7th UCAAT User Conference on Advanced Automated Testing

ETSI

Bordeaux, 22-24 October 2019

THE ETSI TEST DESCRIPTION LANGUAGE (TDL)

Philip Makedonski, Andreas Ulrich (on behalf of MTS-TDL Working Group)

Where does TDL come from?

- European Telecommunication Standards Institute (ETSI)
  - develops standards and test specifications for ICT to facilitate interoperability
  - domains: fixed, mobile, radio, aeronautical, broadcast and internet technologies
- Technical Committee on Methods for Testing and Specification (TC MTS)
  - standardising test and specification methods and languages, guidelines, frameworks
  - Testing and Test Control Notation version 3 (TTCN-3)
  - Test Description Language (TDL)
- Centre for Testing and Interoperability (CTI)
  - evaluates test specification technologies
  - provides hands-on support and assistance to TCs and projects

Where does TDL come from?

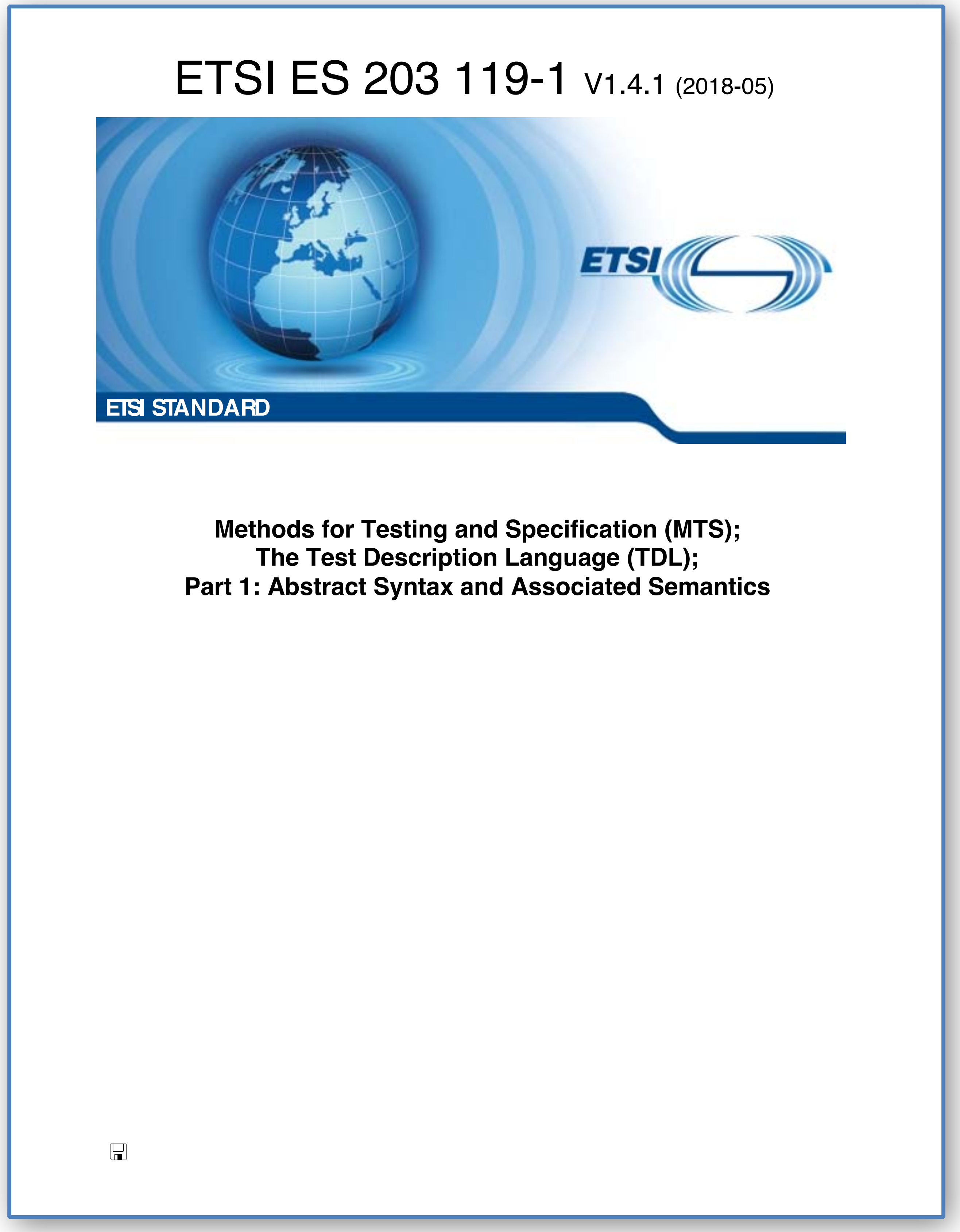
- **Agile**
  - support of test-driven / behaviour-driven development
  - derive scenario-based tests from user stories
  - address different stakeholders through multiple representations

- **Models**
  - describe test-related interfaces, configurations, behaviour, and data
  - generate of abstract tests from test specifications
  - integrate into model-driven software development processes

- **Automation**
  - common and frequently used test patterns
  - clearly defined execution semantics
  - generation of concrete (executable) tests from test specifications

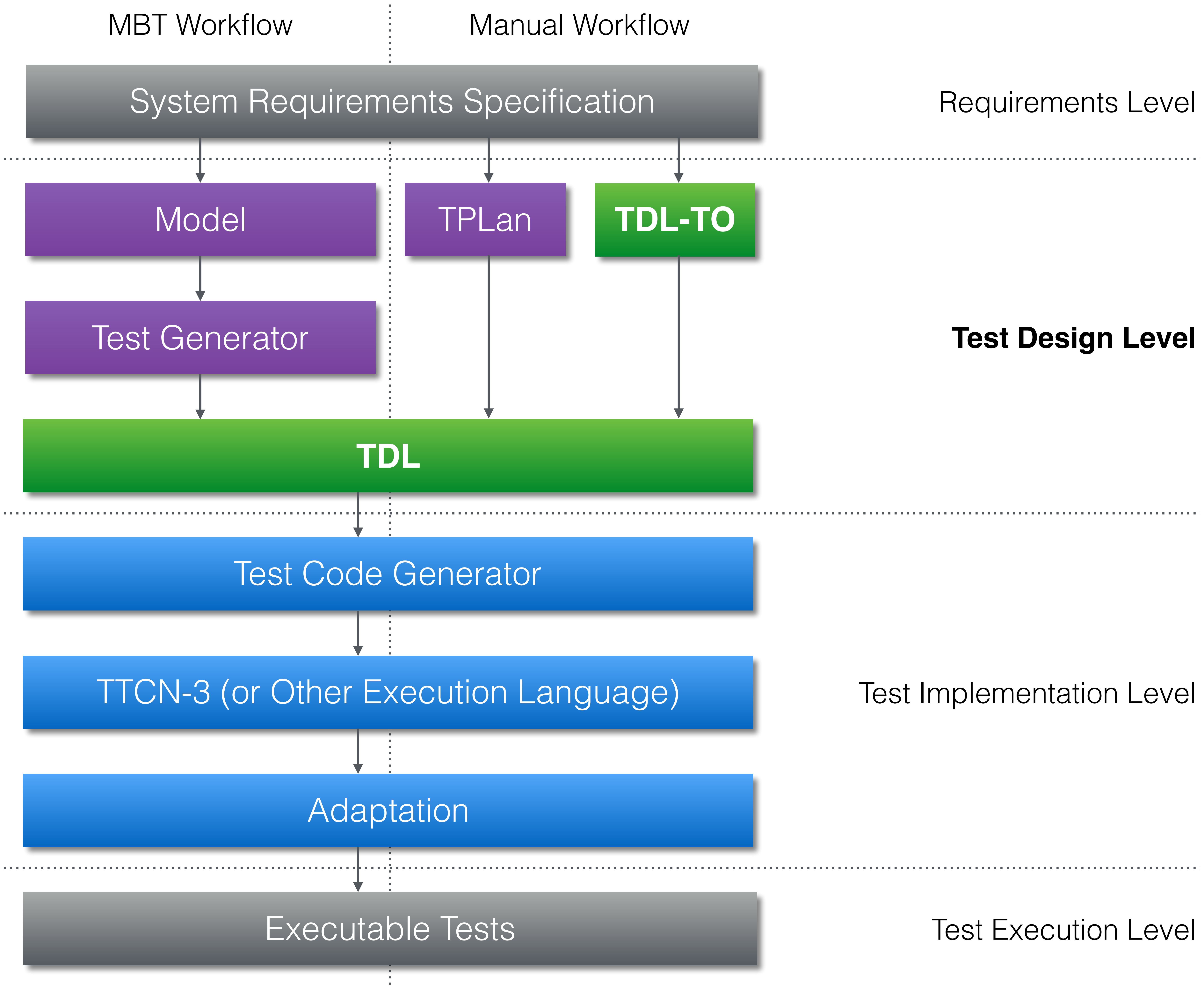
What is TDL?

- Test Description Language
  - Design, documentation, and representation of formalised test descriptions
  - Scenario-based approach
- Standardised at ETSI by TC MTS
  - STF 454 (2013)
  - STF 476 (2014)
  - STF 492 (2015-2016)
  - STF 522 (2017)



ETSI ES 203 119-1 V1.4.1 (2018-05)

Methods for Testing and Specification (MTS);
The Test Description Language (TDL);
Part 1: Abstract Syntax and Associated Semantics


MBT Workflow
Manual Workflow
System Requirements Specification
Requirements Level
Model
TPLan
TDL-TO
Test Generator
Test Design Level
TDL
Test Code Generator
TTCN-3 (or Other Execution Language)
Test Implementation Level
Adaptation
Executable Tests
Test Execution Level

What is TDL?

ETSI ES 203 119-1 V1.4.1 (2018-05)

Methods for Testing and Specification (MTS);
The Test Description Language (TDL);
Part 1: Abstract Syntax and Associated Semantics

Part 1: MM
Meta Model
and Semantics

+

ETSI ES 203 119-2 V1.3.1 (2018-05)

Methods for Testing and Specification (MTS);
The Test Description Language (TDL);
Part 2: Graphical Syntax

ETSI ES 203 119-3 V1.3.1 (2018-05)

Methods for Testing and Specification (MTS);
The Test Description Language (TDL);
Part 3: Exchange Format

ETSI ES 203 119-4 V1.3.1 (2018-05)

Methods for Testing and Specification (MTS);
The Test Description Language (TDL);
Part 4: Structured Test Objective Specification (Extension)

Part 2: GR
Graphical
Syntax

Part 3: XF
Exchange
Format

Part 4: TO
Structured
Test Objective
Specification

ETSI ES 203 119-5 V1.1.1 (2018-05)

Methods for Testing and Specification (MTS);
The Test Description Language (TDL);
Part 5: UML profile for TDL

ETSI ES 203 119-6 V1.1.1 (2018-06)

Methods for Testing and Specification (MTS);
The Test Description Language (TDL);
Part 6: Mapping to TTCN-3

ETSI ES 203 119-7 V1.1.1 (2018-05)

Methods for Testing and Specification (MTS);
The Test Description Language (TDL);
Part 7: Extended Test Configurations

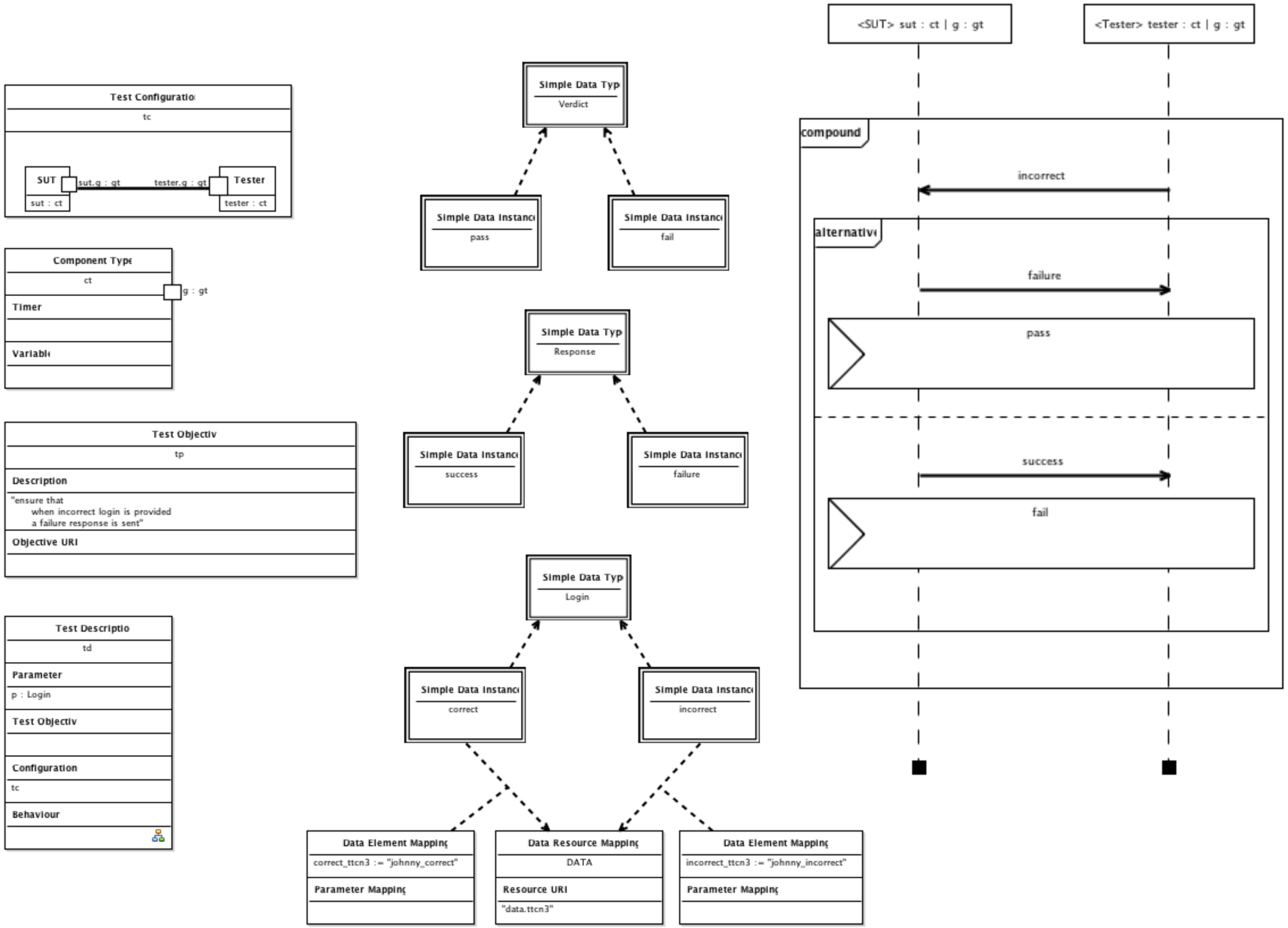
Part 5:
UML Profile
for TDL

Part 6:
Mapping
to TTCN-3

Part 7:
Extended
Test Configurations

What is TDL?

- TDL main ingredients
  - Test data
  - Test configuration
  - Test behaviour
  - Test objectives
  - Time



Structured Test Objectives with TDL-TO

- Requirements to be tested
- Behaviour-driven approach
- Prose syntax

TP Id	TP/GEONW/FDV/BAH/BV/01
Test Objective	Check defined values of default Gn parameters in the basic header
Reference	
PICS Selection	PICS_F1

Initial Conditions

Given

```
with {
  the IUT entity being in the initial state
}
```

Expected Behaviour

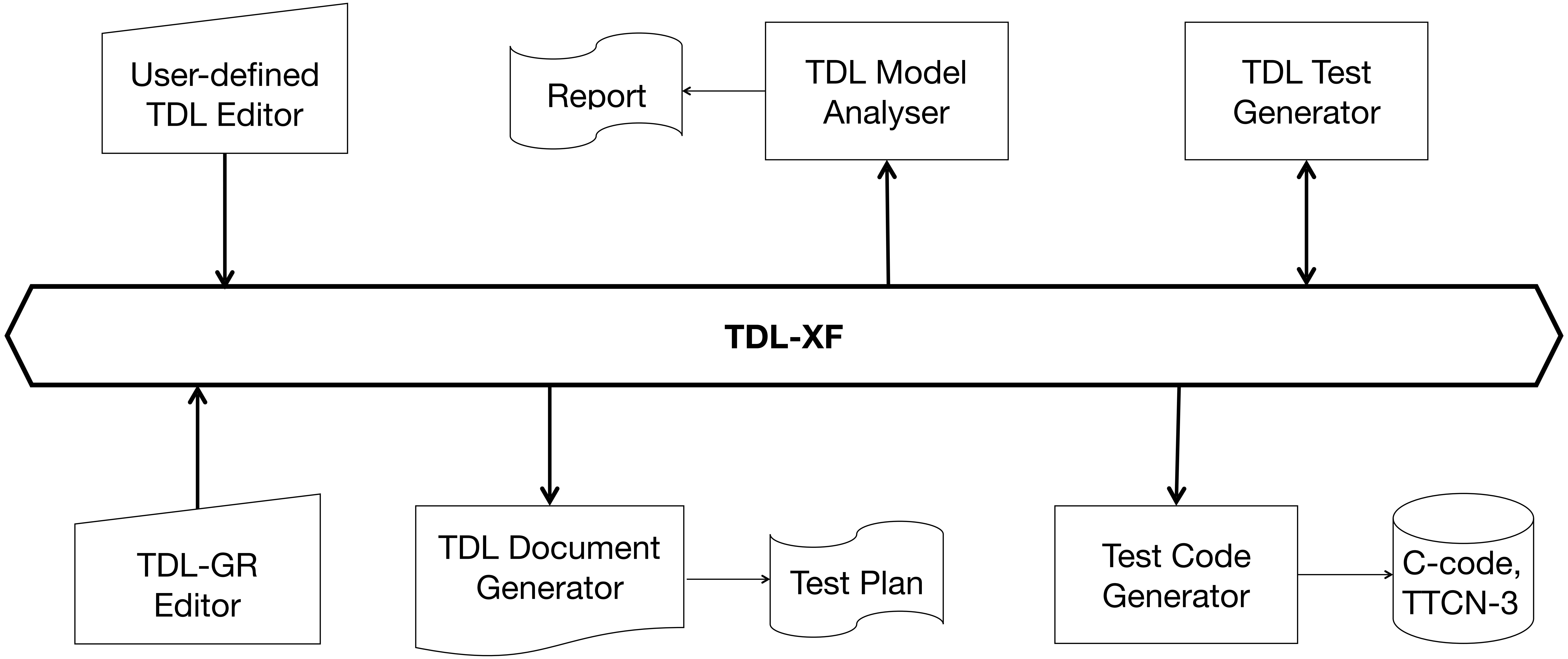
When

Then

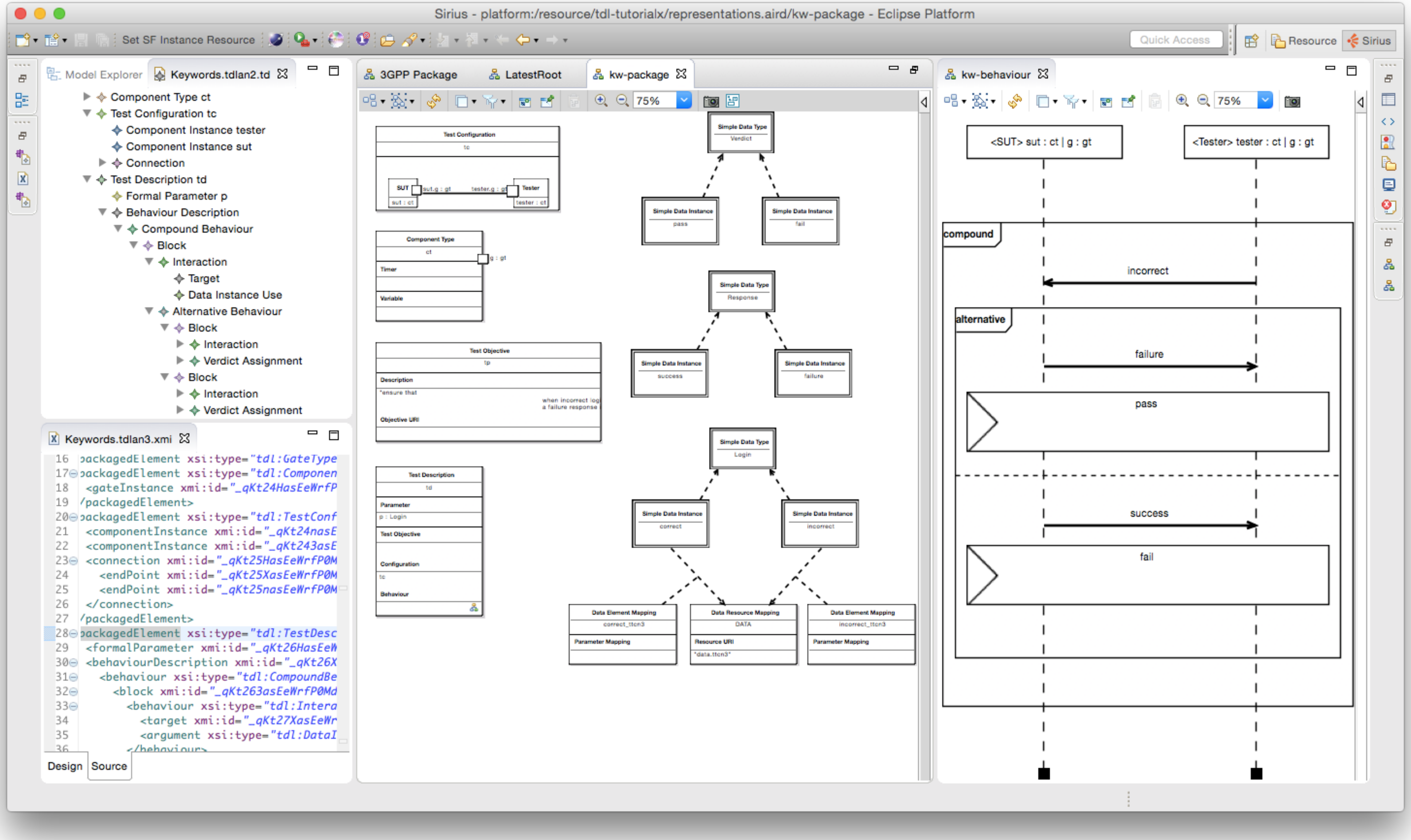
```
ensure that{
  when {
    the IUT entity is requested to send a "GUC packet"
  }
  then {
    the IUT entity sends a "GUC packet" containing
      BasicHeader containing
        "version field" indicating value "itsGnProtocolVersion MIB parameter" ,
        "RHL field" indicating value "itsGnDefaultHopLimit MIB parameter"
      ;
    ;
  }
}
```

Final Conditions

TDL Pipelines



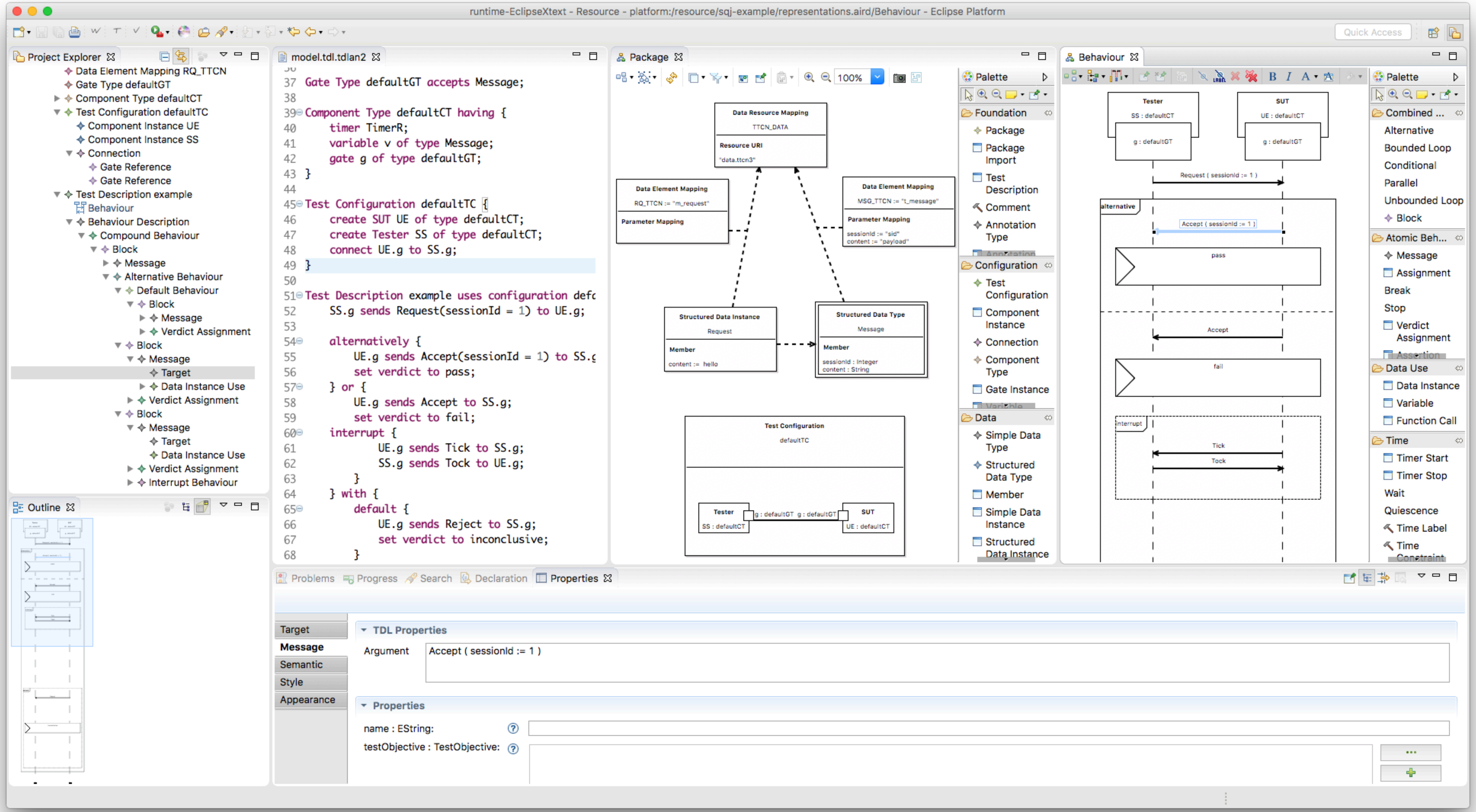
TDL Open Source Project (TOP)



```
▶ Component Type ct
▼ Test Configuration tc
    Component Instance tester
    Component Instance sut
  ▶ Connection
▼ Test Description td
    Formal Parameter p
  ▼ Behaviour Description
    ▼ Compound Behaviour
      ▼ Block
        ▼ Interaction
            Target
            Data Instance Use
        ▼ Alternative Behaviour
          ▼ Block
            ▶ Interaction
            ▶ Verdict Assignment
          ▼ Block
            ▶ Interaction
            ▶ Verdict Assignment
```

```
packagedElement xsi:type="tdl:GateType
packagedElement xsi:type="tdl:Componen
 <gateInstance xmi:id="_qKt24HasEeWrfP
/packagedElement>
packagedElement xsi:type="tdl:TestConf
 <componentInstance xmi:id="_qKt24nasE
 <componentInstance xmi:id="_qKt243asE
 <connection xmi:id="_qKt25HasEeWrfP0M
   <endPoint xmi:id="_qKt25XasEeWrfP0M
   <endPoint xmi:id="_qKt25nasEeWrfP0M
 </connection>
/packagedElement>
packagedElement xsi:type="tdl:TestDesc
 <formalParameter xmi:id="_qKt26HasEeW
 <behaviourDescription xmi:id="_qKt26X
   <behaviour xsi:type="tdl:CompoundBe
     <block xmi:id="_qKt263asEeWrfP0Md
       <behaviour xsi:type="tdl:Intera
         <target xmi:id="_qKt27XasEeWr
         <argument xsi:type="tdl:DataI
       </behaviour>
```

TDL Open Source Project (TOP)



- Data Element Mapping RQ_TTCN
- Gate Type defaultGT
- Component Type defaultCT
- Test Configuration defaultTC
  - Component Instance UE
  - Component Instance SS
  - Connection
    - Gate Reference
    - Gate Reference
- Test Description example
  - Behaviour
  - Behaviour Description
    - Compound Behaviour
      - Block
        - Message
        - Alternative Behaviour
          - Default Behaviour
            - Block
              - Message
              - Verdict Assignment
          - Block
            - Message
              - Target
              - Data Instance Use
            - Verdict Assignment
          - Block
            - Message
              - Target
              - Data Instance Use
            - Verdict Assignment
            - Interrupt Behaviour

```
Gate Type defaultGT accepts Message;

Component Type defaultCT having {
    timer TimerR;
    variable v of type Message;
    gate g of type defaultGT;
}

Test Configuration defaultTC {
    create SUT UE of type defaultCT;
    create Tester SS of type defaultCT;
    connect UE.g to SS.g;
}

Test Description example uses configuration defc
    SS.g sends Request(sessionId = 1) to UE.g;

    alternatively {
        UE.g sends Accept(sessionId = 1) to SS.g
        set verdict to pass;
    } or {
        UE.g sends Accept to SS.g;
        set verdict to fail;
    interrupt {
            UE.g sends Tick to SS.g;
            SS.g sends Tock to UE.g;
        }
    } with {
        default {
            UE.g sends Reject to SS.g;
            set verdict to inconclusive;
        }
```

Target

Message

Semantic

Style

Appearance

TDL Properties

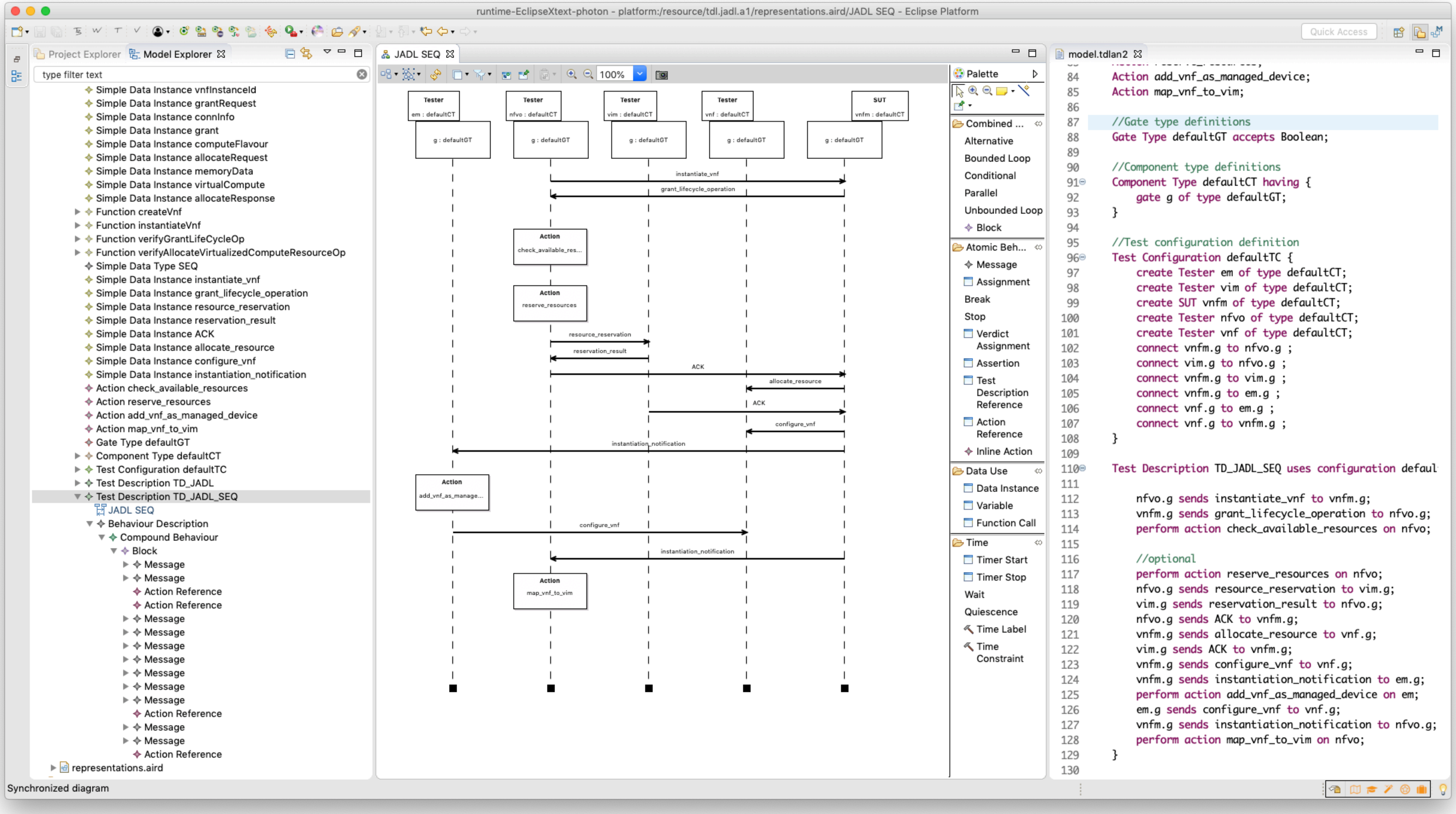
Argument: Accept (sessionId := 1)

Properties

name : EString:

testObjective : TestObjective:

TDL Open Source Project (TOP)

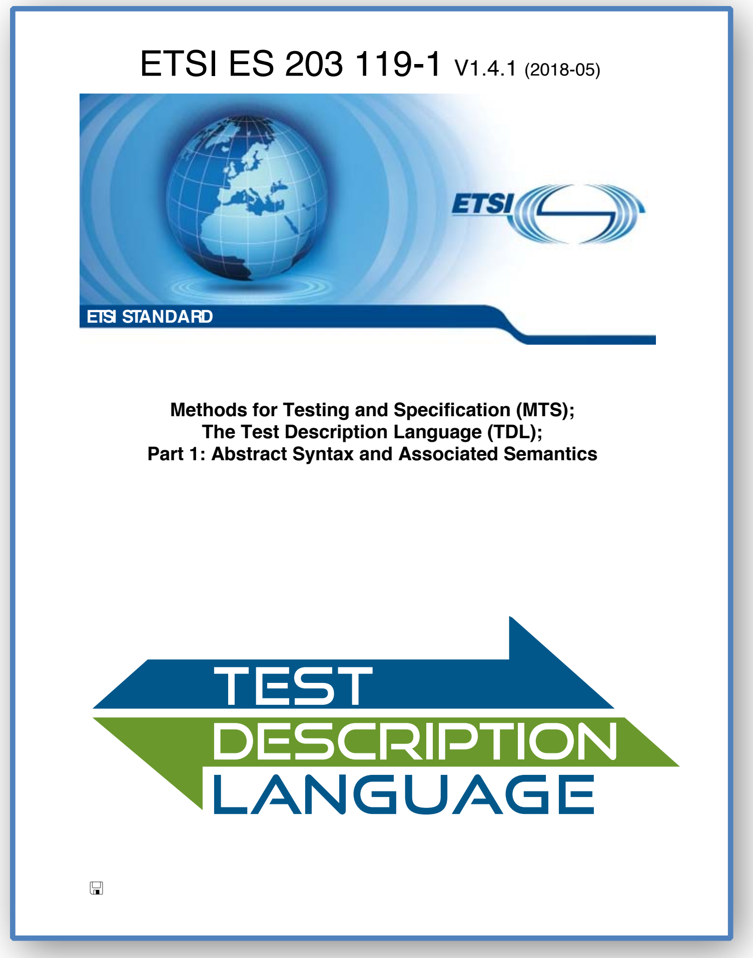


```
    Action add_vnf_as_managed_device;
    Action map_vnf_to_vim;

    //Gate type definitions
    Gate Type defaultGT accepts Boolean;

    //Component type definitions
    Component Type defaultCT having {
        gate g of type defaultGT;
    }

    //Test configuration definition
    Test Configuration defaultTC {
        create Tester em of type defaultCT;
        create Tester vim of type defaultCT;
        create SUT vnfm of type defaultCT;
        create Tester nfvo of type defaultCT;
        create Tester vnf of type defaultCT;
        connect vnfm.g to nfvo.g ;
        connect vim.g to nfvo.g ;
        connect vnfm.g to vim.g ;
        connect vnfm.g to em.g ;
        connect vnf.g to em.g ;
        connect vnf.g to vnfm.g ;
    }

    Test Description TD_JADL_SEQ uses configuration defaul

        nfvo.g sends instantiate_vnf to vnfm.g;
        vnfm.g sends grant_lifecycle_operation to nfvo.g;
        perform action check_available_resources on nfvo;

        //optional
        perform action reserve_resources on nfvo;
        nfvo.g sends resource_reservation to vim.g;
        vim.g sends reservation_result to nfvo.g;
        nfvo.g sends ACK to vnfm.g;
        vnfm.g sends allocate_resource to vnf.g;
        vim.g sends ACK to vnfm.g;
        vnfm.g sends configure_vnf to vnf.g;
        vnfm.g sends instantiation_notification to em.g;
        perform action add_vnf_as_managed_device on em;
        em.g sends configure_vnf to vnf.g;
        vnfm.g sends instantiation_notification to nfvo.g;
        perform action map_vnf_to_vim on nfvo;
    }
```

Mapping TDL to TTCN-3

- Test Description Language
  - Design, documentation, representation of formalised test descriptions
  - Scenario-based approach

- Testing and Test Control Notation
  - Specification and implementation of all kinds of black-box tests
  - Component-based approach



ETSI ES 203 119-1 V1.4.1 (2018-05)

ETSI STANDARD

Methods for Testing and Specification (MTS);
The Test Description Language (TDL);
Part 1: Abstract Syntax and Associated Semantics

ETSI ES 203 119-6 V1.1.1 (2018-06)

ETSI STANDARD

Methods for Testing and Specification (MTS);
The Test Description Language (TDL);
Part 6: Mapping to TTCN-3

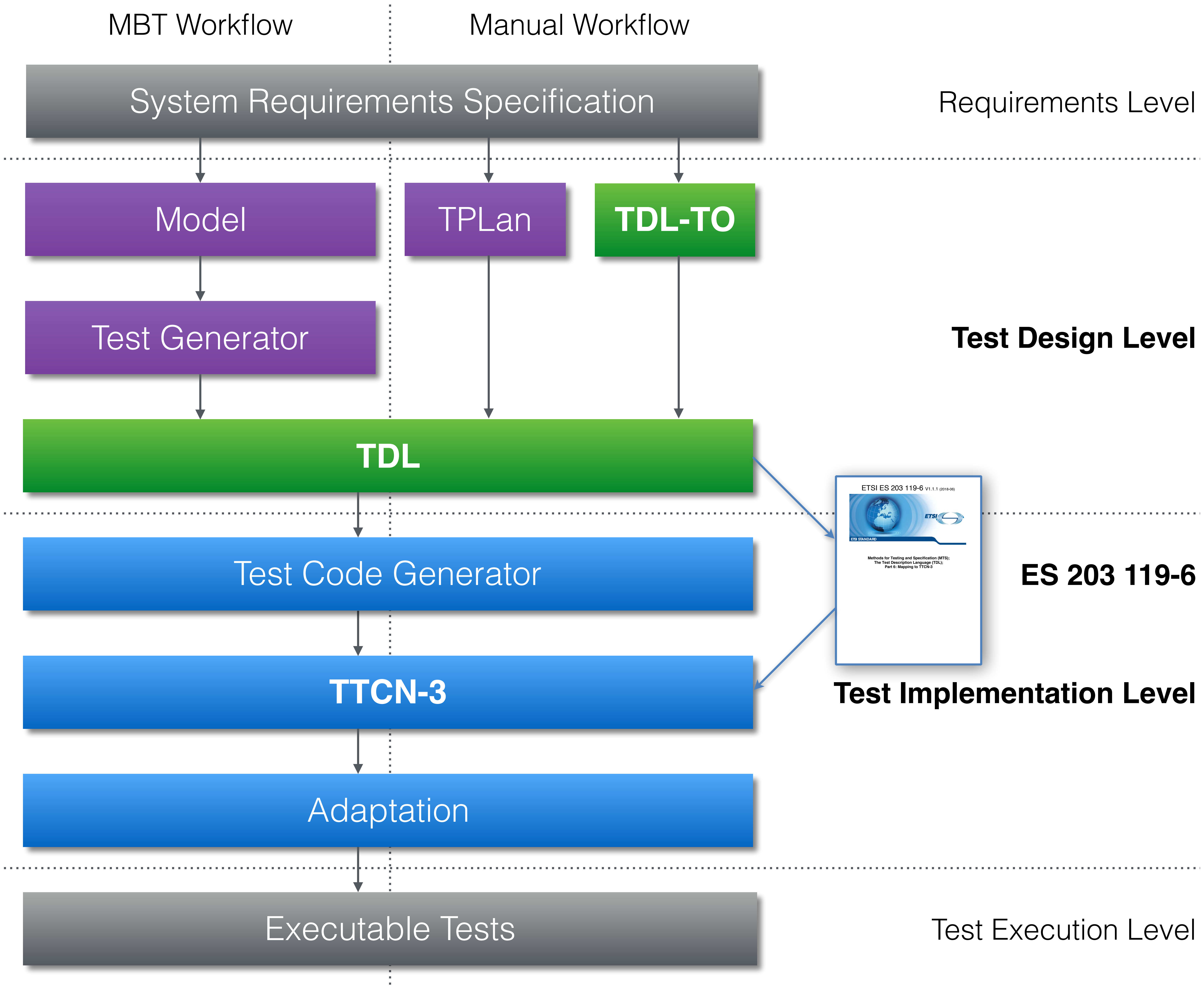
Mapping TDL to TTCN-3

- Establish a connection between TDL and TTCN-3
  - generation of executable tests from test descriptions
  - standardised, ensuring compatibility and consistency
  - re-use existing tools and frameworks for test execution
  - re-use existing TTCN-3 assets (data, behaviour)

ETSI ES 203 119-6 V1.1.1 (2018-06)

ETSI STANDARD

Methods for Testing and Specification (MTS);
The Test Description Language (TDL);
Part 6: Mapping to TTCN-3



MBT Workflow

Manual Workflow

System Requirements Specification

Requirements Level

Model

TPLan

TDL-TO

Test Generator

Test Design Level

TDL

ETSI ES 203 119-6 V1.1.1 (2018-06)

ETSI STANDARD

Methods for Testing and Specification (MTS);
The Test Description Language (TDL);
Part 6: Mapping to TTCN-3

ES 203 119-6

Test Code Generator

TTCN-3

Test Implementation Level

Adaptation

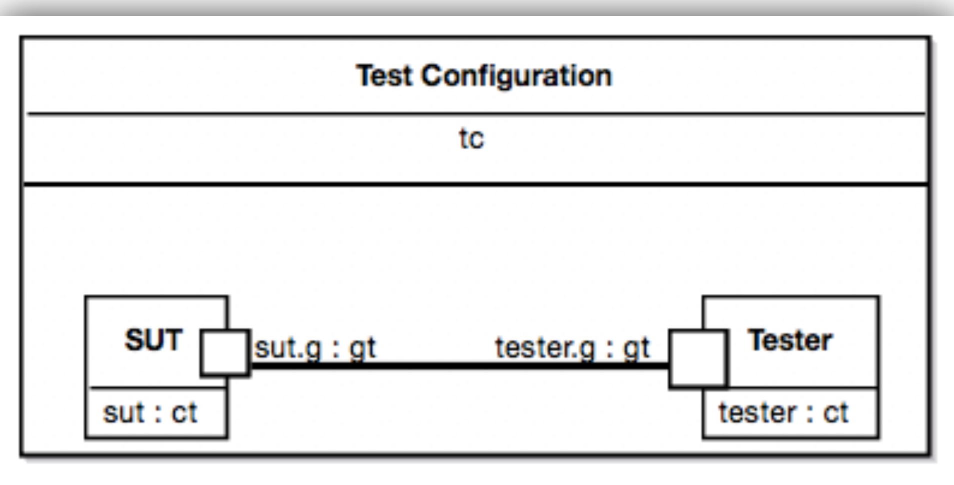
Executable Tests

Test Execution Level

Mapping TDL to TTCN-3

```
Gate Type gt accepts Login, Response;

Component Type ct having {
      gate g of type gt;
}

Test Configuration tc {
      create Tester tester of type ct;
      create SUT sut of type ct;
      connect tester.g to sut.g;
}
```

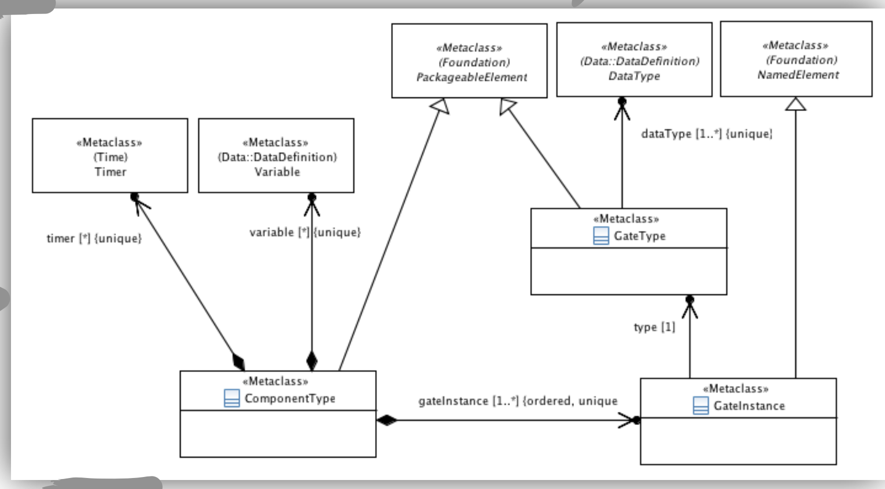
Xtext



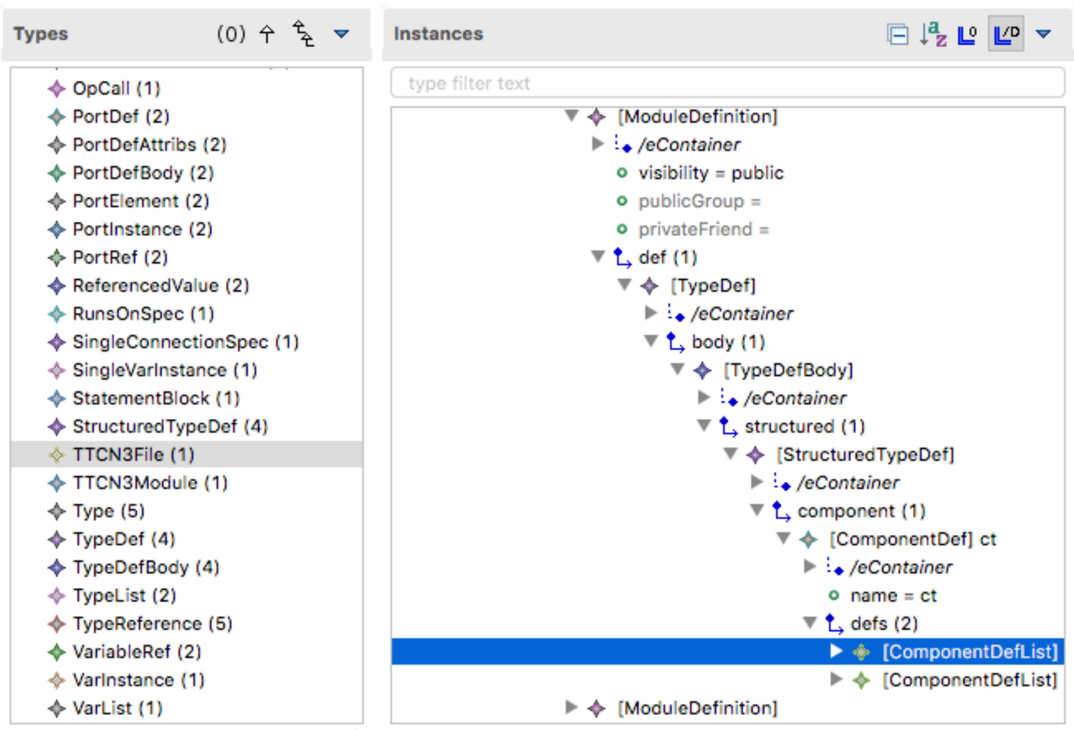
Sirius

```
<packagedElement xsi:type="tdl:ComponentType"
             xmi:id="_qKt233asEeWrfP0MdfQNpg"
             name="ct">
  <gateInstance xmi:id="_qKt24HasEeWrfP0MdfQNpg"
            name="g"
            type="_qKt23nasEeWrfP0MdfQNpg"/>
</packagedElement>
```

emf
ECLIPSE MODELING FRAMEWORK



epsilon



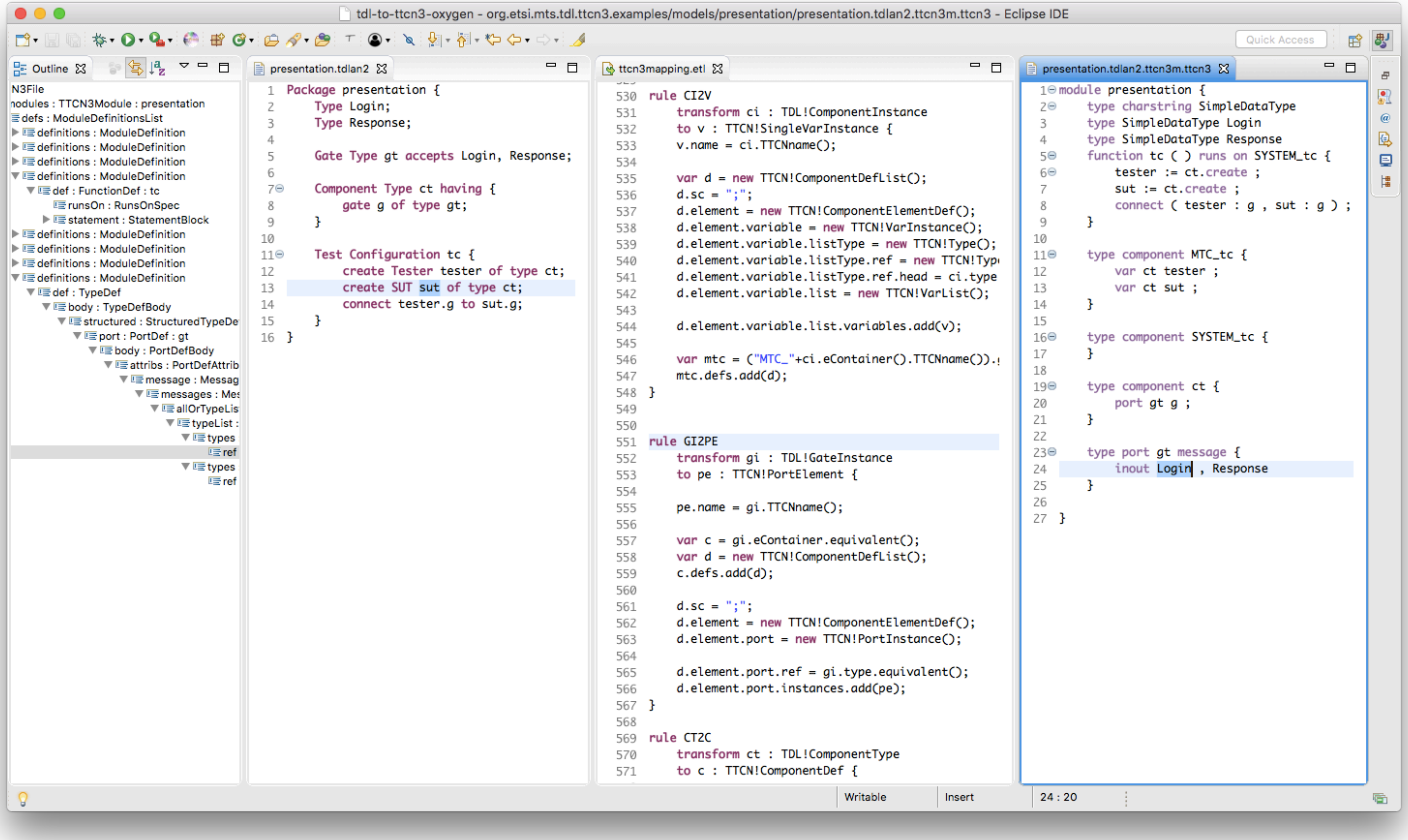
OpCall (1)
PortDef (2)
PortDefAttribs (2)
PortDefBody (2)
PortElement (2)
PortInstance (2)
PortRef (2)
ReferencedValue (2)
RunsOnSpec (1)
SingleConnectionSpec (1)
SingleVarInstance (1)
StatementBlock (1)
StructuredTypeDef (4)
TTCN3File (1)
TTCN3Module (1)
Type (5)
TypeDef (4)
TypeDefBody (4)
TypeList (2)
TypeReference (5)
VariableRef (2)
VarInstance (1)
VarList (1)

```
[ModuleDefinition]
  /eContainer
  visibility = public
  publicGroup =
  privateFriend =
  def (1)
    [TypeDef]
      /eContainer
      body (1)
        [TypeDefBody]
          /eContainer
          structured (1)
            [StructuredTypeDef]
              /eContainer
              component (1)
                [ComponentDef] ct
                  /eContainer
                  name = ct
                  defs (2)
                    [ComponentDefList]
                    [ComponentDefList]
[ModuleDefinition]
```

Xtext

```
type port gt_to_map message {
  //port type for SUT-Tester connections
  inout Login, Response
}

type port gt_to_connect message {
  //port type for Tester-Tester connections
  inout Login, Response
}

type component MTC_CT {
  //component type for MTC
  //variable for the PTC(s)
  var ct TESTER_tester;
}

type component ct {
  port gt_to_map g_to_map;
  port gt_to_connect g_to_connect;
}

function tc() runs on MTC_CT {
  // Test Configuration tc, mappings, connections
  TESTER_tester := ct.create;
  map (TESTER_tester:g_to_map,system:g_to_map);
}
```

Mapping TDL to TTCN-3


tdl-to-ttcn3-oxygen - org.etsi.mts.tdl.ttcn3.examples/models/presentation/presentation.tdlan2.ttcn3m.ttcn3 - Eclipse IDE
Quick Access
Outline
presentation.tdlan2
ttcn3mapping.etl
presentation.tdlan2.ttcn3m.ttcn3
Writable
Insert
24 : 20

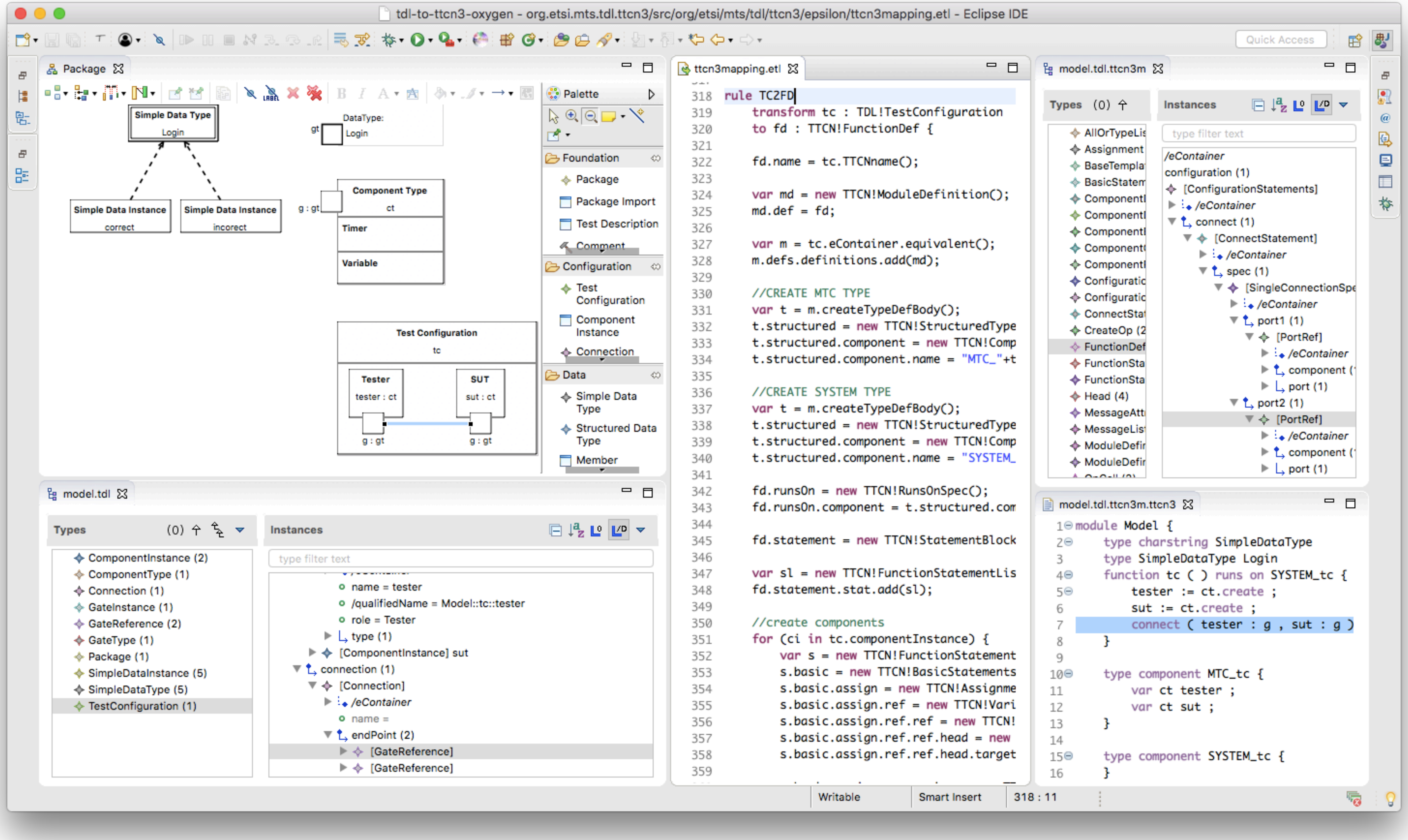
```
N3File
nodules : TTCN3Module : presentation
 defs : ModuleDefinitionsList
  ▶ definitions : ModuleDefinition
  ▶ definitions : ModuleDefinition
  ▶ definitions : ModuleDefinition
  ▼ definitions : ModuleDefinition
    ▼ def : FunctionDef : tc
        runsOn : RunsOnSpec
      ▶ statement : StatementBlock
  ▶ definitions : ModuleDefinition
  ▶ definitions : ModuleDefinition
  ▶ definitions : ModuleDefinition
  ▼ definitions : ModuleDefinition
    ▼ def : TypeDef
      ▼ body : TypeDefBody
        ▼ structured : StructuredTypeDe
          ▼ port : PortDef : gt
            ▼ body : PortDefBody
              ▼ attribs : PortDefAttrib
                ▼ message : Messag
                  ▼ messages : Mes
                    ▼ allOrTypeLis
                      ▼ typeList :
                        ▼ types
                            ref
                        ▼ types
                            ref
```

```
Package presentation {
    Type Login;
    Type Response;

    Gate Type gt accepts Login, Response;

    Component Type ct having {
        gate g of type gt;
    }

    Test Configuration tc {
        create Tester tester of type ct;
        create SUT sut of type ct;
        connect tester.g to sut.g;
    }
}
```

```
rule CI2V
    transform ci : TDL!ComponentInstance
    to v : TTCN!SingleVarInstance {
    v.name = ci.TTCNname();

    var d = new TTCN!ComponentDefList();
    d.sc = ";";
    d.element = new TTCN!ComponentElementDef();
    d.element.variable = new TTCN!VarInstance();
    d.element.variable.listType = new TTCN!Type();
    d.element.variable.listType.ref = new TTCN!Typ
    d.element.variable.listType.ref.head = ci.type
    d.element.variable.list = new TTCN!VarList();

    d.element.variable.list.variables.add(v);

    var mtc = ("MTC_"+ci.eContainer().TTCNname()).
    mtc.defs.add(d);
}


rule GI2PE
    transform gi : TDL!GateInstance
    to pe : TTCN!PortElement {

    pe.name = gi.TTCNname();

    var c = gi.eContainer.equivalent();
    var d = new TTCN!ComponentDefList();
    c.defs.add(d);

    d.sc = ";";
    d.element = new TTCN!ComponentElementDef();
    d.element.port = new TTCN!PortInstance();

    d.element.port.ref = gi.type.equivalent();
    d.element.port.instances.add(pe);
}

rule CT2C
    transform ct : TDL!ComponentType
    to c : TTCN!ComponentDef {
```

```
module presentation {
    type charstring SimpleDataType
    type SimpleDataType Login
    type SimpleDataType Response
    function tc ( ) runs on SYSTEM_tc {
        tester := ct.create ;
        sut := ct.create ;
        connect ( tester : g , sut : g ) ;
    }

    type component MTC_tc {
        var ct tester ;
        var ct sut ;
    }

    type component SYSTEM_tc {
    }

    type component ct {
        port gt g ;
    }

    type port gt message {
        inout Login , Response
    }

}
```

Mapping TDL to TTCN-3



tdl-to-ttcn3-oxygen - org.etsi.mts.tdl.ttcn3/src/org/etsi/mts/tdl/ttcn3/epsilon/ttcn3mapping.etl - Eclipse IDE

ttcn3mapping.etl

```
rule TC2FD
    transform tc : TDL!TestConfiguration
    to fd : TTCN!FunctionDef {

    fd.name = tc.TTCNname();

    var md = new TTCN!ModuleDefinition();
    md.def = fd;

    var m = tc.eContainer.equivalent();
    m.defs.definitions.add(md);

    //CREATE MTC TYPE
    var t = m.createTypeDefBody();
    t.structured = new TTCN!StructuredType
    t.structured.component = new TTCN!Comp
    t.structured.component.name = "MTC_"+t

    //CREATE SYSTEM TYPE
    var t = m.createTypeDefBody();
    t.structured = new TTCN!StructuredType
    t.structured.component = new TTCN!Comp
    t.structured.component.name = "SYSTEM_

    fd.runsOn = new TTCN!RunsOnSpec();
    fd.runsOn.component = t.structured.com

    fd.statement = new TTCN!StatementBlock

    var sl = new TTCN!FunctionStatementLis
    fd.statement.stat.add(sl);

    //create components
    for (ci in tc.componentInstance) {
        var s = new TTCN!FunctionStatement
        s.basic = new TTCN!BasicStatements
        s.basic.assign = new TTCN!Assignme
        s.basic.assign.ref = new TTCN!Vari
        s.basic.assign.ref.ref = new TTCN!
        s.basic.assign.ref.ref.head = new
        s.basic.assign.ref.ref.head.target
```

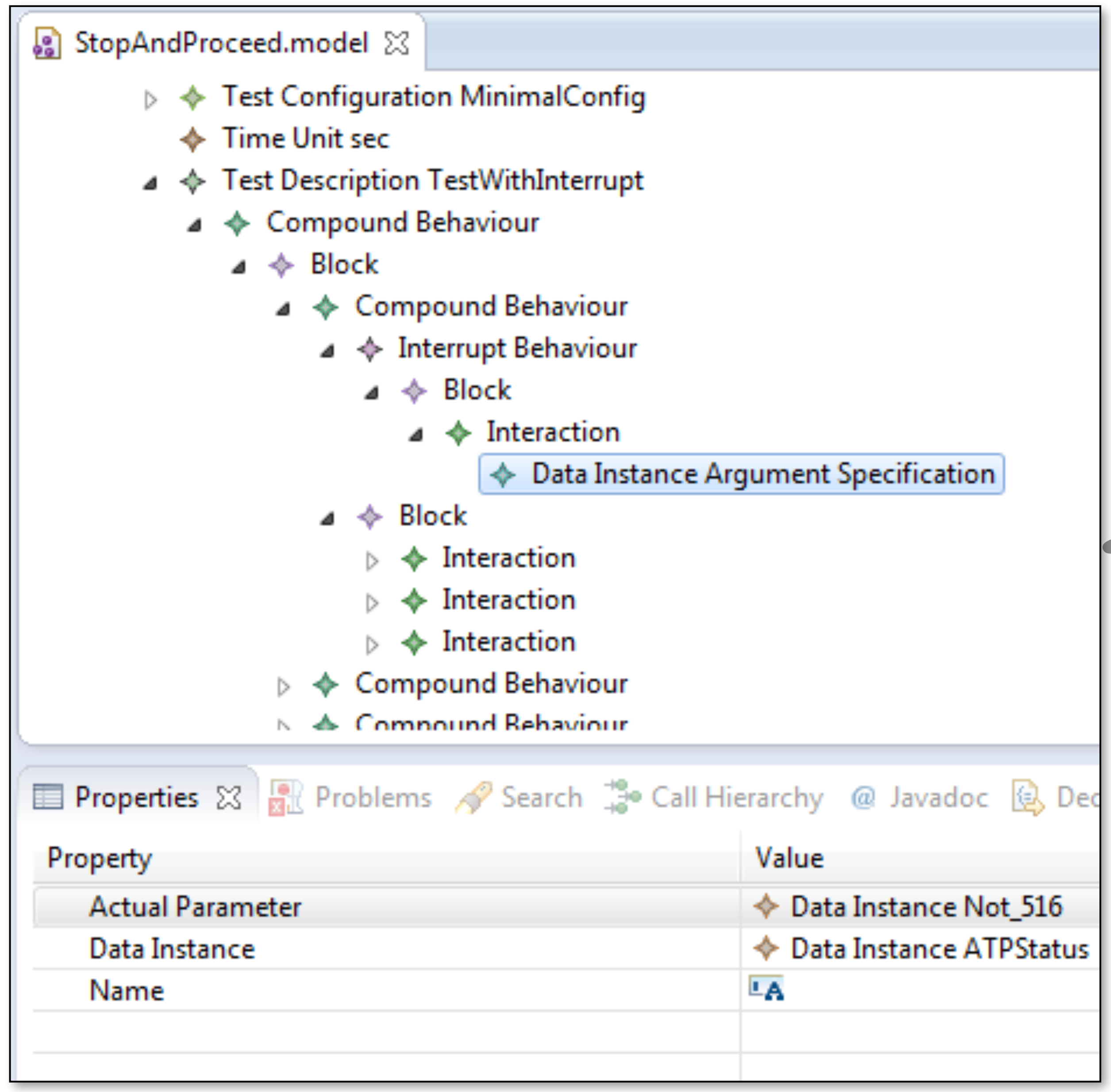
Writable | Smart Insert | 318 : 11

model.tdl.ttcn3m.ttcn3

```
module Model {
    type charstring SimpleDataType
    type SimpleDataType Login
    function tc ( ) runs on SYSTEM_tc {
        tester := ct.create ;
        sut := ct.create ;
        connect ( tester : g , sut : g )
    }

    type component MTC_tc {
        var ct tester ;
        var ct sut ;
    }

    type component SYSTEM_tc {
    }
```

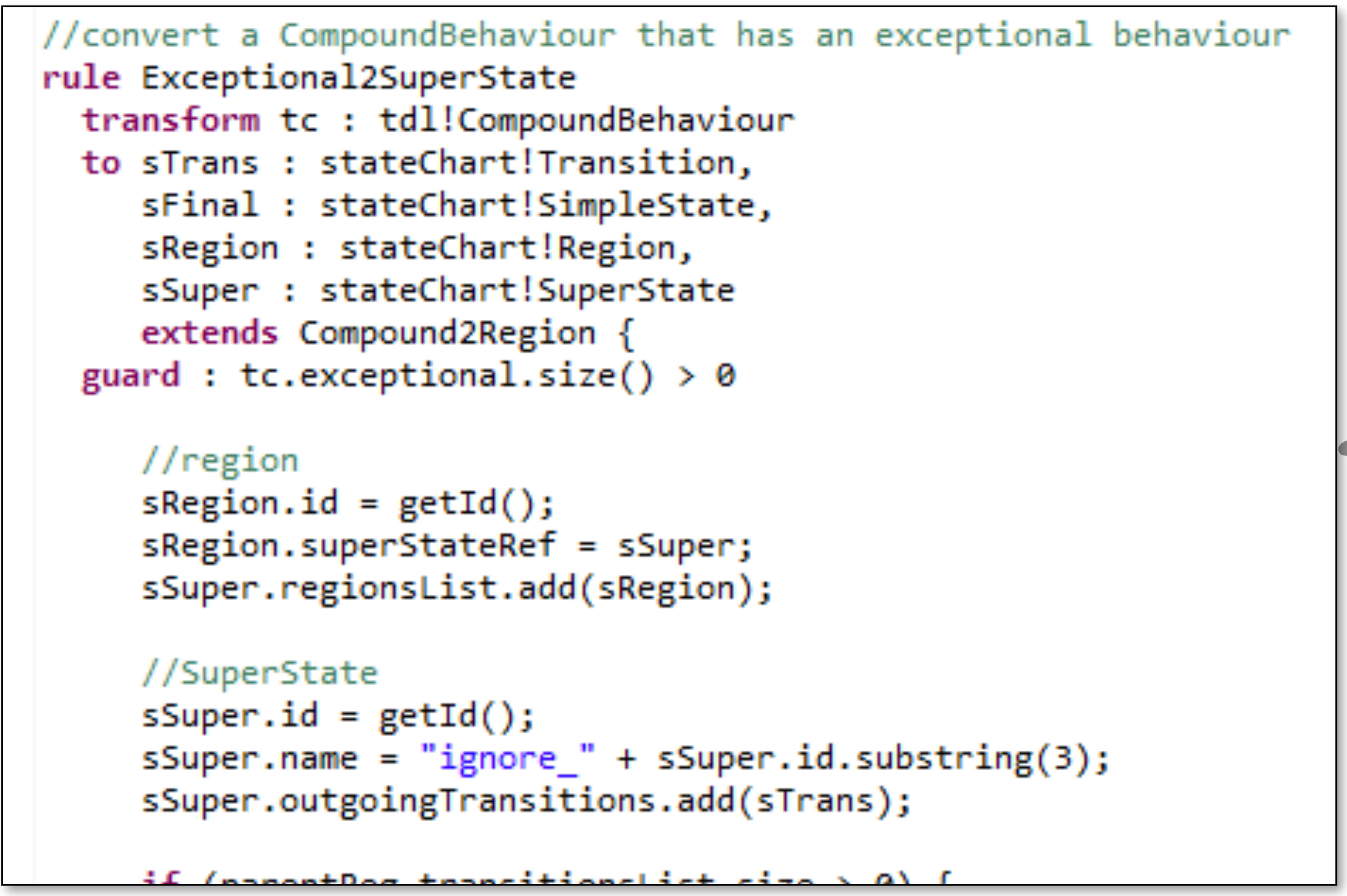
Sirius MoDisco epsilon MoDisco Xtext

Mapping TDL to…

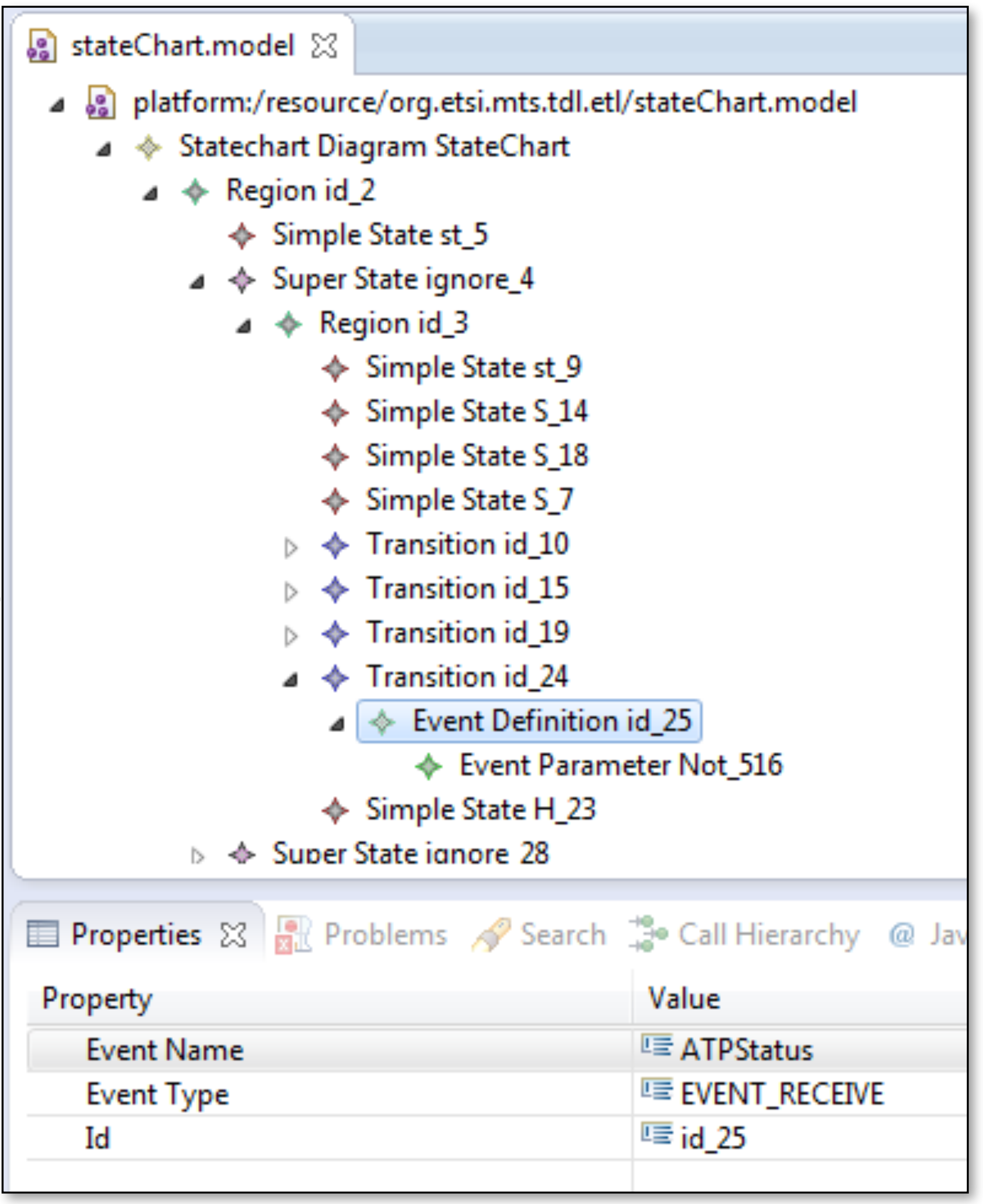


- Test Configuration MinimalConfig
- Time Unit sec
- Test Description TestWithInterrupt
  - Compound Behaviour
    - Block
      - Compound Behaviour
        - Interrupt Behaviour
          - Block
            - Interaction
              - Data Instance Argument Specification
        - Block
          - Interaction
          - Interaction
          - Interaction
      - Compound Behaviour
      - Compound Behaviour

Property	Value
Actual Parameter	Data Instance Not_516
Data Instance	Data Instance ATPStatus
Name	



```
//convert a CompoundBehaviour that has an exceptional behaviour
rule Exceptional2SuperState
  transform tc : tdl!CompoundBehaviour
  to sTrans : stateChart!Transition,
     sFinal : stateChart!SimpleState,
     sRegion : stateChart!Region,
     sSuper : stateChart!SuperState
     extends Compound2Region {
  guard : tc.exceptional.size() > 0

     //region
     sRegion.id = getId();
     sRegion.superStateRef = sSuper;
     sSuper.regionsList.add(sRegion);

     //SuperState
     sSuper.id = getId();
     sSuper.name = "ignore_" + sSuper.id.substring(3);
     sSuper.outgoingTransitions.add(sTrans);

     if (parentReg.transitionsList.size > 0) {
```

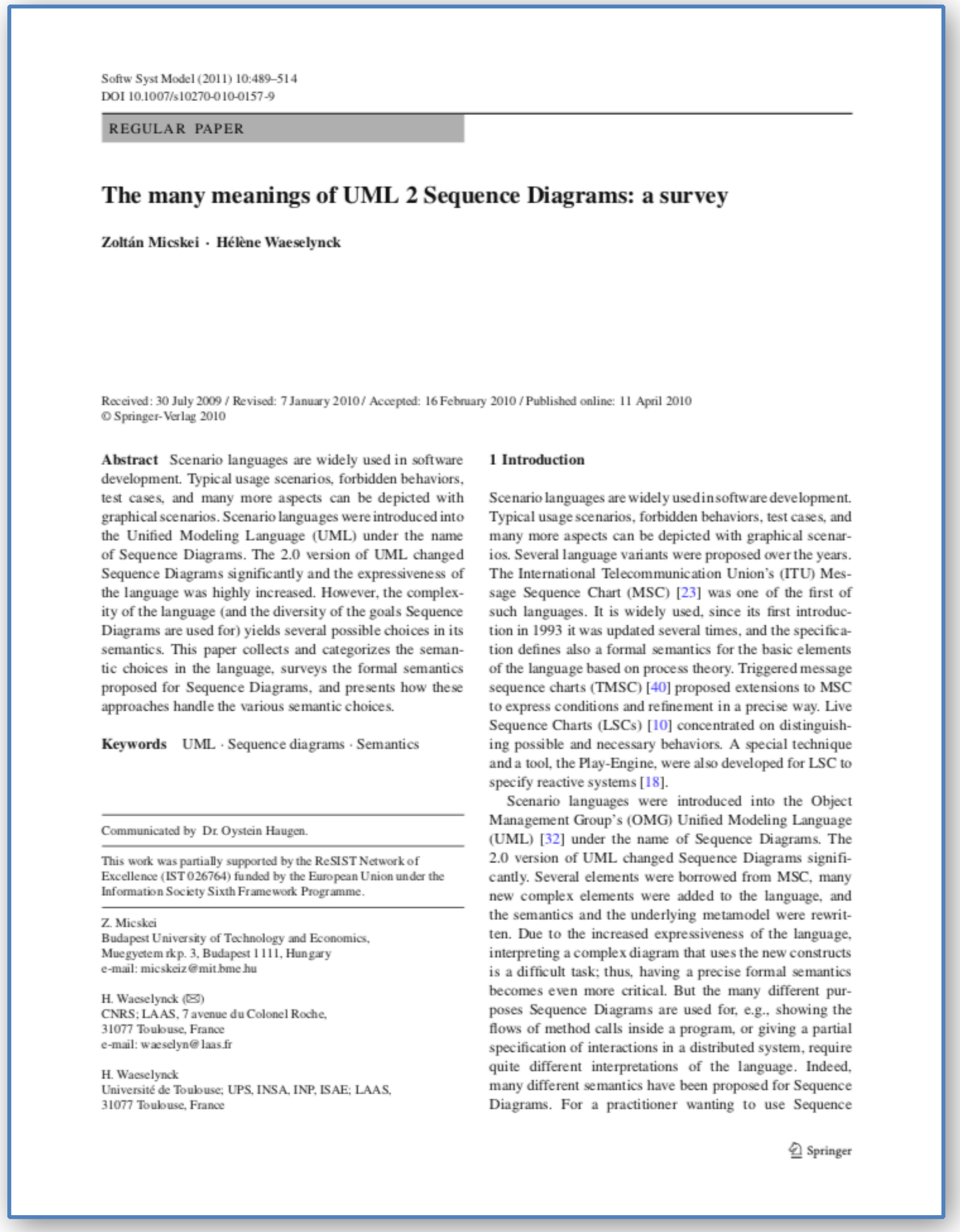


- platform:/resource/org.etsi.mts.tdl.etl/stateChart.model
  - Statechart Diagram StateChart
    - Region id_2
      - Simple State st_5
      - Super State ignore_4
        - Region id_3
          - Simple State st_9
          - Simple State S_14
          - Simple State S_18
          - Simple State S_7
          - Transition id_10
          - Transition id_15
          - Transition id_19
          - Transition id_24
            - Event Definition id_25
              - Event Parameter Not_516
          - Simple State H_23
      - Super State ignore_28

Property	Value
Event Name	ATPStatus
Event Type	EVENT_RECEIVE
Id	id_25

Why not UML / UTP?

- Semantic fuzz of UML
  - different notations
  - different interpretations
- UML Testing Profile (UTP)
  - extension of UML to support (model-based) testing
  - wide scope of modelling notations inherited from UML
  - may still not capture all needs
  - further profiles needed, e.g. MARTE



Softw Syst Model (2011) 10:489–514
DOI 10.1007/s10270-010-0157-9

REGULAR PAPER

The many meanings of UML 2 Sequence Diagrams: a survey

Zoltán Micskei · Hélène Waeselynck

Received: 30 July 2009 / Revised: 7 January 2010 / Accepted: 16 February 2010 / Published online: 11 April 2010

Abstract Scenario languages are widely used in software development. Typical usage scenarios, forbidden behaviors, test cases, and many more aspects can be depicted with graphical scenarios. Scenario languages were introduced into the Unified Modeling Language (UML) under the name of Sequence Diagrams. The 2.0 version of UML changed Sequence Diagrams significantly and the expressiveness of the language was highly increased. However, the complexity of the language (and the diversity of the goals Sequence Diagrams are used for) yields several possible choices in its semantics. This paper collects and categorizes the semantic choices in the language, surveys the formal semantics proposed for Sequence Diagrams, and presents how these approaches handle the various semantic choices.

Keywords UML · Sequence diagrams · Semantics

Communicated by Dr. Oystein Haugen.

This work was partially supported by the ReSIST Network of Excellence (IST 026764) funded by the European Union under the Information Society Sixth Framework Programme.

Z. Micskei
Budapest University of Technology and Economics,
Muegyetem rkp. 3, Budapest 1111, Hungary
e-mail: micskeiz@mit.bme.hu

H. Waeselynck (✉)
CNRS; LAAS, 7 avenue du Colonel Roche,
31077 Toulouse, France
e-mail: waeselyn@laas.fr

H. Waeselynck
Université de Toulouse; UPS, INSA, INP, ISAE; LAAS,
31077 Toulouse, France

1 Introduction

Scenario languages are widely used in software development. Typical usage scenarios, forbidden behaviors, test cases, and many more aspects can be depicted with graphical scenarios. Several language variants were proposed over the years. The International Telecommunication Union's (ITU) Message Sequence Chart (MSC) [23] was one of the first of such languages. It is widely used, since its first introduction in 1993 it was updated several times, and the specification defines also a formal semantics for the basic elements of the language based on process theory. Triggered message sequence charts (TMSC) [40] proposed extensions to MSC to express conditions and refinement in a precise way. Live Sequence Charts (LSCs) [10] concentrated on distinguishing possible and necessary behaviors. A special technique and a tool, the Play-Engine, were also developed for LSC to specify reactive systems [18].

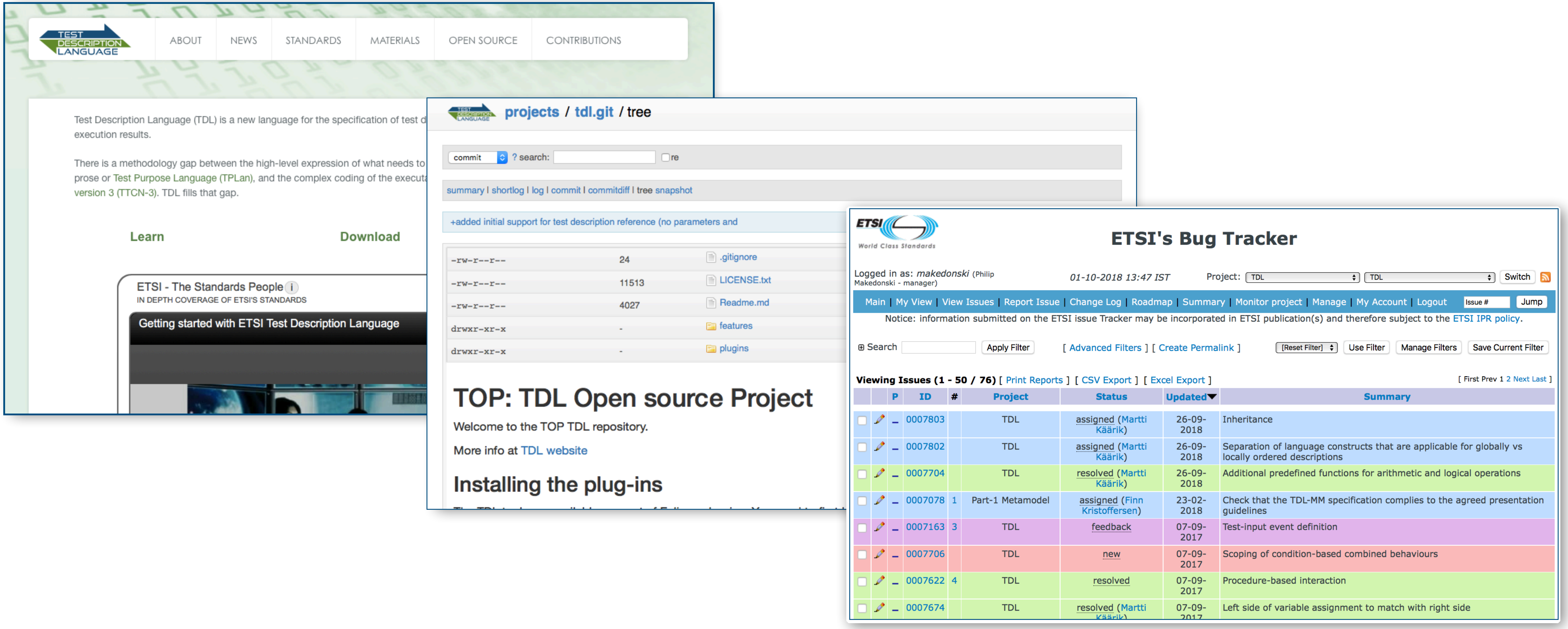
Scenario languages were introduced into the Object Management Group's (OMG) Unified Modeling Language (UML) [32] under the name of Sequence Diagrams. The 2.0 version of UML changed Sequence Diagrams significantly. Several elements were borrowed from MSC, many new complex elements were added to the language, and the semantics and the underlying metamodel were rewritten. Due to the increased expressiveness of the language, interpreting a complex diagram that uses the new constructs is a difficult task; thus, having a precise formal semantics becomes even more critical. But the many different purposes Sequence Diagrams are used for, e.g., showing the flows of method calls inside a program, or giving a partial specification of interactions in a distributed system, require quite different interpretations of the language. Indeed, many different semantics have been proposed for Sequence Diagrams. For a practitioner wanting to use Sequence

Springer

TDL so far…

- A standardised approach for the design of test descriptions
  - graphical, textual, and user-defined syntaxes, common exchange format
  - first extensions: test purposes with TDL-TO, extended test configurations
- Design-first approach
  - higher level test design before rushing towards detailed test code
  - facilitate better quality of tests and higher productivity in testing
- Harmonise and ease development of tools for scenario-based testing
  - editors mapped to TDL meta-model, e.g. graphical, textual
  - model-based re-usable back-end tools, e.g. code and documentation generators
  - Eclipse ecosystem enables quick and low-cost tool development

What would you like to see in TDL?


TEST DESCRIPTION LANGUAGE
ABOUT
NEWS
STANDARDS
MATERIALS
OPEN SOURCE
CONTRIBUTIONS

Test Description Language (TDL) is a new language for the specification of test d
execution results.

There is a methodology gap between the high-level expression of what needs to
prose or Test Purpose Language (TPLan), and the complex coding of the executa
version 3 (TTCN-3). TDL fills that gap.

Learn
Download
ETSI - The Standards People
IN DEPTH COVERAGE OF ETSI'S STANDARDS
Getting started with ETSI Test Description Language

projects / tdl.git / tree

commit
? search:
re
summary | shortlog | log | commit | commitdiff | tree snapshot

+added initial support for test description reference (no parameters and

```
-rw-r--r--    24       .gitignore
-rw-r--r--    11513    LICENSE.txt
-rw-r--r--    4027     Readme.md
drwxr-xr-x    -        features
drwxr-xr-x    -        plugins
```

TOP: TDL Open source Project

Welcome to the TOP TDL repository.

More info at TDL website

Installing the plug-ins

ETSI's Bug Tracker

Logged in as: *makedonski* (Philip Makedonski - manager)

01-10-2018 13:47 IST

Project: TDL TDL

Switch
Main | My View | View Issues | Report Issue | Change Log | Roadmap | Summary | Monitor project | Manage | My Account | Logout
Issue #
Jump

Notice: information submitted on the ETSI issue Tracker may be incorporated in ETSI publication(s) and therefore subject to the ETSI IPR policy.

Viewing Issues (1 - 50 / 76) [Print Reports] [CSV Export] [Excel Export]

[First Prev 1 2 Next Last]

		P	ID	#	Project	Status	Updated▼	Summary
□	✎	–	0007803		TDL	assigned (Martti Käärik)	26-09-2018	Inheritance
□	✎	–	0007802		TDL	assigned (Martti Käärik)	26-09-2018	Separation of language constructs that are applicable for globally vs locally ordered descriptions
□	✎	–	0007704		TDL	resolved (Martti Käärik)	26-09-2018	Additional predefined functions for arithmetic and logical operations
□	✎	–	0007078	1	Part-1 Metamodel	assigned (Finn Kristoffersen)	23-02-2018	Check that the TDL-MM specification complies to the agreed presentation guidelines
□	✎	–	0007163	3	TDL	feedback	07-09-2017	Test-input event definition
□	✎	–	0007706		TDL	new	07-09-2017	Scoping of condition-based combined behaviours
□	✎	–	0007622	4	TDL	resolved	07-09-2017	Procedure-based interaction
□	✎	–	0007674		TDL	resolved (Martti Käärik)	07-09-2017	Left side of variable assignment to match with right side

tdl.etsi.org