▪ SPARQL format manipulation and RDF language for querying the data.	▪ Delivers rapid results in triples in ontology. ▪ Returns the results in RDF, Turtle, N3, N-Tripler formats.

Unique Selling Proposition(s):	▪ Easy import/export of ontology is in widely accepted data standards. ▪ Rapid query creation, returns swift responses and results. ▪ Supports SPARQL Protocol and RDF Language ▪ Efficient rapid search through Ontologies ▪ Consistency and validation checks of triples inside the ontology ▪ API and Web Services use OWL (W3C Web Ontology Language) ▪ Ease of use and integration.
Integration constraint(s):	▪ Linux Operating System (recommended for ease of setup) ▪ Platform Independent
Intended user(s):	▪ Data Scientists, Architects, Modelers, Language Engineers, Data Engineers
Provider:	▪ MANTIS, Turkey
Contact point:	▪ Guven Kose guvenkose@mantis.com.tr
Condition(s) for reuse:	▪ License Agreement ▪ Limited time license for academic use
Source Codes	▪ Closed Code
Publications	▪ N/A
Website	▪ www.mantis.com.tr

Latest update: 2017-08-25