6th UCAAT
User Conference on Advanced Automated Testing
ETSI
Paris, 16-18 October 2018

Organizer: TESTING SOLUTIONS & SERVICES

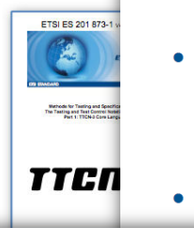
IMPLEMENTING THE STANDARDISED MAPPING OF TDL TO TTCN-3

Philip Makedonski (University of Göttingen)
Jens Grabowski (University of Göttingen)

Overview

Background

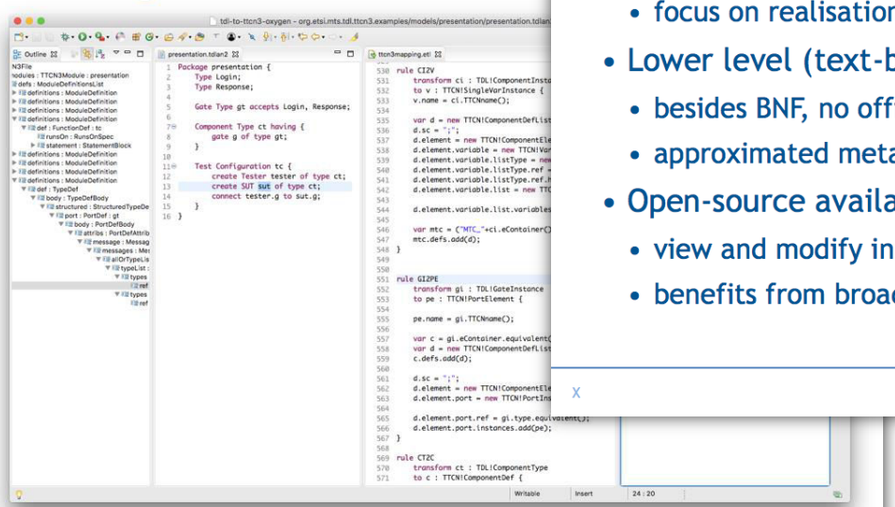
- Test Description Language
  - Design, documentation, representation of formalised test descriptions
  - Scenario-based approach
- Testing and Test Control Notation
  - Specification and implementation of all kinds of black-box tests
  - Component-based approach



Why?

- Highly requested
  - brought up every time the mapping is discusse
- More comprehensive standard validation
  - E
  - V
- Co

Tooling



Xtext epsilon Xtext 6th UCAAT

Experiences

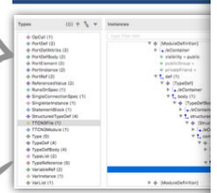
- Standard specification makes things easier!
  - many hard decisions have already been made
  - focus on realisation instead (not always straightforward)
- Lower level (text-based) specification challenging
  - besides BNF, no official meta-model for TTCN-3 available
  - approximated meta-model enables use of available tools
- Open-source availability big boost
  - view and modify internals when necessary
  - benefits from broader upstream ecosystem

6th UCAAT

Implementation: Text-based vs Model-base

```
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;
}
```

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



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

type port gt_to_connect message {
  //port type for Tester-Tester conn
  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);
}
```

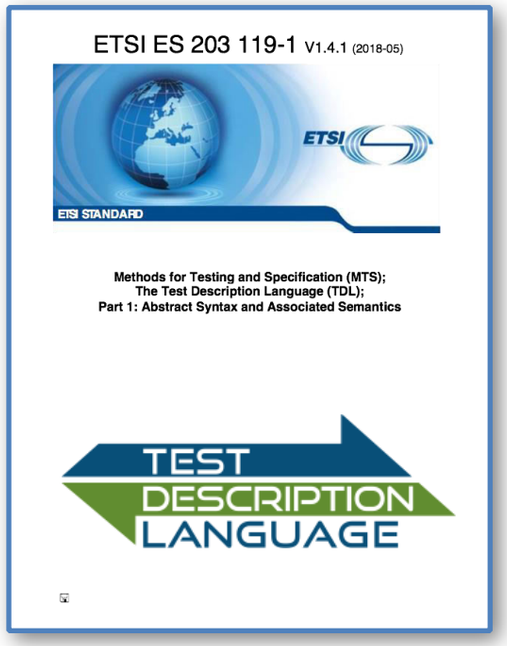
- High-level, non-linear, traceable

6th UCAAT

6th UCAAT

Background

- 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

Background

- 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)

MBT Workflow

Manual Workflow

System Requirements Specification

Requirements Level

Model

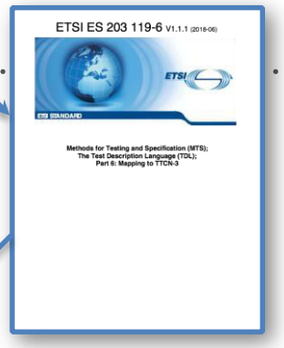
TPLan

TDL-TO

Test Generator

Test Design Level

TDL



ES 203 119-6

Test Code Generator

TTCN-3

Test Implementation Level

Adaptation

Executable Tests

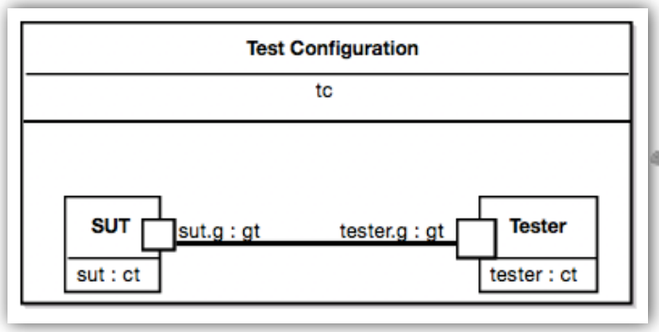
Test Execution Level

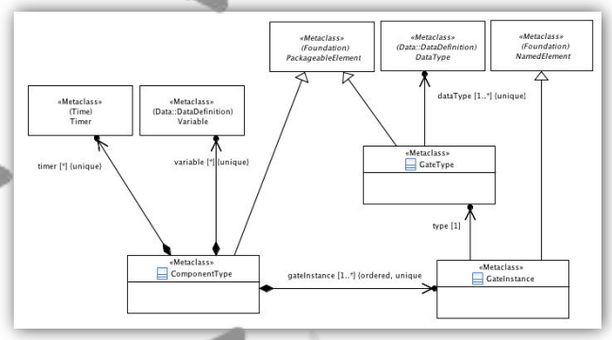
Why?

- Highly requested
  - brought up every time the mapping is discussed
- More comprehensive standard validation
  - built on top of initial proof-of-concept prototype
  - wider application of the mapping, address corner cases
- Collect and report on experiences
  - implementation and validation of the standard
  - application of model-based technologies
  - evolving the standard

Implementation: Text-based

```
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;
}
```





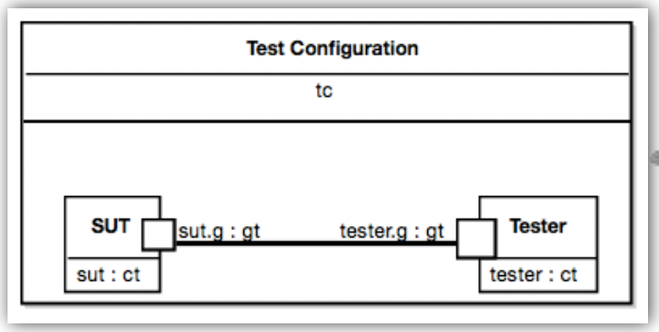

```
<packagedElement xsi:type="tdl:ComponentType"
                 xmi:id="_qKt233asEeWrfP0MdfQNpg"
                 name="ct">
  <gateInstance xmi:id="_qKt24HasEeWrfP0MdfQNpg"
                name="g"
                type="_qKt23nasEeWrfP0MdfQNpg"/>
</packagedElement>
```

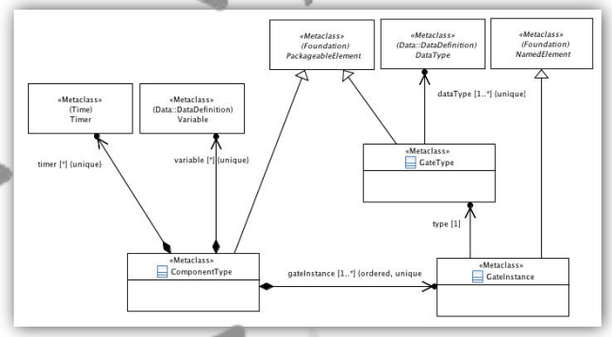
```
type port gt_to_map message {
  //port type for SUT-Tester connections
  inout Login, Response
}

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);
}
```

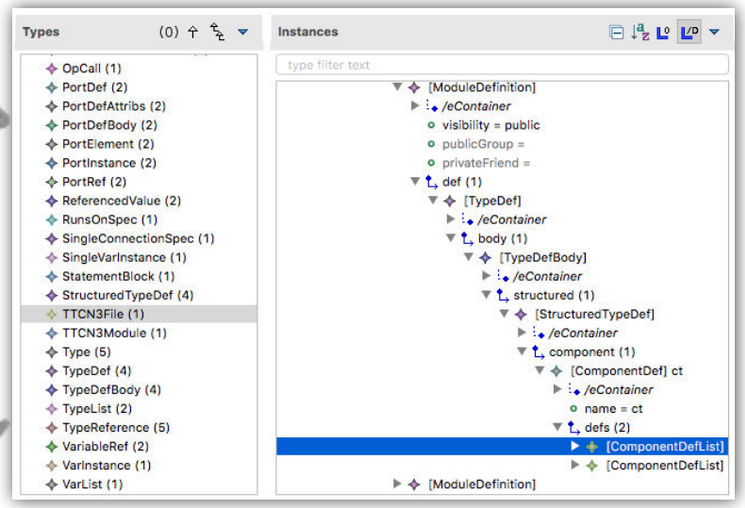
- Linear, complex, limited, messy

Implementation: Text-based vs Model-based

```
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;
}
```








```
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);
}
```

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

- High-level, non-linear, traceable

Model-based Mapping

- Work with higher level structural representation
  - target structure rather complex
  - syntactical details derived automatically
  - non-linear approach for stepwise enrichment
    - traceability and references to equivalent constructs
    - structural validation already during transformation
- But:
  - standard described with text-based mapping
  - no official meta-model for TTCN-3

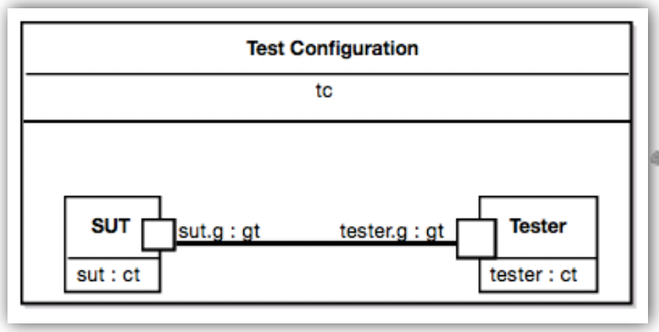
Tooling

- Eclipse + EMF - modelling platform
- Xtext - textual mapping for models
- Epsilon / ETL - model-to-model transformation
- MoDisco - tree-based model editing (optional)
- Sirius - graphical model editing (optional)

- TOP - EMF-based TDL tools
- TRex v2 / t3tools v2 - EMF-based TTCN-3 tools
- Custom automation tools

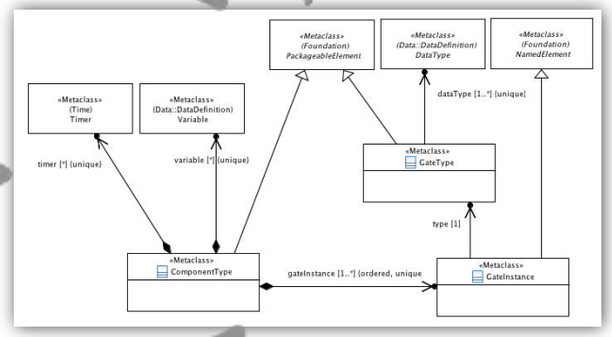
Tooling

```
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

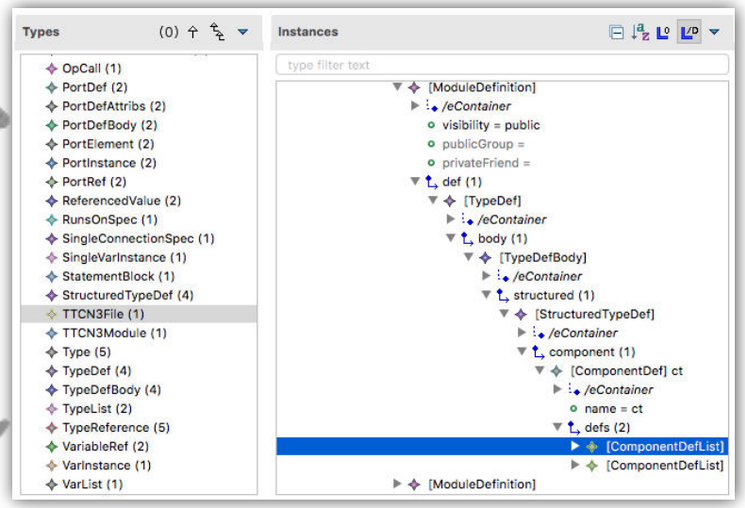


emf
ECLIPSE MODELING FRAMEWORK

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

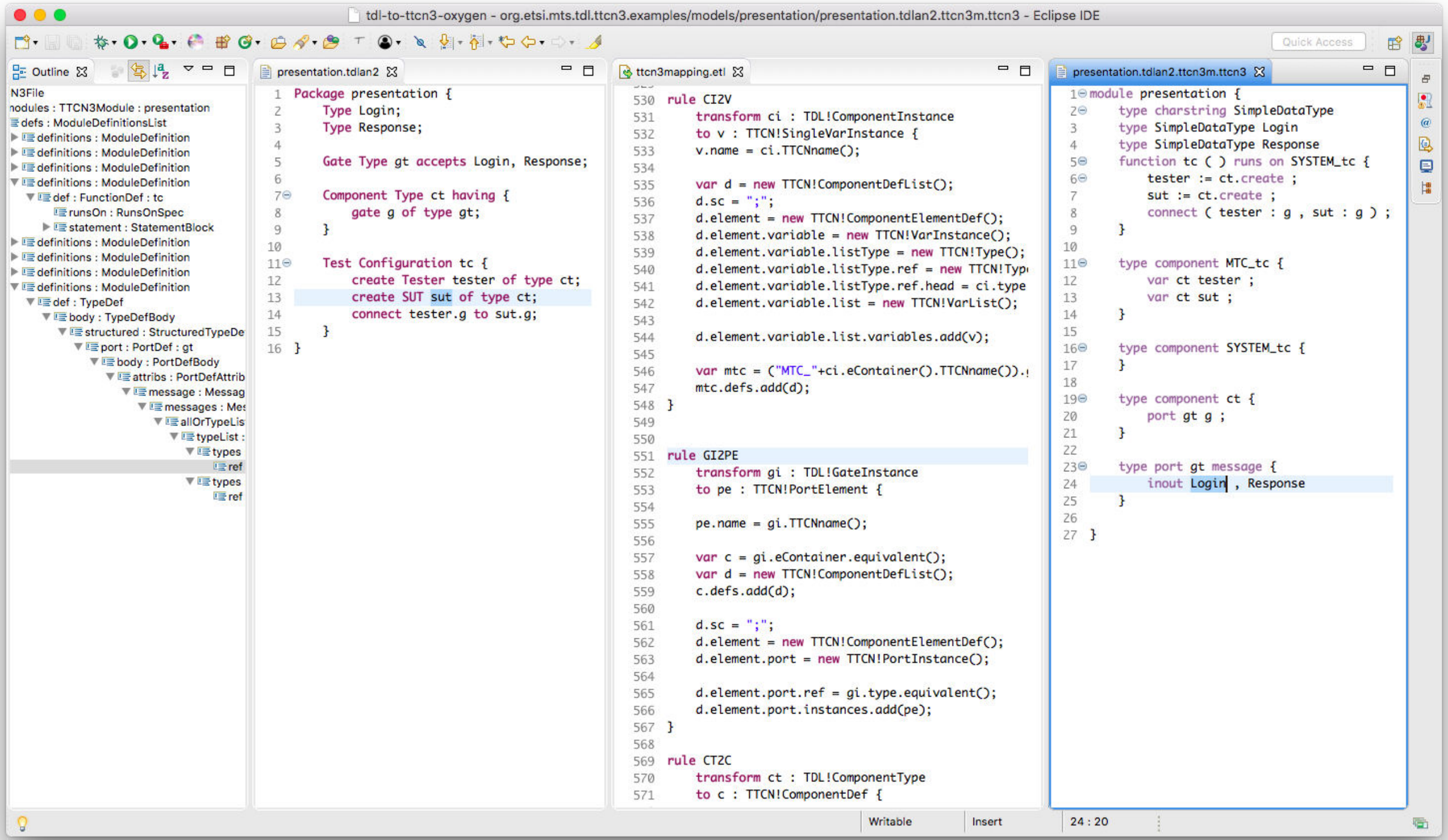
epsilon

Xtext



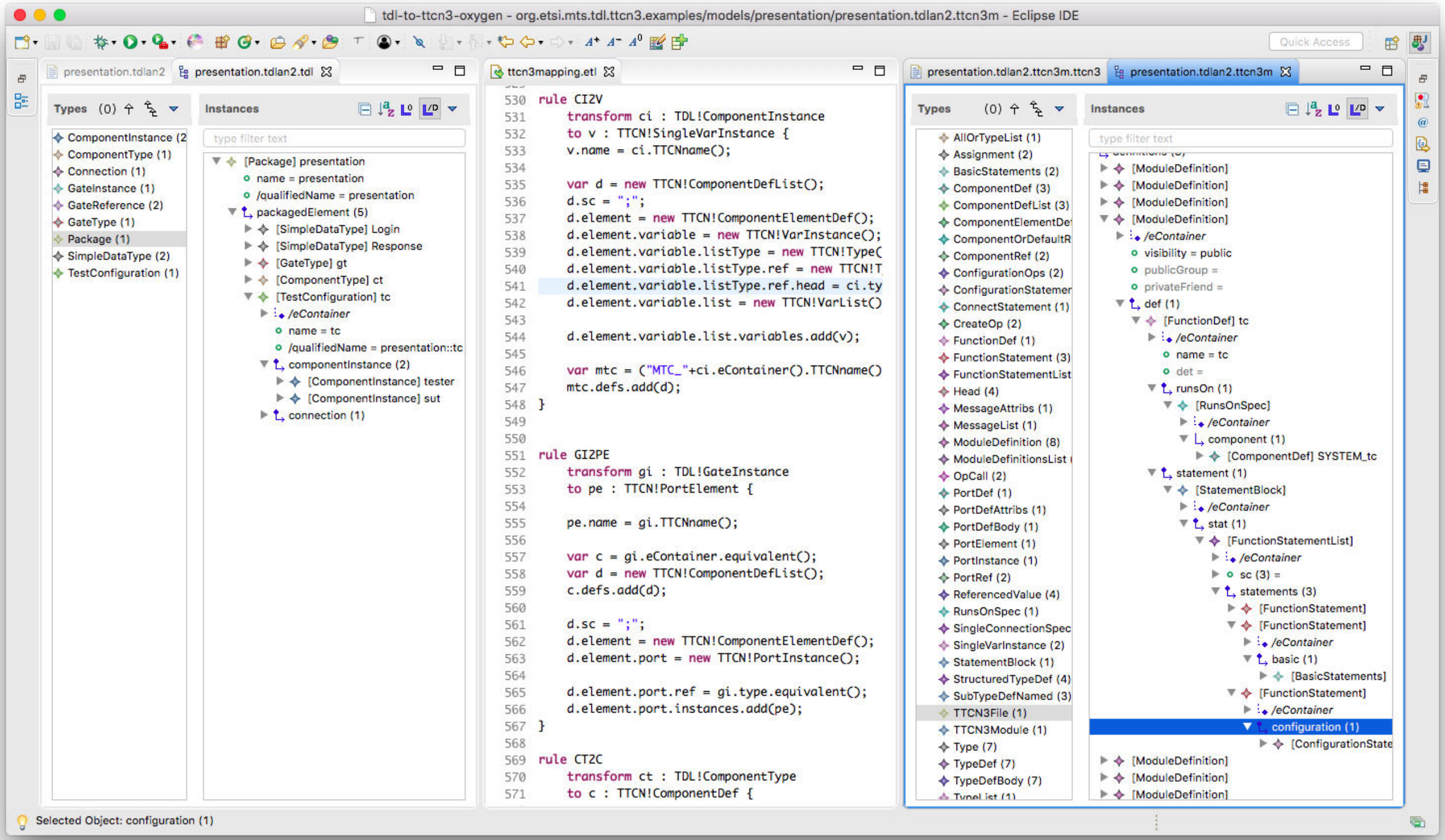

```
type port gt_to_map message {
  //port type for SUT-Tester connections
  inout Login, Response
}

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);
}
```

Tooling

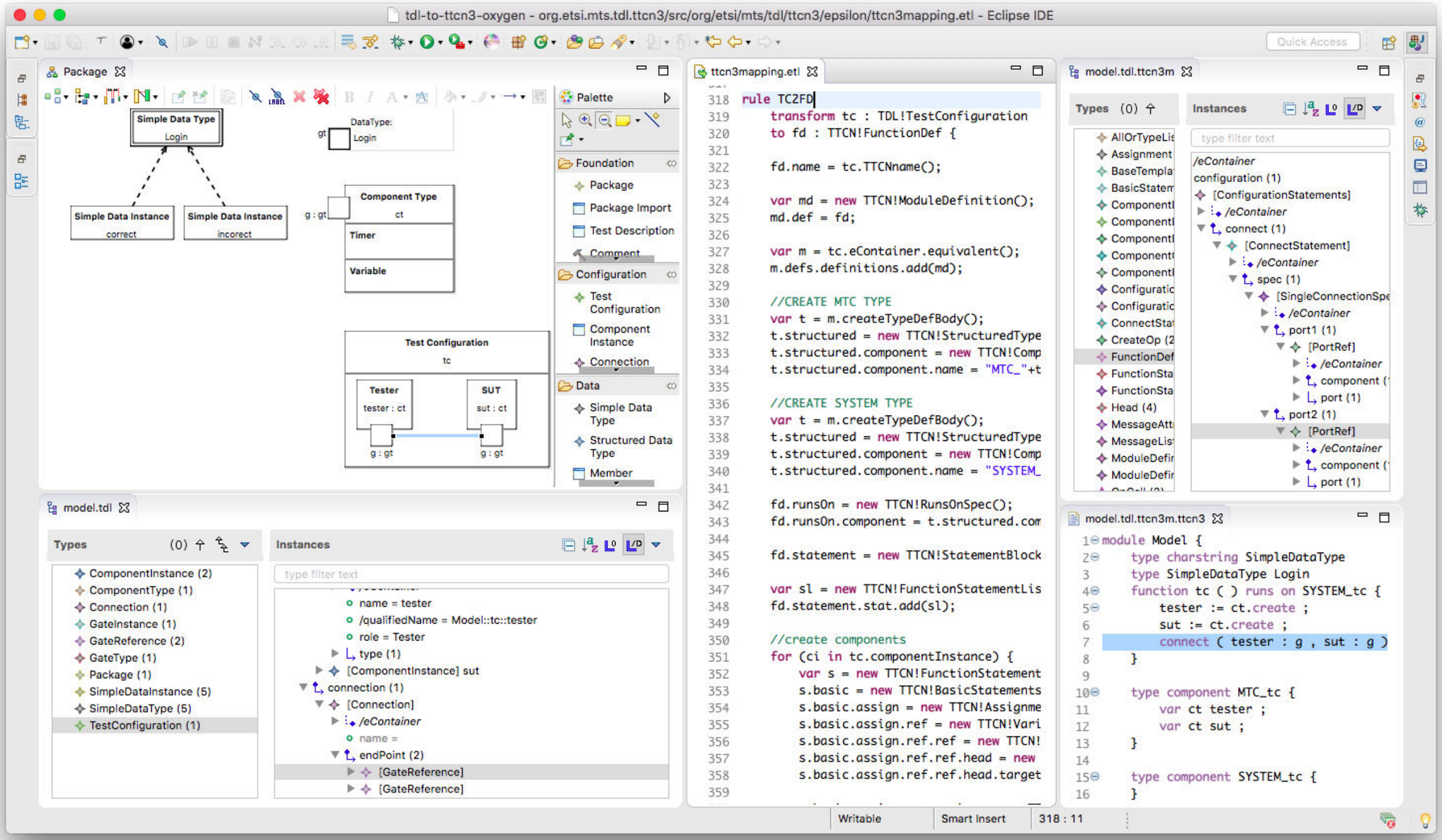


Xtext epsilon Xtext 6th UCAAT

Tooling


tdl-to-ttcn3-oxygen - org.etsi.mts.tdl.ttcn3.examples/models/presentation/presentation.tdlan2.ttcn3m - Eclipse IDE
Quick Access
presentation.tdlan2
presentation.tdlan2.tdl
Types (0)
Instances
type filter text
ComponentInstance (2
ComponentType (1)
Connection (1)
GateInstance (1)
GateReference (2)
GateType (1)
Package (1)
SimpleDataType (2)
TestConfiguration (1)
[Package] presentation
name = presentation
/qualifiedName = presentation
packagedElement (5)
[SimpleDataType] Login
[SimpleDataType] Response
[GateType] gt
[ComponentType] ct
[TestConfiguration] tc
/eContainer
name = tc
/qualifiedName = presentation::tc
componentInstance (2)
[ComponentInstance] tester
[ComponentInstance] sut
connection (1)
ttcn3mapping.etl
530 rule CI2V
531 transform ci : TDL!ComponentInstance
532 to v : TTCN!SingleVarInstance {
533 v.name = ci.TTCNname();
534
535 var d = new TTCN!ComponentDefList();
536 d.sc = ";";
537 d.element = new TTCN!ComponentElementDef();
538 d.element.variable = new TTCN!VarInstance();
539 d.element.variable.listType = new TTCN!Type(
540 d.element.variable.listType.ref = new TTCN!T
541 d.element.variable.listType.ref.head = ci.ty
542 d.element.variable.list = new TTCN!VarList()
543
544 d.element.variable.list.variables.add(v);
545
546 var mtc = ("MTC_"+ci.eContainer().TTCNname()
547 mtc.defs.add(d);
548 }
549
550
551 rule GI2PE
552 transform gi : TDL!GateInstance
553 to pe : TTCN!PortElement {
554
555 pe.name = gi.TTCNname();
556
557 var c = gi.eContainer.equivalent();
558 var d = new TTCN!ComponentDefList();
559 c.defs.add(d);
560
561 d.sc = ";";
562 d.element = new TTCN!ComponentElementDef();
563 d.element.port = new TTCN!PortInstance();
564
565 d.element.port.ref = gi.type.equivalent();
566 d.element.port.instances.add(pe);
567 }
568
569 rule CT2C
570 transform ct : TDL!ComponentType
571 to c : TTCN!ComponentDef {
presentation.tdlan2.ttcn3m.ttcn3
presentation.tdlan2.ttcn3m
Types (0)
Instances
type filter text
AllOrTypeList (1)
Assignment (2)
BasicStatements (2)
ComponentDef (3)
ComponentDefList (3)
ComponentElementDe
ComponentOrDefaultR
ComponentRef (2)
ConfigurationOps (2)
ConfigurationStatemer
ConnectStatement (1)
CreateOp (2)
FunctionDef (1)
FunctionStatement (3)
FunctionStatementList
Head (4)
MessageAttribs (1)
MessageList (1)
ModuleDefinition (8)
ModuleDefinitionsList
OpCall (2)
PortDef (1)
PortDefAttribs (1)
PortDefBody (1)
PortElement (1)
PortInstance (1)
PortRef (2)
ReferencedValue (4)
RunsOnSpec (1)
SingleConnectionSpec
SingleVarInstance (2)
StatementBlock (1)
StructuredTypeDef (4)
SubTypeDefNamed (3)
TTCN3File (1)
TTCN3Module (1)
Type (7)
TypeDef (7)
TypeDefBody (7)
[ModuleDefinition]
/eContainer
visibility = public
publicGroup =
privateFriend =
def (1)
[FunctionDef] tc
name = tc
det =
runsOn (1)
[RunsOnSpec]
component (1)
[ComponentDef] SYSTEM_tc
statement (1)
[StatementBlock]
stat (1)
[FunctionStatementList]
sc (3) =
statements (3)
[FunctionStatement]
basic (1)
[BasicStatements]
configuration (1)
Selected Object: configuration (1)

Tooling


tdl-to-ttcn3-oxygen - org.etsi.mts.tdl.ttcn3/src/org/etsi/mts/tdl/ttcn3/epsilon/ttcn3mapping.etl - Eclipse IDE
Quick Access
Package
Simple Data Type
Login
DataType:
gt
Login
Simple Data Instance
correct
Simple Data Instance
incorect
Component Type
ct
g : gt
Timer
Variable
Test Configuration
tc
Tester
tester : ct
g : gt
SUT
sut : ct
g : gt
Palette
Foundation
Package
Package Import
Test Description
Comment
Configuration
Test Configuration
Component Instance
Connection
Data
Simple Data Type
Structured Data Type
Member
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
model.tdl.ttcn3m
Types (0)
Instances
type filter text
/eContainer
configuration (1)
[ConfigurationStatements]
/eContainer
connect (1)
[ConnectStatement]
/eContainer
spec (1)
port1 (1)
[PortRef]
/eContainer
port (1)
port2 (1)
[PortRef]
/eContainer
port (1)
model.tdl
Types (0)
Instances
type filter text
ComponentInstance (2)
ComponentType (1)
Connection (1)
GateInstance (1)
GateReference (2)
GateType (1)
Package (1)
SimpleDataInstance (5)
SimpleDataType (5)
TestConfiguration (1)
name = tester
/qualifiedName = Model::tc::tester
role = Tester
type (1)
[ComponentInstance] sut
connection (1)
[Connection]
/eContainer
name =
endPoint (2)
[GateReference]
[GateReference]
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 {
}
Writable
Smart Insert
318 : 11

Experiences

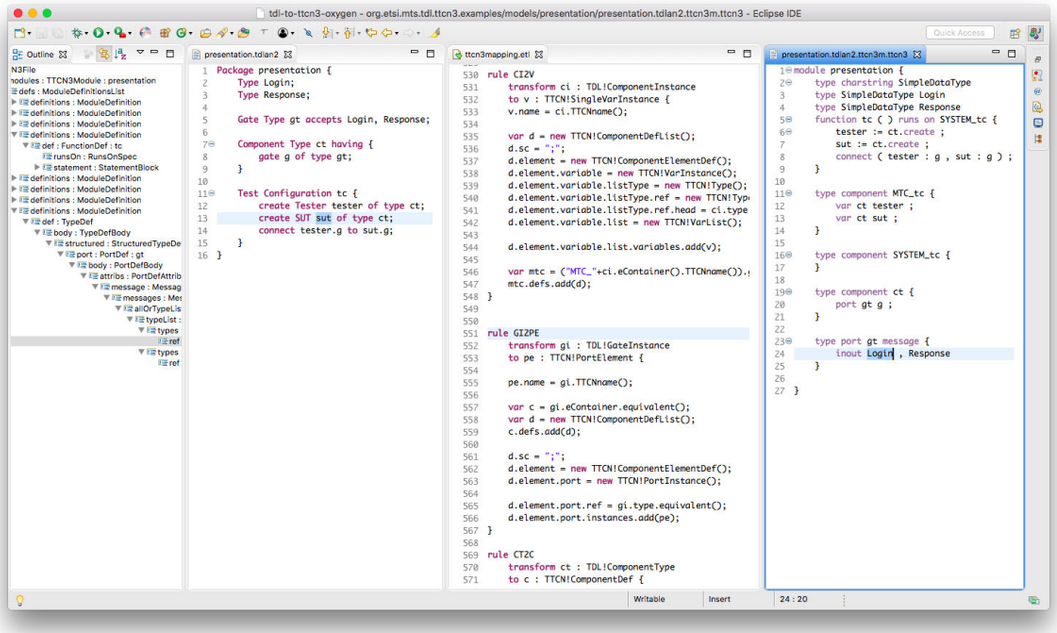
- Standard specification makes things easier!
  - many hard decisions have already been made
  - focus on realisation instead (not always straightforward)
- Lower level (text-based) specification challenging
  - besides BNF, no official meta-model for TTCN-3 available
  - approximated meta-model enables use of available tools
- Open-source availability big boost
  - view and modify internals when necessary
  - benefits from broader upstream ecosystem

Experiences

- Model-based approach
  - focus on essential parts - transformation logic
  - building blocks provided by the platform
  - convenient stepwise enrichment
- Custom tooling to streamline repetitive tasks
  - translation between different formats
    - expected TTCN-3 to model (for analysis)
    - TDLan to model
    - model to model transformations
    - model to TTCN-3

Conclusion

- Current status
  - ~80% of specification covered
  - still a prototype
  - open for contributions



- Future work
  - further refinement towards 100% specification coverage
  - comprehensive testing and evaluation
  - TTCN-3 to TDL?

tdl.etsi.org

6th UCAAT