ETSI TR 103 119 V1.4.1 (2023-09)

**Methods for Testing and Specification (MTS);
The Test Description Language (TDL);
Reference Implementation**

Reference

RTR/MTS-TDL103119v141

Keywords

MBT

ETSI

650 Route des Lucioles
F-06921 Sophia Antipolis Cedex - FRANCE

Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Siret N° 348 623 562 00017 - APE 7112B
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° w061004871

Contents

Intellectual Property Rights

Essential patents

IPRs essential or potentially essential to normative deliverables may have been declared to ETSI. The declarations pertaining to these essential IPRs, if any, are publicly available for **ETSI members and non-members**, and can be found in ETSI SR 000 314: *"Intellectual Property Rights (IPRs); Essential, or potentially Essential, IPRs notified to ETSI in respect of ETSI standards"*, which is available from the ETSI Secretariat. Latest updates are available on the ETSI Web server (https://ipr.etsi.org/).

Pursuant to the ETSI Directives including the ETSI IPR Policy, no investigation regarding the essentiality of IPRs, including IPR searches, has been carried out by ETSI. No guarantee can be given as to the existence of other IPRs not referenced in ETSI SR 000 314 (or the updates on the ETSI Web server) which are, or may be, or may become, essential to the present document.

Trademarks

The present document may include trademarks and/or tradenames which are asserted and/or registered by their owners. ETSI claims no ownership of these except for any which are indicated as being the property of ETSI, and conveys no right to use or reproduce any trademark and/or tradename. Mention of those trademarks in the present document does not constitute an endorsement by ETSI of products, services or organizations associated with those trademarks.

DECT™, **PLUGTESTS™**, **UMTS™** and the ETSI logo are trademarks of ETSI registered for the benefit of its Members. **3GPP™** and **LTE™** are trademarks of ETSI registered for the benefit of its Members and of the 3GPP Organizational Partners. **oneM2M™** logo is a trademark of ETSI registered for the benefit of its Members and of the oneM2M Partners. **GSM**® and the GSM logo are trademarks registered and owned by the GSM Association.

Foreword

This Technical Report (TR) has been produced by ETSI Technical Committee Methods for Testing and Specification (MTS).

NOTE: Eclipse™, Xtext™, Sirius™, EMF™, Papyrus™, GMF™, Epsilon™, EVL™ are the trade names of a product supplied by the Eclipse Foundation. OMG®, XMI™, UML™, OCL™, MOF™ are the trade names of a product supplied by Object Management Group®. This information is given for the convenience of users of the present document and does not constitute an endorsement by ETSI of the product named.

The present document is complementary to the multi-part deliverable covering the Test Description Language (TDL). Full details of the entire series can be found in part 1 of the multi-part deliverable [i.13].

Modal verbs terminology

In the present document "**should**", "**should not**", "**may**", "**need not**", "**will**", "**will not**", "**can**" and "**cannot**" are to be interpreted as described in clause 3.2 of the ETSI Drafting Rules (Verbal forms for the expression of provisions).

"**must**" and "**must not**" are **NOT** allowed in ETSI deliverables except when used in direct citation.

1 Scope

The present document summarizes technical aspects related to the implementation of TDL within the TDL Open Source Project (TOP). It describes the implementation details needed for the further development and integration of the tools. It also provides usage instructions for end users.

The following tools and components are covered in the present document:

- implementation of the TDL meta-model;
- editor for the graphical representation format of TDL;
- editor for the textual representation format of TDL;
- multiple other types of TDL model editors;
- facilities for checking the semantic validity of models according to the constraints specified in the TDL meta-model;
- implementation and tool-support for the mapping TDL elements to TTCN-3 code;
- implementation and tool-support for the importing of data definitions from OpenAPI™ and ASN.1 specifications;
- implementation and tool-support for execution of TDL models;
- implementation of the UML profile for TDL; and
- editor supporting the creation and manipulation of UML models applying the UML profile for TDL.

NOTE: The implementation of the UML profile for TDL and the corresponding editor descriptions are not aligned with the referenced versions of the TDL specification parts, but are related to an earlier release of the TDL specification parts.

2 References

2.1 Normative references

Normative references are not applicable in the present document.

2.2 Informative references

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references, only the cited version applies. For non-specific references, the latest version of the referenced document (including any amendments) applies.

NOTE: While any hyperlinks included in this clause were valid at the time of publication, ETSI cannot guarantee their long term validity.

The following referenced documents are not necessary for the application of the present document but they assist the user with regard to a particular subject area.

[i.1] Eclipse Foundation™: Eclipse IDE Website (last visited 15.06.2023).

[i.2] Eclipse Foundation™: Eclipse Xtext™ Website (last visited 15.06.2023).

[i.3] Eclipse Foundation™: Eclipse Sirius™ Website (last visited 15.06.2023).

[i.4] Eclipse Foundation™: Eclipse Modeling Framework (EMF™) Website (last visited 15.06.2023).

[i.5] Eclipse Foundation™: Eclipse Papyrus™ Modeling Environment Website (last visited 15.06.2023).

[i.6] Eclipse Foundation™: UML™ Profiles Repository Website (last visited 15.06.2023).

[i.7] Eclipse Foundation™: Graphical Modeling Framework (GMF™) Website (last visited 15.06.2023).

[i.8] "Object Constraint Language™ (OMG® OCL™), Version 2.4", formal/2014-02-03.

[i.9] Eclipse Foundation™: Eclipse OCL™ Website (last visited 15.06.2023).

[i.10] Plutext Pty Ltd: Docx4j Website (last visited 15.06.2023).

[i.11] "OMG® XML™ Metadata Interchange (XMI™) Specification", Version 2.4.2, formal/ 2014-04-04.

[i.12] Eclipse Foundation™: Epsilon™ Validation Language (EVL™) Website (last visited 15.06.2023).

[i.13] ETSI ES 203 119-1: "Methods for Testing and Specification (MTS); The Test Description Language (TDL); Part 1: Abstract Syntax and Associated Semantics".

[i.14] ETSI ES 203 119-2: "Methods for Testing and Specification (MTS); The Test Description Language (TDL); Part 2: Graphical Syntax".

[i.15] ETSI ES 203 119-3: "Methods for Testing and Specification (MTS); The Test Description Language (TDL); Part 3: Exchange Format".

[i.16] ETSI ES 203 119-4: "Methods for Testing and Specification (MTS); The Test Description Language (TDL); Part 4: Structured Test Objective Specification (Extension)".

[i.17] ETSI ES 203 119-5: "Methods for Testing and Specification (MTS); The Test Description Language (TDL); Part 5: UML Profile for TDL".

[i.18] ETSI ES 203 119-6: "Methods for Testing and Specification (MTS); The Test Description Language (TDL); Part 6: Mapping to TTCN-3".

[i.19] ETSI ES 203 119-7: "Methods for Testing and Specification (MTS); The Test Description Language (TDL); Part 7: Extended Test Configurations".

[i.20] ETSI ES 203 119-8: "Methods for Testing and Specification (MTS); The Test Description Language (TDL); Part 8: Textual Syntax".

[i.21] ETSI EG 203 130 (V1.1.1): "Methods for Testing and Specification (MTS); Model-Based Testing (MBT); Methodology for standardized test specification development".

[i.22] The Apache Software Foundation: Apache POI Website (last visited 15.06.2023).

[i.23] ETSI: The TDL Website (last visited 15.06.2023).

[i.24] ETSI: The TDL Open Source Project Website (last visited 15.06.2023).

[i.25] ETSI TS 136 321: "LTE; Evolved Universal Terrestrial Radio Access (E-UTRA); Medium Access Control (MAC) protocol specification (3GPP TS 36.321)".

[i.26] ETSI TS 103 029: "IMS Network Testing (INT); IMS & EPC Interoperability test descriptions (3GPP Release 10)".

[i.27] ETSI TS 129 214 (V15.6.0): "Universal Mobile Telecommunications System (UMTS); LTE; Policy and charging control over Rx reference point (3GPP TS 29.214 version 15.6.0 Release 15)".

[i.28] Javadoc documentation generator for Java™.

[i.29] Junit testing framework.

[i.30] Guice dependency injection framework.

[i.31]	OpenAPI™ Specification, Version 3.0.3.
[i.32]	ETSI EG 203 647 (V1.1.1): "Methods for Testing and Specification (MTS); Methodology for RESTful APIs specifications and testing".
[i.33]	IETF draft-bhutton-json-schema-00: "JSON Schema: A Media Type for Describing JSON Documents", December 8, 2020.
[i.34]	Recommendation ITU-T X.680: "Information technology - Abstract Syntax Notation One (ASN.1): Specification of basic notation". 02/2021.
[i.35]	ETSI TS 103 666-1 (V15.0.0): "Smart Secure Platform (SSP); Part 1: General characteristics (Release 15)".
[i.36]	Recommendation ITU-T X.681: "Information technology - Abstract Syntax Notation One (ASN.1): Information object specification". 02/2021.
[i.37]	ETSI TS 103 597-3 (V1.1.1): "Methods for Testing and Specification (MTS); Test Specification for MQTT; Part 3: Performance Tests".
[i.38]	ISO 8601: "Date and time format".

3 Definition of terms, symbols and abbreviations

3.1 Terms

For the purposes of the present document, the following terms apply:

abstract syntax: graph structure representing a TDL specification in an independent form of any particular encoding

concrete syntax: particular representation of a TDL specification, encoded in a textual, graphical, tabular or any other format suitable for the users of this language

meta-model: modelling elements representing the abstract syntax of a language

System Under Test (SUT): role of a component within a test configuration whose behaviour is validated when executing a test description

TDL model: instance of the TDL meta-model

TDL specification: representation of a TDL model given in a concrete syntax

3.2 Symbols

Void.

3.3 Abbreviations

For the purposes of the present document, the following abbreviations apply:

API	Application Programming Interface
ASN.1	Abstract Syntax Notation One
EBNF	Extended Backus-Naur Form
EMF	Eclipse Modelling Framework
EVL	Epsilon Validation Language
GMF	Graphical Modelling Framework
MBT	Model-Based Testing
MOF	Meta-Object Facility
OCL	Object Constraint Language
OMG	Object Management Group®r

SUT	System Under Test
TDL	Test Description Language
TOP	TDL Open Source Project
UML	Unified Modelling Language
URI	Unified Resource Identifier
XMI	eXtensible Markup Language Metadata Interchange

4 Basic Principles

4.1 Introduction

To accelerate the adoption of TDL, an implementation of TDL is provided within TOP in order to lower the barrier to entry for both users and tool vendors in getting started with using TDL. The implementation comprises graphical and textual editors, as well as validation facilities, transformation functionalities, and other tools. In addition, the UML profile for TDL and supporting editing facilities are implemented in order to enable application of TDL in UML-based working environments and model-based testing approaches.

4.2 Implementation Scope

The implementation scope includes a graphical editor according to ETSI ES 203 119-2 [i.14] based on the Eclipse platform [i.12] and related technologies, covering essential constructs of TDL. For creating and manipulating models, textual editors for ETSI ES 203 119-8 [i.20] and ETSI ES 203 119-1 [i.13], annex B are implemented based on the Eclipse platform and related technologies. The applicability of general purpose model editing facilities provided by the Eclipse platform and related technologies is discussed as well.

For tools that need to import and export TDL models according to ETSI ES 203 119-3 [i.15], corresponding facilities are implemented based on the Eclipse platform and related technologies. These facilities can be used to transform textual representations based on ETSI ES 203 119-8 [i.20] and ETSI ES 203 119-1 [i.13] into XMI [i.11] serializations according to ETSI ES 203 119-3 [i.15] and can be integrated in custom tooling that builds on the Eclipse platform.

An implementation of ETSI ES 203 119-4 [i.16] includes a dedicated textual editor for structured test objectives, which can be integrated in the textual editor for TDL. The implementation also includes facilities for exporting structured test objectives to Word™ documents using customisable tabular templates.

An implementation of the UML profile for TDL includes a specification of the TDL UML profile abstract syntax according to the mapping from the TDL meta-model to TDL stereotypes and UML meta-classes in ETSI ES 203 119-5 [i.17]. It is integrated with the open source UML modelling environment Eclipse Papyrus [i.5] as an open TDL UML profile implementation.

An implementation of ETSI ES 203 119-6 [i.18] includes a partial prototypical implementation of the TDL to TTCN-3 mapping based on the Eclipse platform.

Additional functionalities supporting the importing of data definitions from OpenAPI™ and ASN.1 specifications are also provided as a prototype.

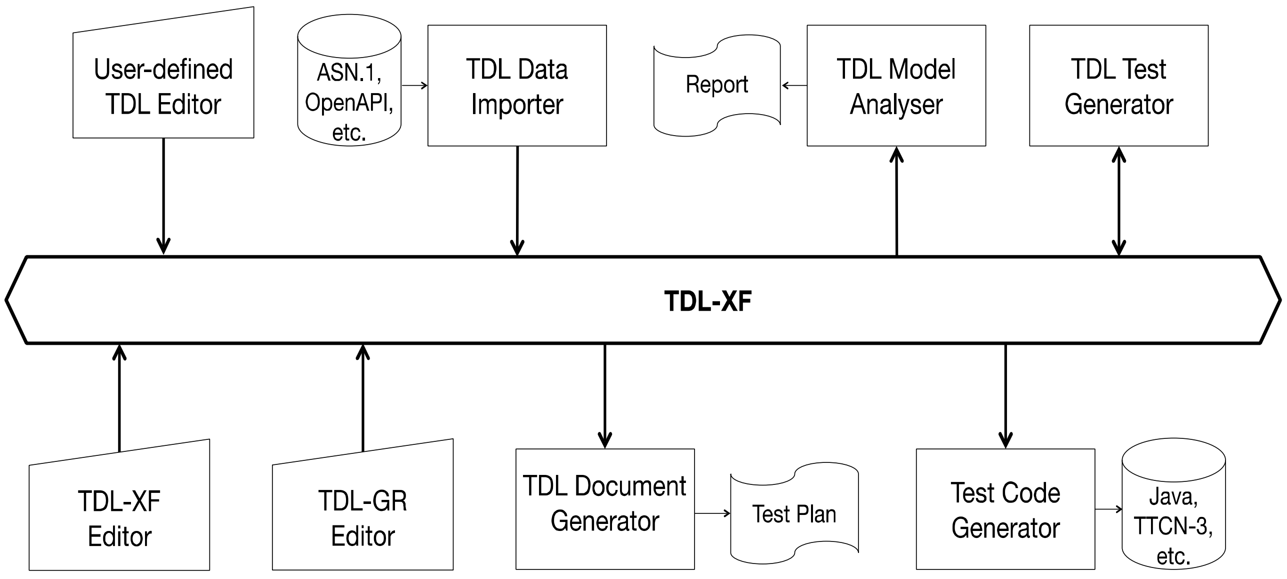


Figure 4.2-1: TDL tool infrastructure

A schematic overview of the implementation is shown in Figure 4.2-1. The TDL exchange format specified in ETSI ES 203 119-3 [i.15] serves as a bridge between the different tool components. Textual editors enable the creation and manipulation of TDL models. Data importers enable the integration and use of existing data specifications in TDL. The graphical editor is used to edit and visualize TDL models as diagrams. Documentation generation, in particular for structured test objectives, can be plugged in to produce Word documents for presenting parts of a TDL model in a format suitable for standardization documents. Test code generation, e.g. for TTCN-3 can be plugged in to produce executable TTCN-3 code or TTCN-3 skeletons to be refined afterwards.

The implementation is published as part of the TOP [i.24] on the TDL [i.23].

4.3 Document Structure

The present document contains three main technical clauses focusing on relevant technical details. The Graphical Representation editor implementing ETSI ES 203 119-2 [i.14], as well as related facilities implementing ETSI ES 203 119-1 [i.13], ETSI ES 203 119-3 [i.15] and ETSI ES 203 119-4 [i.16] are described in clause 5. Illustrative examples and guidelines for the use of TDL to address common use cases with the help of the TOP are described in clause 6. The UML Profile Editor implementing ETSI ES 203 119-5 [i.17] is described in clause 7. The use of TDL with external data specifications is discussed in clause 8. The implementation of an execution environment for the testing of RESTful API services with TDL is outlined in clause 9.

NOTE: The UML Profile Editor for TDL complies to an earlier release of the TDL specification parts.

5 Graphical Representation Editor

5.1 Scope and Requirements

TDL graphical editor implementation has two major requirements. The main objective is to provide means to visualize TDL models according to the graphical notation. The second objective is to facilitate layout of diagrams in a way that is suitable for documentation. For the second purpose, it is essential to provide graphical editing capabilities. Although often provided by modelling frameworks, the ability to graphically edit the underlying models (that is, to create new elements and set their properties) is not considered essential for this implementation.

Eclipse provides several graphical modelling tools to help build editors. Sirius [i.3] was chosen for its declarative approach that provides separation between meta-model mappings and implementations of graphical elements. With the existence of predefined common graphical elements, such as containers and connectors, the effort of implementing a graphical editor with a custom syntax in Sirius is only spent on the parts that diverge from those common elements.

Another area that requires a custom implementation is the layout of graphical elements. This covers both the absolute placement of nodes on the diagram as well as the size and internal contents of each node. Due to the rather hierarchical nature of the TDL graphical syntax, several additional base graphical elements are introduced. Some peculiar limitations of Sirius have also been identified prior to the implementation, which also need appropriate workarounds. The goal of implementing a diagram layout is to automate diagram creation to the extent that the sizes and contents of graphical elements are adjusted by layout algorithms while the absolute placement of diagram elements is solved by using built in layout implementations. This will guarantee that only minimal user interaction with the diagram editor is needed for achieving the desired layouts.

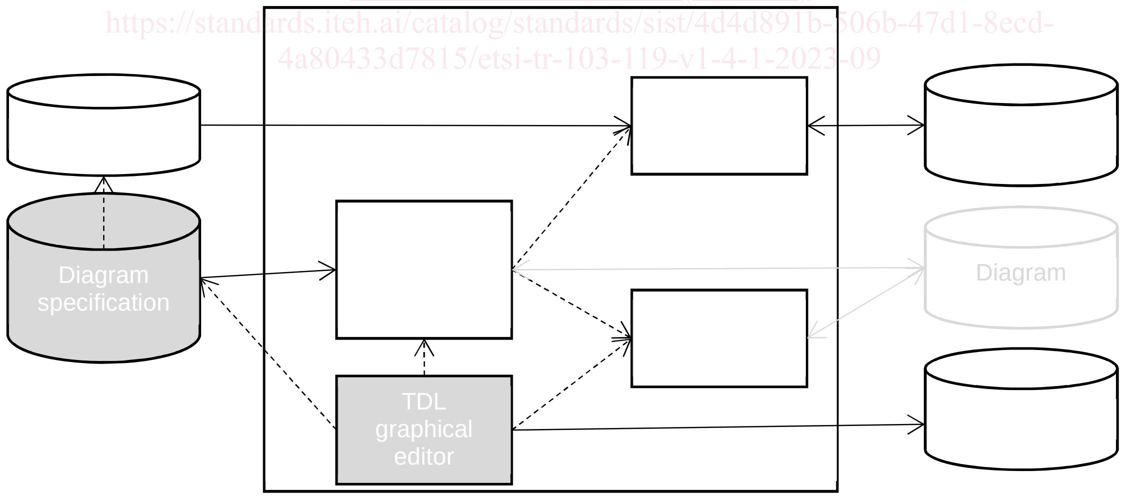
Diagram export for documentation purposes is provided by the framework. The implementation can provide complimentary export to the Word® document format.

Due to the peculiarities and intended use of structured test objectives, it was determined that instead of graphical shapes that can be exported as images, the graphical representation are realized as tables exported directly in a Word document according to user-defined templates. These tables can then be manipulated further as necessary to fit in within an existing document.

5.2 Architecture and Technology Foundation

5.2.1 Graphical Editor

The TDL graphical editor is built on top of the Eclipse platform to benefit from its wide range of modelling tools. The main Eclipse projects that are used as basis for this implementation are shown in Figure 5.2.1-1. Sirius is a technology that allows declarative creation of graphical editors that work with EMF models. It uses GMF [i.7] to create visual diagram elements and link those to model objects. Model management and serialization is done by EMF [i.4].



NOTE: Components with grey background are part of the implementation that is covered by the present document.

Figure 5.2.1-1: Dependencies and data flows of the TDL graphical editor

Every EMF model is based on a meta-model that is defined in terms of meta-modelling system named Ecore. The TDL meta-model in UML format was converted to an Ecore meta-model (TDL Ecore) using the Papyrus UML and EMF facilities. Furthermore, Java code for the TDL meta-model was generated based on the TDL meta-model.

Sirius creates diagram editors by interpreting diagram specification files. These files contain TDL meta-model references in the form of Java or OCL [i.8] queries. OCL support is provided by the Eclipse OCL project [i.9], Java queries are references to classes that are part of the TDL graphical editor and editor source code. Diagram specifications also contain definitions of Sirius specific styles that are applied to model objects when rendering them on diagrams. Since the TDL graphical editor requires customized shapes, it has dependencies on both the Sirius API and the Eclipse GMF. Several extensions to GMF classes have been implemented in Sirius in order to configure shapes according to the customized styles. GMF facilities are then used to export the diagrams as images.

Some of the labels in the graphical shapes, in particular labels related to data specification and data use have a complex structure. For their realization, facilities provided by Xtext [i.1] are used to serialize model fragments related to data use as text according to an annotated EBNF grammar derived from the formal label specifications in ETSI ES 203 119-2 [i.14].

5.2.2 Structured Test Objective Representation

Structured test objectives are exported as tables in a Word document according to user-defined templates. The export relies on facilities provided by Xtext as well as the Apache POI library [i.22] (previously the Docx4j library [i.10] was used) providing API for manipulating Word documents. The exporting facilities take a Word document containing one or more templates in the form of tables with placeholders and a TDL model containing one or more structured test objectives as input. The user has to provide the name of the desired template as an additional input. For a given TDL specification, the selected template is used to generate a tabular representation for every structured test objective. The placeholders in the template are replaced by the content serialized from the corresponding TDL element according to Xtext mappings in a similar manner as the labels for the TDL graphical editor. Existing packaging structures within the TDL specification are used to organize the generated tabular representations with corresponding headings. The generation process is sketched in Figure 5.2.2-1. The generated tables in the new Word document can be further manipulated or merged into an existing document containing additional information. Additional templates may be defined by the users to suit their specific needs.

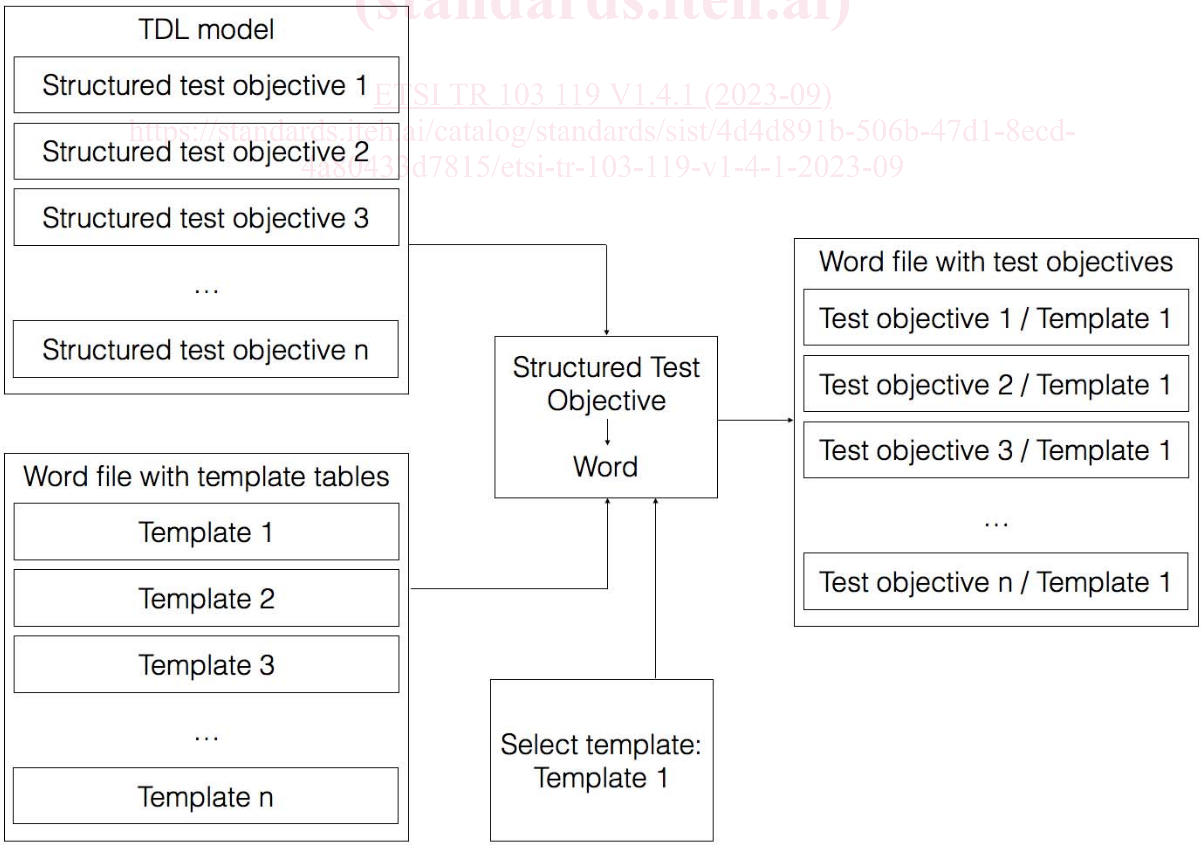

TDL model
Structured test objective 1
Structured test objective 2
Structured test objective 3
…
Structured test objective n
Word file with template tables
Template 1
Template 2
Template 3
…
Template n
Structured Test Objective
Word
Select template: Template 1
Word file with test objectives
Test objective 1 / Template 1
Test objective 2 / Template 1
Test objective 3 / Template 1
…
Test objective n / Template 1

Figure 5.2.2-1: Structured test objective generation process

5.3 Implemented Facilities

5.3.1 Creating Models

Overview

Model instances are the primary artefacts for TDL. They carry the semantic information. In a modelling environment there are various means for creating, viewing, and manipulating model instances of a particular meta-model. Comprehensive modelling environments typically provide generic facilities that enable working with model instances of arbitrary meta-models, provided the meta-model is known. Generic facilities provide sufficient capabilities for performing basic tasks on model instances. However, due to their generic nature, they are often cumbersome to work with, lack support for certain features that are not expressed in the meta-model directly (unless customized), and do not provide domain-specific features, such as syntactical customization beyond basic adaptations.

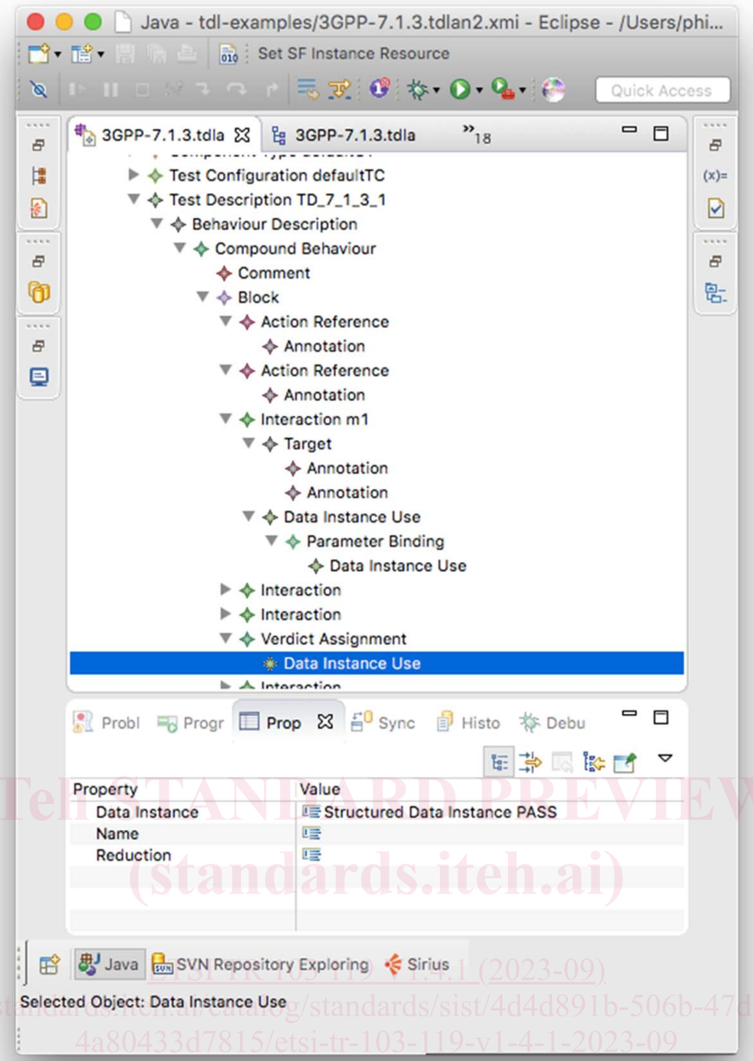
Custom syntax implementations address some of the shortcomings of generic model editors. Such implementations enable the specification of a customized representation of a model instance in a format that is tailored to a specific group of users. There may be multiple custom syntax implementations mapped to the same meta-model, serving different stakeholders or even different purposes for the same stakeholder. Custom syntax implementations may cover only a subset of the meta-model, restricting the access to certain features that are not relevant for specific stakeholders or purposes. Modelling environments provide platforms for the realization of custom syntax implementations. Custom syntax implementations may rely on secondary artefacts that store the concrete representation of the TDL model instance.

TDL model instances may be produced automatically by tools. The exchange format for TDL enables the interoperability of tools producing model instances and tools for manipulating model instances.

Generic Model Editors

The EMF provides facilities for generating basic tree editors for a given meta-model, which can then be customized to an extent while still remaining within the tree editor paradigm. In addition, the EMF also provides generic reflective model editors which provide quick access to model instances of any meta-model. An example of such an editor for TDL is shown in Figure 5.3.1-1. The example includes a tree-based editor for manipulating the overall structure of a model on top and a detailed property view for manipulating individual properties on the bottom.

Extensions to the EMF platform, such as MoDisco, include additional generic facilities such as the MoDisco Model Browser which provides faceted browsing and editing of model instances. Faceted browsing provides filtering by type, as well as deep navigation across references. In addition, MoDisco also includes tabular views on different parts of the model for a quick overview across multiple dimensions. An example for a TDL model is illustrated in Figure 5.3.1-1. The example includes a faceted browser on the top for navigating and manipulating the overall structure of a model, as well as individual properties of model elements. On the left side of the faceted browser, model elements can be filtered by type. Below the faceted browser, a tabular editor provides more compact representation of multiple model elements at the same level in a model tree, such as the behaviour elements of a block. The property view on the bottom part of the example still allows the manipulation of properties of selected model elements.



```
▶ Test Configuration defaultTC
▼ Test Description TD_7_1_3_1
  ▼ Behaviour Description
    ▼ Compound Behaviour
        Comment
      ▼ Block
        ▼ Action Reference
            Annotation
        ▼ Action Reference
            Annotation
        ▼ Interaction m1
          ▼ Target
              Annotation
              Annotation
          ▼ Data Instance Use
            ▼ Parameter Binding
                Data Instance Use
        ▶ Interaction
        ▶ Interaction
        ▼ Verdict Assignment
            Data Instance Use
```

Property	Value
Data Instance	Structured Data Instance PASS
Name	
Reduction	

Selected Object: Data Instance Use

Figure 5.3.1-1: Example of reflective model editor