5th UCAAT User Conference on Advanced Automated Testing

ETSI World Class Standards

Berlin, 11-13 October 2017

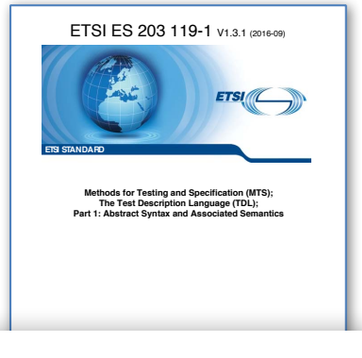
FROM TDL TO TTCN-3: A STEP BY STEP TUTORIAL

Philip Makedonski, Gusztav Adamis, Martti Käärik, Finn Kristoffersen, György Réthy

Overview

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)

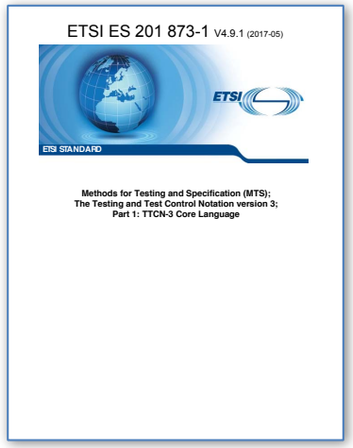


What is TTCN-3?

- Testing and Test Control Notation
  - Specification and implementation of all kinds of black-box tests
  - Platform independent link between modelling and execution
  - Component-based approach

MTS

ork



5th UCAAT

Mapping TDL to TTCN-3

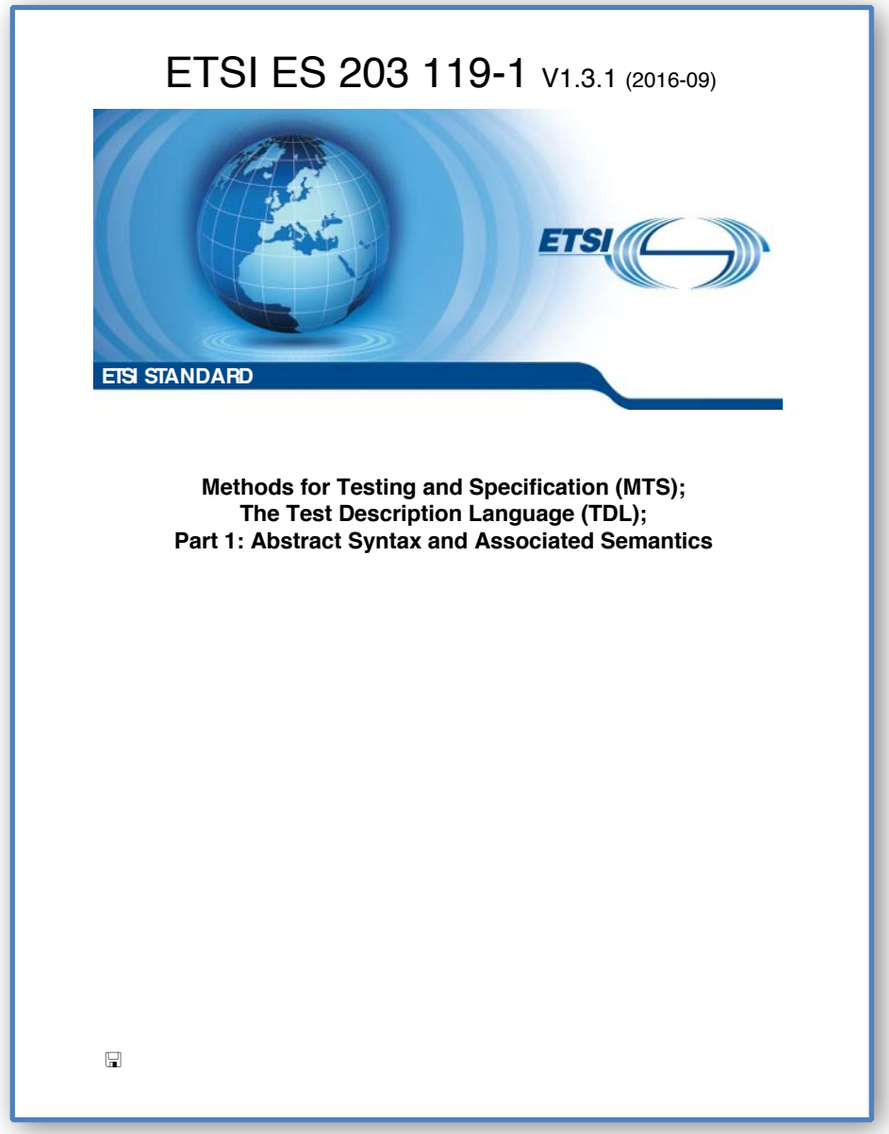
- Establish a connection between TDL and TTCN-3
  - generation of executable tests from test descriptions
  - standardised, ensuring compatibility and consistency
  - re-use existing tools and frameworks for test execution
  - re-use existing TTCN-3 assets (data, behaviour)

5th UCAAT

5th UCAAT

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)

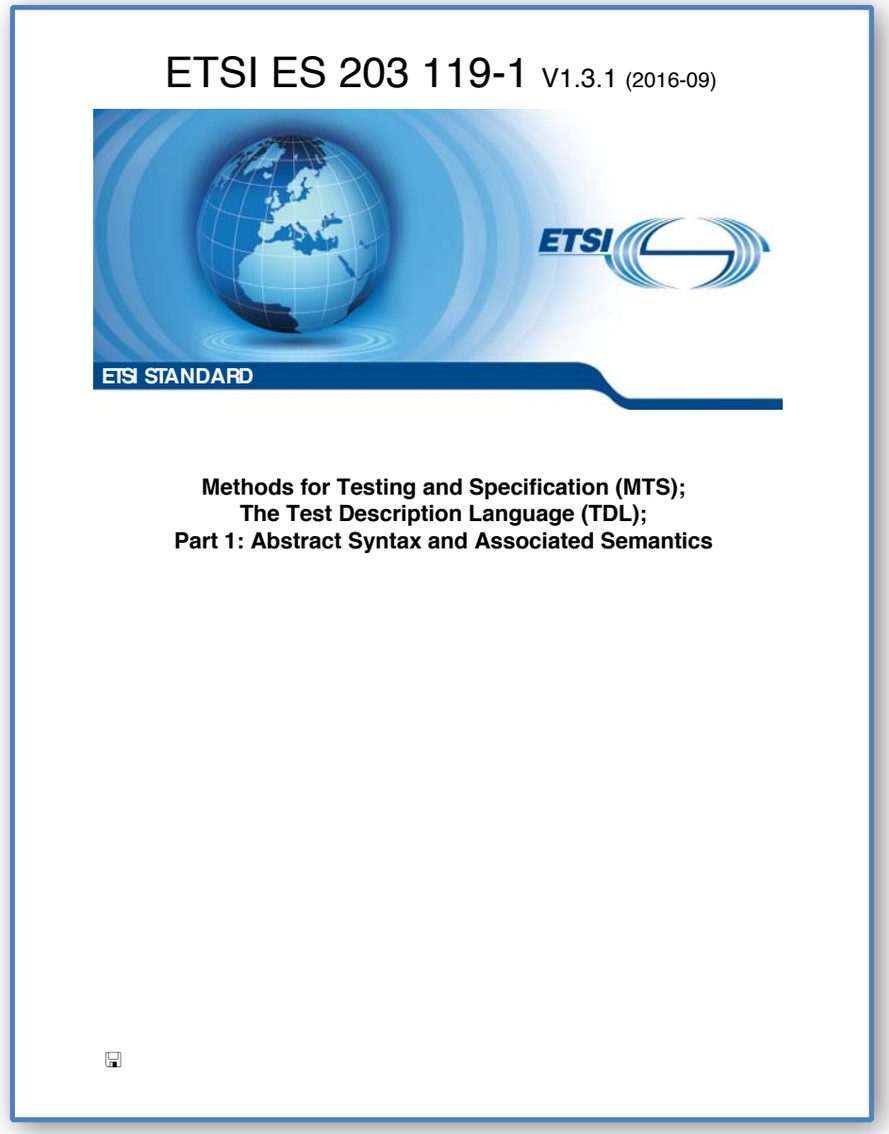

ETSI ES 203 119-1 V1.3.1 (2016-09)

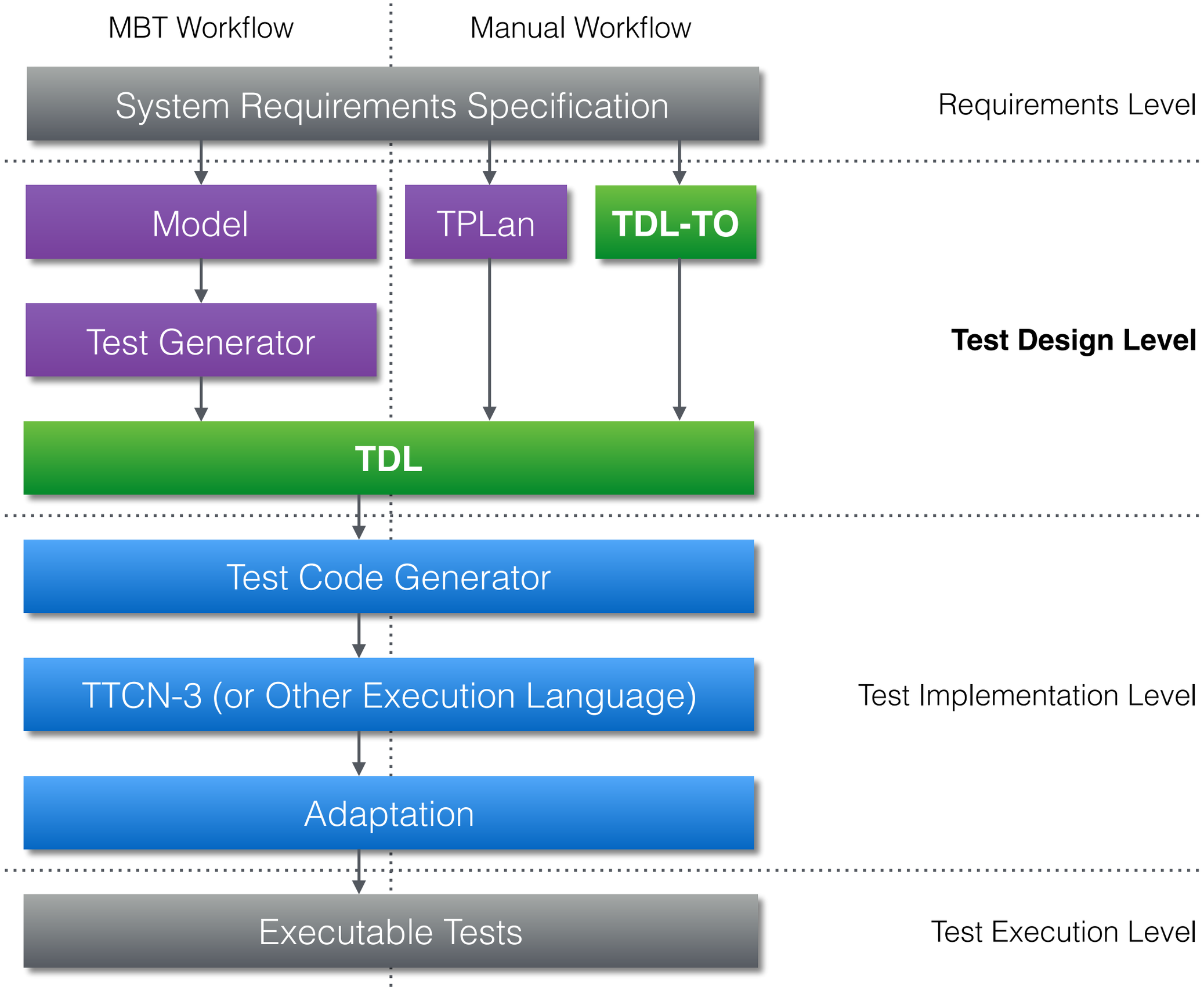
ETSI STANDARD

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

What is TDL?

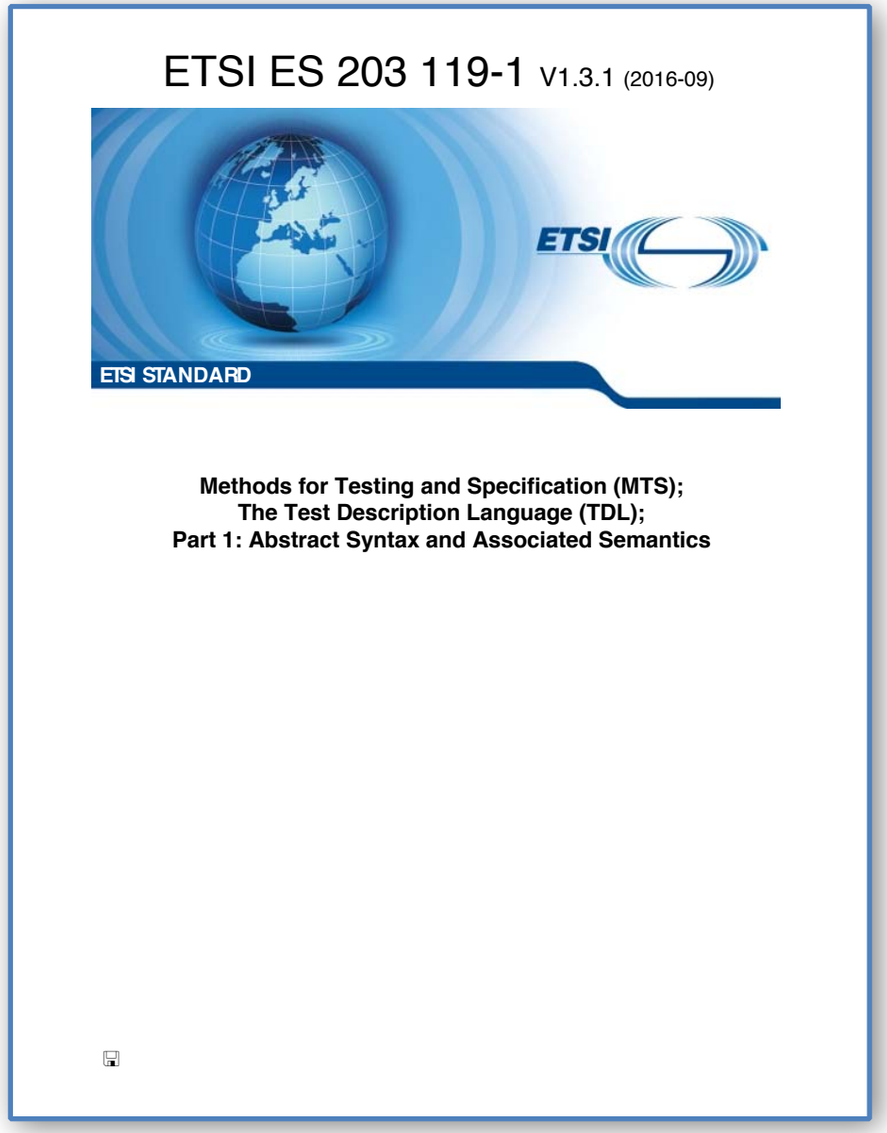
- Design, documentation, representation?
  - ease development and review
  - improve productivity and quality
  - both industry and standardisation
  - reduce implementation details




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?

- Scenario-based?
  - describe interactions with a system
  - attach test objectives to scenarios
  - derive and automate tests
- Reactive, distributed, real-time
  - common black-box testing concepts
  - domain adaptation
  - agile development

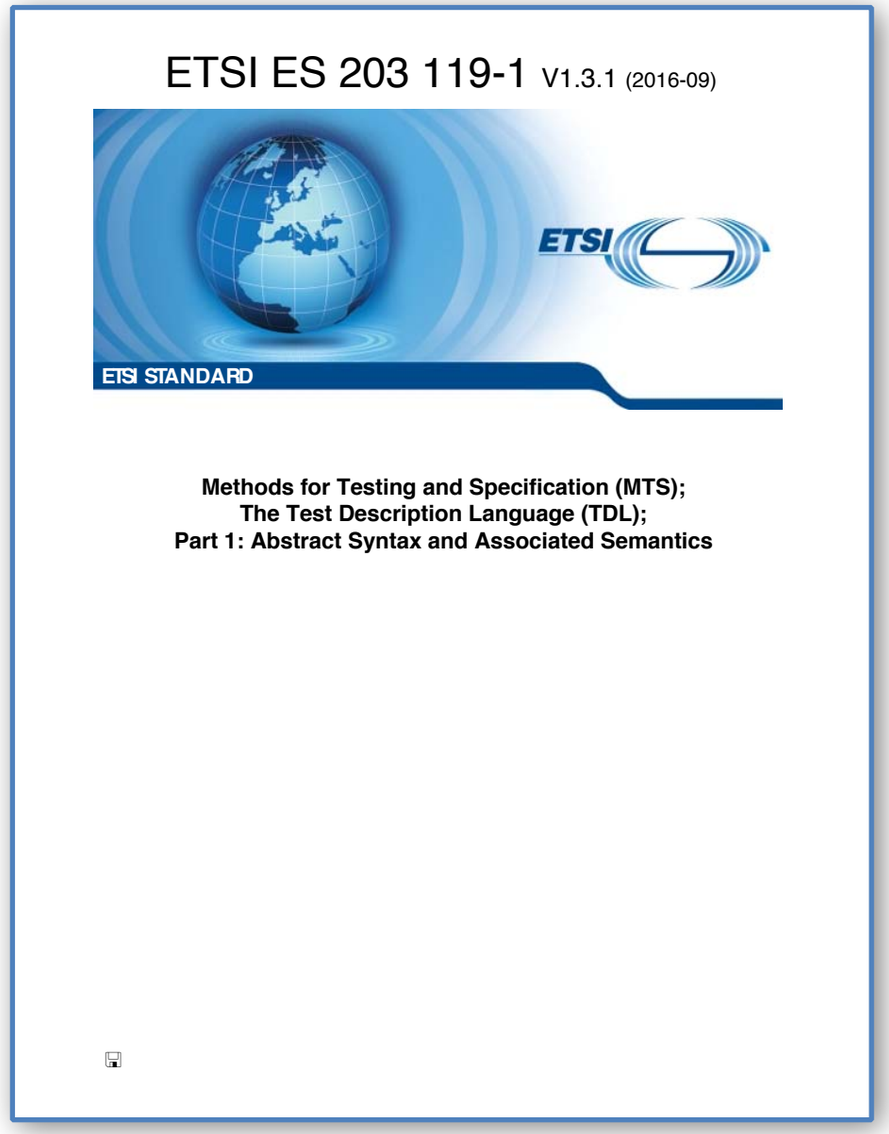

ETSI ES 203 119-1 V1.3.1 (2016-09)

ETSI STANDARD

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

What is TDL?

- Standardised?
  - canonical reference
  - stable documentation
  - clear semantics
  - interoperability and independence
  - updated with user needs
  - maintenance commitment

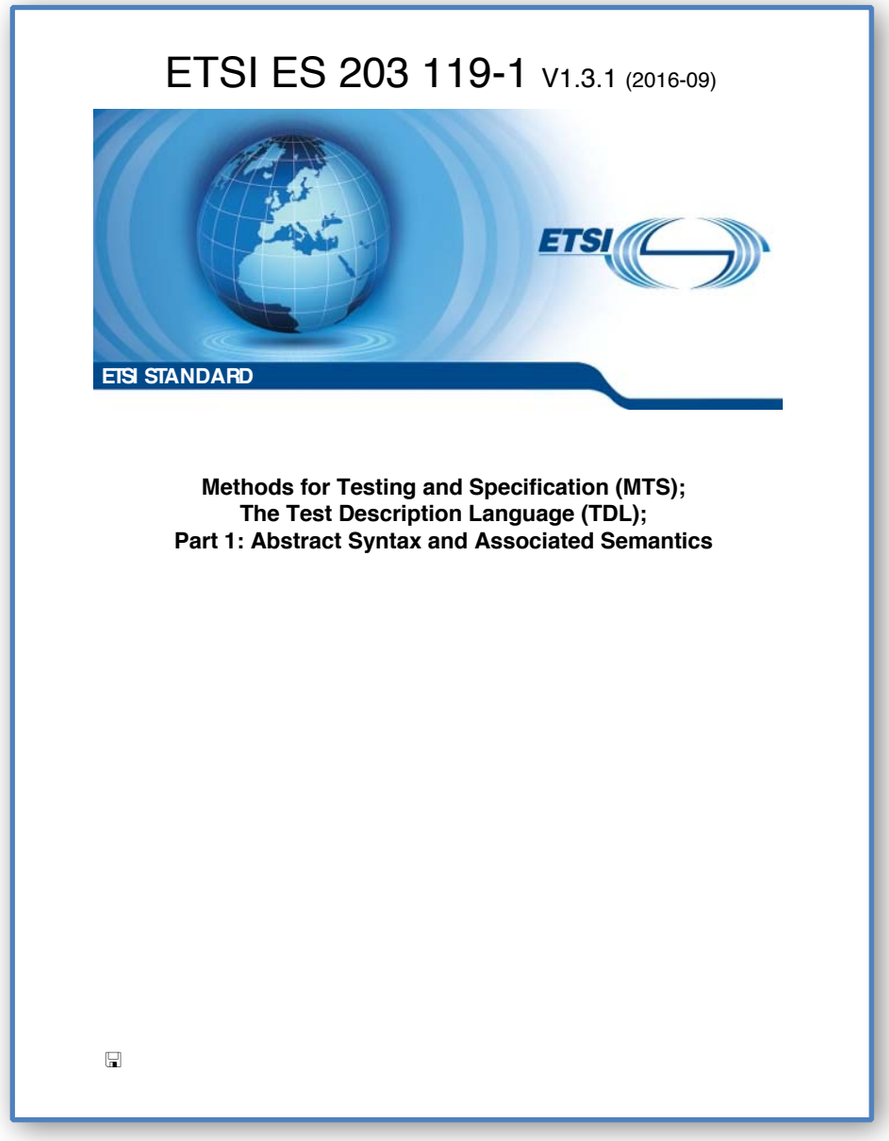

ETSI ES 203 119-1 V1.3.1 (2016-09)

ETSI STANDARD

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

What is TDL?

- Contributions from:
  - Siemens AG, Ericsson Hungary
  - Fraunhofer FOKUS, ETSI CTI
  - CEA, University of Göttingen
  - OU Elvior, Cinderella ApS
- Guidance:
  - Steering Group, TC MTS


ETSI ES 203 119-1 V1.3.1 (2016-09)

ETSI STANDARD

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

What is TDL?

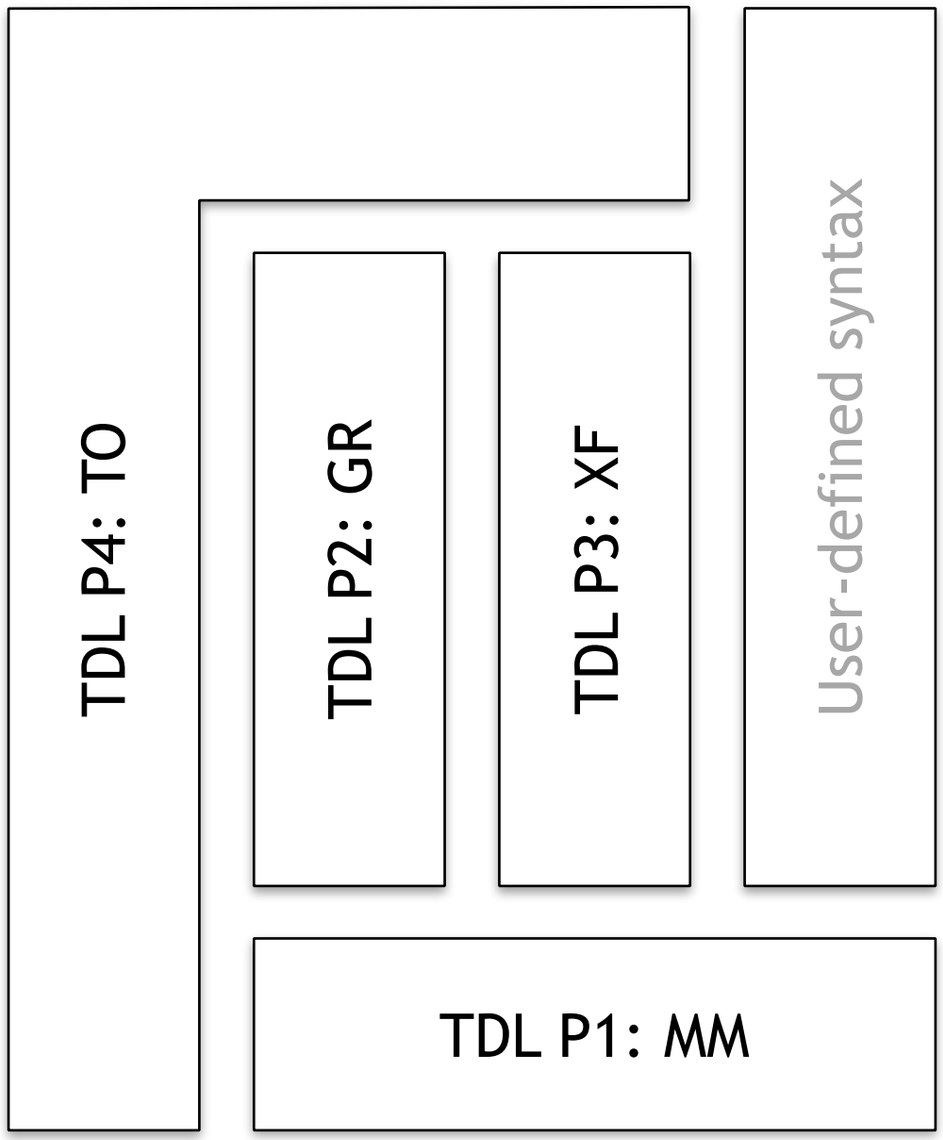
Part 1: MM Meta-Model and Semantics

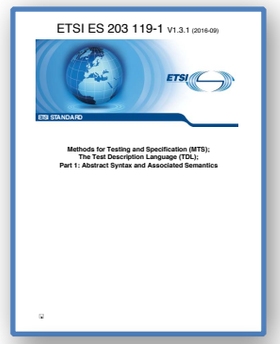
Part 2: GR Graphical Syntax

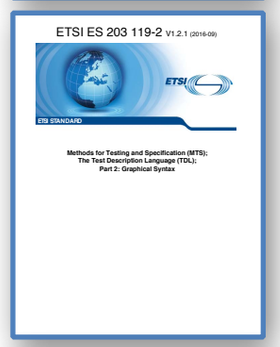
Part 3: XF Exchange Format

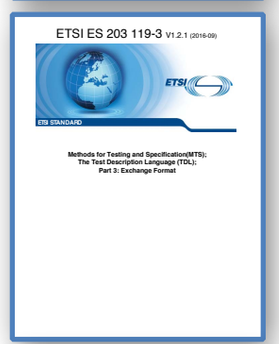
Part 4: TO Structured Test Objective Specification

What is TDL?

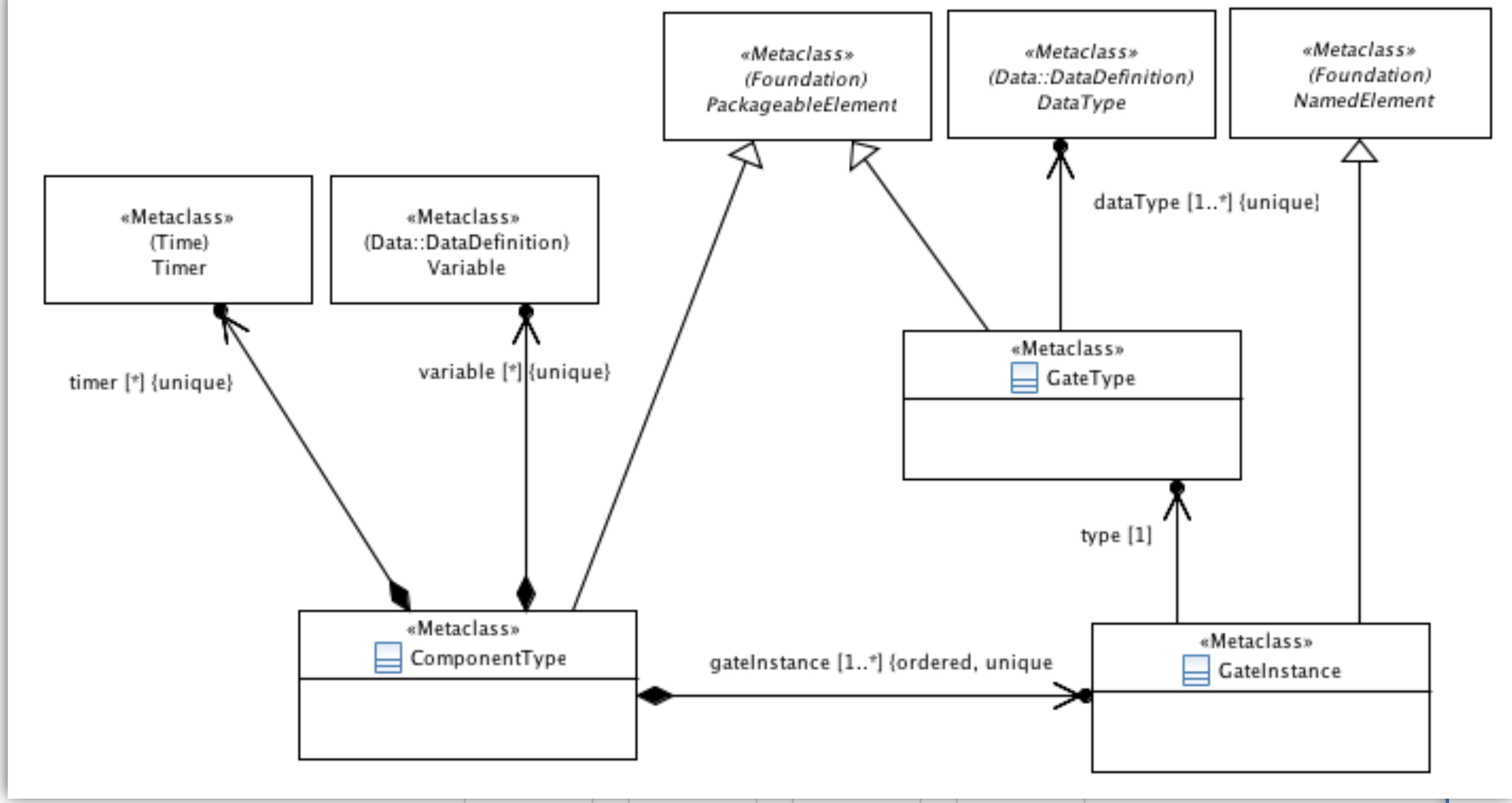

TDL P4: TO
TDL P2: GR
TDL P3: XF
User-defined syntax
TDL P1: MM


ETSI ES 203 119-1 V1.3.1 (2016-09)
ETSI STANDARD
Methods for Testing and Specification (MTS);
The Test Description Language (TDL);
Part 1: Abstract Syntax and Associated Semantics


ETSI ES 203 119-2 V1.2.1 (2016-09)
ETSI STANDARD
Methods for Testing and Specification (MTS);
The Test Description Language (TDL);
Part 2: Graphical Syntax

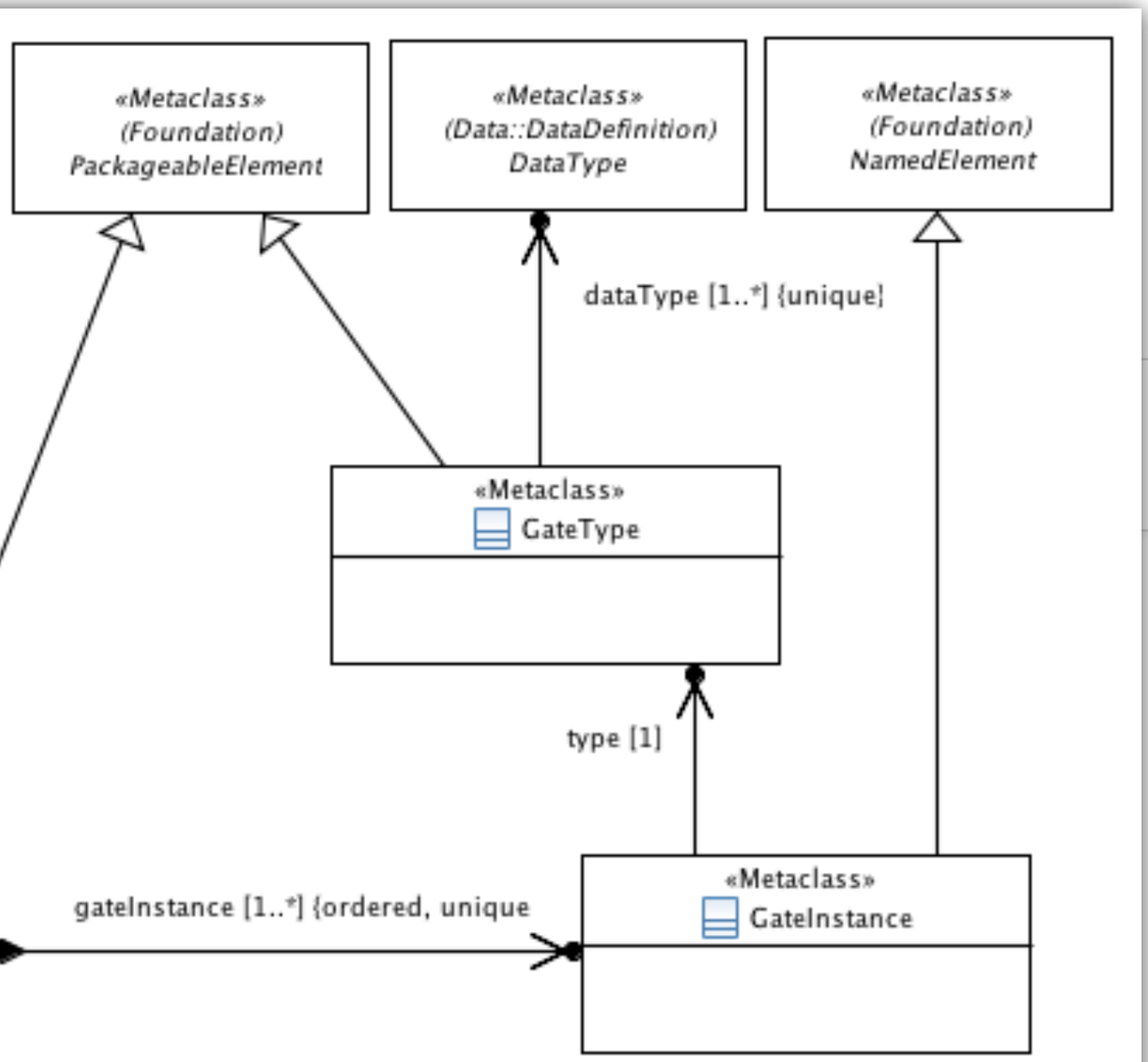

ETSI ES 203 119-3 V1.2.1 (2016-09)
ETSI STANDARD
Methods for Testing and Specification(MTS);
The Test Description Language (TDL);
Part 3: Exchange Format


ETSI ES 203 119-4 V1.2.1 (2016-09)
ETSI STANDARD
Methods for Testing and Specification (MTS);
The Test Description Language (TDL);
Part 4: Structured Test Objective Specification (Extension)


«Metaclass»
(Foundation)
PackageableElement
«Metaclass»
(Data::DataDefinition)
DataType
«Metaclass»
(Foundation)
NamedElement
«Metaclass»
(Time)
Timer
«Metaclass»
(Data::DataDefinition)
Variable
dataType [1..*] {unique}
timer [*] {unique}
variable [*] {unique}
«Metaclass»
GateType
type [1]
«Metaclass»
ComponentType
gateInstance [1..*] {ordered, unique
«Metaclass»
GateInstance

TDL P1: MM


ETSI ES 203 119-1 V1.3.1 (2016-09)
ETSI ES 203 119-2 V1.2.1 (2016-09)
ETSI ES 203 119-3 V1.2.1 (2016-09)
ETSI ES 203 119-4 V1.2.1 (2016-09)
ETSI STANDARD
Methods for Testing and Specification (MTS);
The Test Description Language (TDL);
Part 4: Structured Test Objective Specification (Extension)



Semantics

A 'GateType' represents a type of communication points, called 'ComponentInstance's. A 'GateType' specifies the 'DataType's t both directions.

Generalization

- PackageableElement

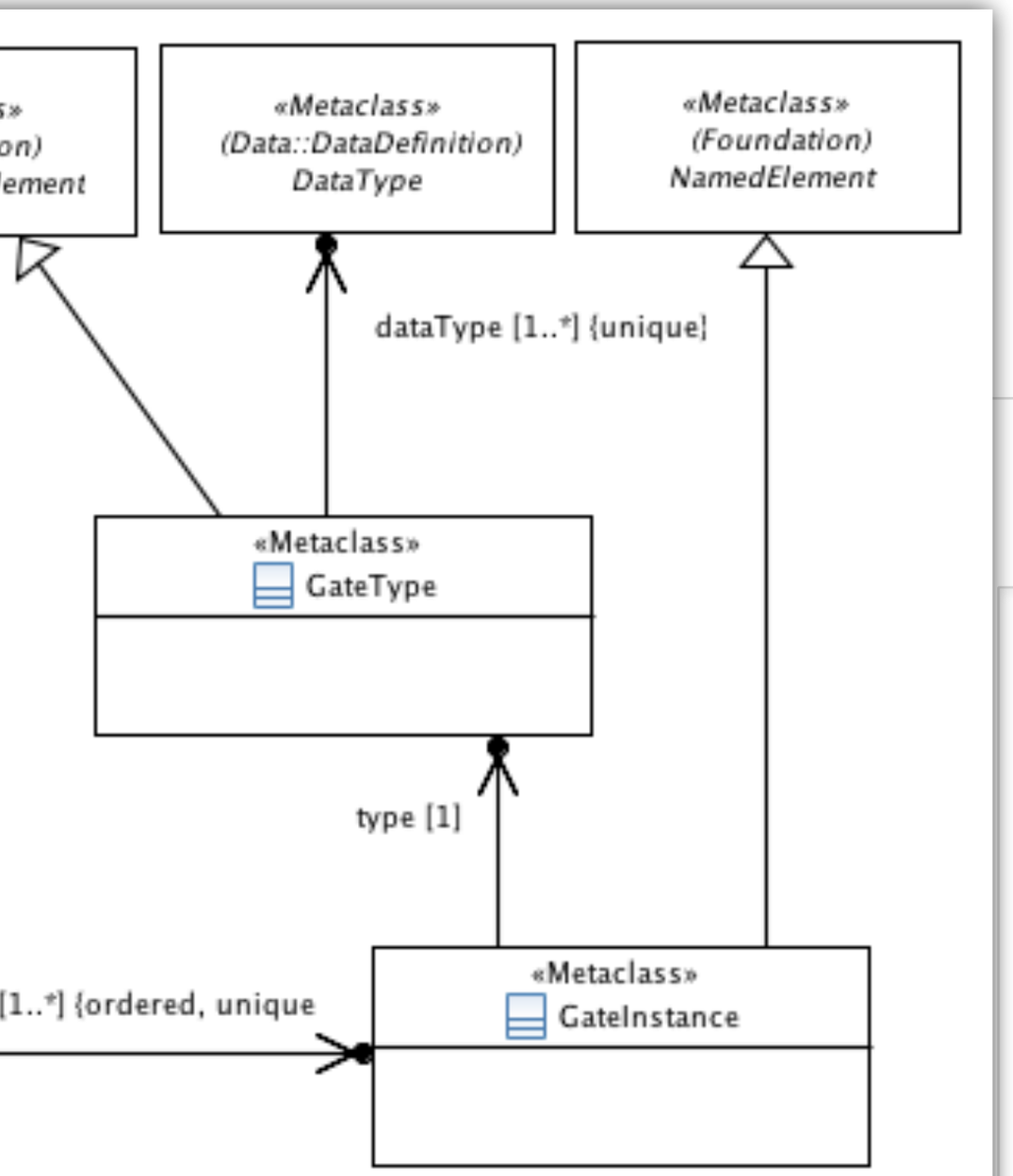
Properties

- dataType: DataType [1..*] {unique}
  The 'DataType's that can be exchanged via 'GateInstan shall adhere to the 'DataType's that are allowed to be e

Constraints

There are no constraints specified.

TDL P1: MM



6.4.2 GateType

Concrete Graphical Notation

GATETYPENAMELABEL
Data Type: DATATYPELISTLABEL

Formal Description

context GateType
GATETYPENAMELABEL **::=** self.name
DATATYPELISTLABEL **::=** self.dataType.name->**separator(',')**

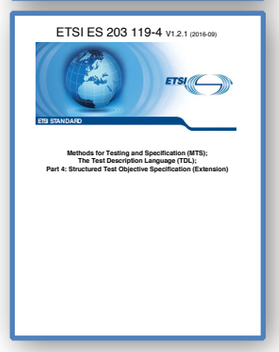
Comments

No comments.

TDL P2: GR

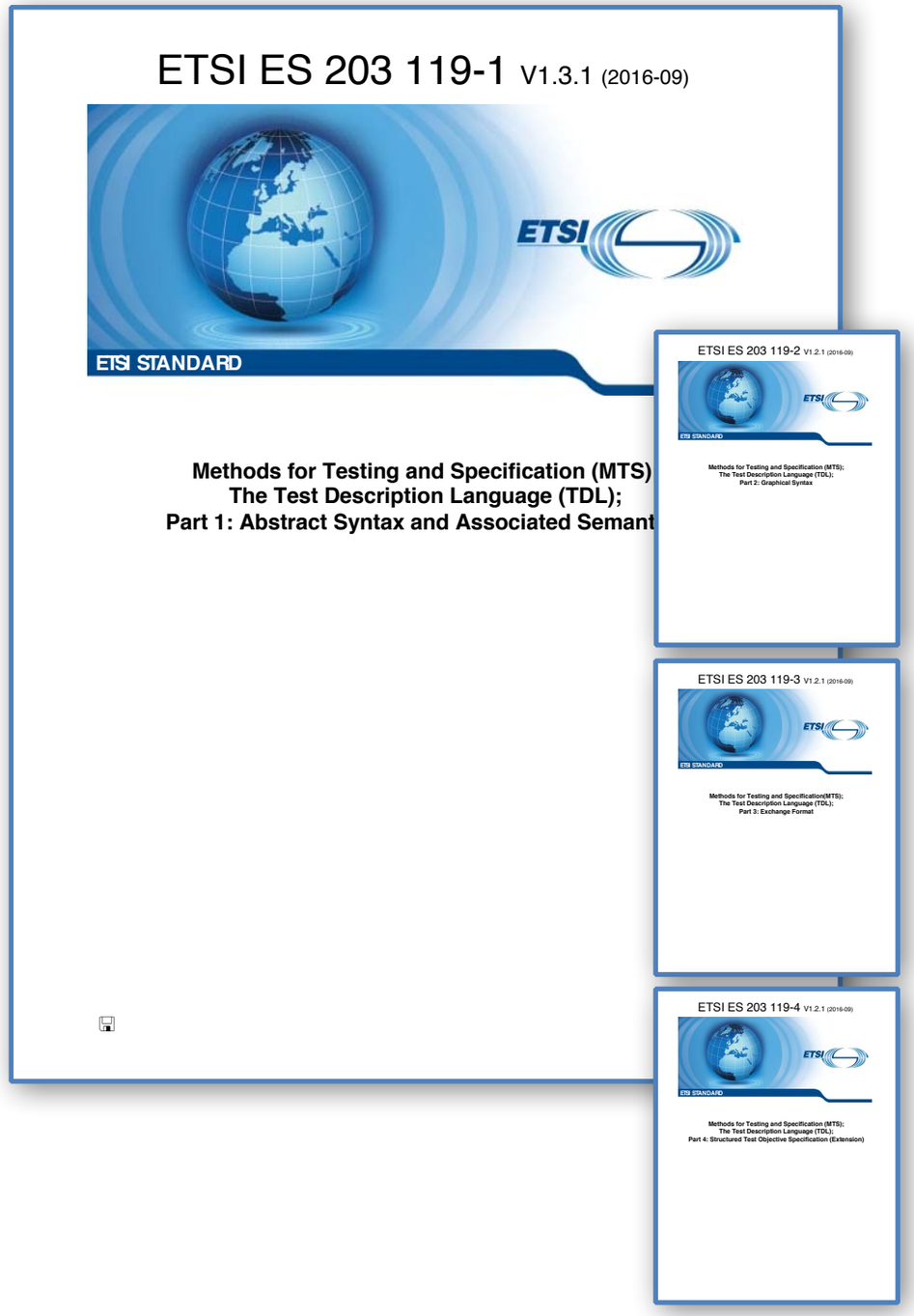
TDL P1: MM

Radio
Data Type: Message, Signal



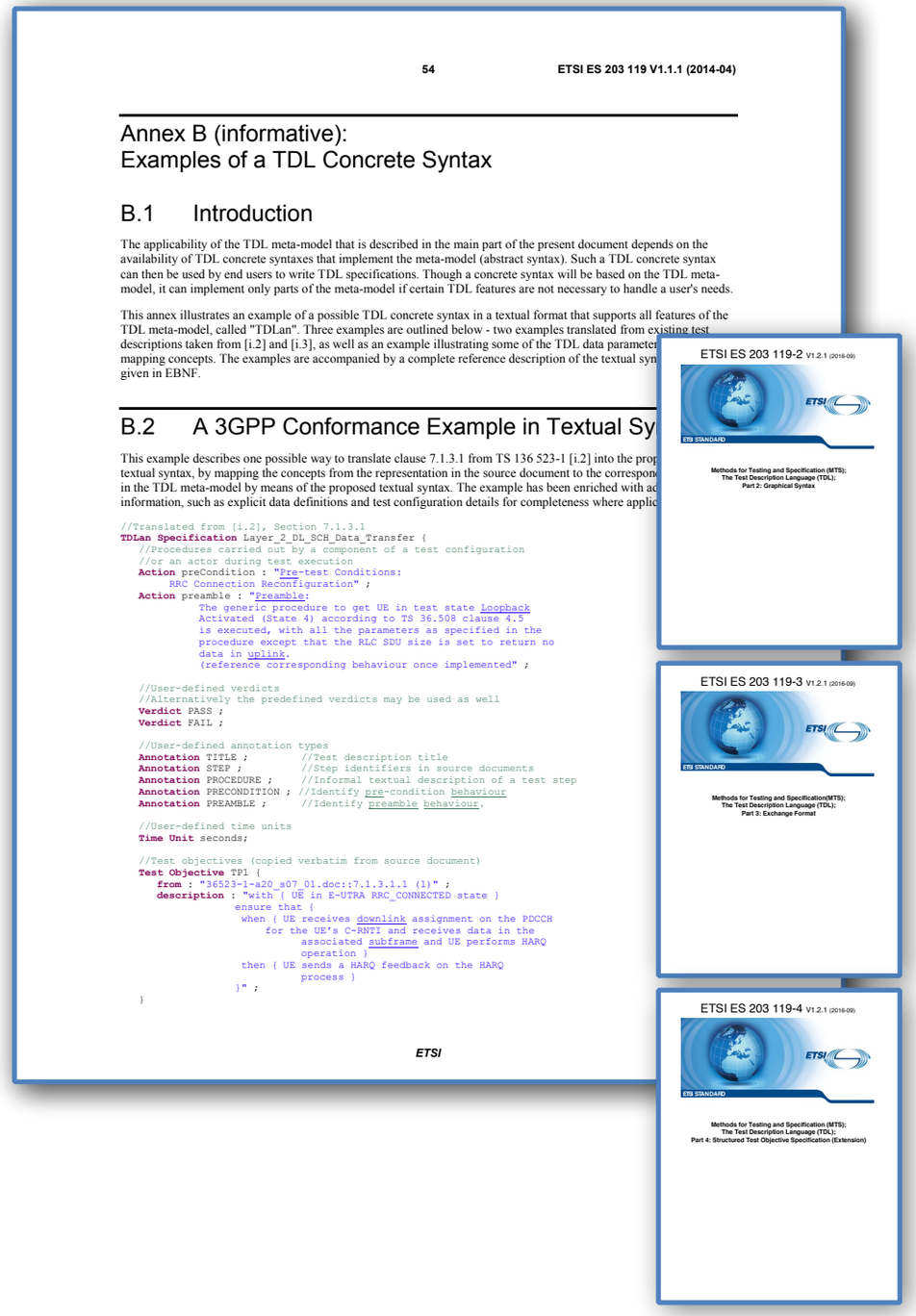
What is TDL? Part 1: MM

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



What is TDL? Part 1: MM

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



What is TDL? Part 1: MM

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

What is TDL? Part 1: MM

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

What is TDL? Part 1: MM

- Test data
  - data definition and data use
  - abstract types and instances
  - composed by using parameters
  - functions and actions
  - mappable to concrete data
  - variables and special values

What is TDL? Part 1: MM

Test Design

```
Type Login;
Login correct;
Login incorrect;

Use "data.ttcn3" as DATA ;
Map correct to "johnny_correct" in DATA as correct_ttcn3;
Map incorrect to "johnny_incorrect" in DATA as incorrect_ttcn3;
```

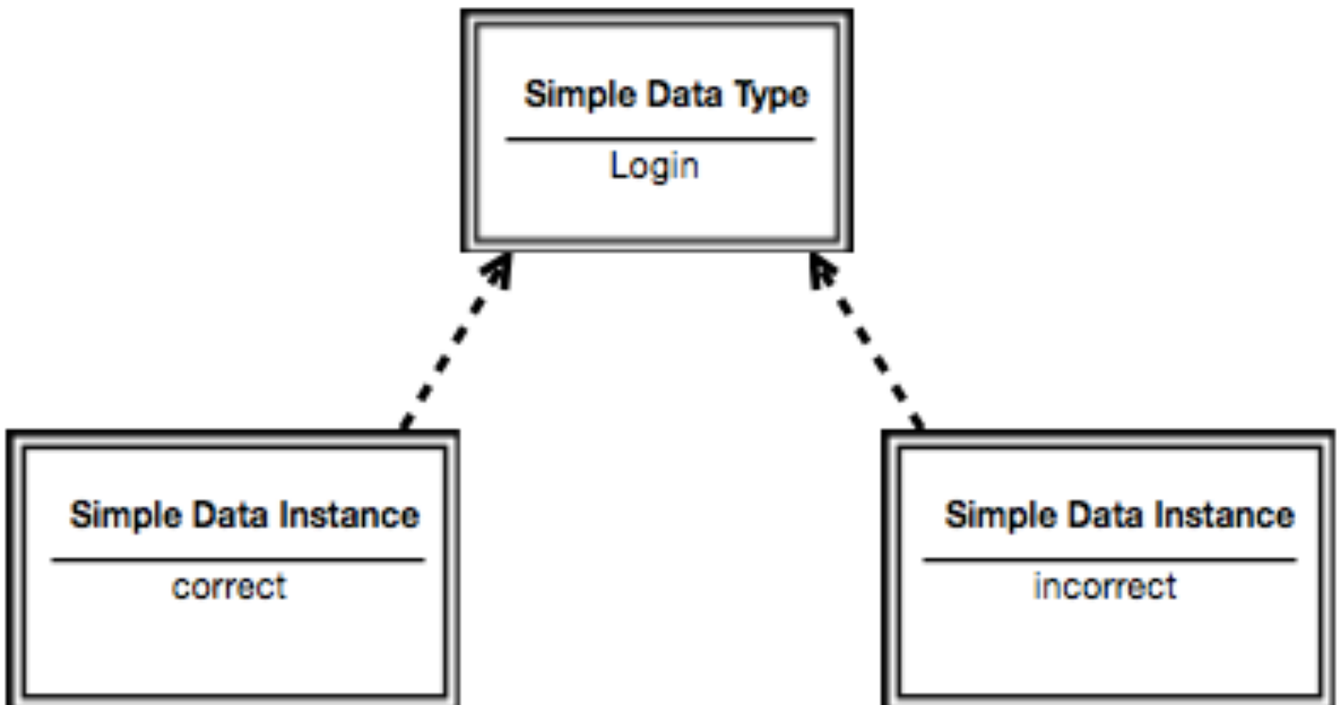
```
template Login johnny_correct := {
    user := "johnny",
    password := "apple",
    hint := "seed",
    id := 1000
}
template Login johnny_incorrect := {
    user := "johnny",
    password := "orange",
    hint := "second favourite fruit",
    id := 2000
}
```

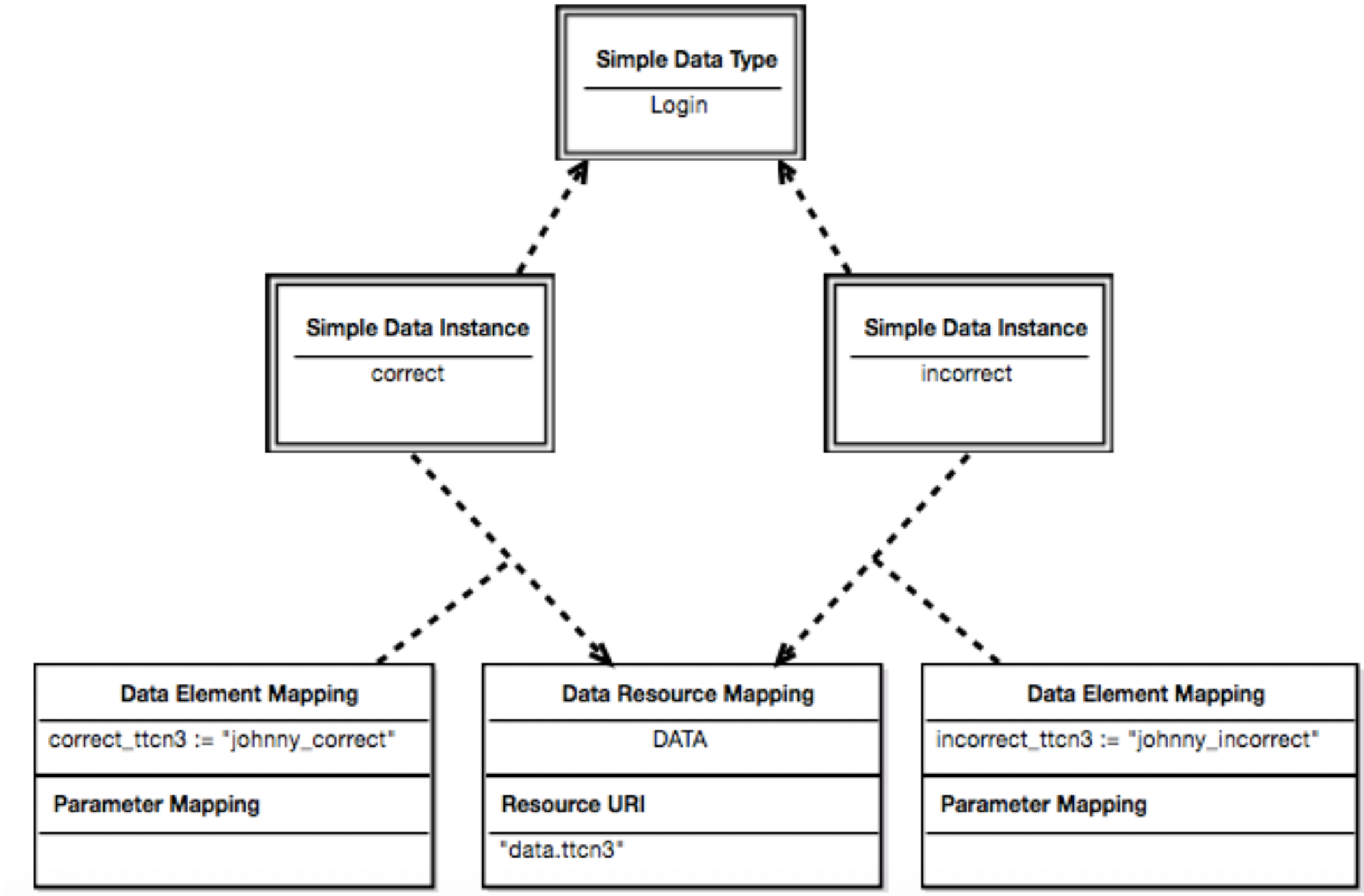
Test Implementation

```
type record Login {
  charstring user,
  charstring password,
  charstring hint,
  integer id
} with {
  encode "xpath=//div[@id='login']";
  encode (user) "relative=/div/dd[3]";
  encode (password) "relative=/div/dd[4]";
};
```

What is TDL? Part 1: MM

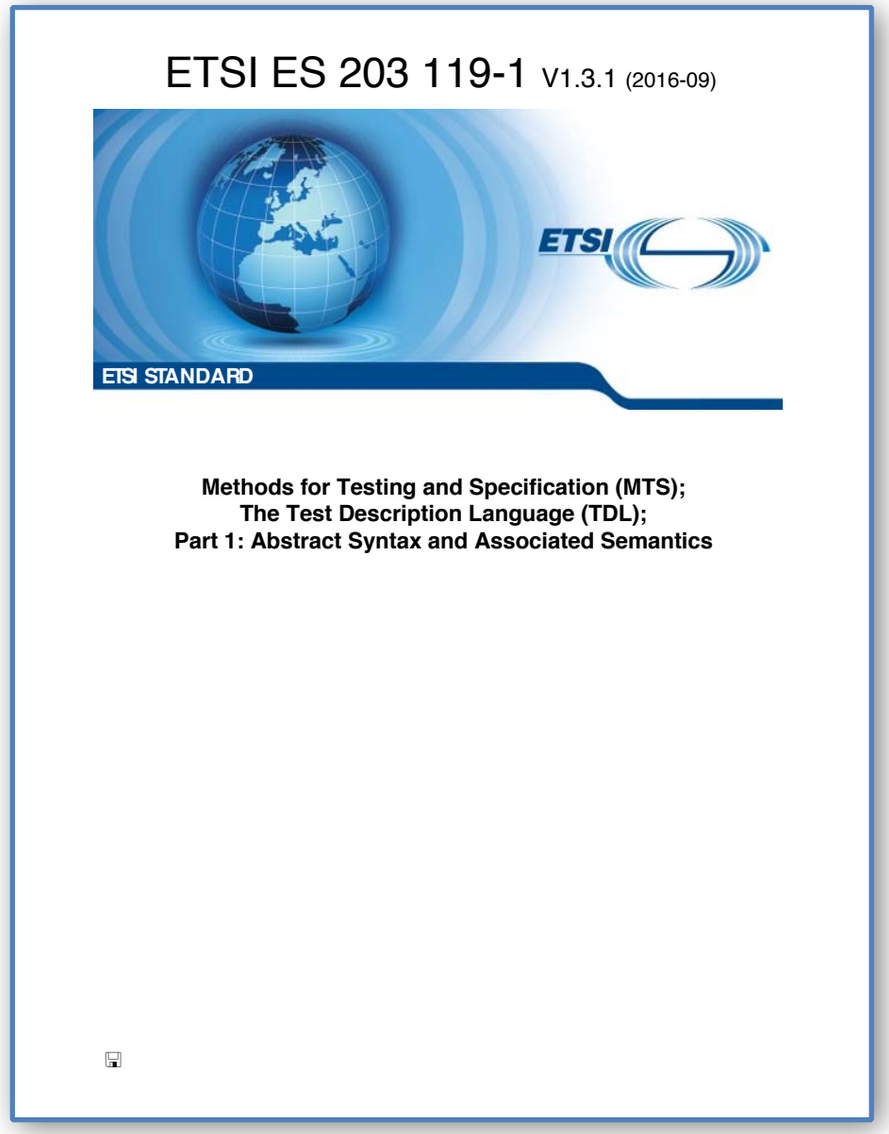
```
Type Login;
Login correct;
Login incorrect;

Use "data.ttcn3" as DATA ;
Map correct to "johnny_correct" in DATA as correct_ttcn3;
Map incorrect to "johnny_incorrect" in DATA as incorrect_ttcn3;
```




Simple Data Type
Login
Simple Data Instance
correct
Simple Data Instance
incorrect
Data Element Mapping
correct_ttcn3 := "johnny_correct"
Parameter Mapping
Data Resource Mapping
DATA
Resource URI
"data.ttcn3"
Data Element Mapping
incorrect_ttcn3 := "johnny_incorrect"
Parameter Mapping

What is TDL? Part 1: MM

- Test configuration
  - typed components and gates
  - timers and variables
  - connections among gates
  - component roles

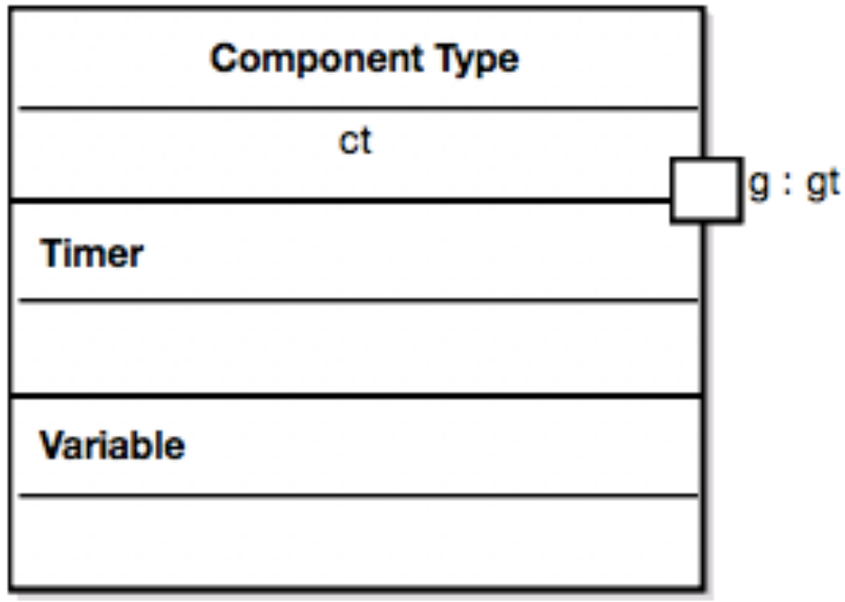

ETSI ES 203 119-1 V1.3.1 (2016-09)

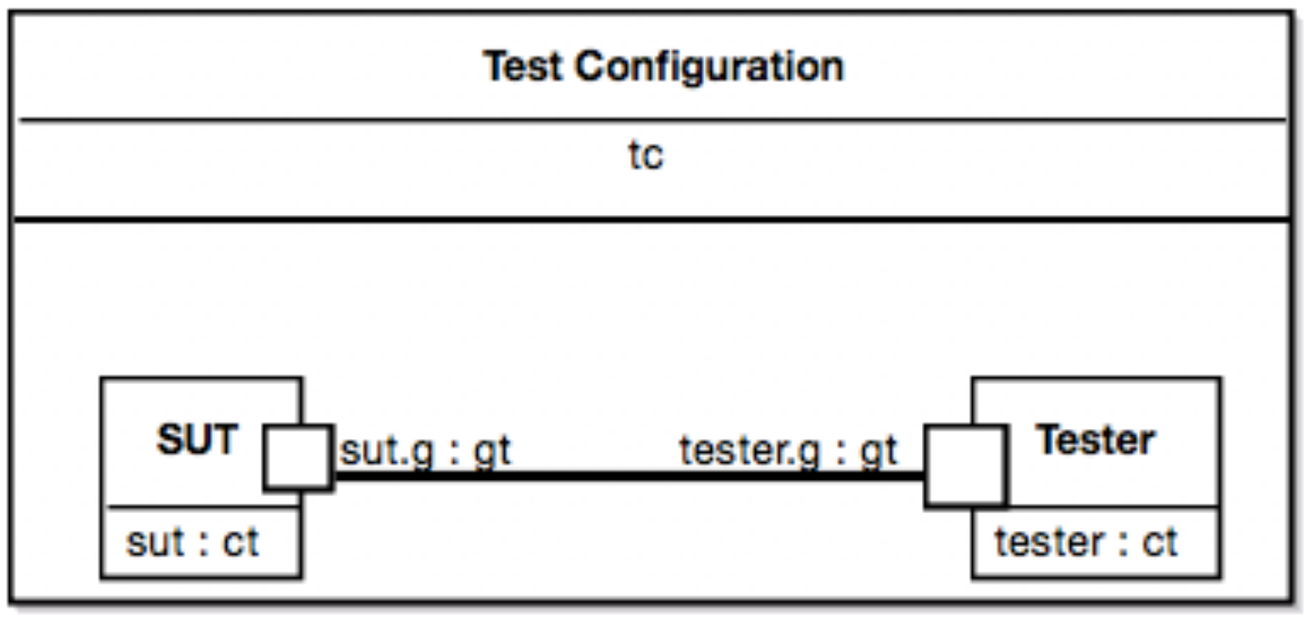
ETSI STANDARD

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

What is TDL? Part 1: MM

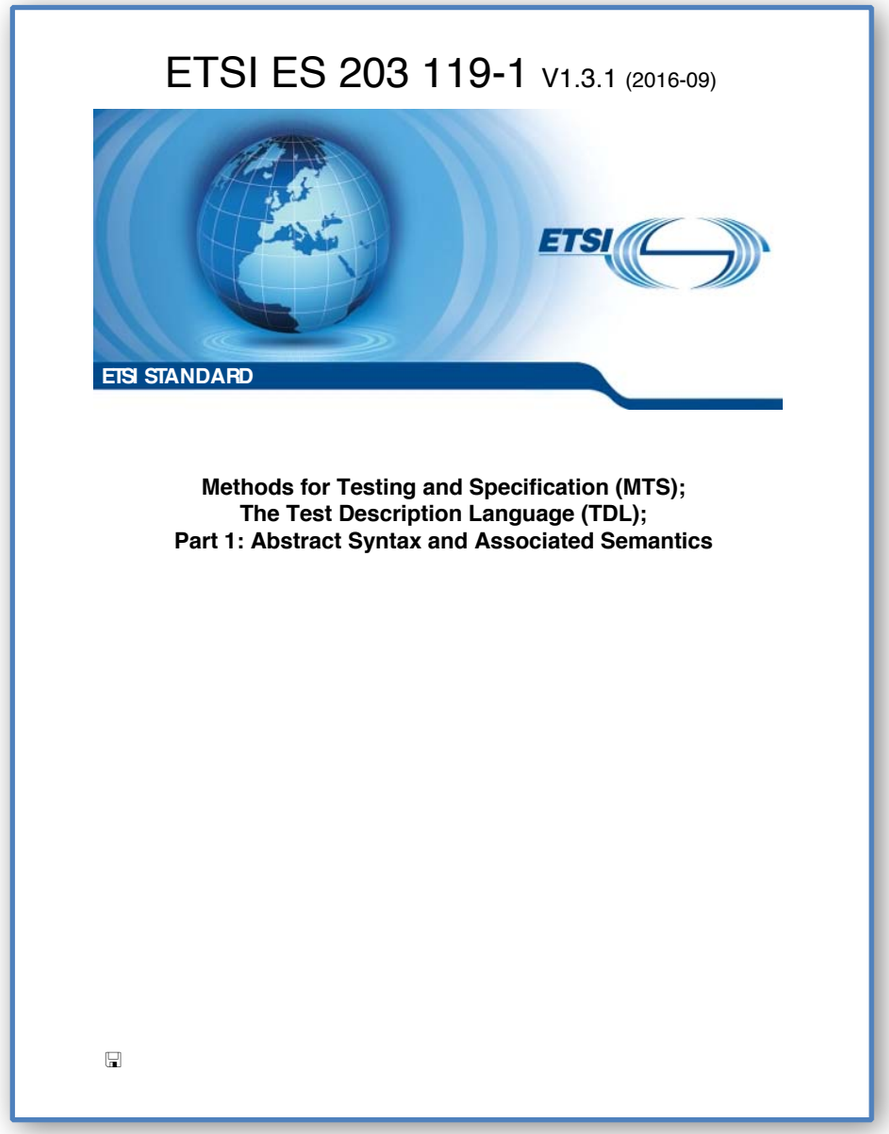
```
Gate Type gt accepts Login, Response;

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;
}
```





What is TDL? Part 1: MM

- Test behaviour
  - defines expected behaviour
  - failure upon deviations by default
  - actions and interactions
  - alternative, parallel, iterative, conditional
  - defaulting, interrupting, breaking

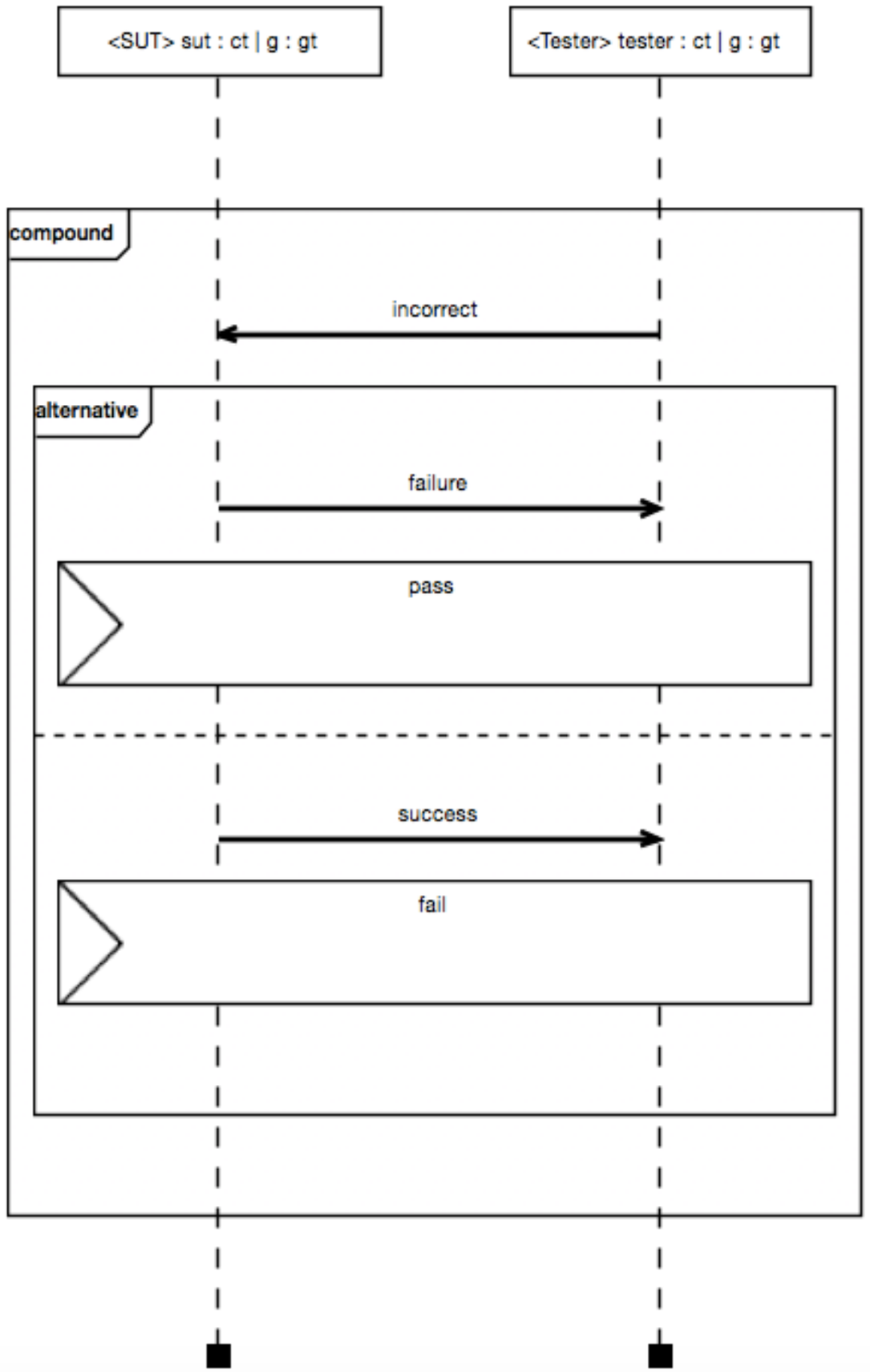


What is TDL? Part 1: MM

```
Test Description td (p of type Login)
  uses configuration tc {
    tester.g sends incorrect to sut.g;
    alternatively {
        sut.g sends failure to tester.g with {
            test objectives : tp;
        };
        set verdict to pass;
    } or {
        sut.g sends success to tester.g;
        set verdict to fail;
    }
}
```

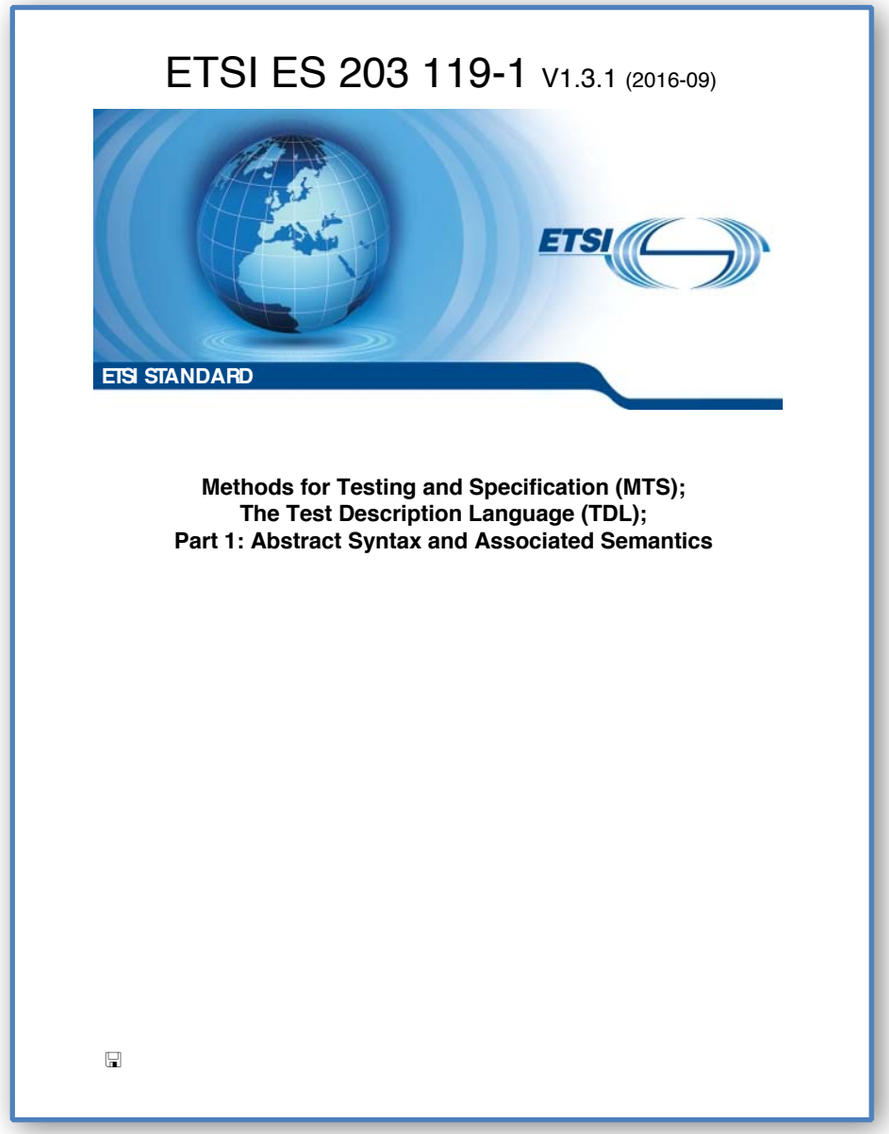
or simply (relying on the default semantics):

```
Test Description td_default (p of type Login)
  uses configuration tc {
    tester.g sends incorrect to sut.g;
    sut.g sends failure to tester.g with {
        test objectives : tp;
    };
}
```



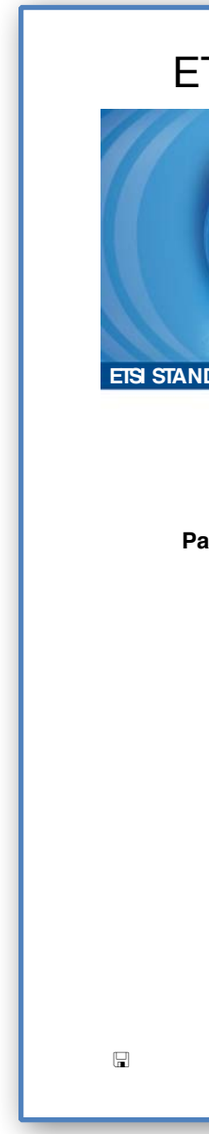
What is TDL? Part 1: MM

- Test objectives
  - may be attached to
    - behaviour (atomic or compound)
    - whole test description
  - contain description and reference



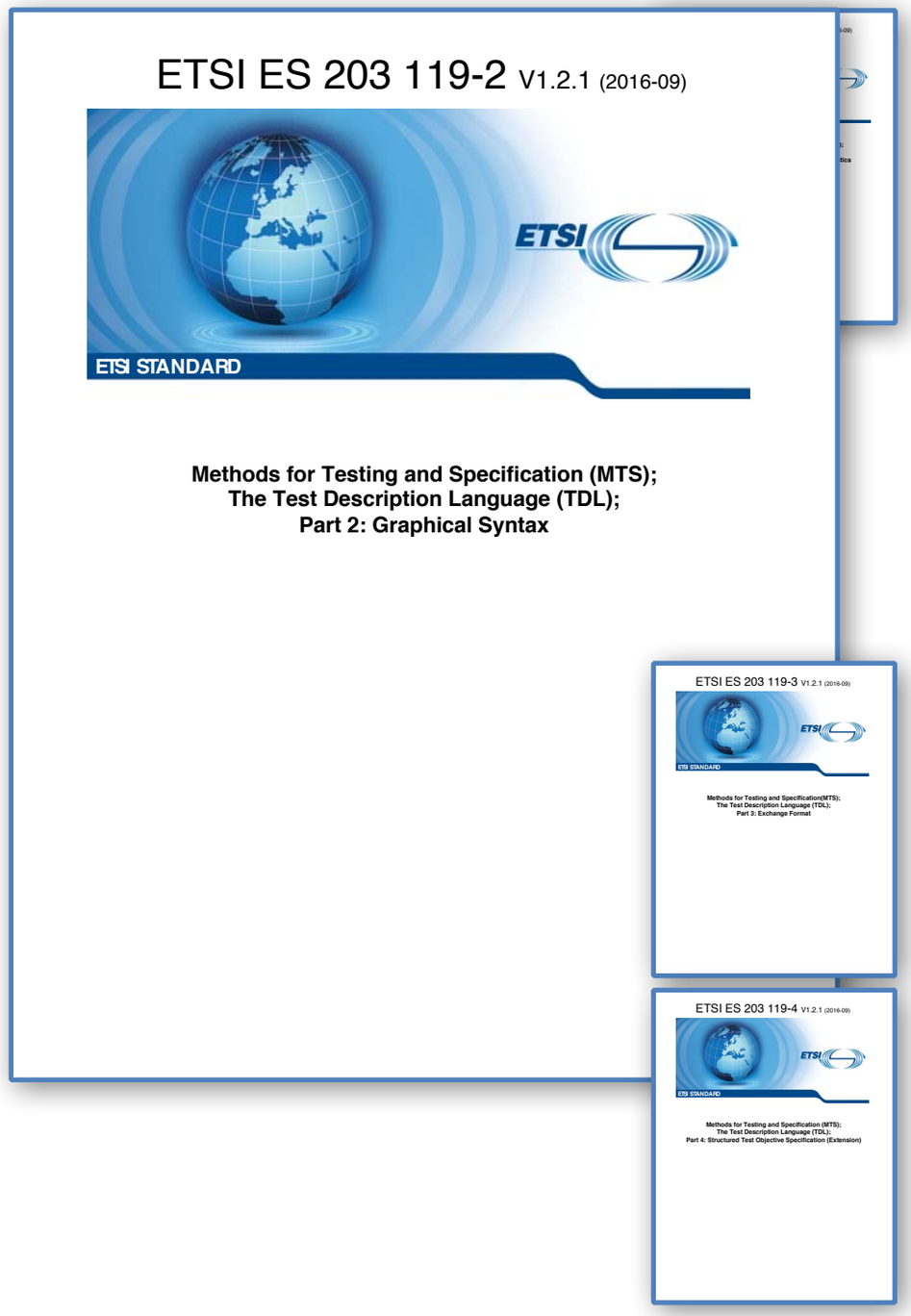
What is TDL? Part 1: MM

```
Test Objective tp {
    description : "ensure that
                   when incorrect login is provided
                   a failure response is sent";
}
Test Description td (p of type Login)
  uses configuration tc {
    tester.g sends incorrect to sut.g;
    alternatively {
        sut.g sends failure to tester.g with {
            test objectives : tp;
        };
        set verdict to pass;
    } or {
        sut.g sends success to tester.g;
        set verdict to fail;
    }
}
```



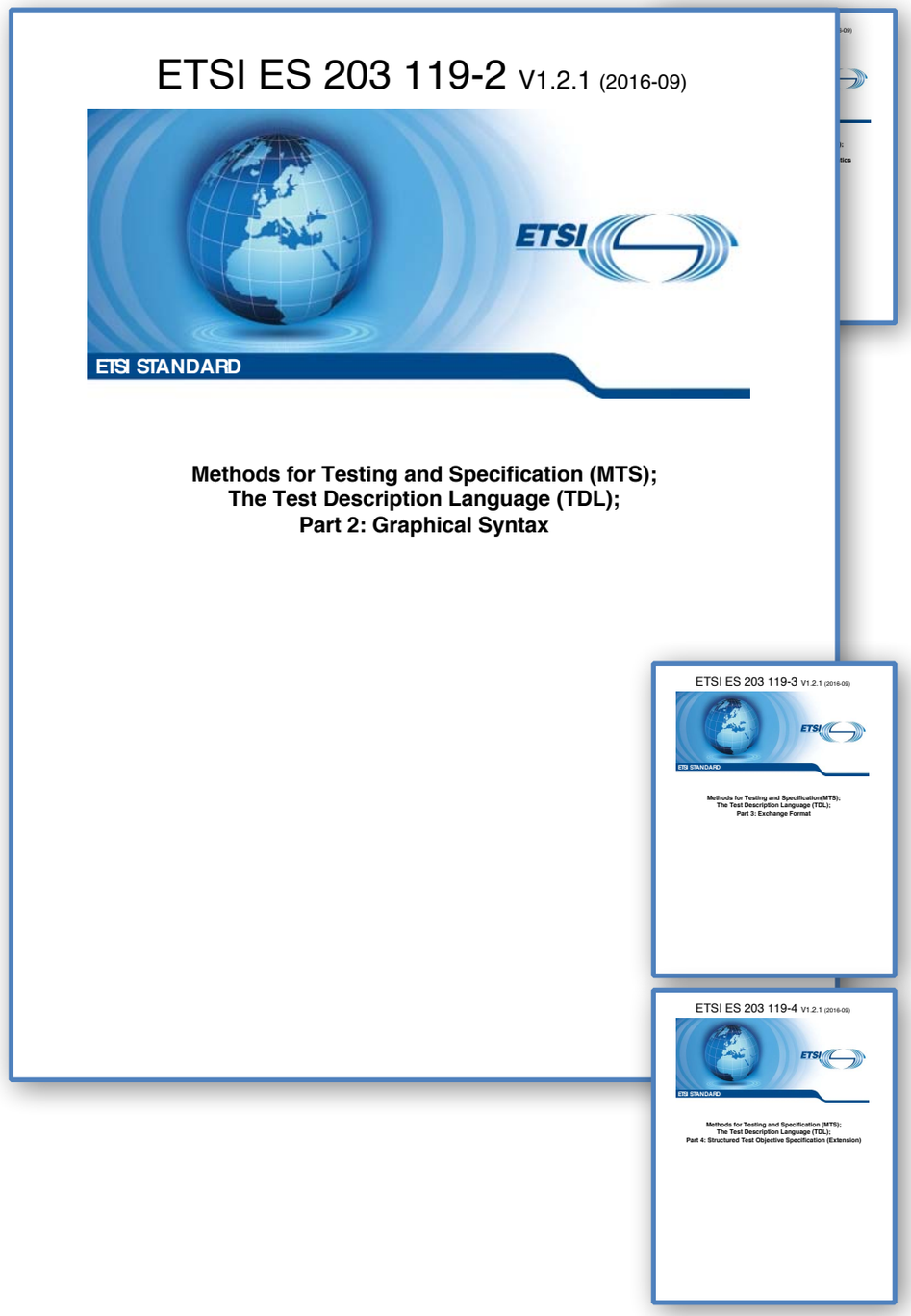
What is TDL? Part 2: GR

- Graphical languages
  - common in (test) modelling
  - ease communication
- TDL Graphical Syntax
  - hybrid graphical language
  - simple shapes, compartments
  - textual visualisation of contents

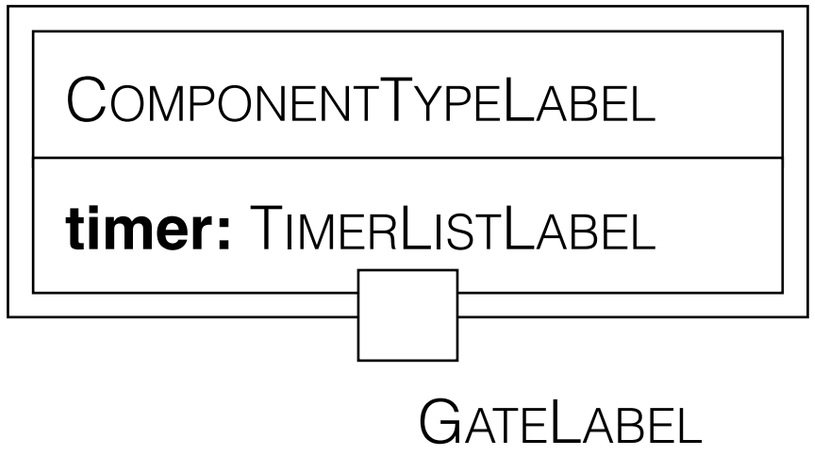


What is TDL? Part 2: GR

- Aligned with UML
  - distinct where semantics differ
- One diagram to rule them all!
- BNF-like label specification
- Considers both ease of use and implementation
- Prototyped with Sirius



What is TDL? Part 2: GR

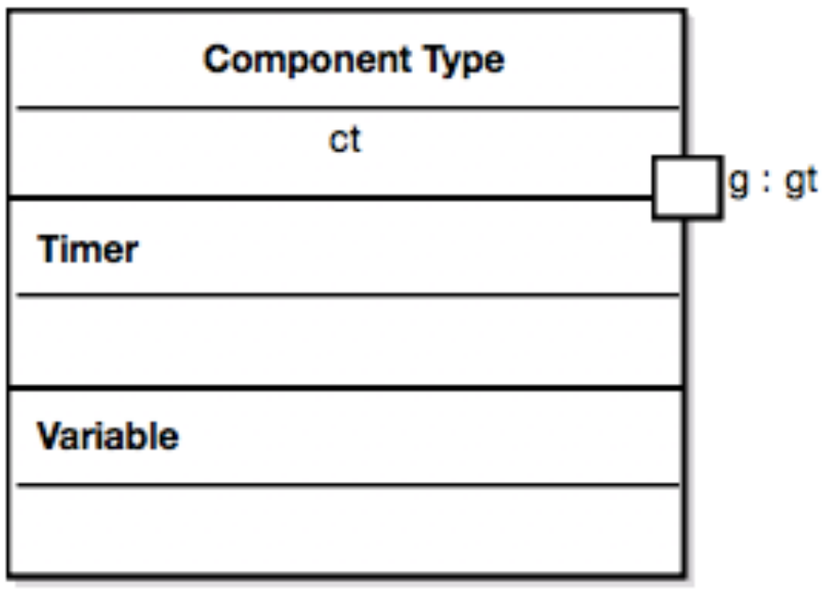


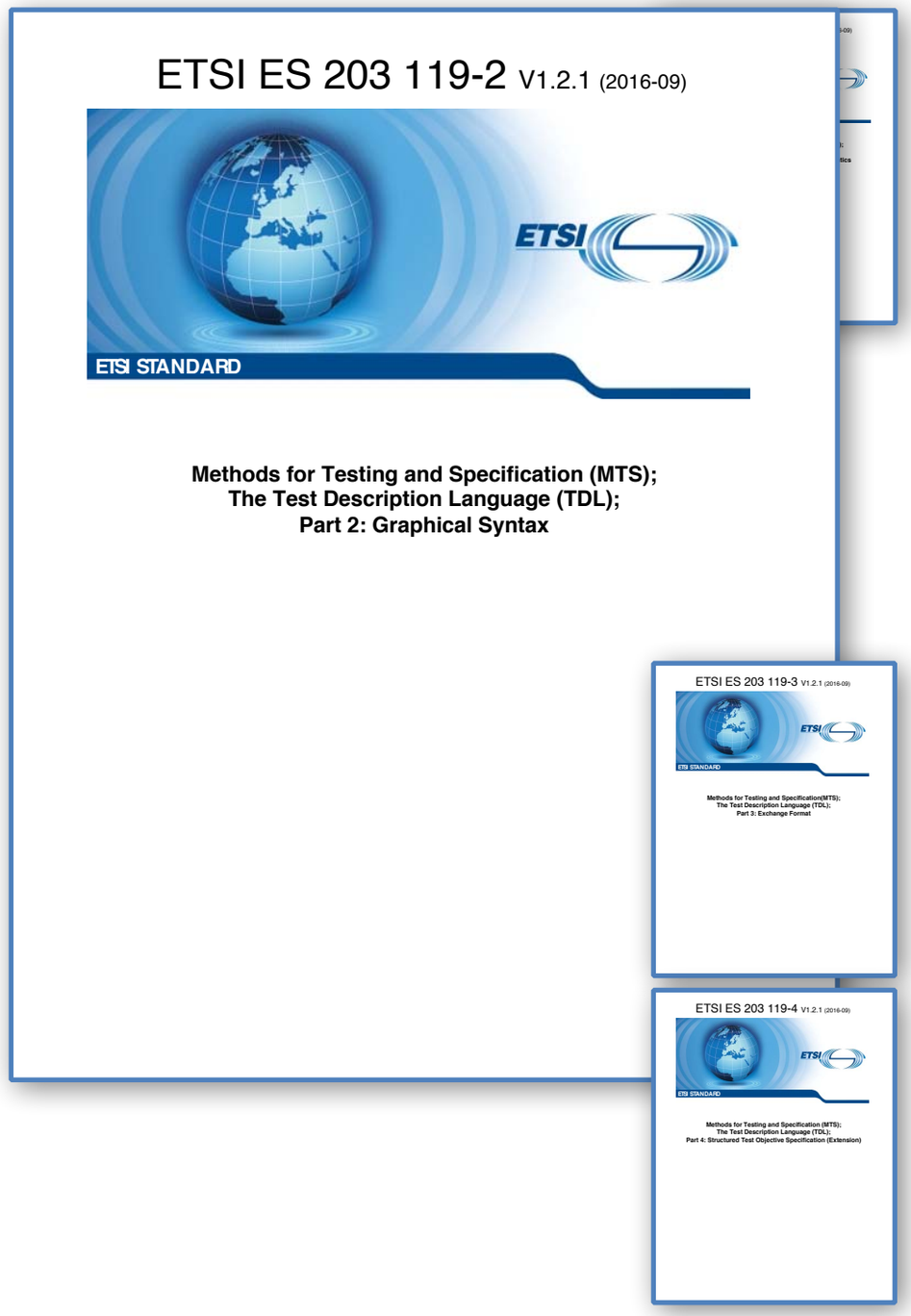
context: ComponentType

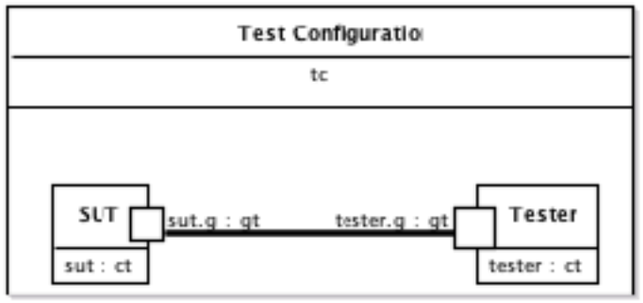
COMPONENTTYPELABEL ::= self.name

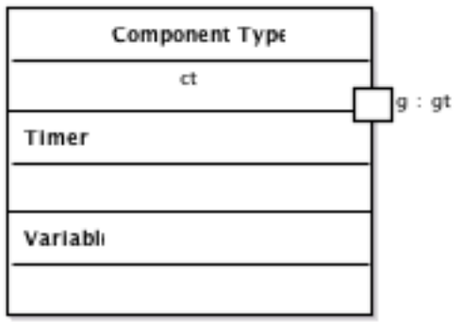
TIMERLISTLABEL ::= self.timer.name

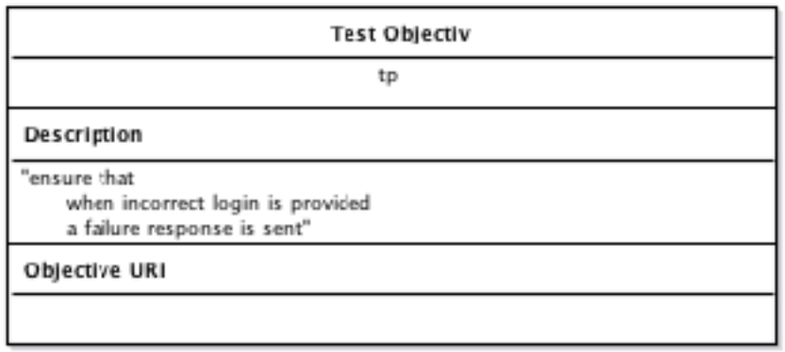
…

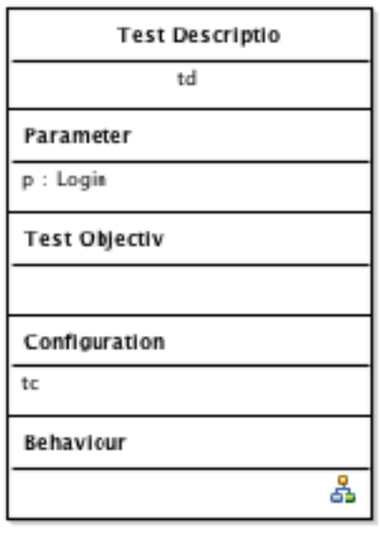


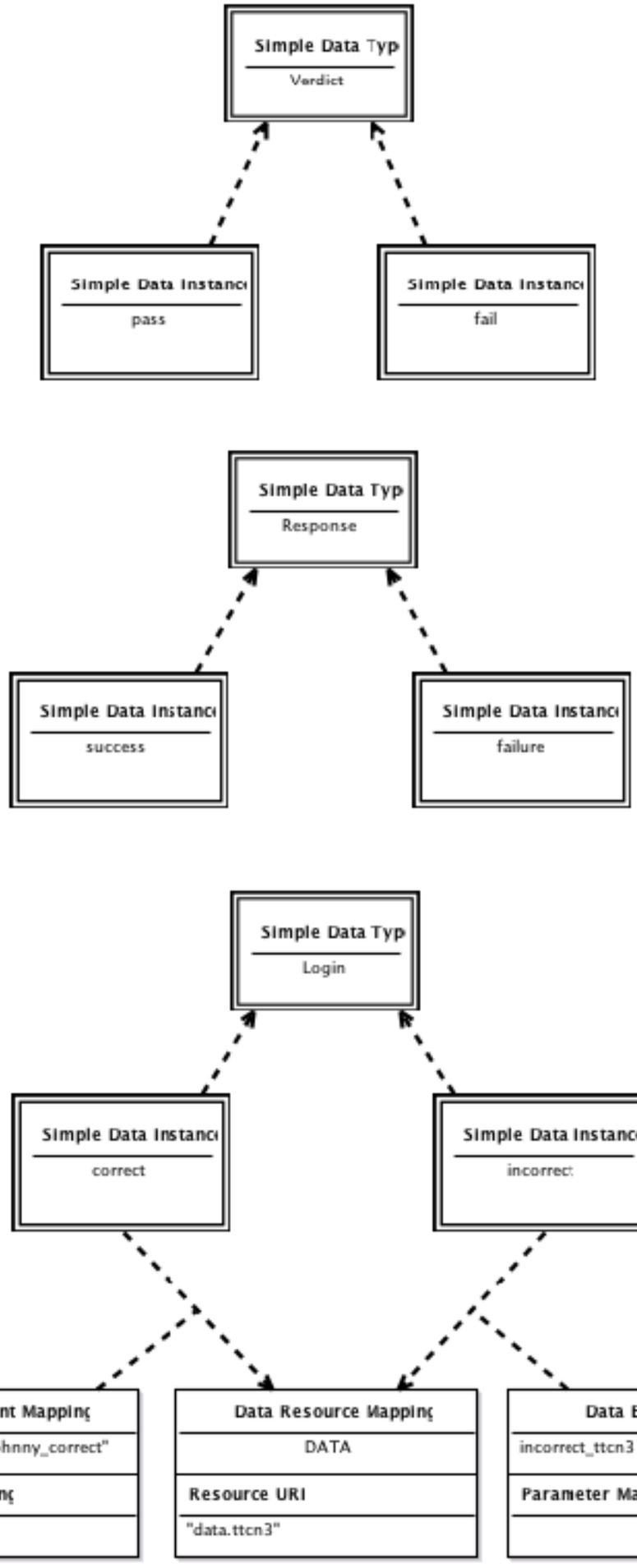


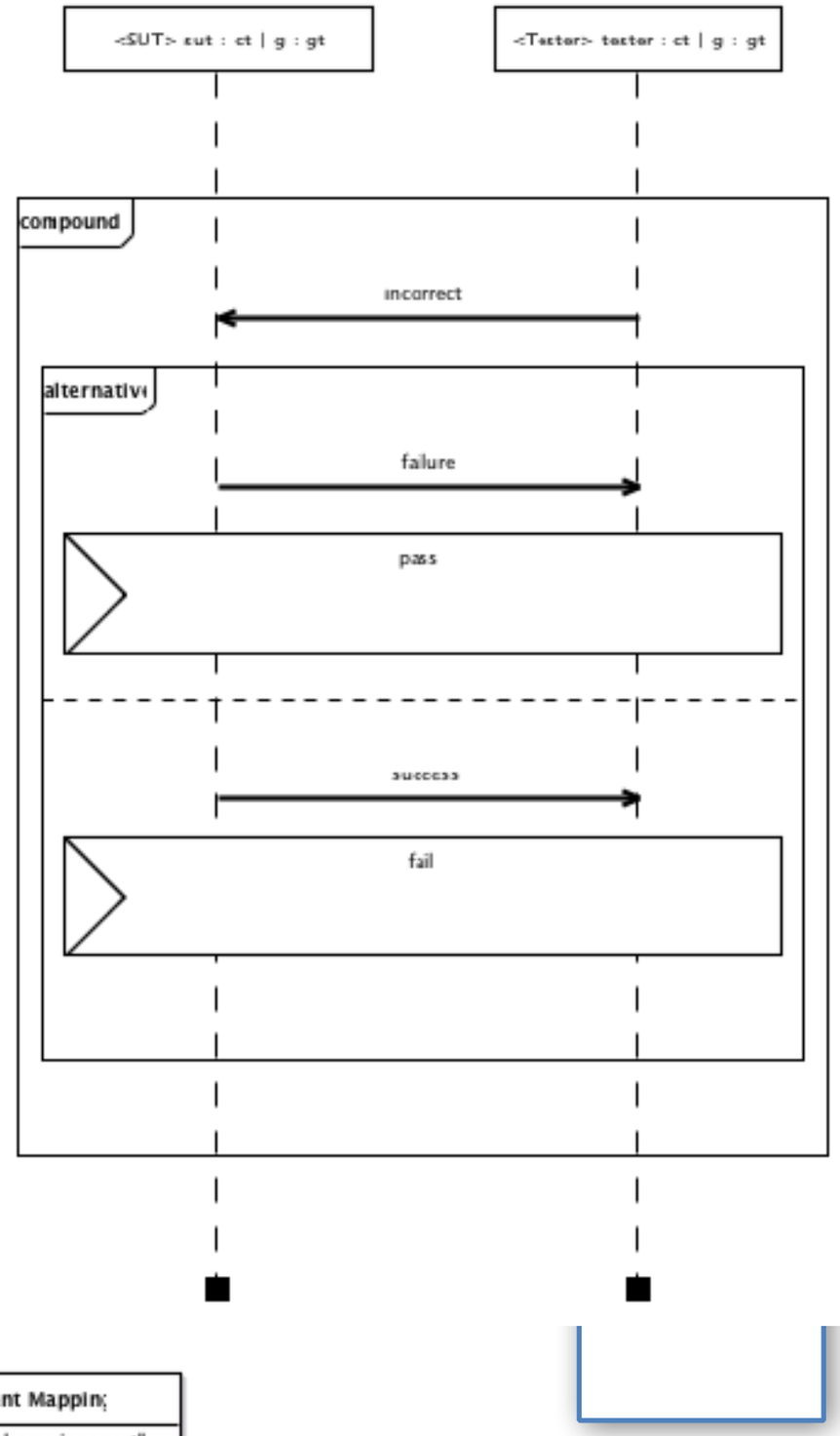

Test Configuratio
tc
SUT
sut : ct
sut.g : gt
tester.g : gt
Tester
tester : ct


Component Type
ct
g : gt
Timer
Variabl


Test Objectiv
tp
Description
"ensure that
when incorrect login is provided
a failure response is sent"
Objective URI

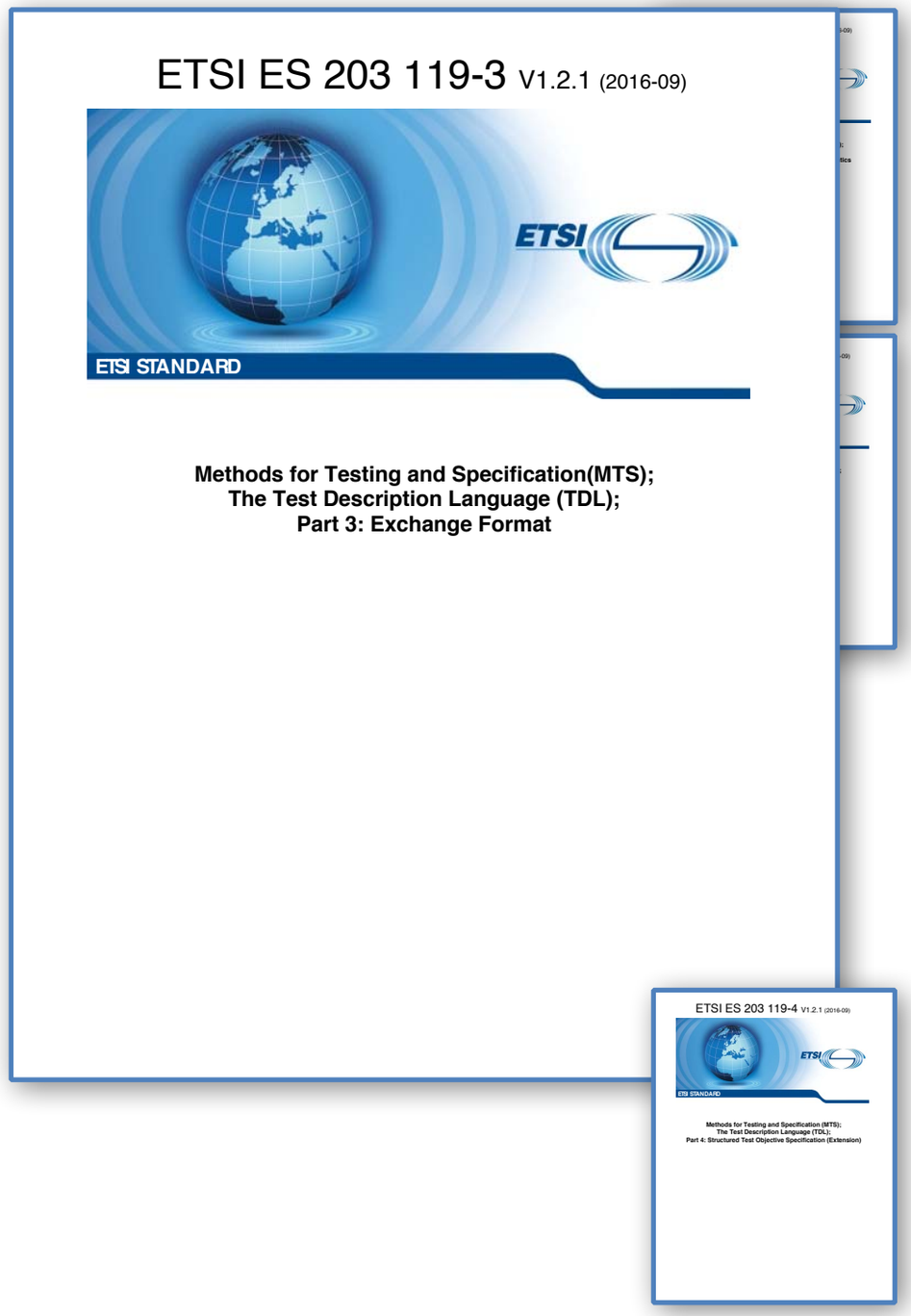

Test Descriptio
td
Parameter
p : Login
Test Objectiv
Configuration
tc
Behaviour


Simple Data Typ
Verdict
Simple Data Instanc
pass
Simple Data Instanc
fail
Simple Data Typ
Response
Simple Data Instanc
success
Simple Data Instanc
failure
Simple Data Typ
Login
Simple Data Instanc
correct
Simple Data Instanc
incorrect
Data Element Mapping
correct_ttcn3 := "johnny_correct"
Parameter Mapping
Data Resource Mapping
DATA
Resource URI
"data.ttcn3"
Data Element Mapping
incorrect_ttcn3 := "johnny_incorrect"
Parameter Mapping


<SUT> sut : ct | g : gt
<Tester> tester : ct | g : gt
compound
incorrect
alternativ
failure
pass
success
fail

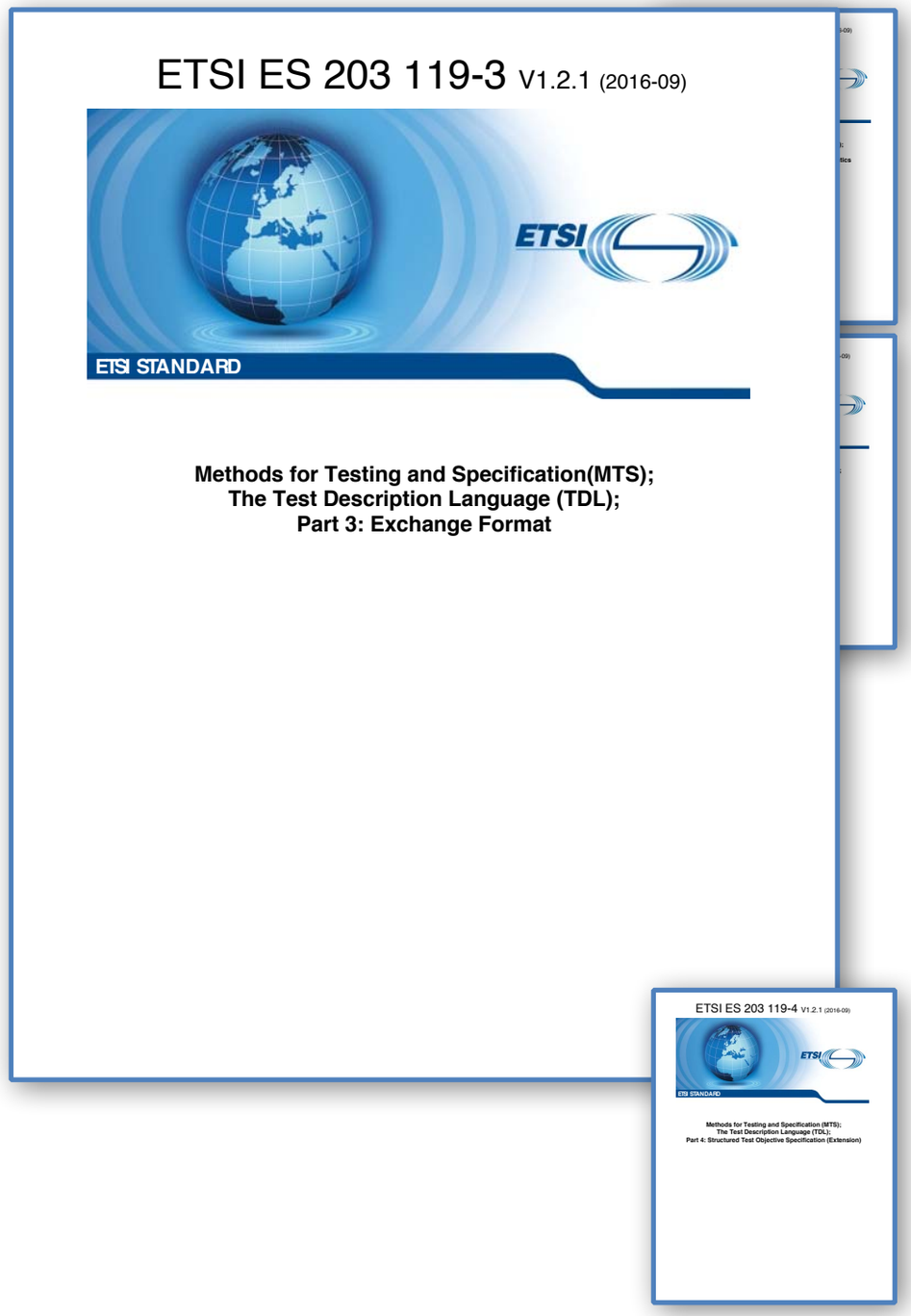
What is TDL? Part 3: XF

- Based on OMG XMI
  - XML: Metadata Interchange
  - Serialisation of MOF models
  - Exchange among MOF tools
- XMI concerns
  - complex, many options



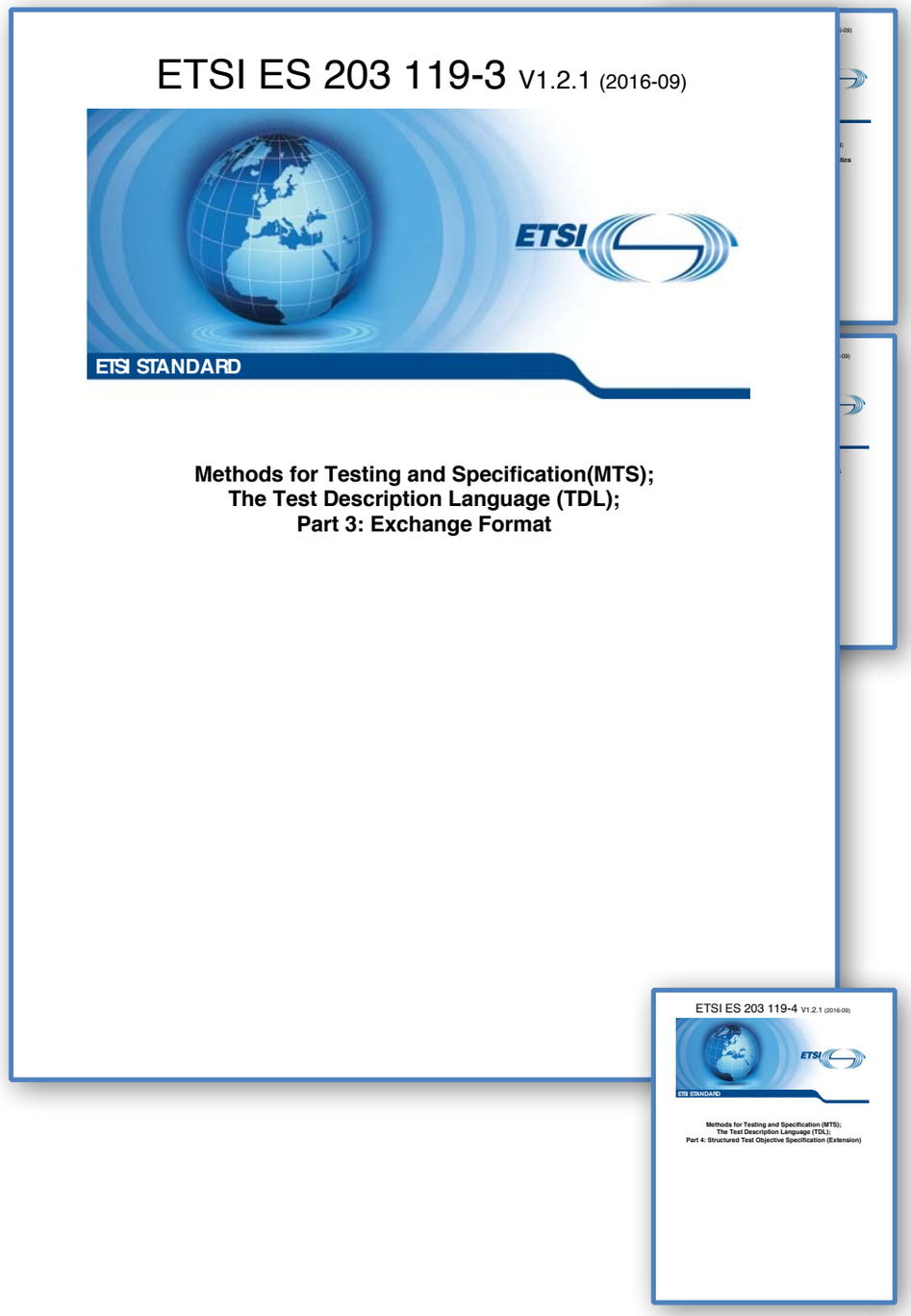
What is TDL? Part 3: XF

- TDL specific XMI structure
  - exchange of TDL models
  - canonical TDL XMI structure
    - meta-class representations
    - multiplicity, associations, inheritance
- restrict flexibility of XMI
- syntactical validity only!

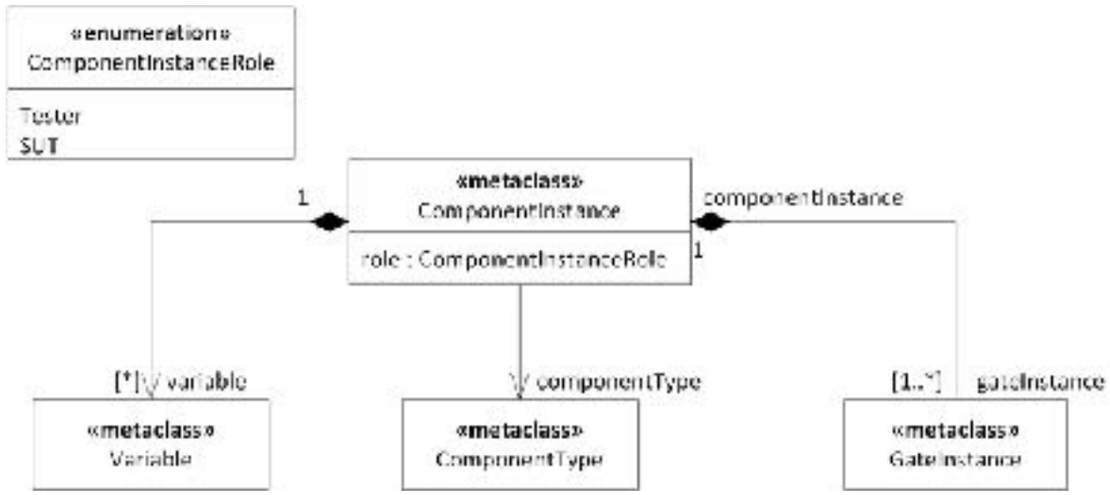


What is TDL? Part 3: XF

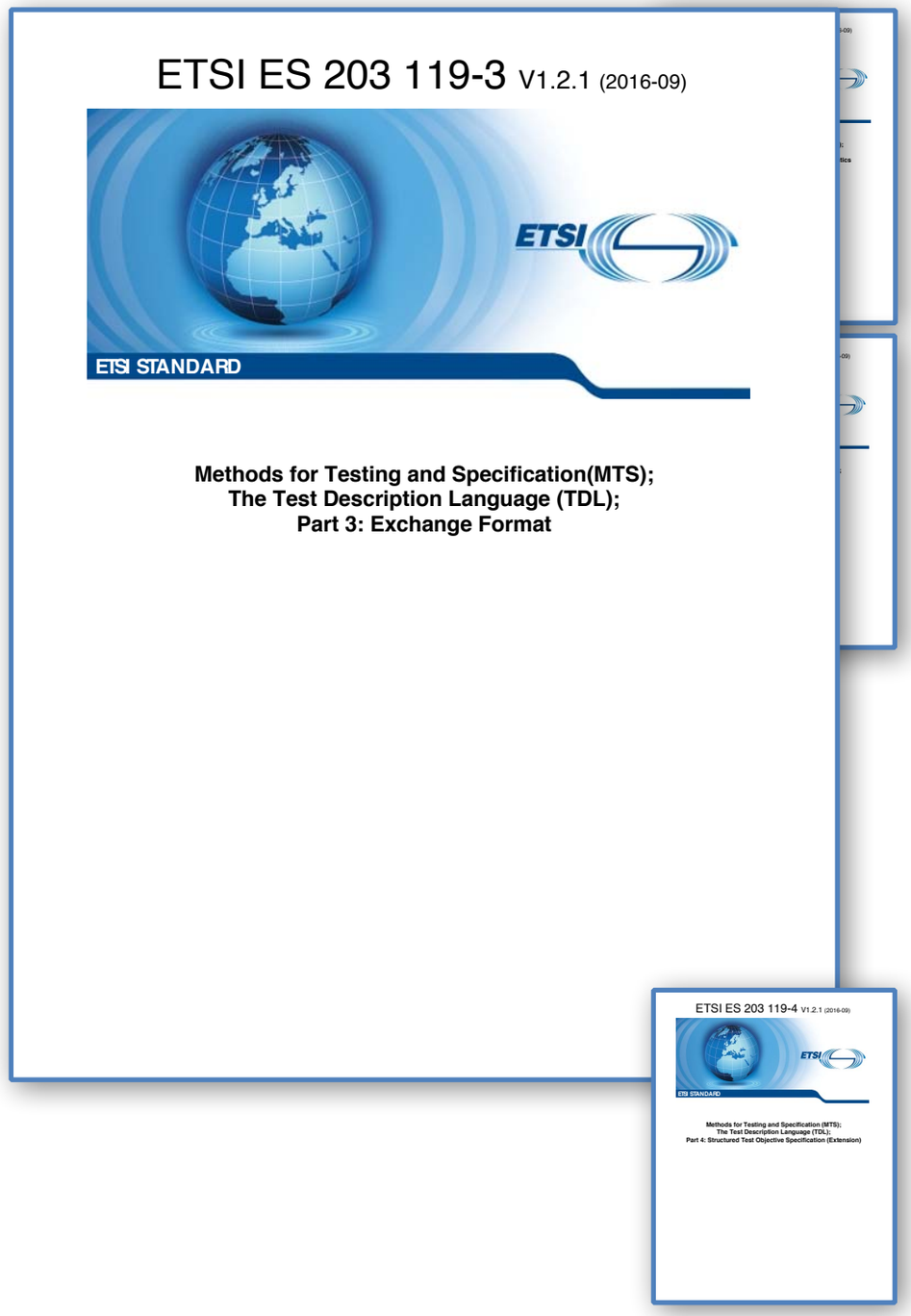
- Syntactical validity only?
  - two-step validation
  - syntax: XMI Schema
  - semantics: MOF model validation



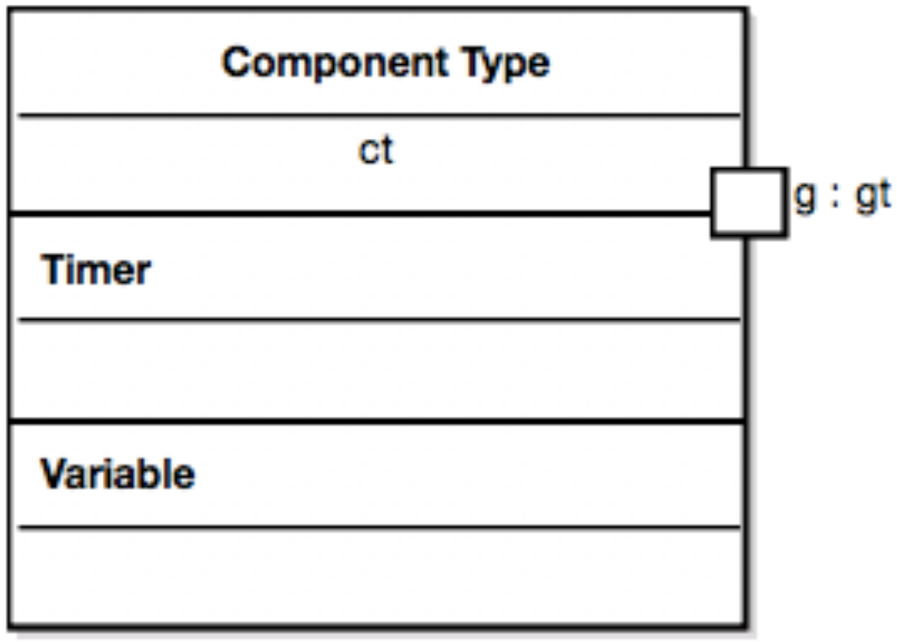
What is TDL? Part 3: XF

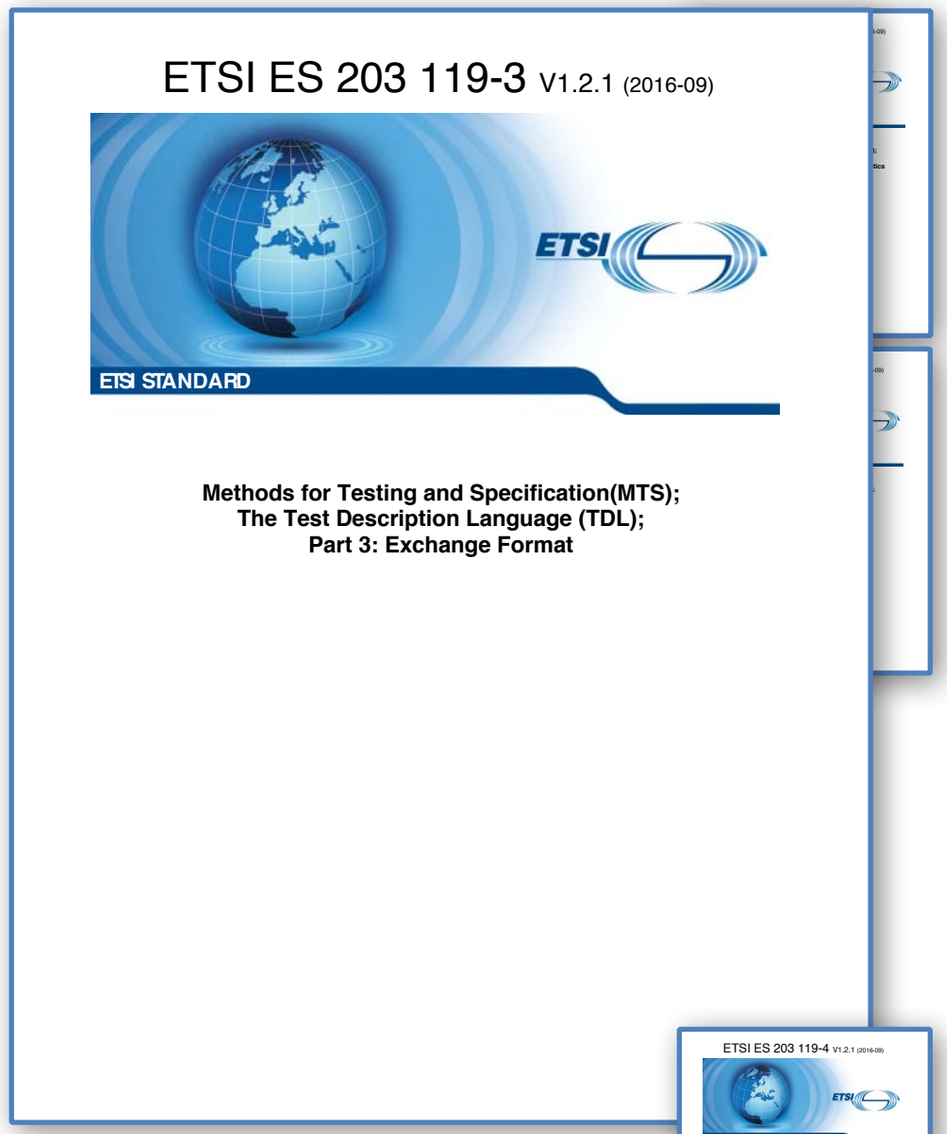


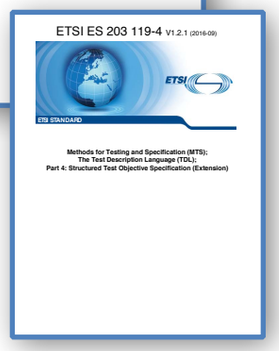

```
<xsd:complexType name="ComponentInstance">
   <xsd:complexContent>
     <xsd:extension base="tdl:Element">
       <xsd:choice maxOccurs="unbounded" minOccurs="0">
         <xsd:element name="gateInstance" type="tdl:GateInstance"/>
         <xsd:element name="variable" type="tdl:Variable"/>
       </xsd:choice>
       <xsd:attribute name="componentType" type="xsd:anyURI">
       <xsd:attribute name="role" type="tdl:ComponentInstanceRole">
     </xsd:extension>
   </xsd:complexContent>
</xsd:complexType>
```



What is TDL? Part 3: XF



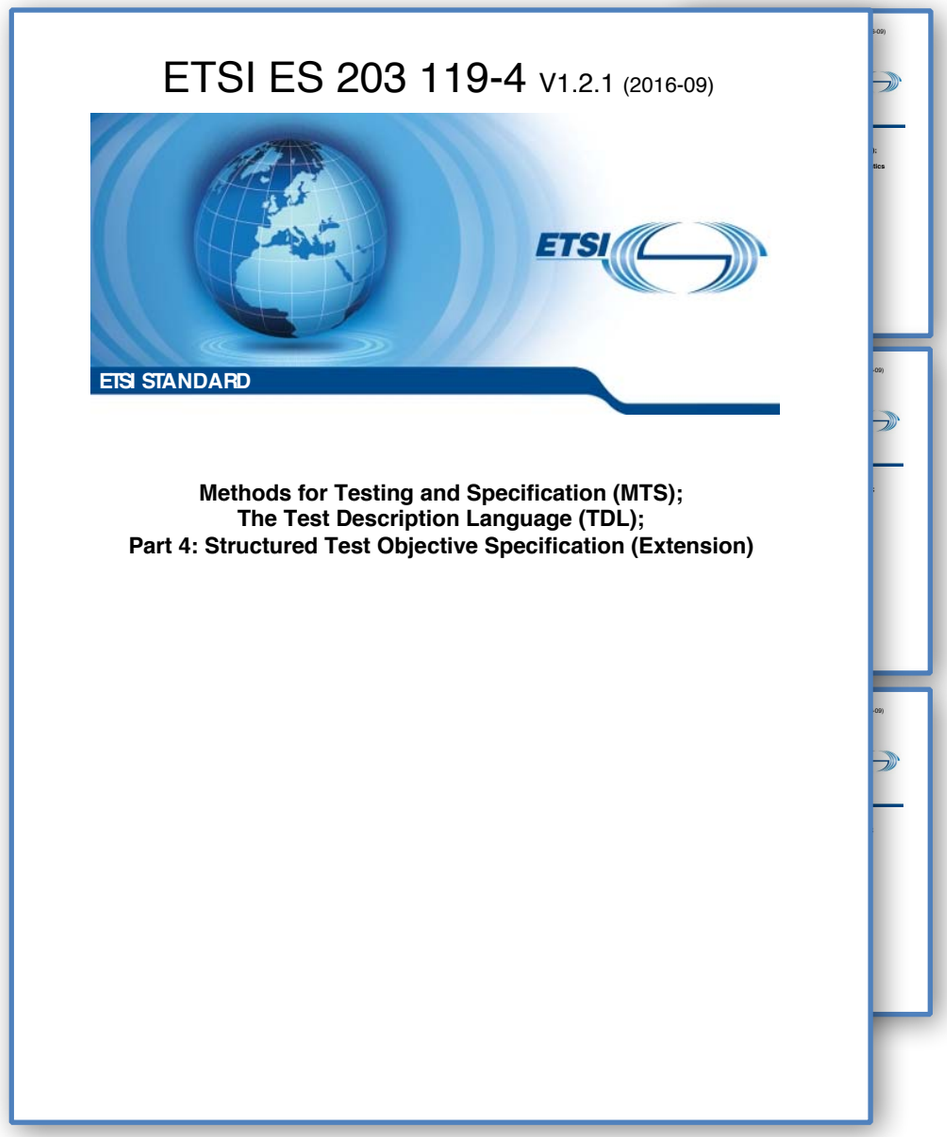





```
<packagedElement xsi:type="tdl:ComponentType" xmi:id="_qKt233asEeWrfP0MdfQNpg" name="ct">
  <gateInstance xmi:id="_qKt24HasEeWrfP0MdfQNpg" name="g" type="_qKt23nasEeWrfP0MdfQNpg"/>
</packagedElement>
```

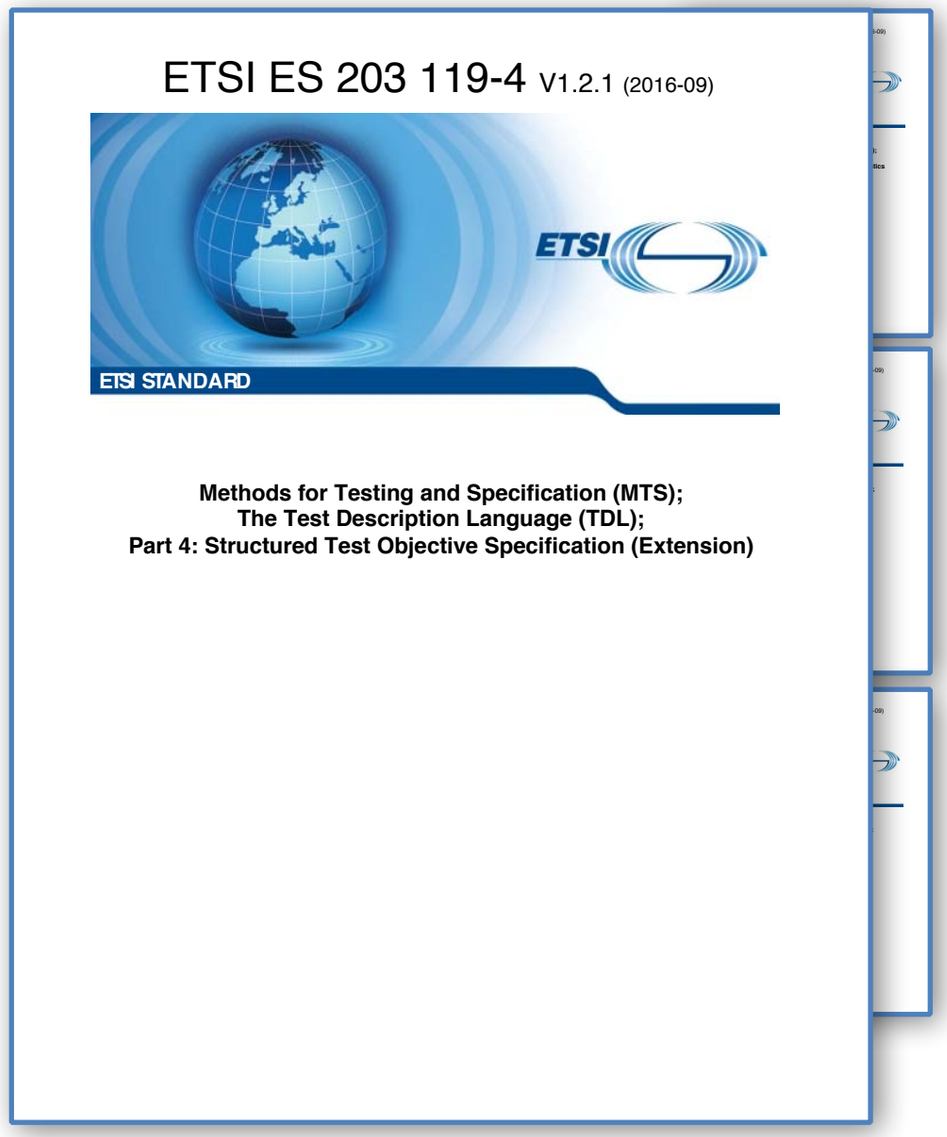
What is TDL? Part 4: TO

- Based on TPLan
  - refine test objectives
  - formalise specification
  - integrate and unify test description and test purpose specification



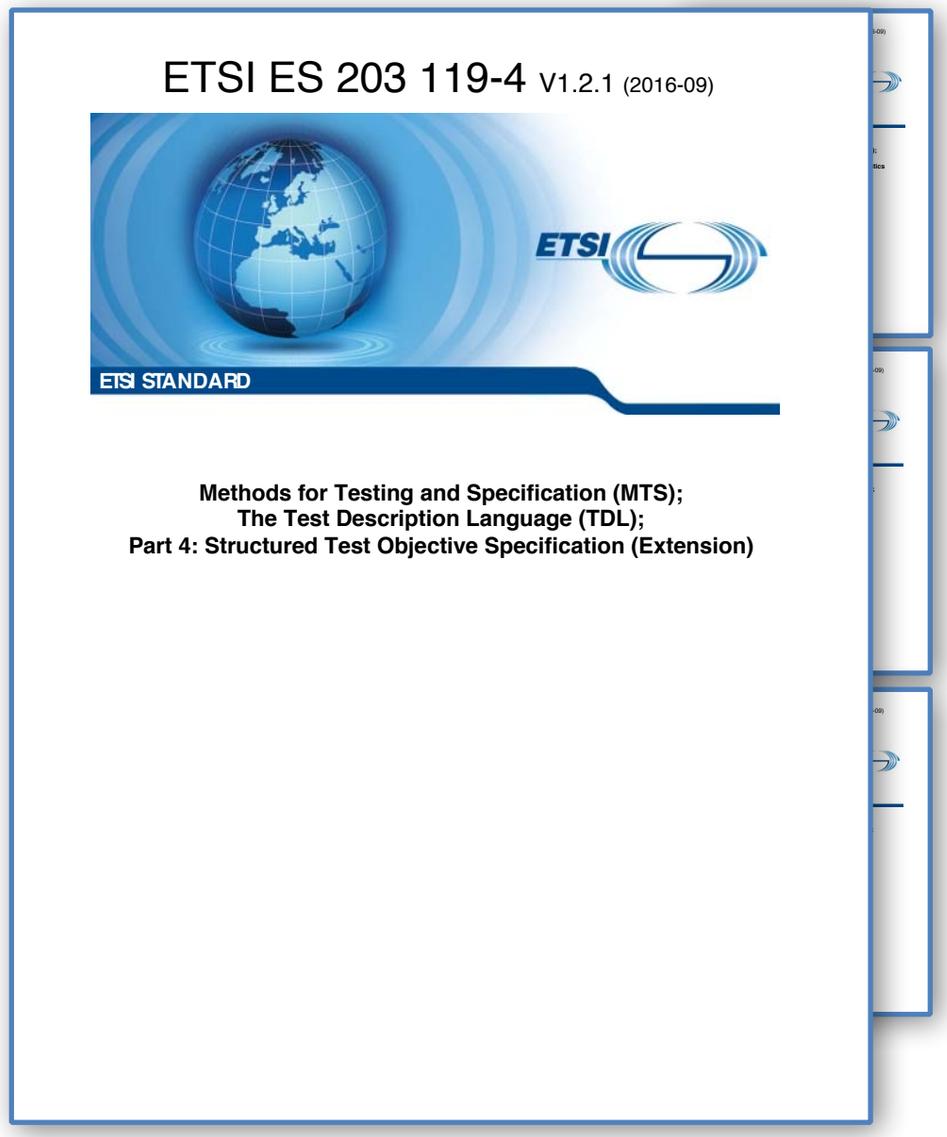
What is TDL? Part 4: TO

- Base Standard Specification
- Identification of Requirements
- Creation of ICS/IFS
- Definition of TSS
- Specification of Test Purposes
- Specification of Test Descriptions
- Specification of Test Cases
- Validation



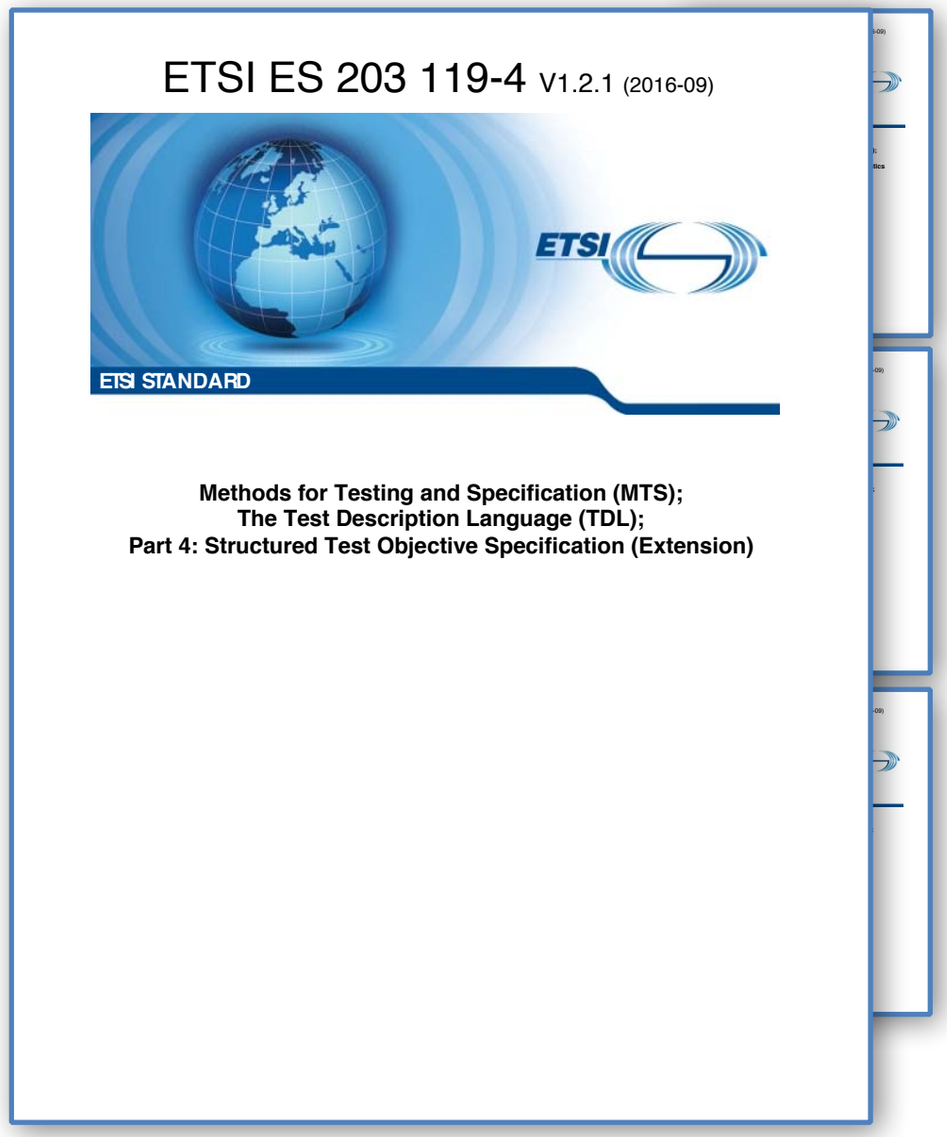
What is TDL? Part 4: TO

- Base Standard Specification
- Identification of Requirements
- Creation of ICS/IFS
- Definition of TSS
- Specification of Test Purposes
- Specification of Test Descriptions
- Specification of Test Cases
- Validation



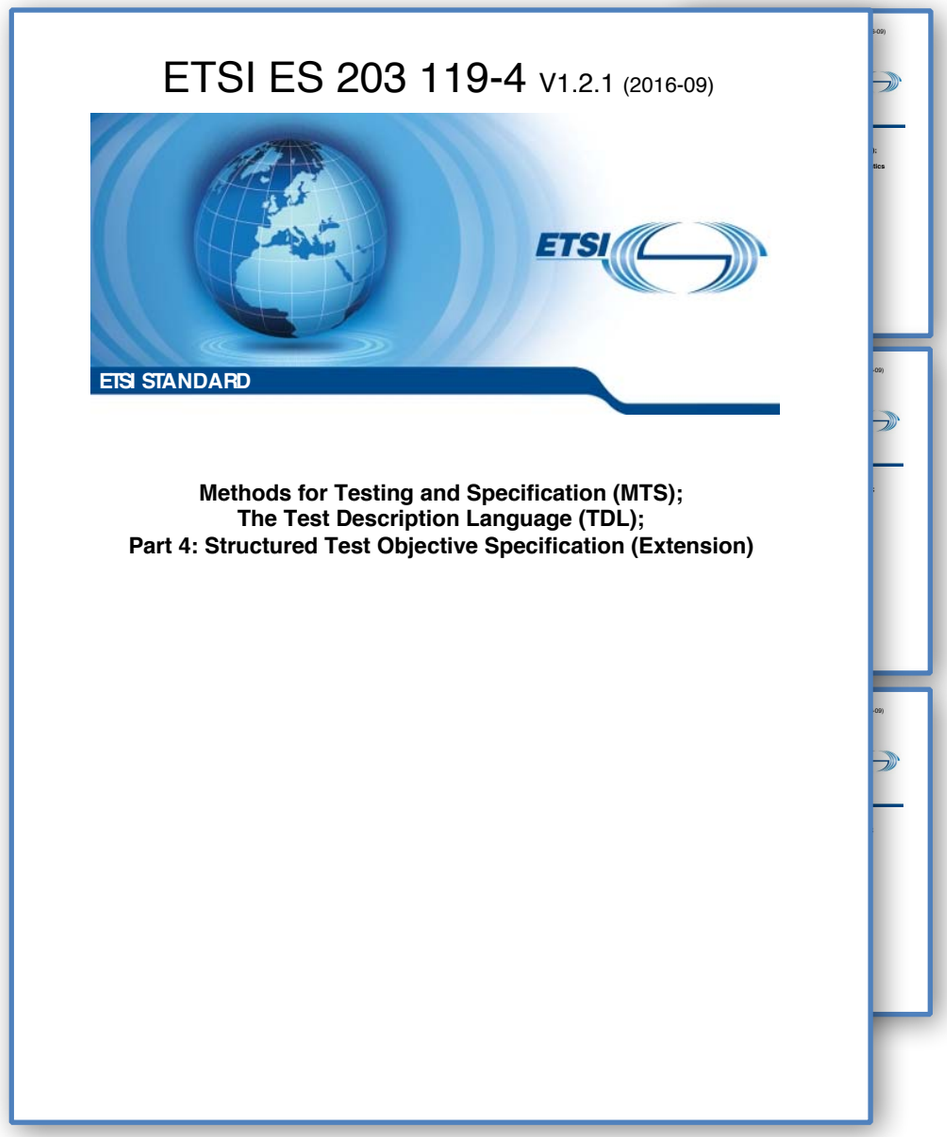
What is TDL? Part 4: TO

- Base Standard Specification
- Identification of Requirements
- Creation of ICS/IFS
- Definition of TSS
- Specification of Test Purposes
- Specification of Test Descriptions
- Specification of Test Cases
- Validation

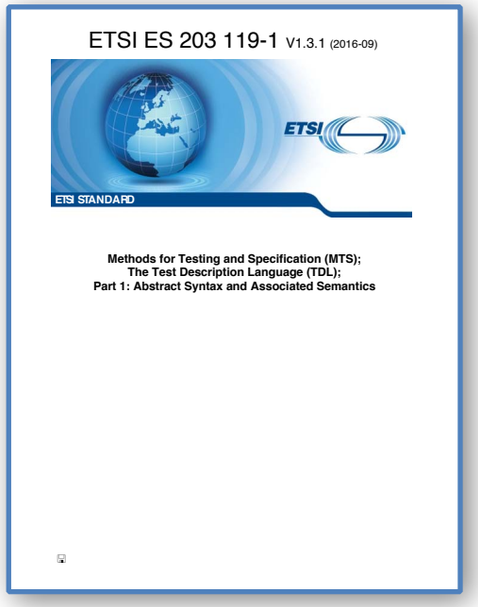


What is TDL? Part 4: TO

```
Test Purpose {
    TP Id "TP/CAM/INA/DOP/BV/02"
    Test objective "Checks that CAM message includes
                    DoorOpen information 30s after closed"
    Reference "TS 102 637-2 [1], clauses 7.1 and 7.2"
    PICS Selection PICS_PUBTRANSVEH
    Initial conditions
    with {
        the IUT entity having reached an initial_state
        and
        the IUT entity having sent a valid CAM message
            containing DoorOpen TaggedValue;
    }
    Expected behaviour
    ensure that {
        when {
            the door entity is closed
        }
        then {
            the IUT entity sends a new CAM message
                containing DoorOpen TaggedValue;
        }
    }
}
```

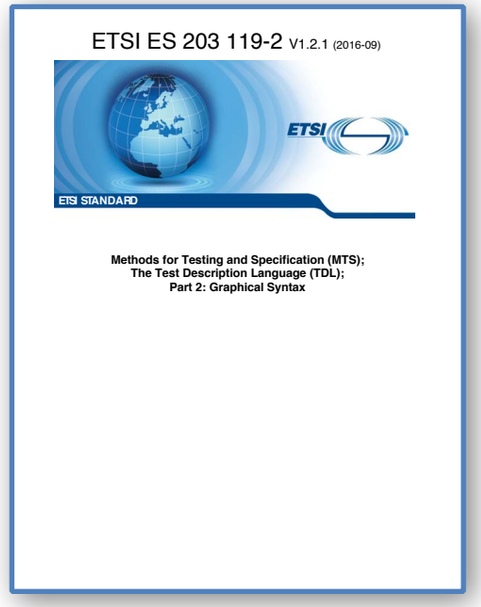


What is TDL?


ETSI ES 203 119-1 V1.3.1 (2016-09)

ETSI STANDARD

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


ETSI ES 203 119-2 V1.2.1 (2016-09)

ETSI STANDARD

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


ETSI ES 203 119-3 V1.2.1 (2016-09)

ETSI STANDARD

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

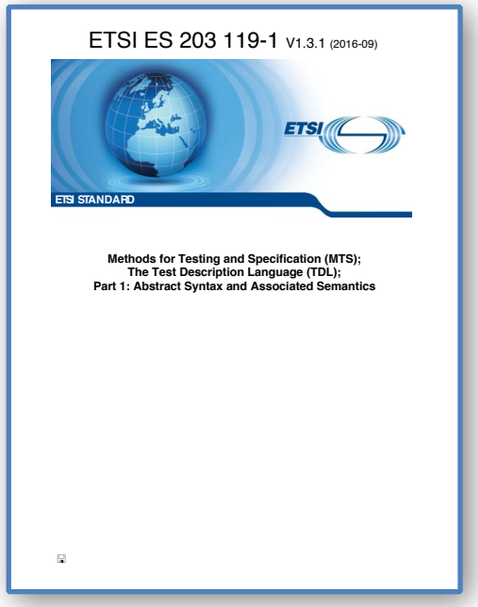

ETSI ES 203 119-4 V1.2.1 (2016-09)

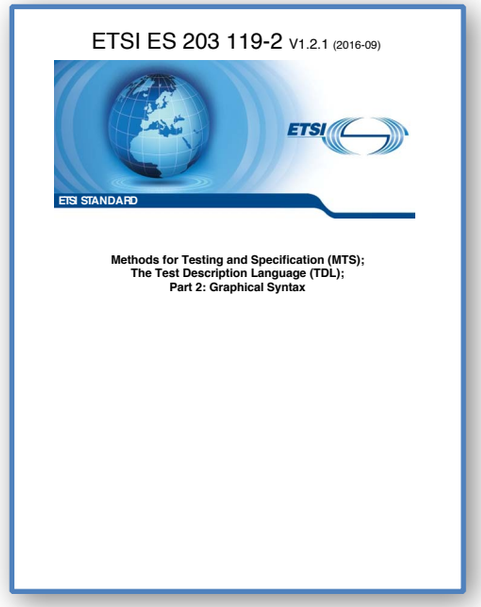
ETSI STANDARD

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

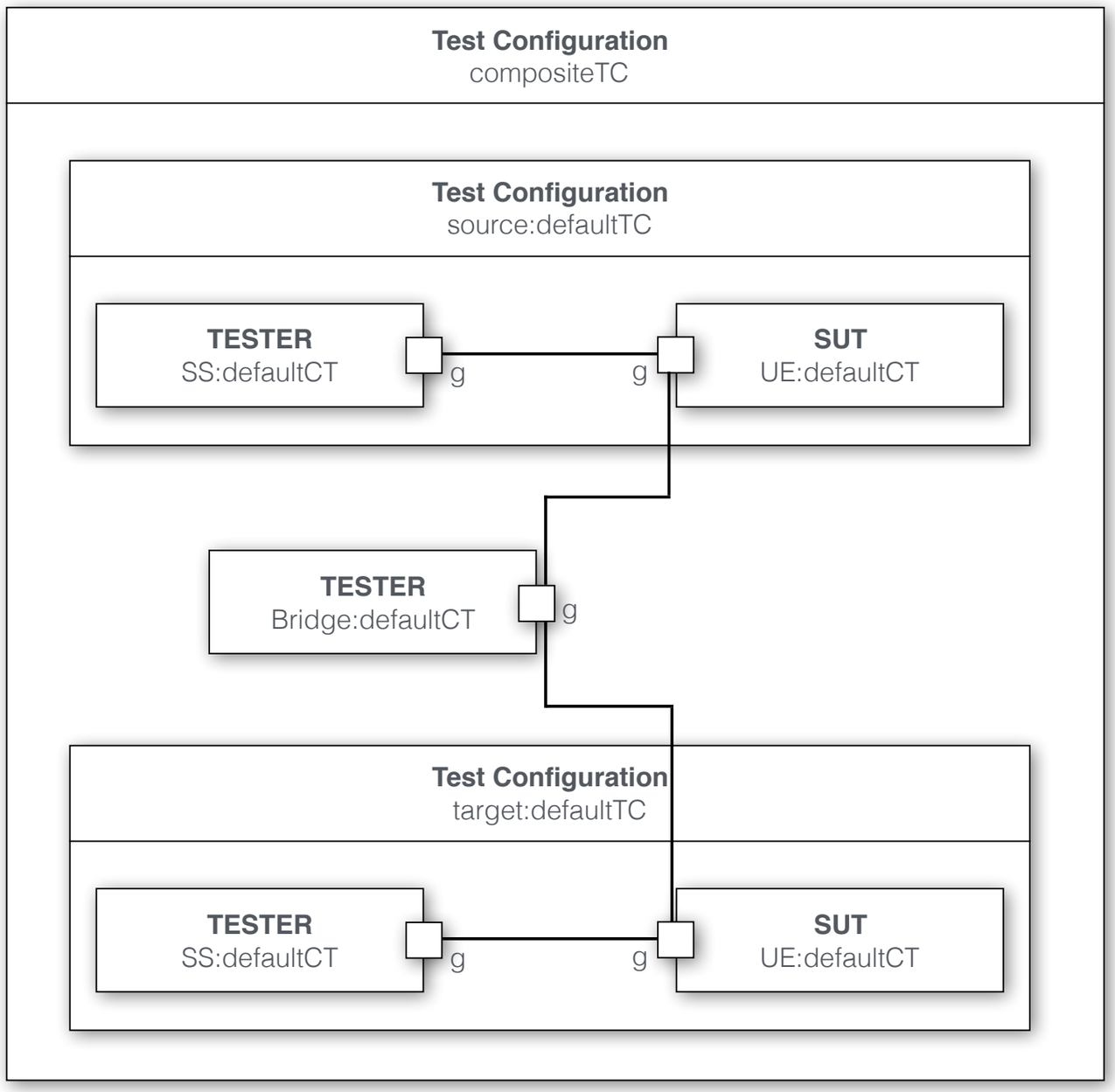
What is new TDL?

- Part 1: New features
  - collections, procedures
  - local ordering option
- Part 5: UML Profile
  - previously included in Part 1
- Part 6: Mapping to TTCN-3
  - coming up next
- Part 7: Extended Configurations
  - instantiate existing configurations
  - reuse and extend





What is new TDL?




ETSI ES 203 119-1 V1.3.1 (2016-09)

ETSI STANDARD

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


ETSI ES 203 119-2 V1.2.1 (2016-09)

ETSI STANDARD

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


ETSI ES 203 119-3 V1.2.1 (2016-09)

ETSI STANDARD

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

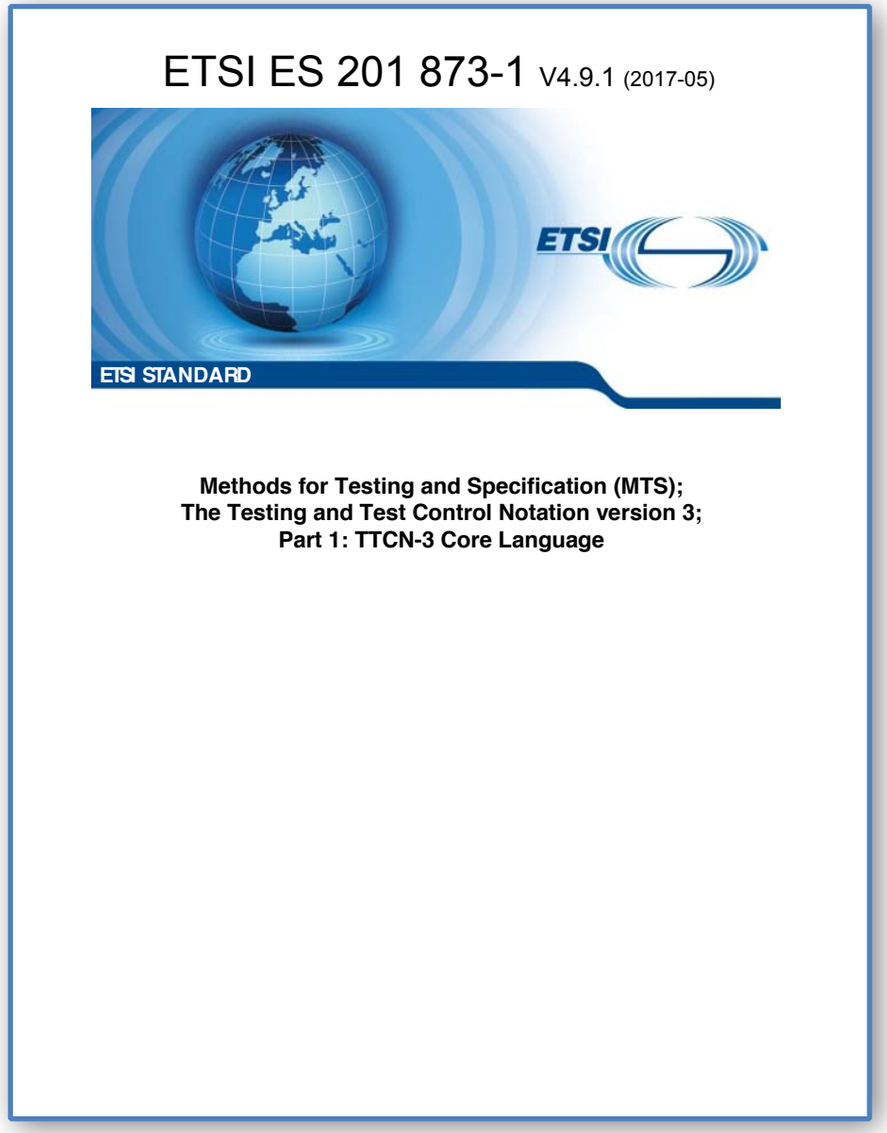

ETSI ES 203 119-4 V1.2.1 (2016-09)

ETSI STANDARD

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

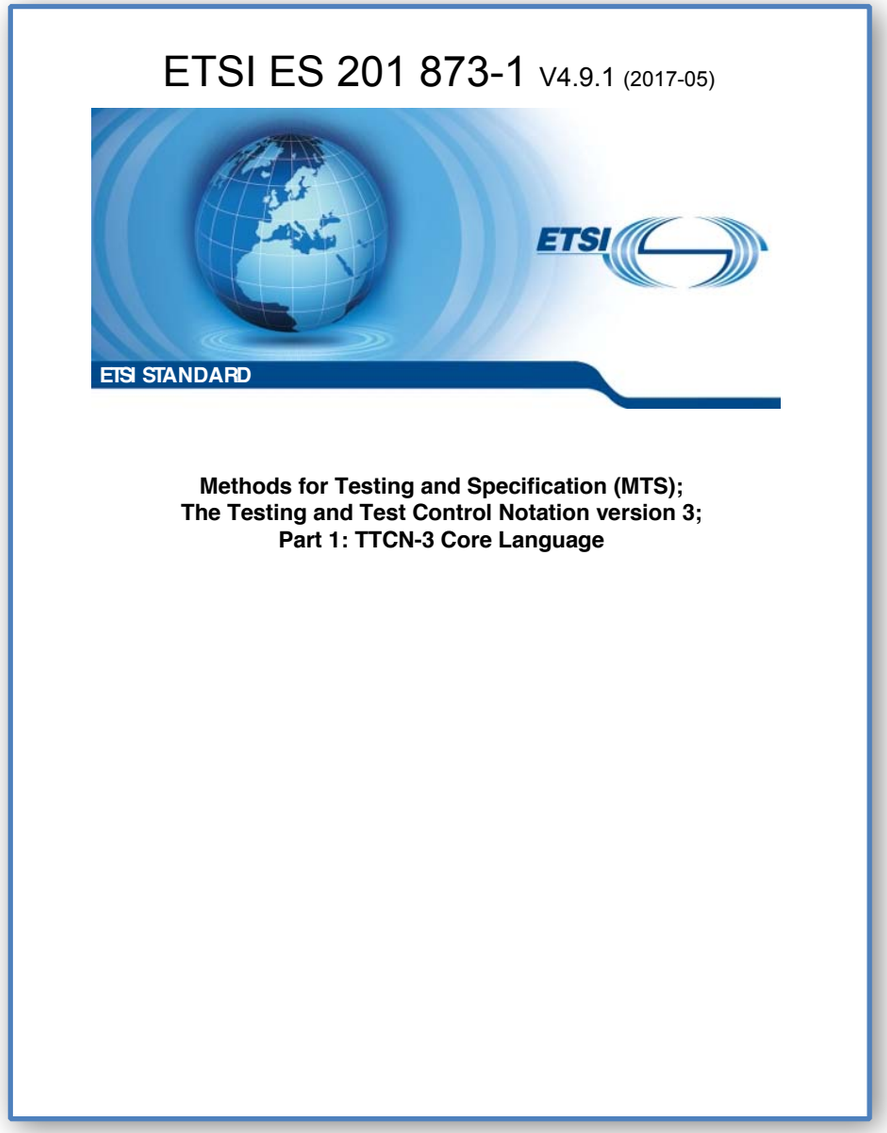
What is TTCN-3?

- Testing and Test Control Notation
  - Specification and implementation of all kinds of black-box tests
  - Platform independent link between modelling and execution
  - Component-based approach
- Standardised at ETSI by TC MTS
  - 15+ years of maintenance work



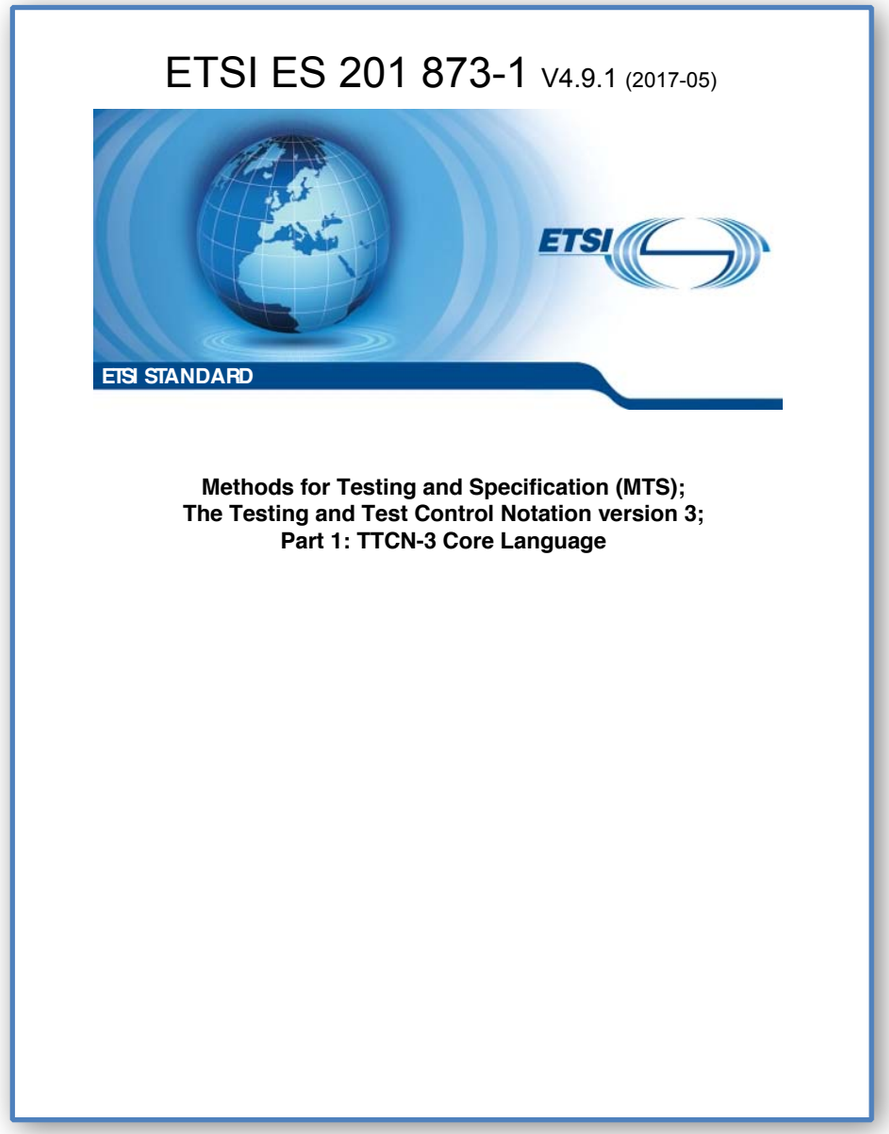
What is TTCN-3?

- Black-box tests?
  - functional, conformance, interoperability, robustness, load
  - standardisation and certification
- Used in various domains
  - telecommunications
  - automotive
  - railway
  - financial
  - medical

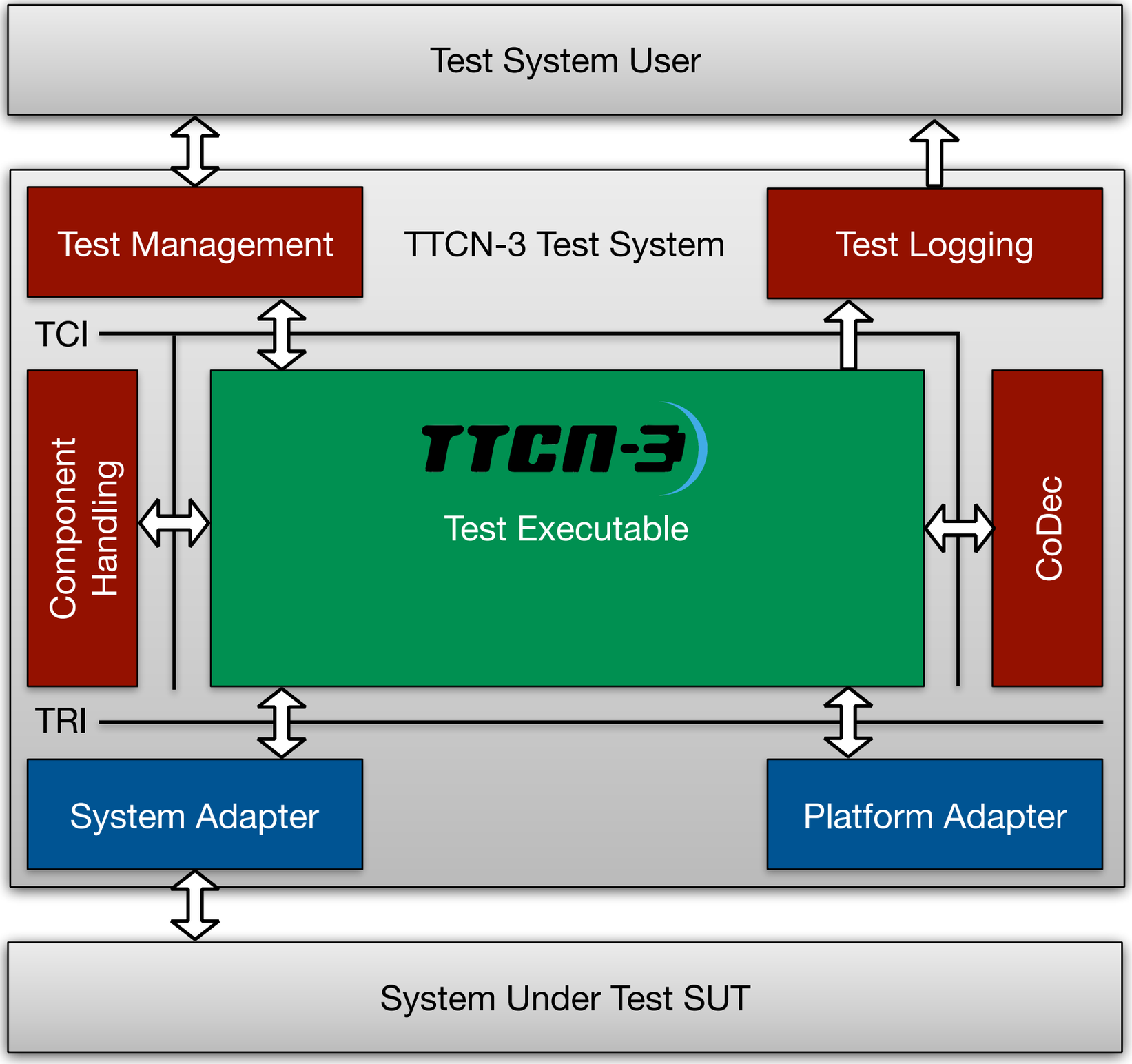


What is TTCN-3?

- Platform independent?
  - standardised core language
  - standardised interfaces
  - not tied to application or interface
  - not tied to tooling
- Requirements
  - test suite
  - compiler / interpreter
  - adapters and codecs
  - execution environment



TCI - TTCN-3 Control Interface
TRI - TTCN-3 Runtime Interface

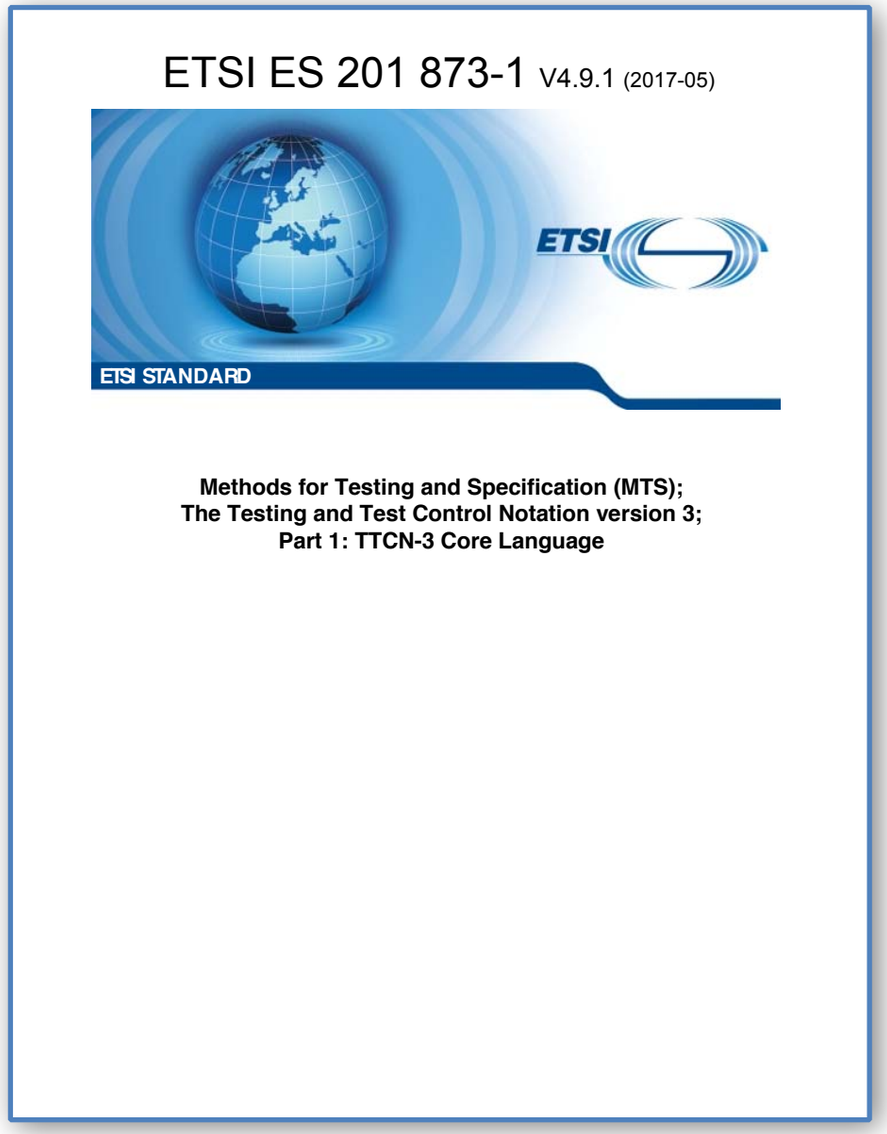


Domain	Real-Time and Performance ES 202 782	Continuous Signals ES 202 786		
Extension	Advanced Parameterisation ES 202 784	Advanced Matching ES 203 022	Static Configurations ES 202 781	Behaviour Types ES 202 785
Presentation	Tabular ES 201 873-2	Graphical ES 201 873-3	Documentation Tags ES 201 873-10	
Core	Core Language ES 201 873-1	Semantics ES 201 873-4		
Execution	Control Interfaces ES 201 873-6	Runtime Interfaces ES 201 873-5	Extended Runtime Interfaces ES 202 789	
Mappings	ASN.1 ES 201 873-7	IDL ES 201 873-8	XML Schema ES 201 873-9	JSON ES 201 873-11

Adapted from Grabowski et al., 2014

What is TTCN-3?

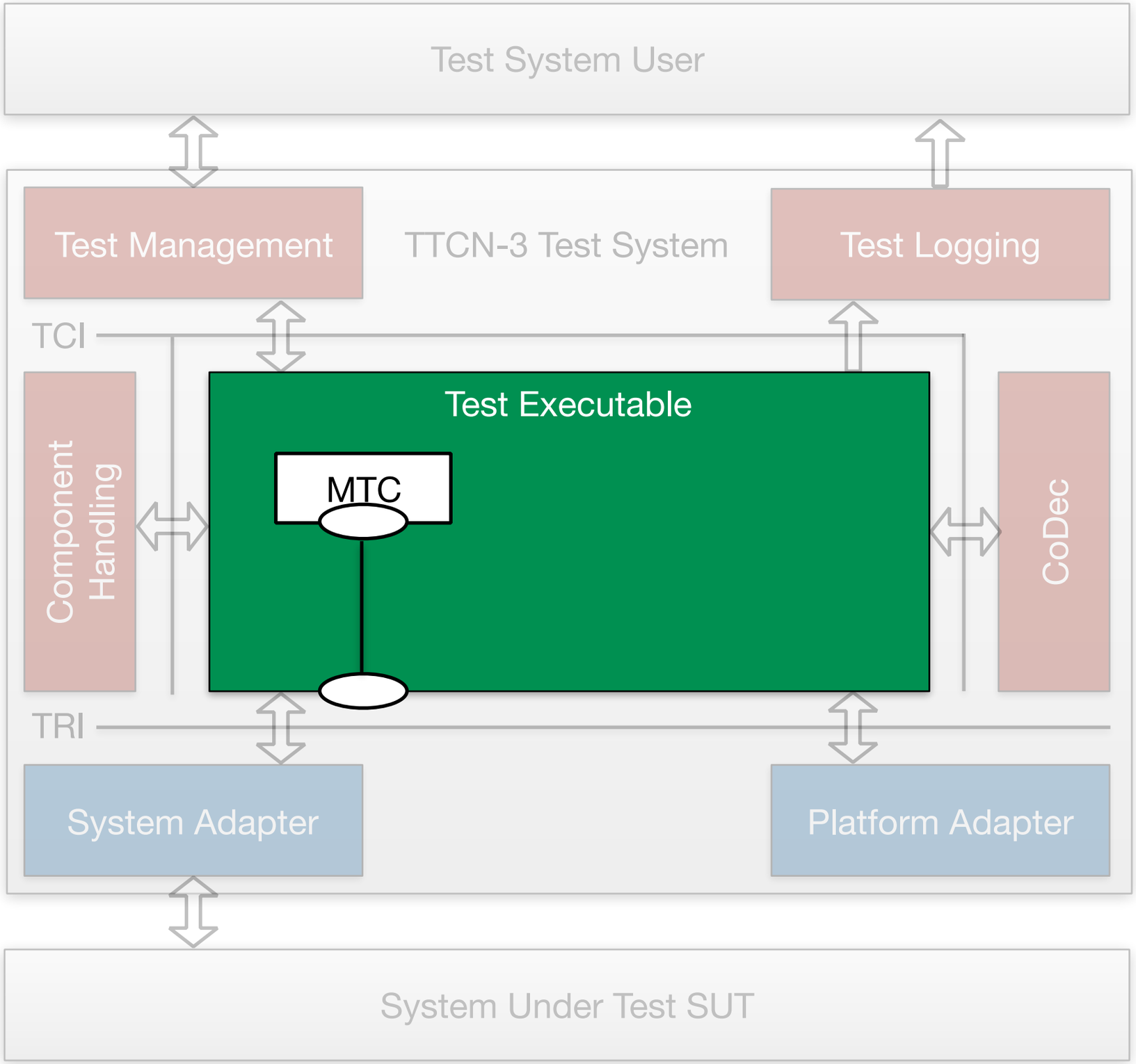
- Component-based?
  - describe behaviour of test system
  - one or more test components
  - interconnected among each other
  - mapped to unified SUT interface


ETSI ES 201 873-1 V4.9.1 (2017-05)

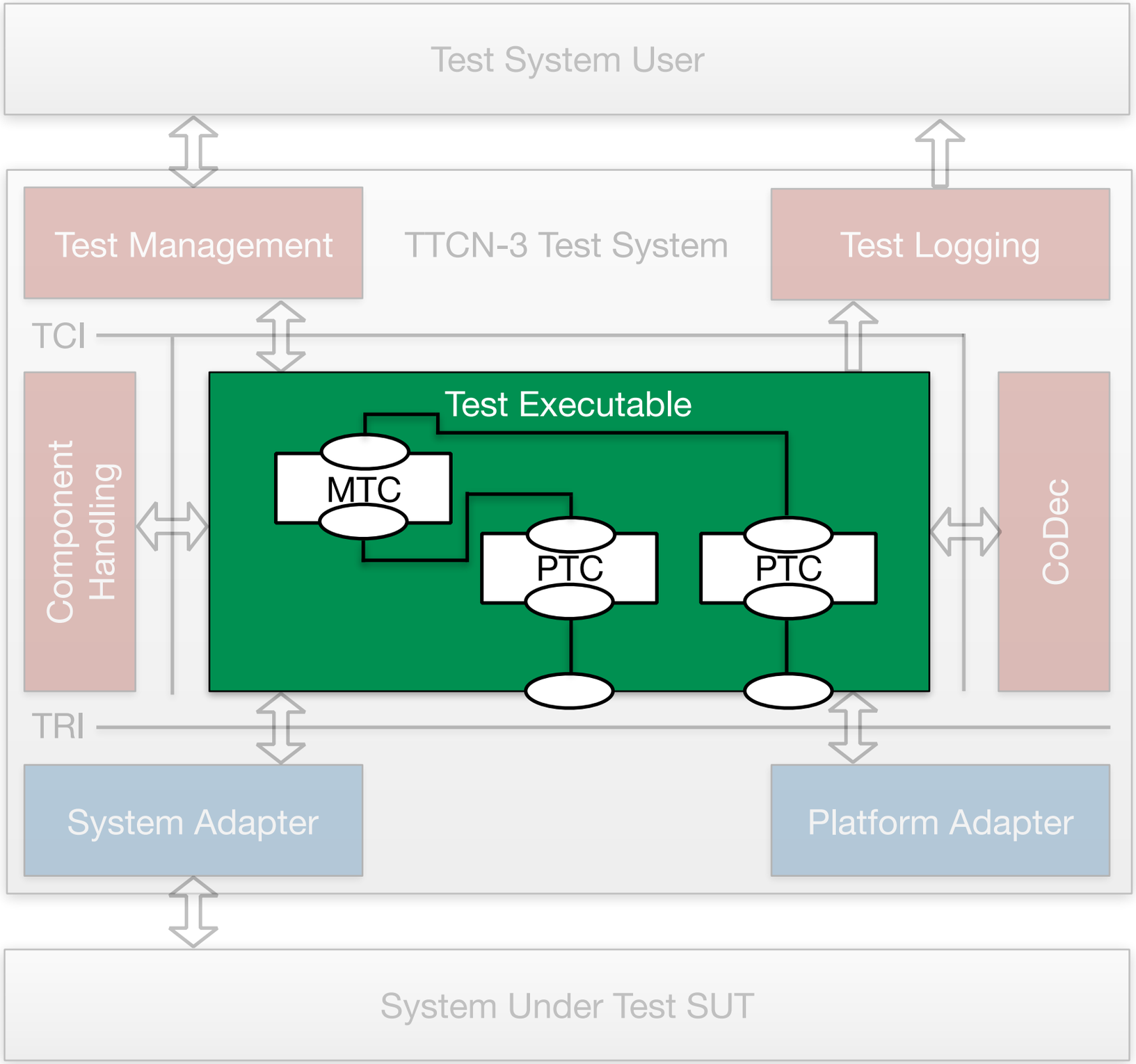
ETSI STANDARD

Methods for Testing and Specification (MTS);
The Testing and Test Control Notation version 3;
Part 1: TTCN-3 Core Language

MTC - Main Test Component
PTC - Parallel Test Component

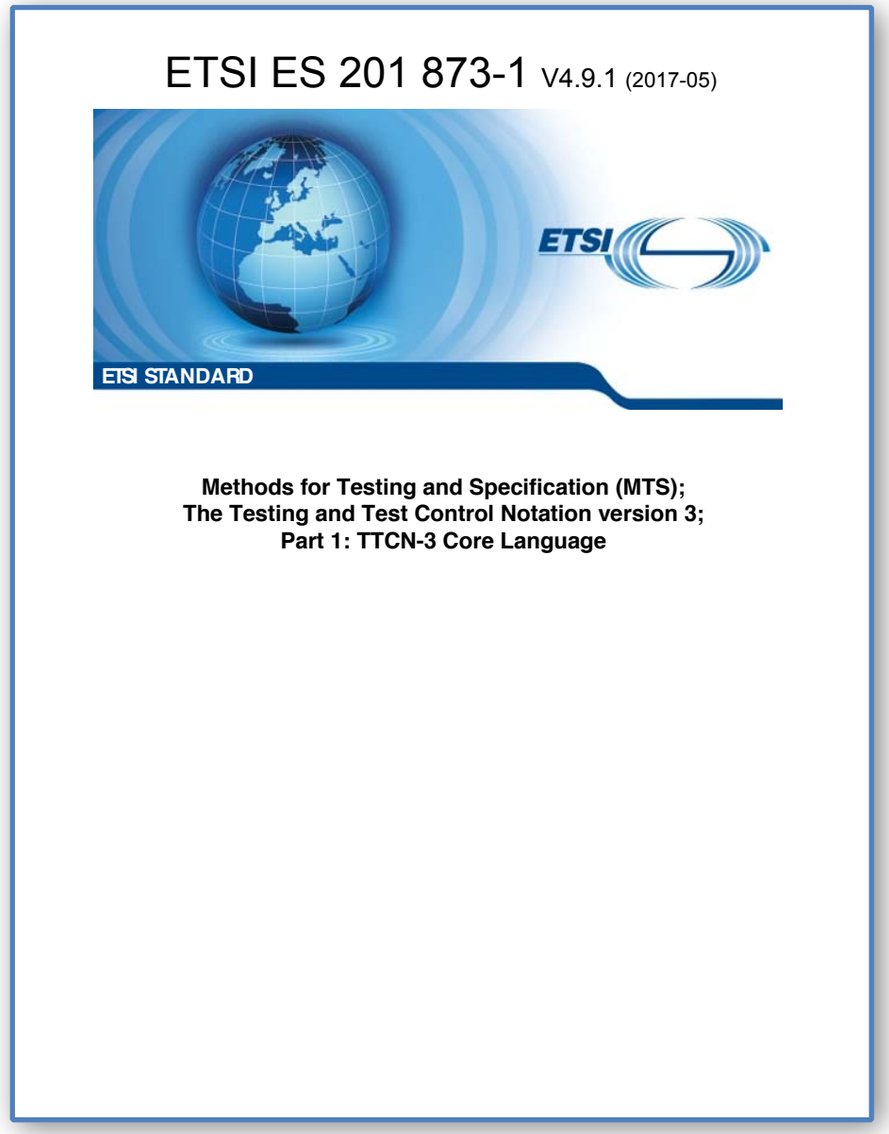


MTC - Main Test Component
PTC - Parallel Test Component



What is TTCN-3?

- Test suite ingredients
  - Data
    - basic, structured, and special types
    - constants, templates, expressions
  - Configuration
    - components, ports, connections
    - dynamic management
  - Behaviour
    - test cases, functions, altsteps
    - defaults and timers
    - optional test execution control



What is TTCN-3?

```
//enumerated data type
type enumerated MSGKind {question, answer}

//structured data type
type record MSG {
    MSGKind kind,
    charstring content
}

//a question template
template MSG readyQuestion := {
    kind := question,
    content := "Ready?"
}

//a generic question template
//the content shall be provided upon use
template MSG p_Question (charstring c) := {
    kind := question,
    content := c
}

//a generic answer template
template MSG p_Answer (charstring c) := {
    kind := answer,
    content := c
}
```

```
//a generic question template
//any question is fine
template MSG anyQuestion := {
    kind := question,
    content := ?
}

//a generic answer template
//any content is fine
template MSG anyAnswer := {
    kind := answer,
    content := ?
}
```

What is TTCN-3?

```
//simple port
type port MSGPort message {
    inout MSG
    //may also support transmission of other types
}

//simple component
type component Client {
    timer patience;
    port MSGPort clientPort
    //may also define multiple ports, variables, timers
}

//simple test case
testcase TC_isServiceReady() runs on Client {
   clientPort.send(p_Question("Ready?"));
   alt {
        [] clientPort.receive(p_Answer("Yes!")) {
            setverdict(pass);
        }
        [] clientPort.receive(p_Answer("No!")) {
            setverdict(fail);
        }
   }
}
```

What is TTCN-3?

```
//simple timed test case
testcase TC_isTimedServiceReady() runs on Client {
    clientPort.send(p_Question("Ready?"));
    patience.start(10.0);
    alt {
        [] clientPort.receive(p_Answer("Yes!")) {
            setverdict(pass);
        }
        [] clientPort.receive(p_Answer("No!")) {
            setverdict(fail);
        }
        [] patience.timeout {
            setverdict(fail);
        }
    }
    patience.stop;
}
```

What is TTCN-3?

```
//simple timed test case for nosy service
testcase TC_isTimedNosyServiceReady() runs on Client {
    clientPort.send(p_Question("Ready?"));
    patience.start(10.0);
    alt {
        [] clientPort.receive(p_Answer("Yes!")) {
            setverdict(pass);
        }
        [] clientPort.receive(p_Answer("No!")) {
            setverdict(fail);
        }
        [] clientPort.receive(anyQuestion) {
            clientPort.send(p_Answer("Yes!"))
            repeat;
        }
        [] patience.timeout {
            setverdict(fail);
        }
    }
    patience.stop;
}
```

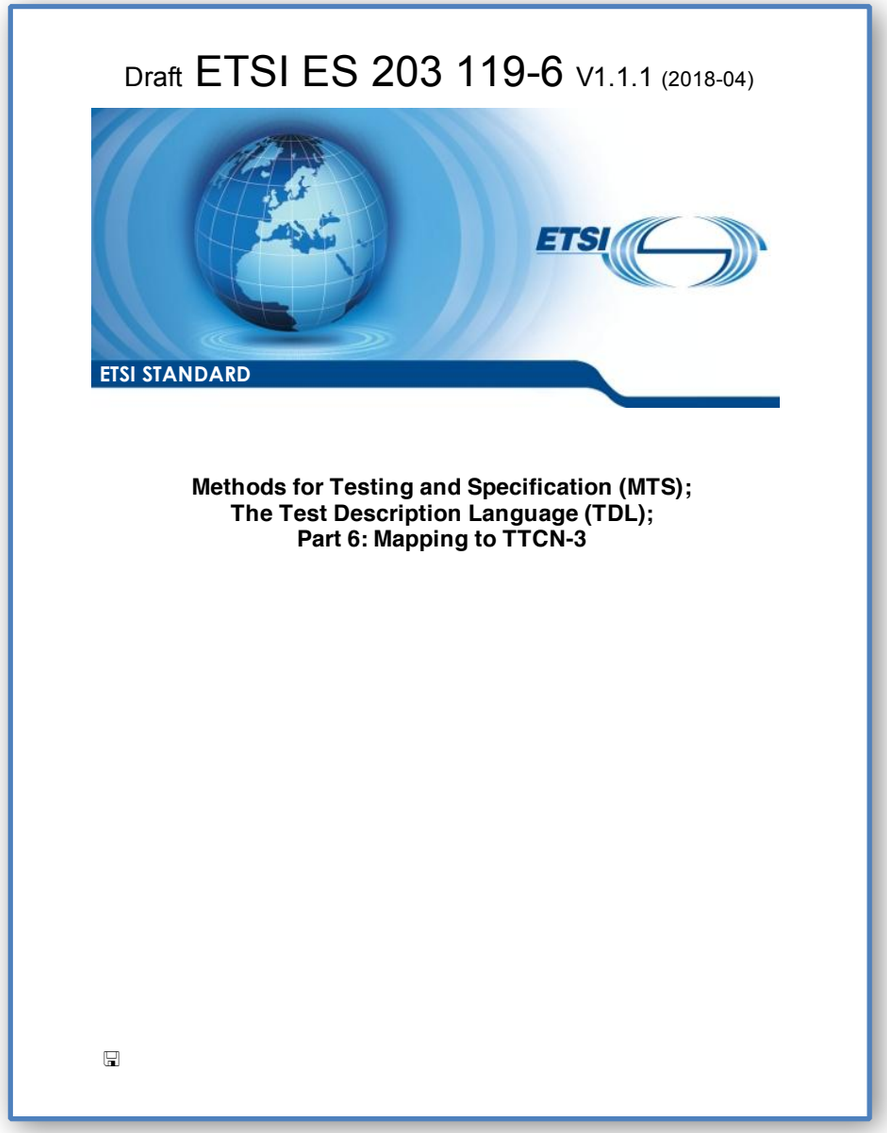
What is TTCN-3?

```
//distributed test case
testcase TC_distributed() runs on Client
  system Service {
    //create components
    var Client client1 := Client.create;
    var Client client2 := Client.create;
    //map / connect components
    map(system:servicePort, client1:clientPort);
    map(system:servicePort, client2:clientPort);

    //start initiate behaviour of components
    client1.start(f_isReady());
    client2.start(f_isReady());

    //wait for components to complete their execution
    all component.done;
}
```

```
//handle timeouts and incoming questions
 altstep impatientYesMan() runs on Client {
     [] clientPort.receive(p_Question(?)) {
         clientPort.send(p_Answer("Yes!"))
         repeat;
     }
     [] patience.timeout {
         setverdict(fail);
     }
 }

//reusable behaviour
//can be executed multiple times
function f_isReady() runs on Client {
    clientPort.send(p_Question("Ready?"));
    patience.start(10.0);
    activate(impatientYesMan());
    alt {
        [] clientPort.receive(p_Answer("Yes!")) {
            setverdict(pass);
        }
        [] clientPort.receive(p_Answer("No!")) {
            setverdict(fail);
        }
    }
    deactivate(impatientYesMan());
}
```

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)


Draft ETSI ES 203 119-6 V1.1.1 (2018-04)

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

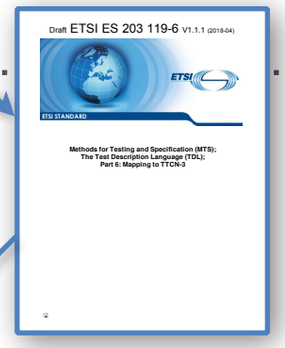
TPLan

TDL-TO

Test Generator

Test Design Level

TDL



ES 203 119-6

Test Code Generator

TTCN-3

Test Implementation Level

Adaptation

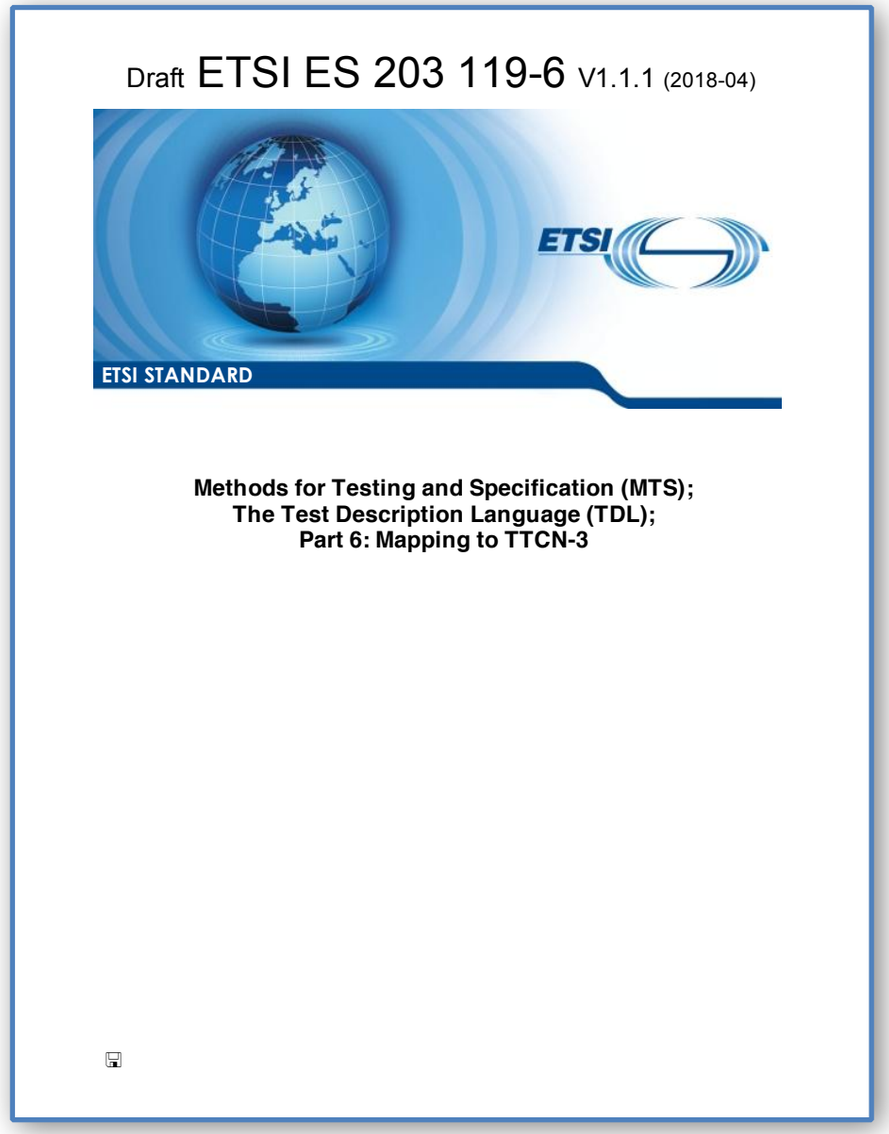
Executable Tests

Test Execution Level

5th UCAAT

Mapping TDL to TTCN-3

- Challenges
  - different levels of abstraction
  - different perspectives
  - different assumptions
    - behaviour
    - configurations
    - data
    - time

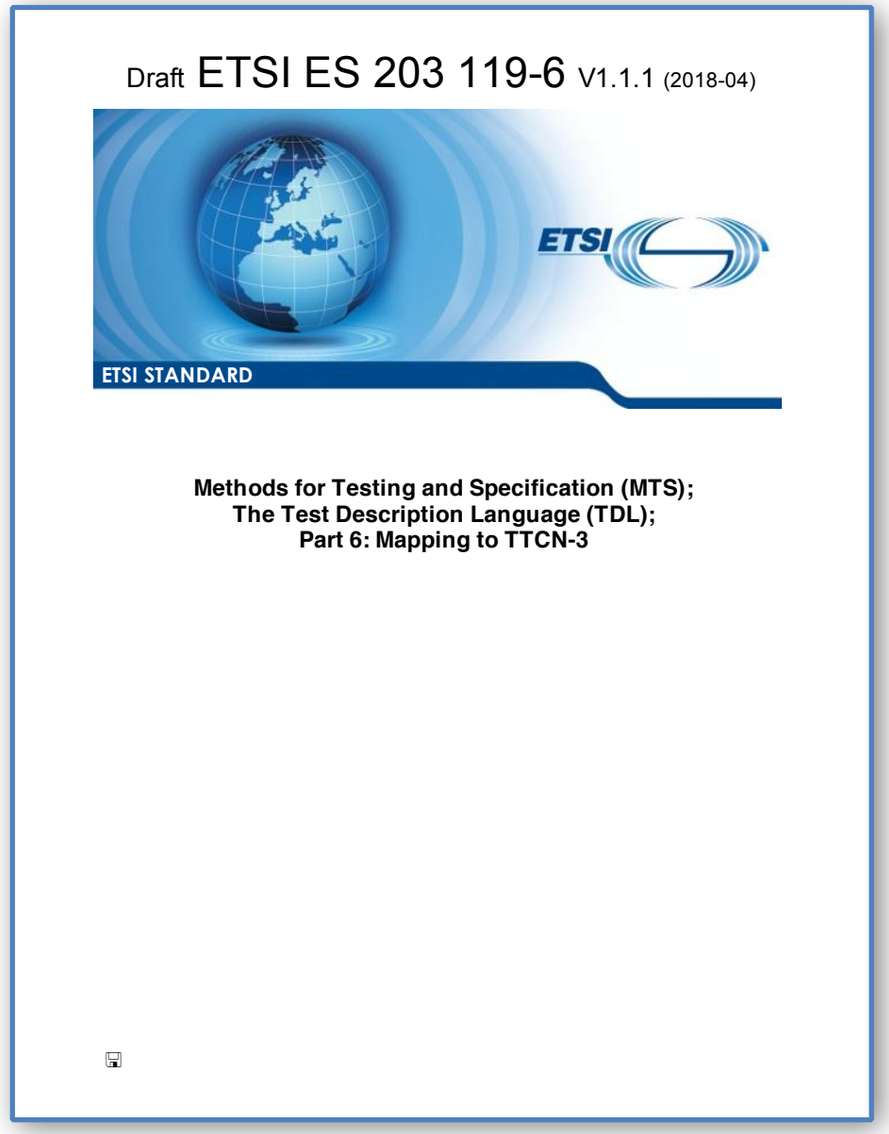

Draft ETSI ES 203 119-6 V1.1.1 (2018-04)

ETSI STANDARD

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

Mapping TDL to TTCN-3

- Levels of abstraction
  - TTCN-3
    - low - close to implementation
    - sufficient for automated execution
    - still abstracts away some details
  - TDL
    - high - test purposes (TO-extension)
    - medium - test design and description
    - low - some implementation details
    - focus on relevant parts at every level

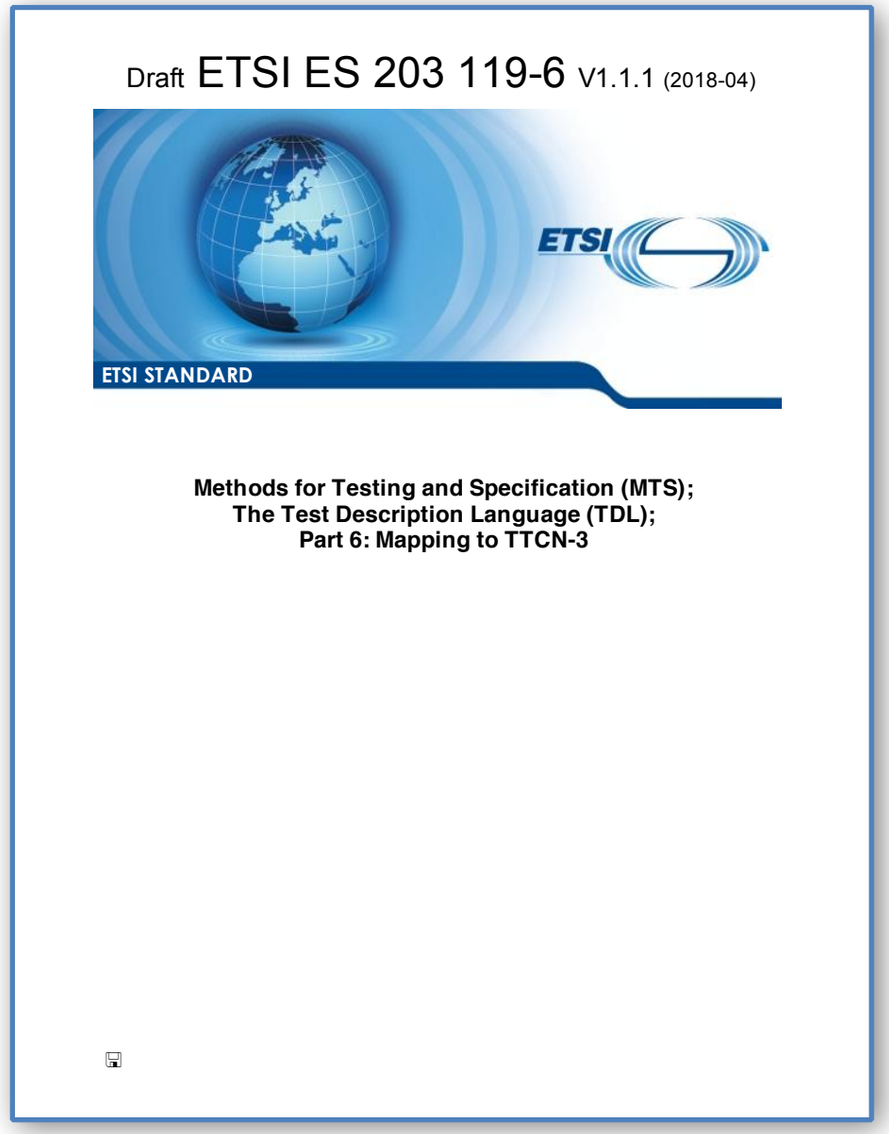

Draft ETSI ES 203 119-6 V1.1.1 (2018-04)

ETSI STANDARD

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

Mapping TDL to TTCN-3

- Perspectives
  - TTCN-3
    - test-system centric (test system view)
    - test components
    - unified SUT interface (ports)
  - TDL
    - system centric (global view)
    - tester and SUT components (roles)
    - describes entire scenario

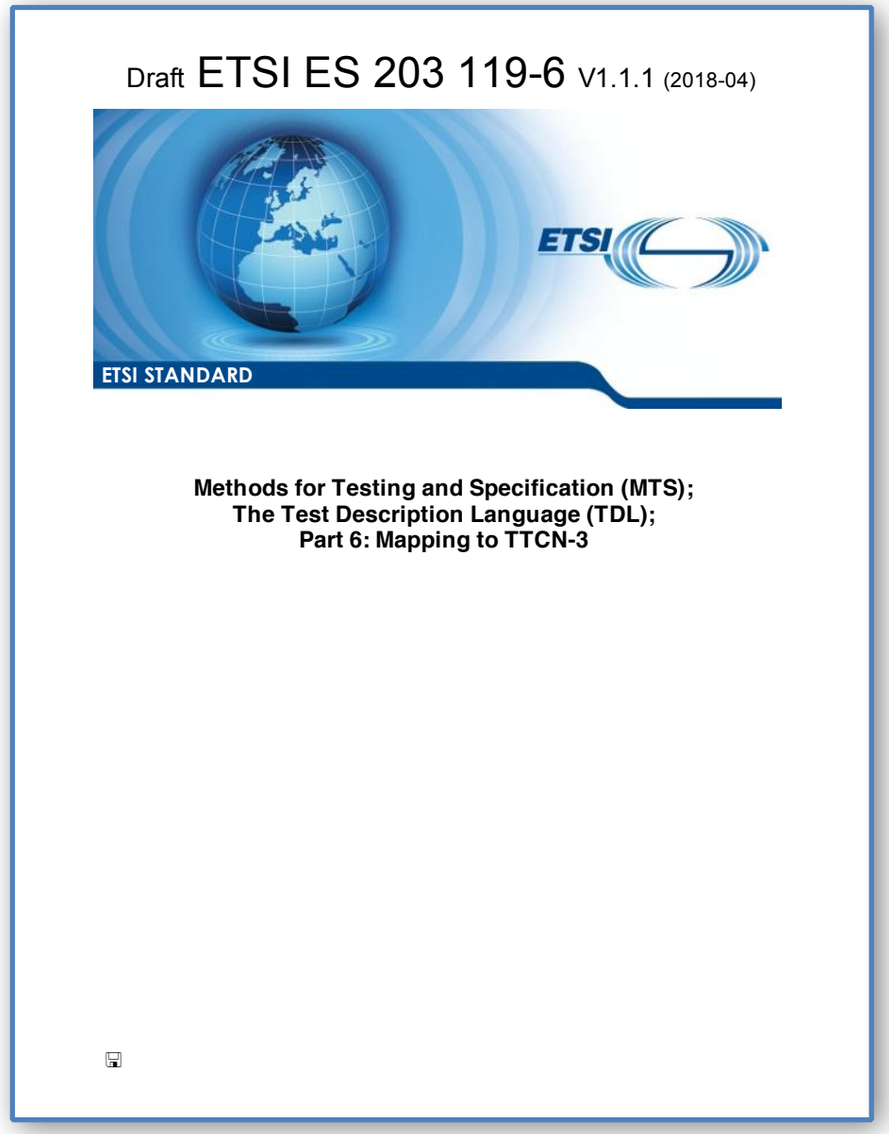

Draft ETSI ES 203 119-6 V1.1.1 (2018-04)

ETSI STANDARD

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

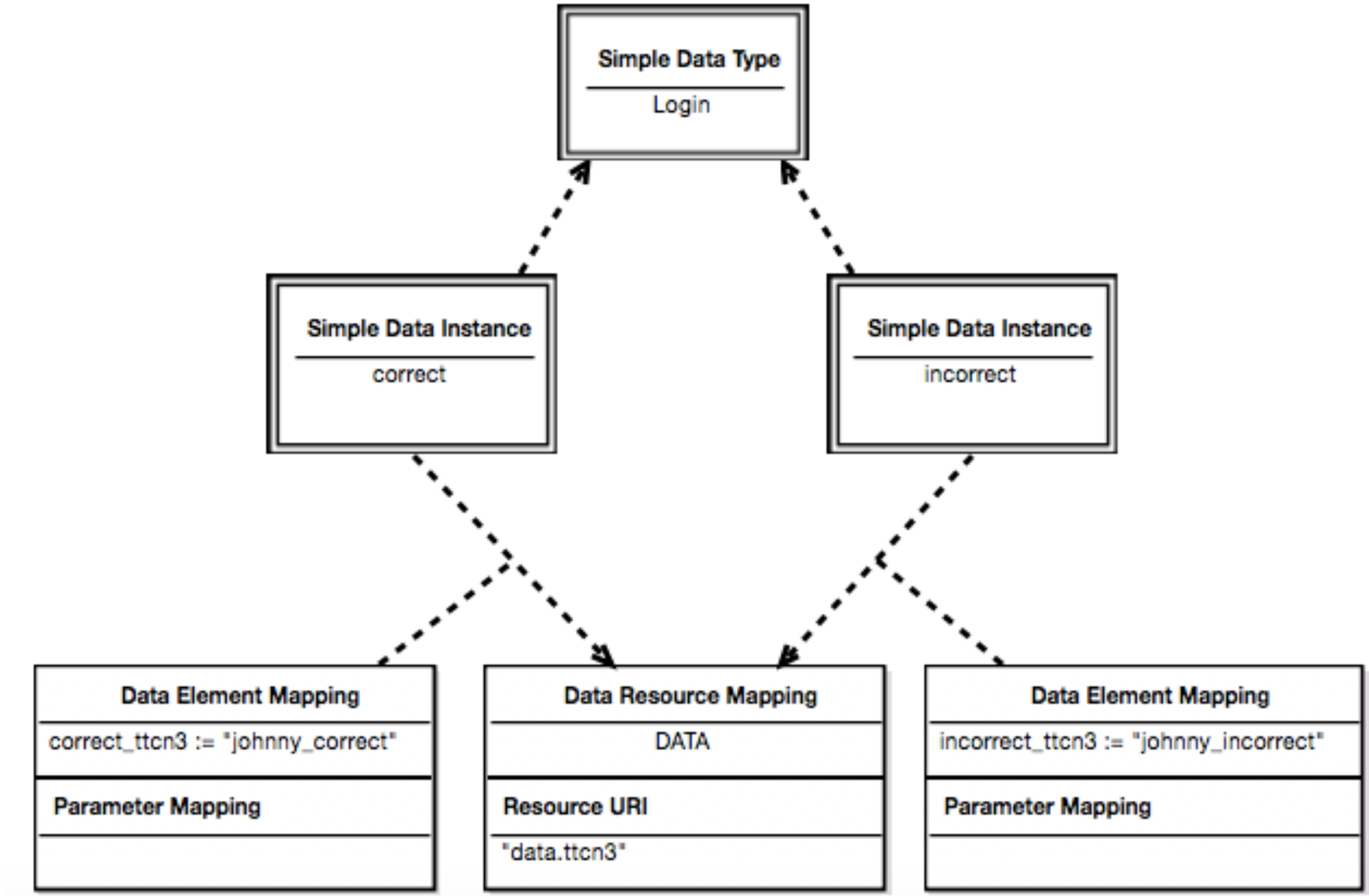
Mapping TDL to TTCN-3

- Assumptions: Data
  - TTCN-3
    - comprehensive type system
    - powerful template mechanism
    - extensive matching operators
  - TDL
    - mappable symbolic elements
    - types and instances
    - wildcards
    - limited direct data manipulation
    - nested arguments for data use


Draft ETSI ES 203 119-6 V1.1.1 (2018-04)

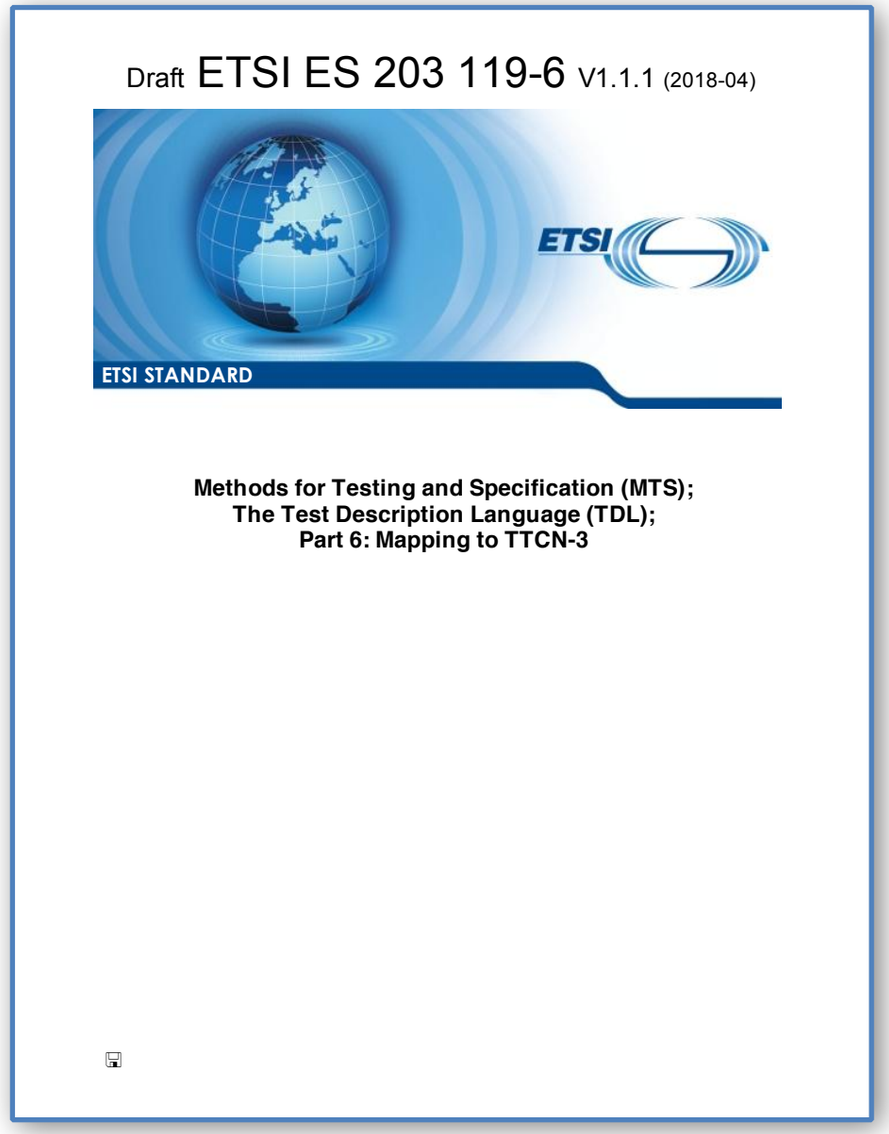
ETSI STANDARD

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


Simple Data Type
Login
Simple Data Instance
correct
Simple Data Instance
incorrect
Data Element Mapping
correct_ttcn3 := "johnny_correct"
Parameter Mapping
Data Resource Mapping
DATA
Resource URI
"data.ttcn3"
Data Element Mapping
incorrect_ttcn3 := "johnny_incorrect"
Parameter Mapping

Mapping TDL to TTCN-3

- Mapping: Data Definition
  - data mappings within TDL required
    - also for all members
    - substituted by respective targets
  - basic generation in case absent
    - charstrings, records, templates
    - functions for functions and actions
  - annotations override assumptions
    - also for variables and parameters


Draft ETSI ES 203 119-6 V1.1.1 (2018-04)

ETSI STANDARD

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

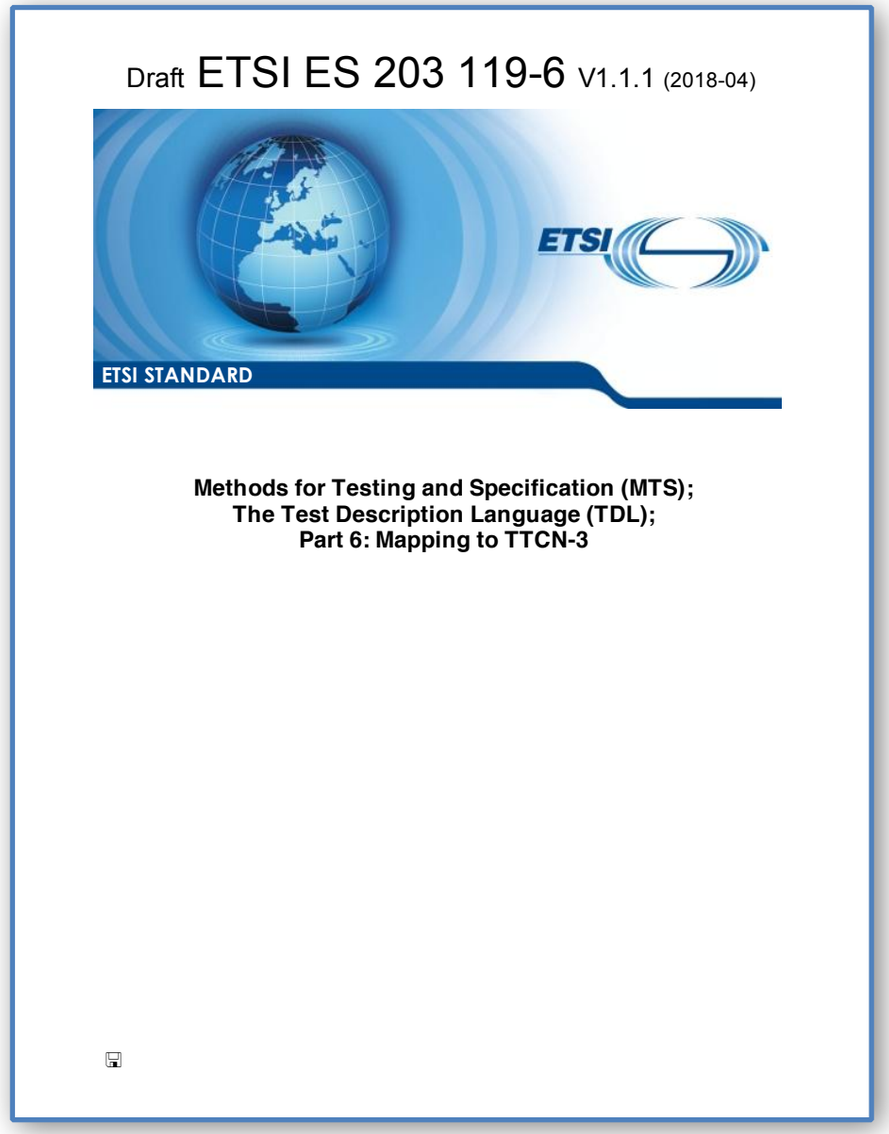
Mapping TDL to TTCN-3: Data definition

```
//data types
Type SESSIONS (id1 of type Integer, id2 of type Integer);
Type MSG (ses of type SESSIONS, content of type String);

//data instances
SESSIONS s1(id1 = 1, id2 = 2);
SESSIONS s2(id1 = 11, id2 = 22);
MSG msg1(ses = s1, content = m1);

//value data instances
SESSIONS c_s1(id1 = 1, id2 = 2) with {VALUE;};
MSG c1(ses = s1, content = c1) with {VALUE;};

Component Type ct having {
    //variables
    variable v1 of type MSG with {VALUE;};
    variable v2 of type MSG;
    gate g of type gt;
}
```

```
//data types
type record SESSIONS {
    integer id1,
    integer id2
}
type record MSG {
    SESSIONS ses,
    charstring content
}

//templates
template SESSIONS s1 := {id1:=1, id2:=2}
template SESSIONS s2 := {id1:=11, id2:=22}
template MSG msg1 := {ses := s1, content := "m1"}

//value -> constant
const SESSIONS c_s1 := {id1:=1, id2:=2}
const MSG c1 := {ses := c_s1, content := "c1"}

type component ct {
    //variables
    var MSG v1;
    var template MSG v2;
    port gt g;
}
```

Mapping TDL to TTCN-3

- Mapping: Data Use
  - treatment as values or templates
    - temporary templates
    - using valueOf
  - modification for arguments
    - inline for first level
    - iterative for nested arguments
  - special values
    - AnyValue -> ?
    - AnyValueOrOmit -> * (optional), ?
    - OmitValue -> omit


Draft ETSI ES 203 119-6 V1.1.1 (2018-04)

ETSI STANDARD

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

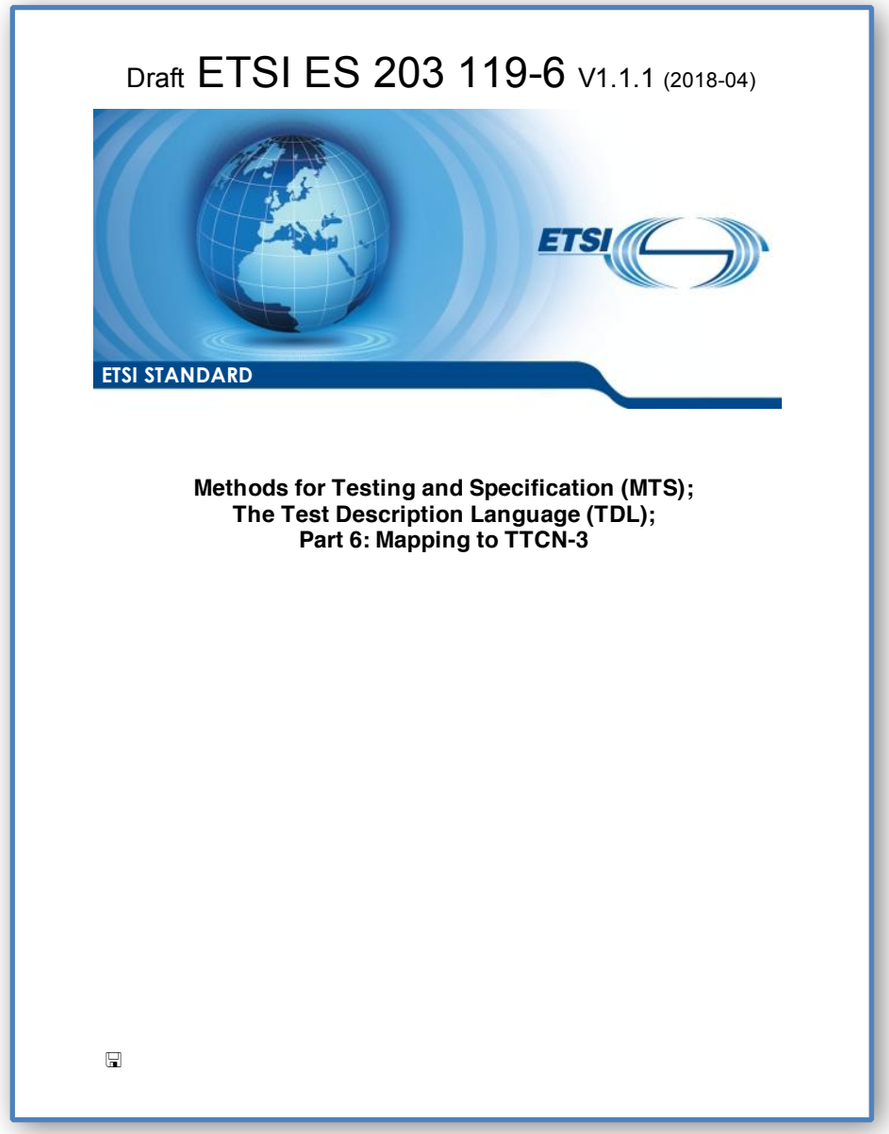
Mapping TDL to TTCN-3: Data use

```
Test Description td uses configuration tc {
    //one level arguments
    tester.g sends msg1(ses = s2) to sut.g;

    //nested arguments
    tester.g sends msg1(ses = s1(id1 = 111)) to sut.g;

    //nested arguments with value
    tester.g sends msg1(ses = c_s1(id1 = 111)) to sut.g;
}
```

```
function td_tester() runs on ct {
    //one level arguments
    g.send(modifies msg1 := {ses := s2});

    //nested arguments
    template SESSIONS t_s1 modifies s1 := {id1:=111};
    g.send(modifies msg1 := {ses := t_s1});

    //nested arguments with constants
    template SESSIONS t_c_s1 := c_s1;
    template SESSIONS t_c_s1_m modifies t_c_s1 :=
        {id1:=111};
    g.send(modifies msg1 := {ses := t_c_s1_m});
}
```

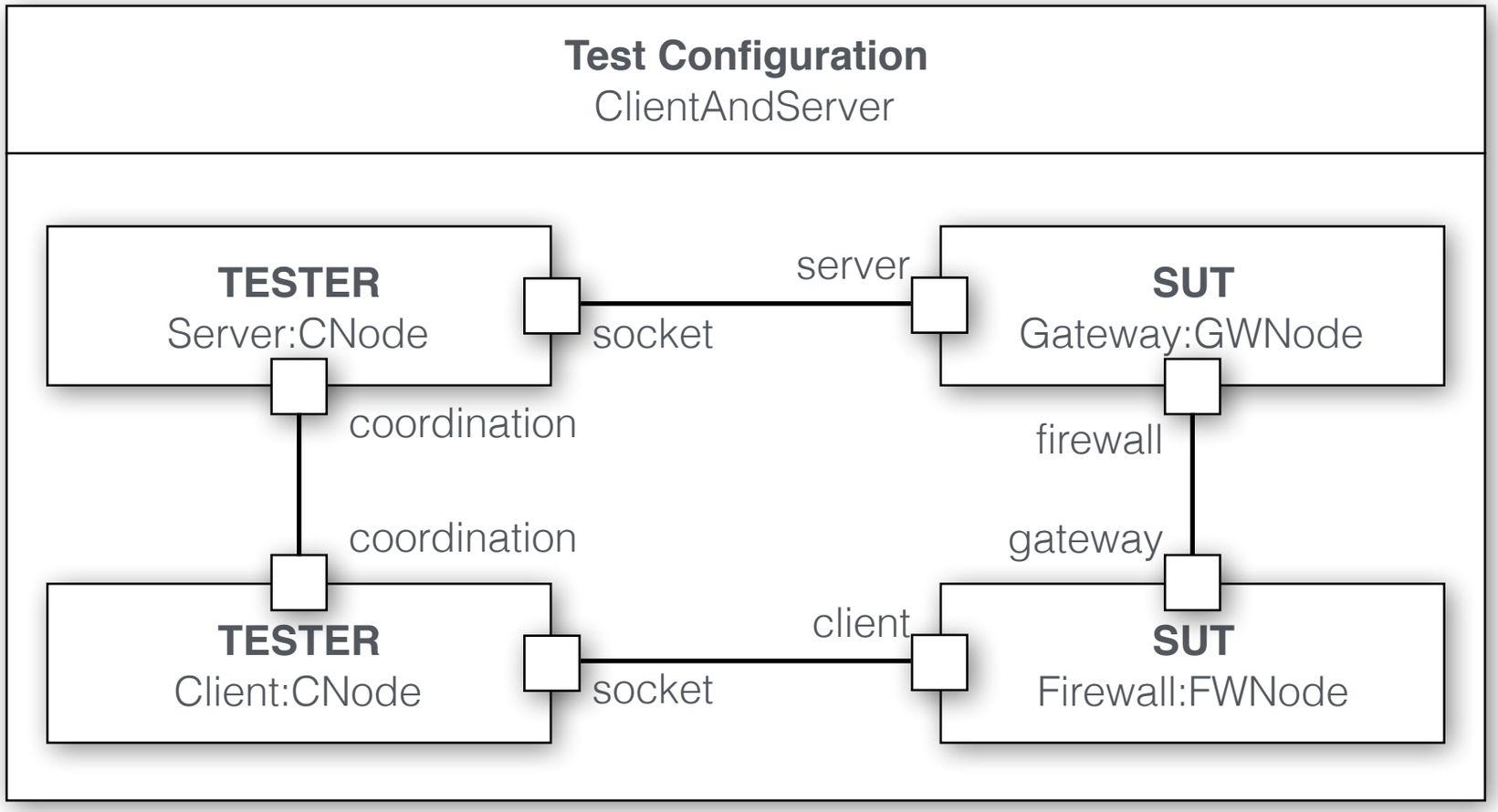
Mapping TDL to TTCN-3

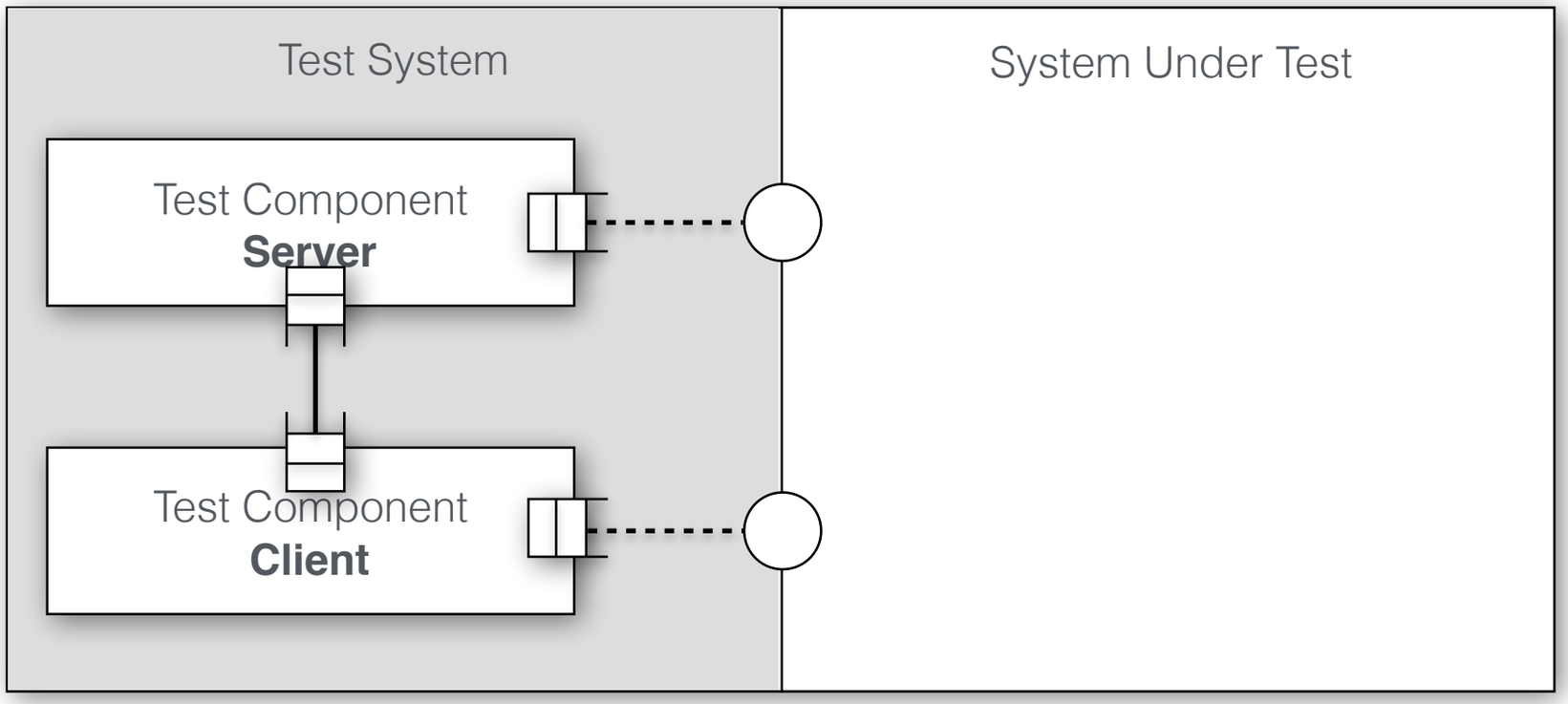
- Assumptions: Configurations
  - TTCN-3
    - dynamic instantiation / management
    - MTC, PTCs, system interface
    - mapping vs connecting ports
    - connection and mapping restrictions
  - TDL
    - static configuration defined upfront
    - holistic view, multiple SUTs


Draft ETSI ES 203 119-6 V1.1.1 (2018-04)

ETSI STANDARD

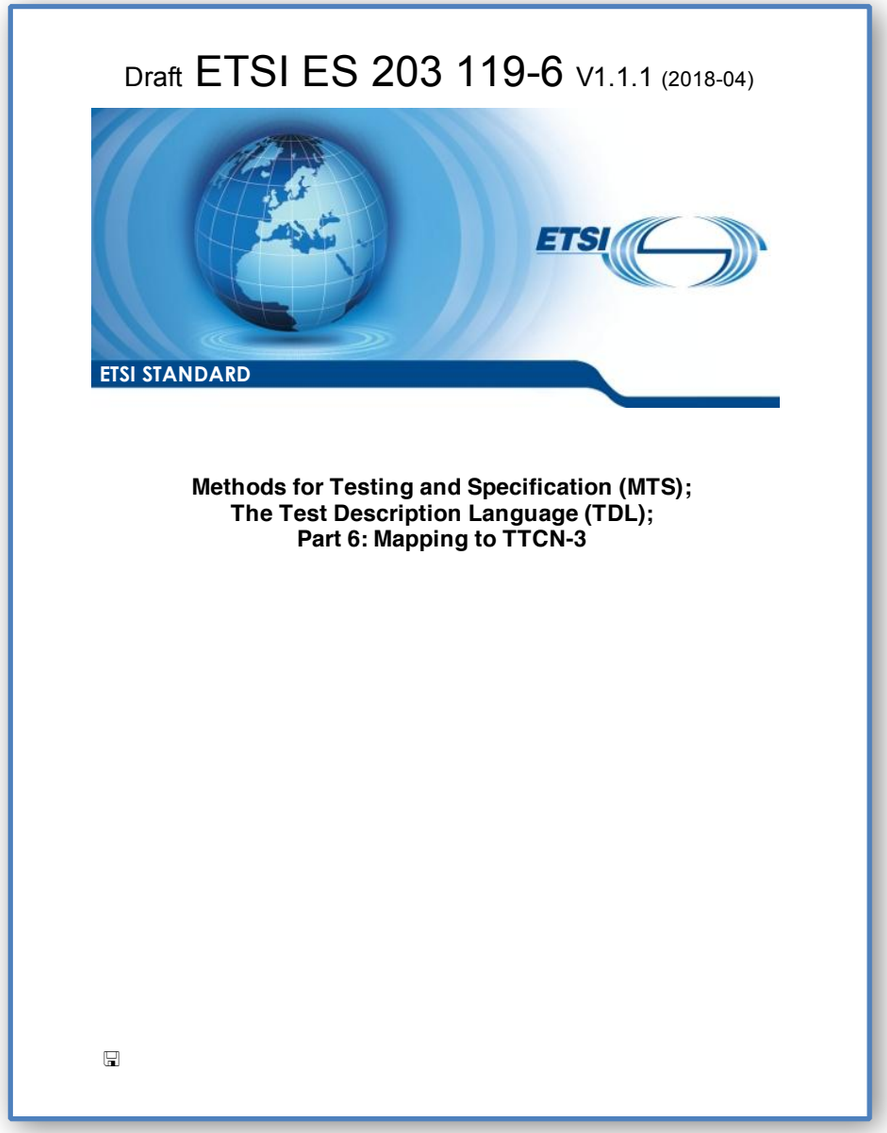
Methods for Testing and Specification (MTS);
The Test Description Language (TDL);
Part 6: Mapping to TTCN-3


Test Configuration
ClientAndServer
TESTER
Server:CNode
socket
server
SUT
Gateway:GWNode
coordination
firewall
coordination
gateway
TESTER
Client:CNode
socket
client
SUT
Firewall:FWNode


Test System
System Under Test
Test Component
Server
Test Component
Client

Mapping TDL to TTCN-3

- Mapping: Configurations
  - port types for each gate type
  - infer unified system interface
  - types for MTC, system components
  - types for tester components
  - creating components
  - map and connect ports
  - respect restrictions in TTCN-3
  - some ports may need to be cloned


Draft ETSI ES 203 119-6 V1.1.1 (2018-04)

ETSI STANDARD

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

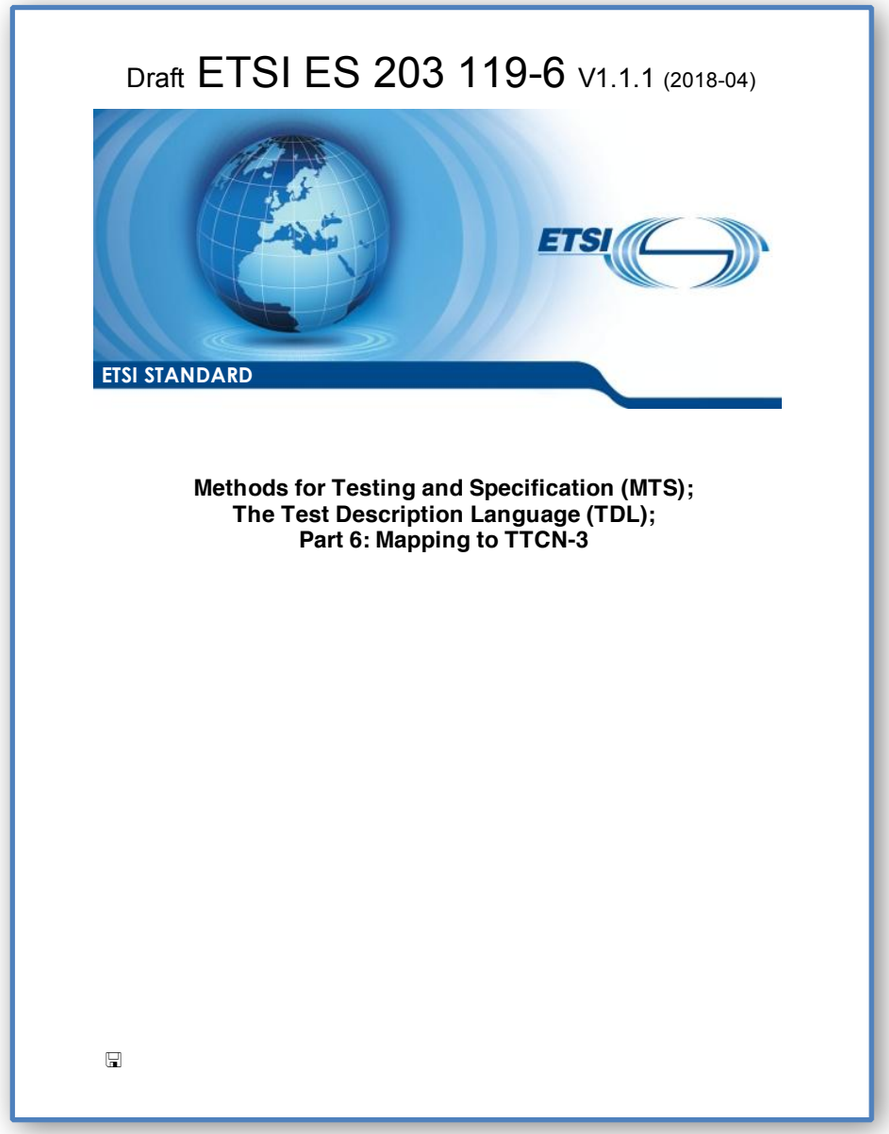
Mapping TDL to TTCN-3: Configurations

```
Gate Type defaultGT accepts
    ACK, PDU, PDCCH, C_RNTI, CONFIGURATION ;

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

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

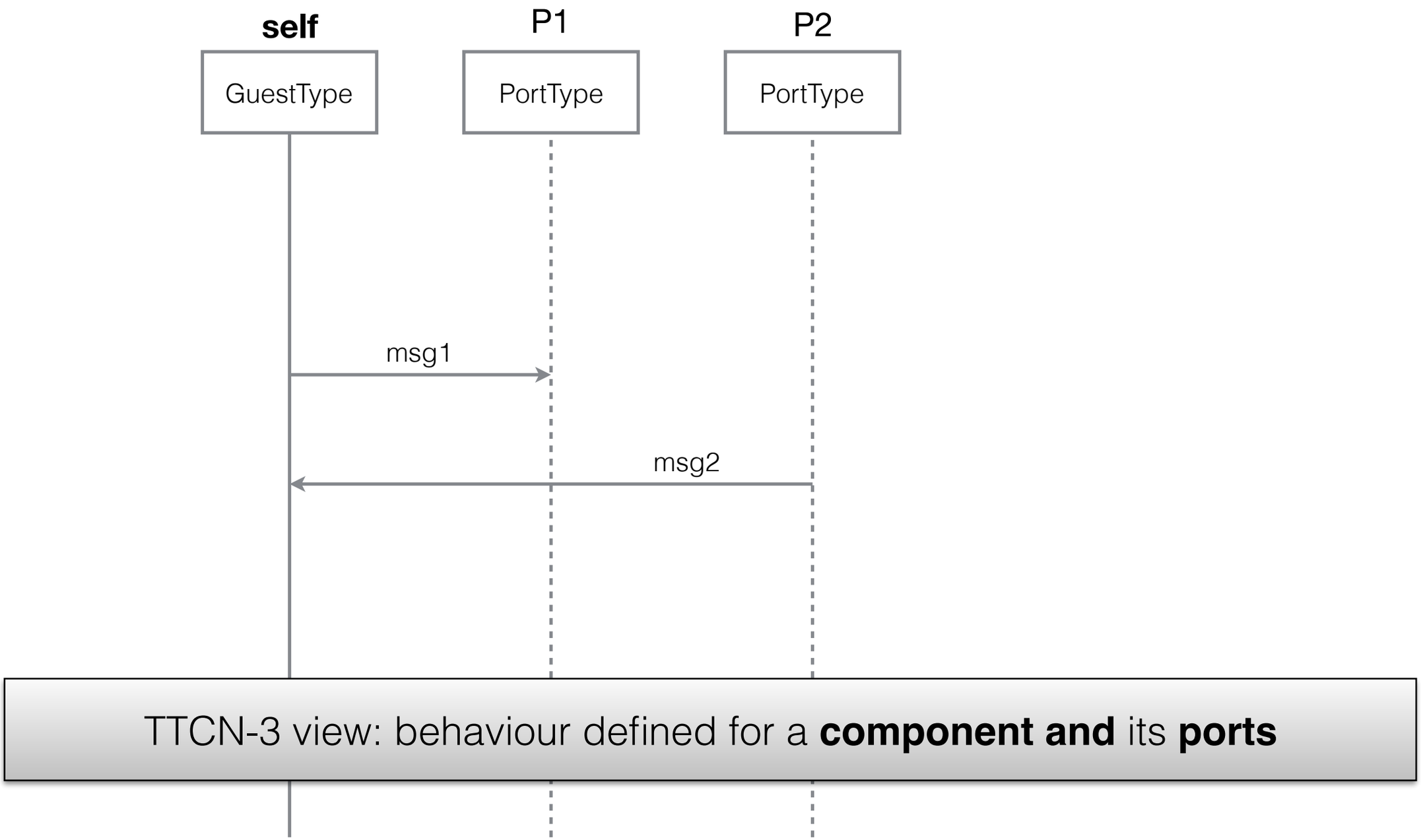
```
type port defaultGT_to_map message {
  //this is a port type for SUT-Tester connections
  inout charstring, PDCCH /* ACK, PDU, C_RNTI, CONFIGURATION ; */
}

type port defaultGT_to_connect message {
  //this is a port type for Tester-Tester connections
  inout charstring, PDCCH /* ACK, PDU, C_RNTI, CONFIGURATION ; */
}

type component MTC_CT {
  //component type for MTC
  //variable for the PTC(s) --TESTER component(s) in TDL
  var defaultCT TESTER_SS;
}

type component defaultCT {
  port defaultGT_to_map g_to_map;
  port defaultGT_to_connect g_to_connect;
}

function defaultTC() runs on MTC_CT {
  // Test Configuration defaultTC, mappings, connections
  TESTER_SS := defaultCT.create;
  map (TESTER_SS:g_to_map,system:g_to_map);
}
```

Mapping TDL to TTCN-3

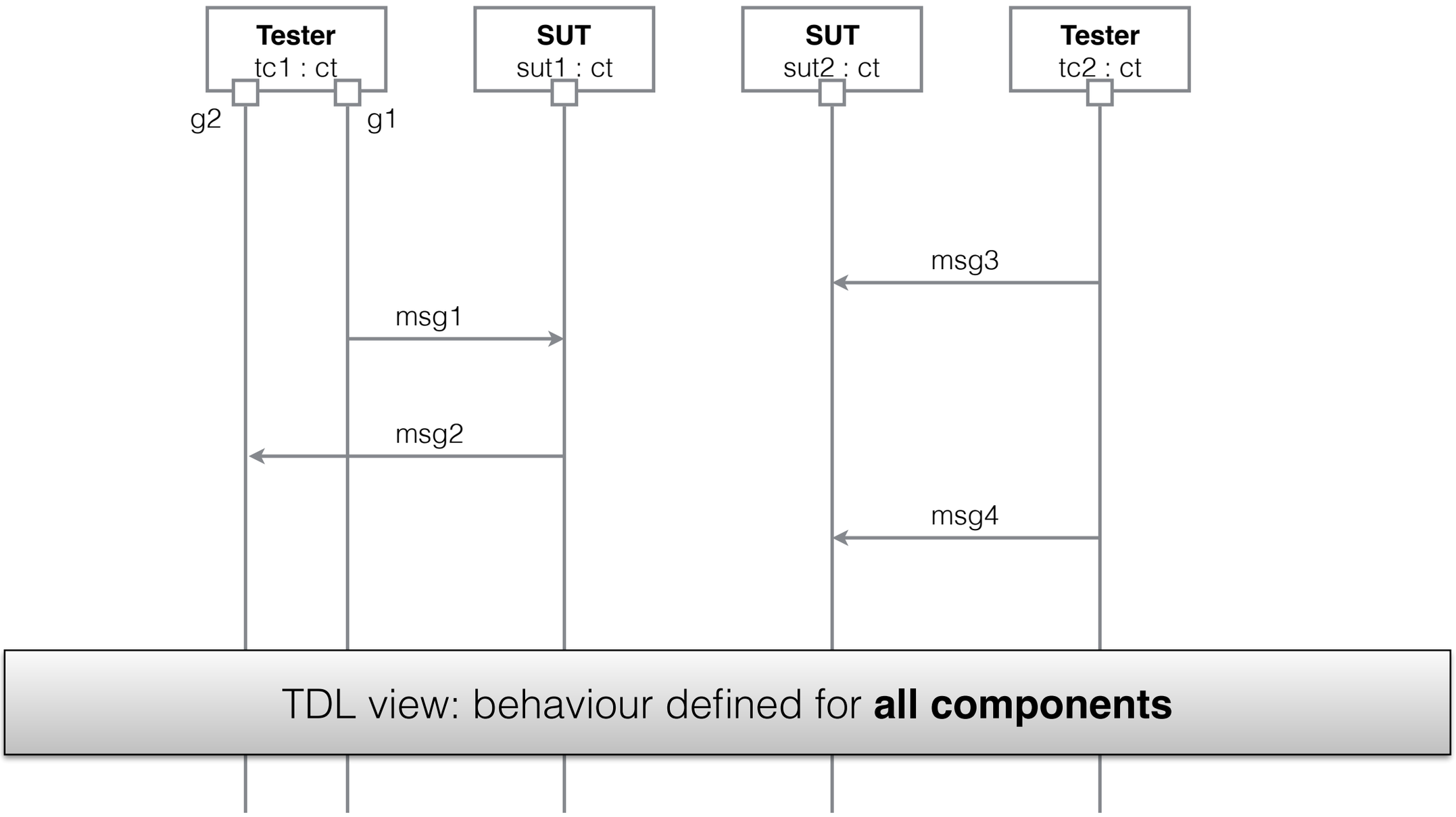
- Assumptions: Behaviour
  - TTCN-3
    - test system view
    - independent concurrent execution
    - explicit synchronisation
    - strictly local behaviours
  - TDL
    - global view
    - global or local ordering
    - implicit or explicit synchronisation
    - global combined behaviours



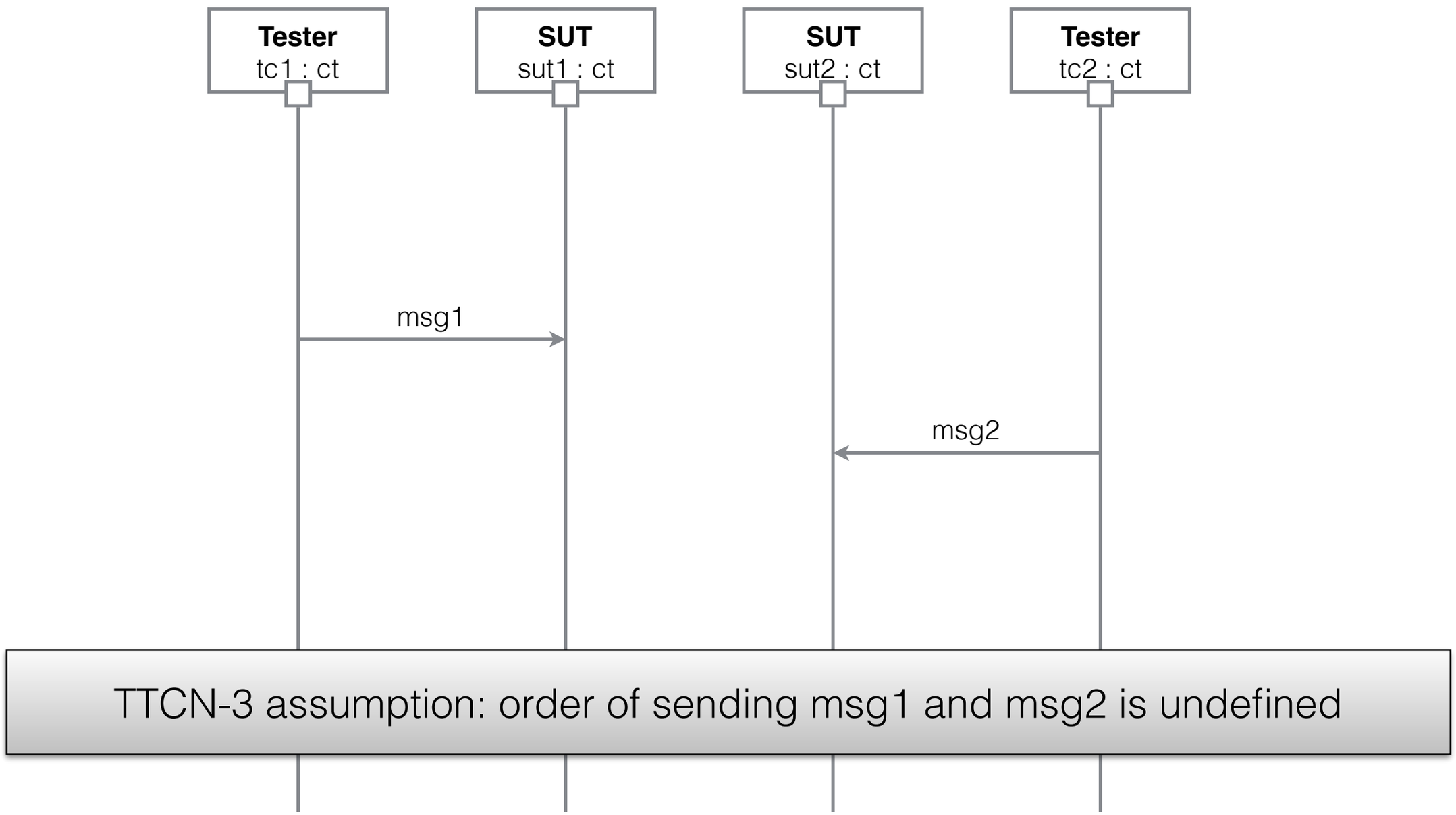
Mapping TDL to TTCN-3: Views



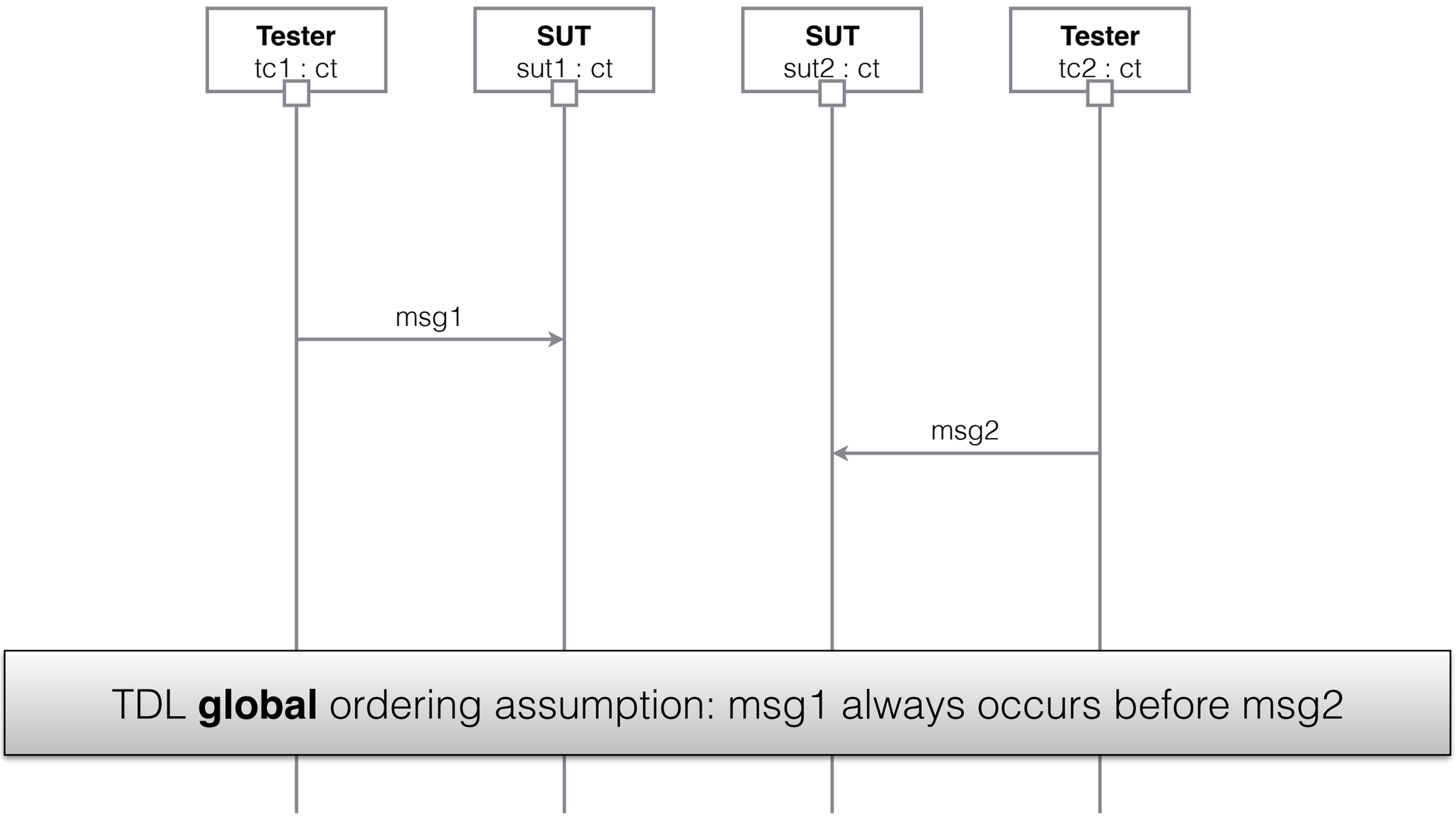
Mapping TDL to TTCN-3: View



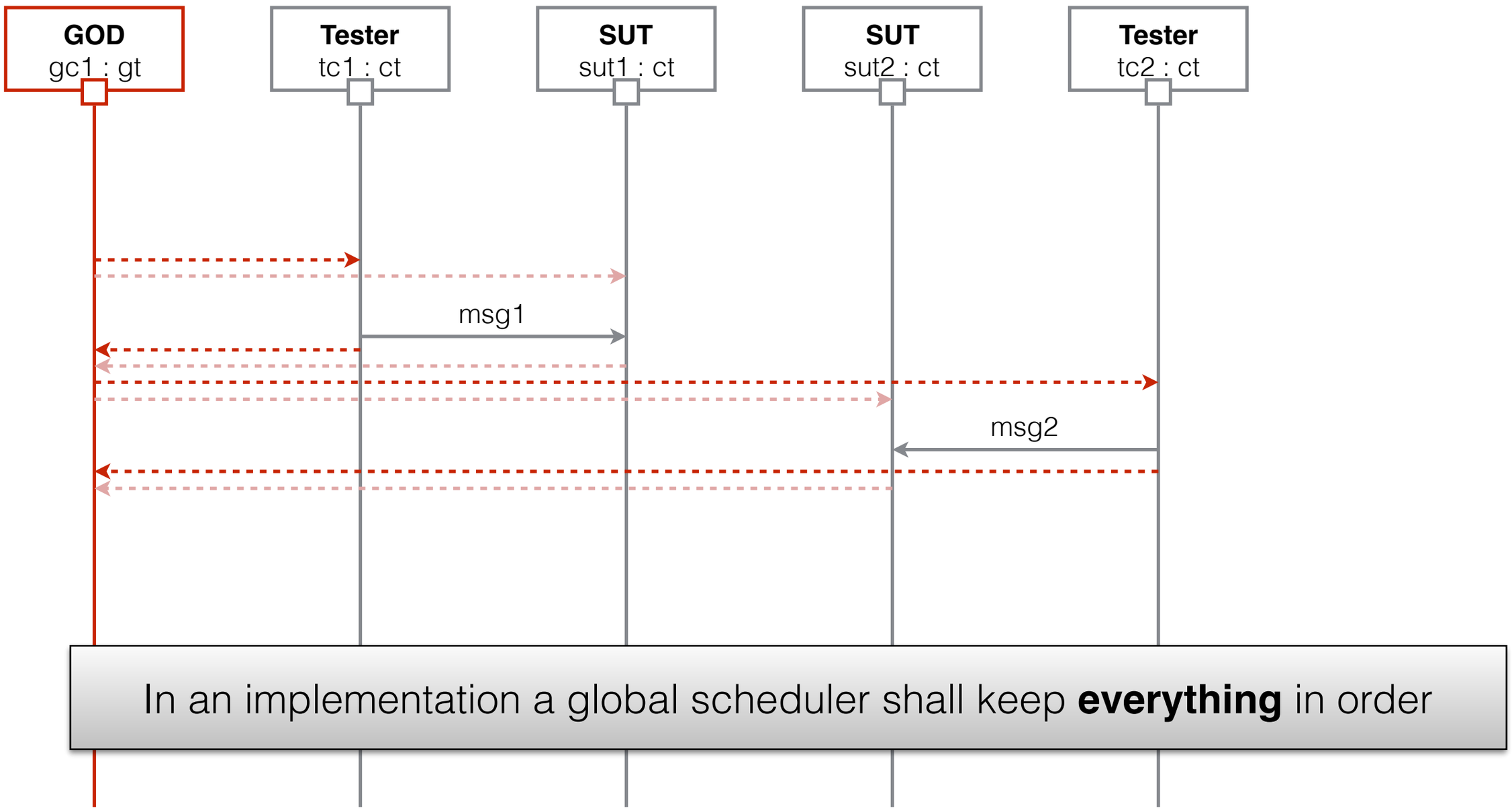
Mapping TDL to TTCN-3: Ordering



Mapping TDL to TTCN-3: Ordering

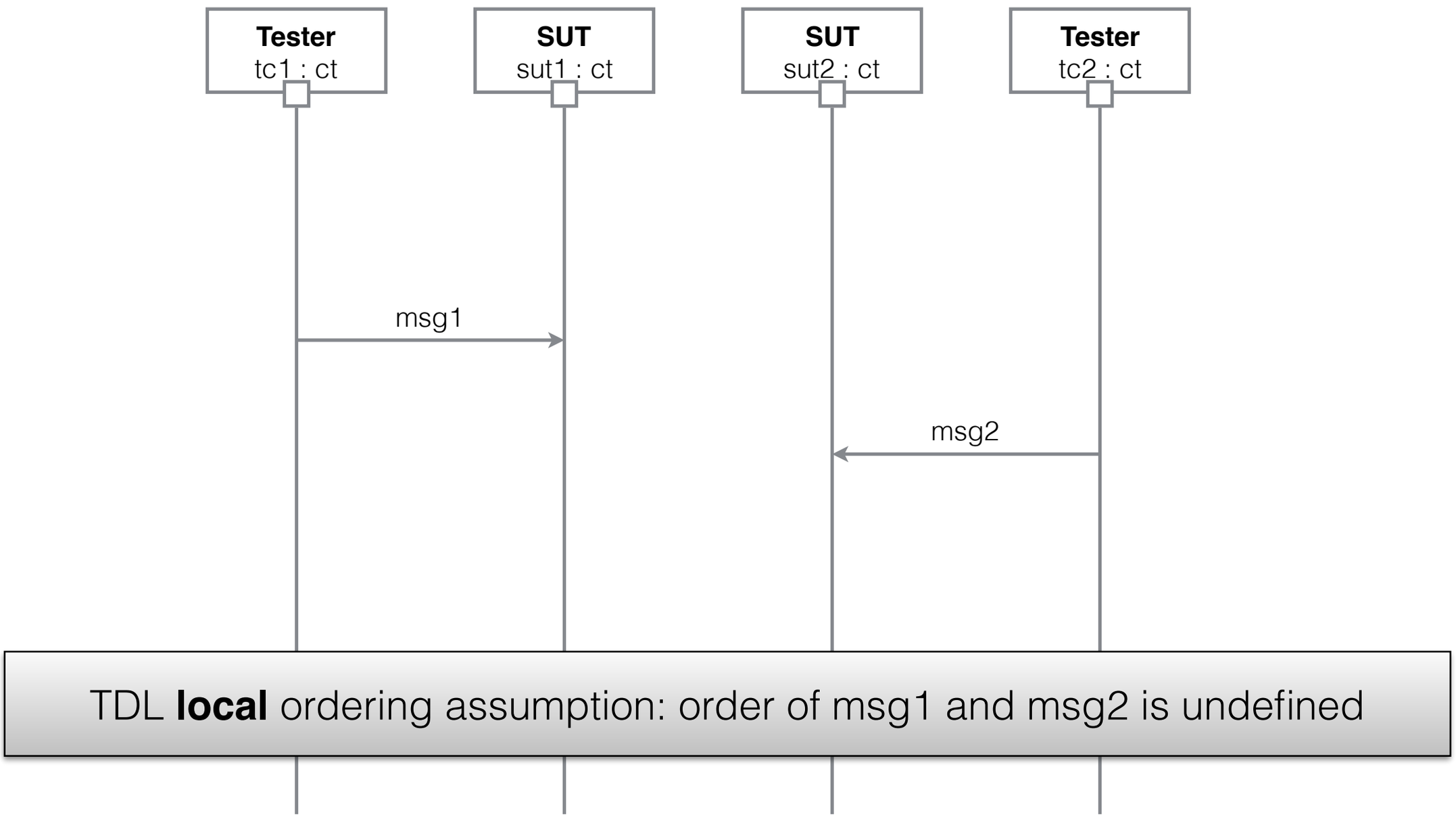


Mapping TDL to TTCN-3: Ordering

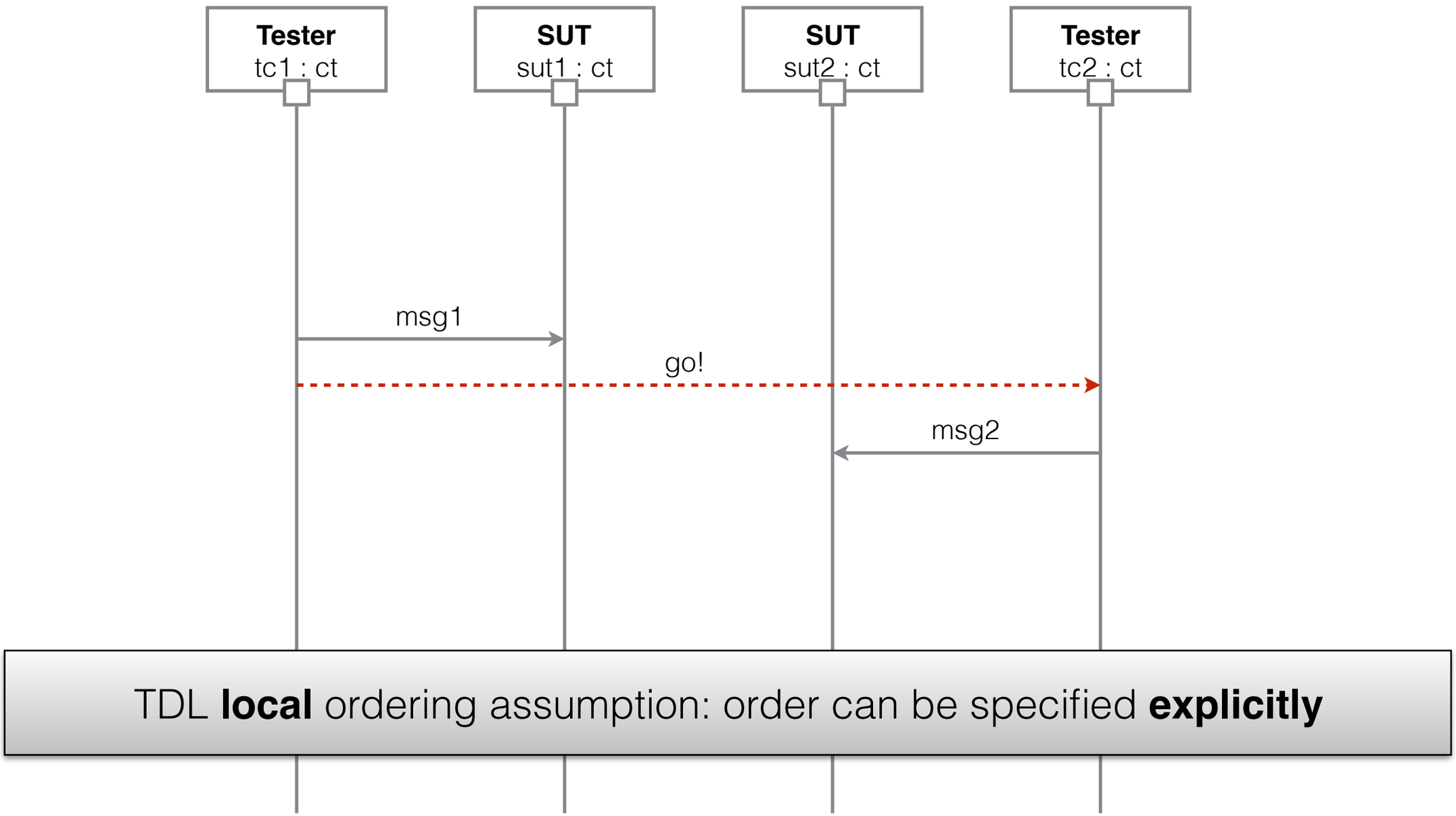


GOD*: Global Order Dispatcher

Mapping TDL to TTCN-3: Ordering



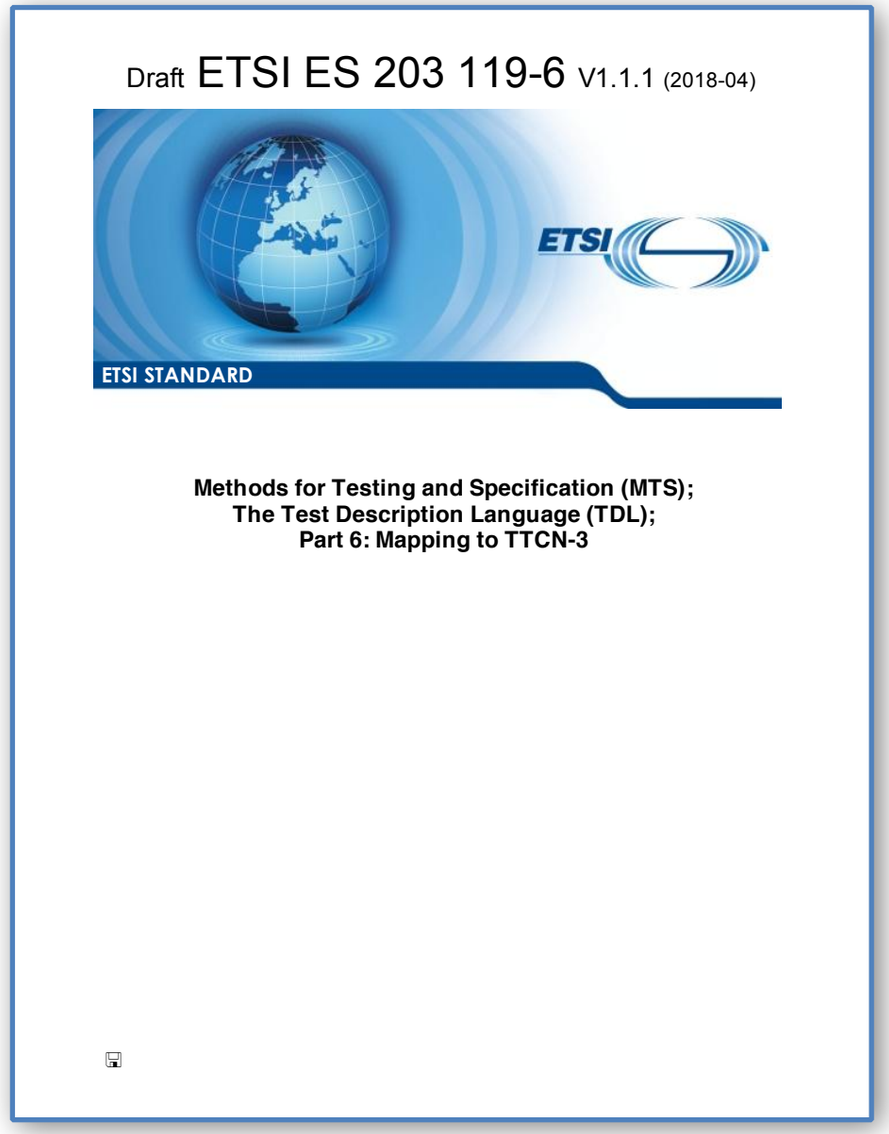
Mapping TDL to TTCN-3: Ordering



TDL **local** ordering assumption: order can be specified **explicitly**

Mapping TDL to TTCN-3

- Mapping: Behaviour
  - capture tester perspective only
  - only locally ordered so far
  - functions for each component
- combined behaviours
  - split for each participating component
- interactions
  - split into test and/or receive
- deviations from behaviour
  - altsteps activated as defaults


Draft ETSI ES 203 119-6 V1.1.1 (2018-04)

ETSI STANDARD

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

Mapping TDL to TTCN-3: Behaviour

```
Test Description Implementation TD_7_1_3_1
  uses configuration defaultTC {

     SS.g sends pdcch (c_rnti=ue) to UE.g;
     SS.g sends mac_pdu to UE.g;
     UE.g sends harq_ack to SS.g with {
         test objectives : TP1 ;
     };

     set verdict to PASS ;
     SS.g sends pdcch (c_rnti=unknown) to UE.g;
     SS.g sends mac_pdu to UE.g;

     alternatively {
         UE.g sends harq_ack to SS.g ;
         set verdict to FAIL ;
     } or {
         gate SS.g is quiet for five ;
         set verdict to PASS ;
     } with {
        test objectives : TP2 ;
     }
}
```

```
altstep to_handle_deviations_from_TDL_description_AS () {
  [] any port.receive {
    setverdict(fail);
    mtc.stop;
  }
  //if nothing happens, a timer shall be started
  //before every receive instruction
  //and the timer must be here
  //or we can leave the timeout for
  //the execute instruction called with the optional
  //timer parameter - but in this case
  //the final verdict will be 'error'
}

altstep quiescence_handler_AS (timer quiescence) {
  //for all quiescence that is not connected to a gate
  [] any port.receive{
     setverdict(fail);
     mtc.stop;
   }
   [] quiescence.timeout {
     setverdict(pass);
   }
}
```

Mapping TDL to TTCN-3: Behaviour

```
Test Description Implementation TD_7_1_3_1
  uses configuration defaultTC {

    SS.g sends pdcch (c_rnti=ue) to UE.g;
    SS.g sends mac_pdu to UE.g;
    UE.g sends harq_ack to SS.g with {
        test objectives : TP1 ;
    };

    set verdict to PASS ;
    SS.g sends pdcch (c_rnti=unknown) to UE.g;
    SS.g sends mac_pdu to UE.g;

    alternatively {
        UE.g sends harq_ack to SS.g ;
        set verdict to FAIL ;
    } or {
        gate SS.g is quiet for five ;
        set verdict to PASS ;
    } with {
       test objectives : TP2 ;
    }
}
```

```
function behaviourOfTESTER_SS() runs on defaultCT {
  timer quiescence;

  activate(to_handle_deviations_from_TDL_description_AS());

  g_to_map.send(modifies pdcch := {c_rnti := ue})
  g_to_map.send(mac_pdu);
  g_to_map.receive(harq_ack);
  setverdict(pass);
  /*Test Objective Statisfied:  TP2 */
  g_to_map.send(modifies pdcch := {c_rnti := unknown});
  g_to_map.send(mac_pdu);

  quiescence.start(five);
  alt{
    [] g_to_map.receive(harq_ack){
       setverdict(fail);
    }
    [] quiescence_handler_AS(quiescence);
    /*Test Objective Statisfied:  TP2 */
  }
}
```

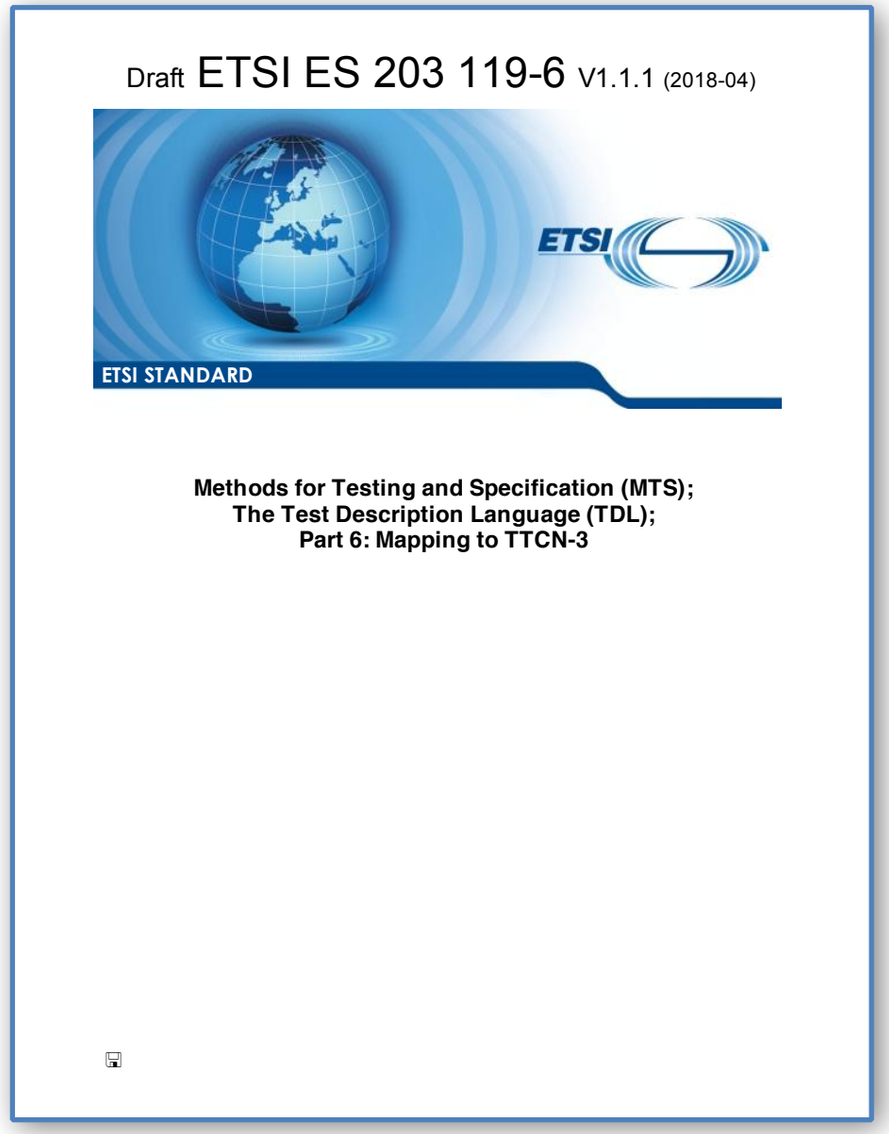
Mapping TDL to TTCN-3: Behaviour

```
Test Description Implementation TD_7_1_3_1
  uses configuration defaultTC {

     SS.g sends pdcch (c_rnti=ue) to UE.g;
     SS.g sends mac_pdu to UE.g;
     UE.g sends harq_ack to SS.g with {
         test objectives : TP1 ;
     };

     set verdict to PASS ;
     SS.g sends pdcch (c_rnti=unknown) to UE.g;  }
     SS.g sends mac_pdu to UE.g;

     alternatively {
         UE.g sends harq_ack to SS.g ;
         set verdict to FAIL ;
     } or {
         gate SS.g is quiet for five ;
          set verdict to PASS ;
     } with {
        test objectives : TP2 ;
     }
}
```

```
testcase TD_7_1_3_1() runs on MTC_CT
     system defaultCT
{
  activate(to_handle_deviations_from_TDL_description_AS());

  defaultTC();

  TESTER_SS.start(behaviourOfTESTER_SS());

  all component.done;
```

Mapping TDL to TTCN-3

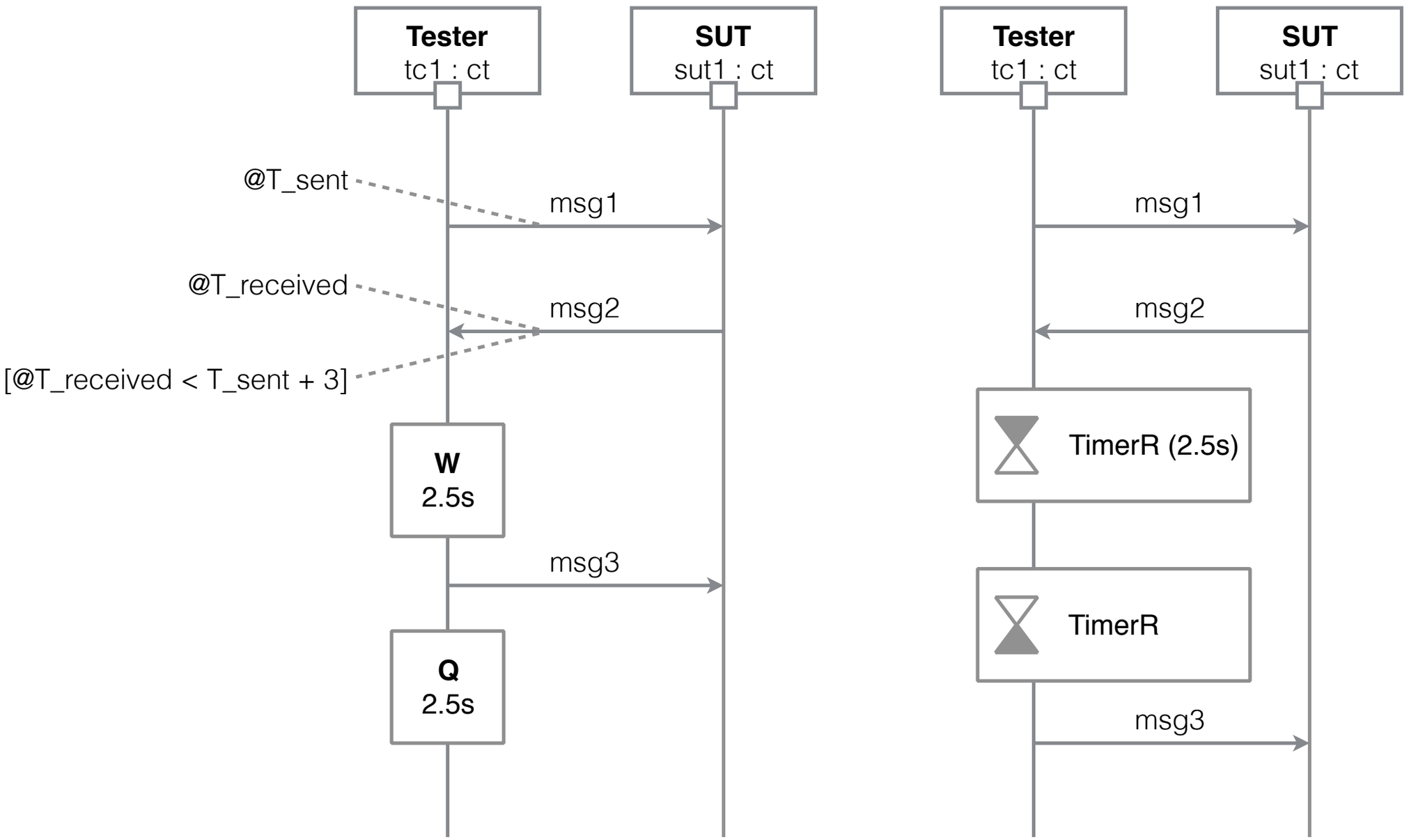
- Assumptions: Time
  - TTCN-3
    - timers and timer operations
    - realtime extension
  - TDL
    - timers and timer operations
    - time operations (wait, quiescence)
    - time labels and time constraints


Draft ETSI ES 203 119-6 V1.1.1 (2018-04)

ETSI STANDARD

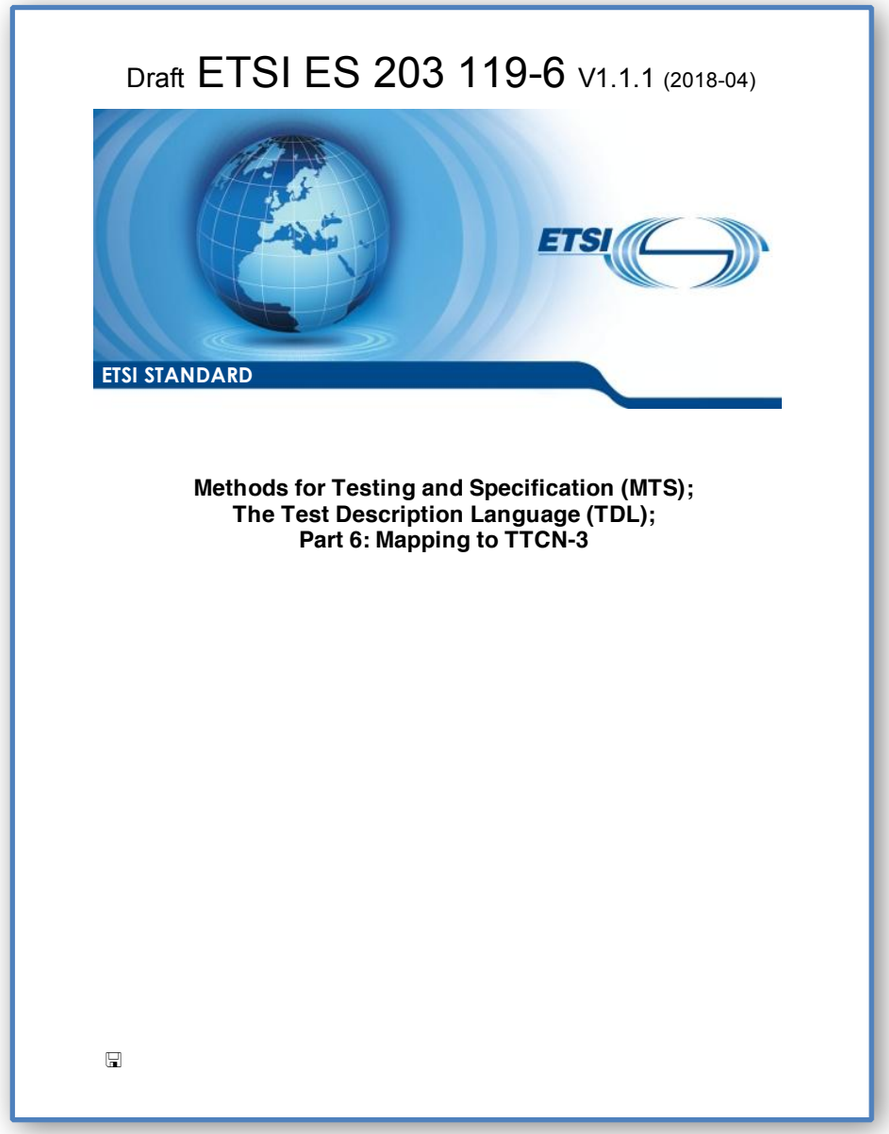
Methods for Testing and Specification (MTS);
The Test Description Language (TDL);
Part 6: Mapping to TTCN-3

Mapping TDL to TTCN-3



Mapping TDL to TTCN-3

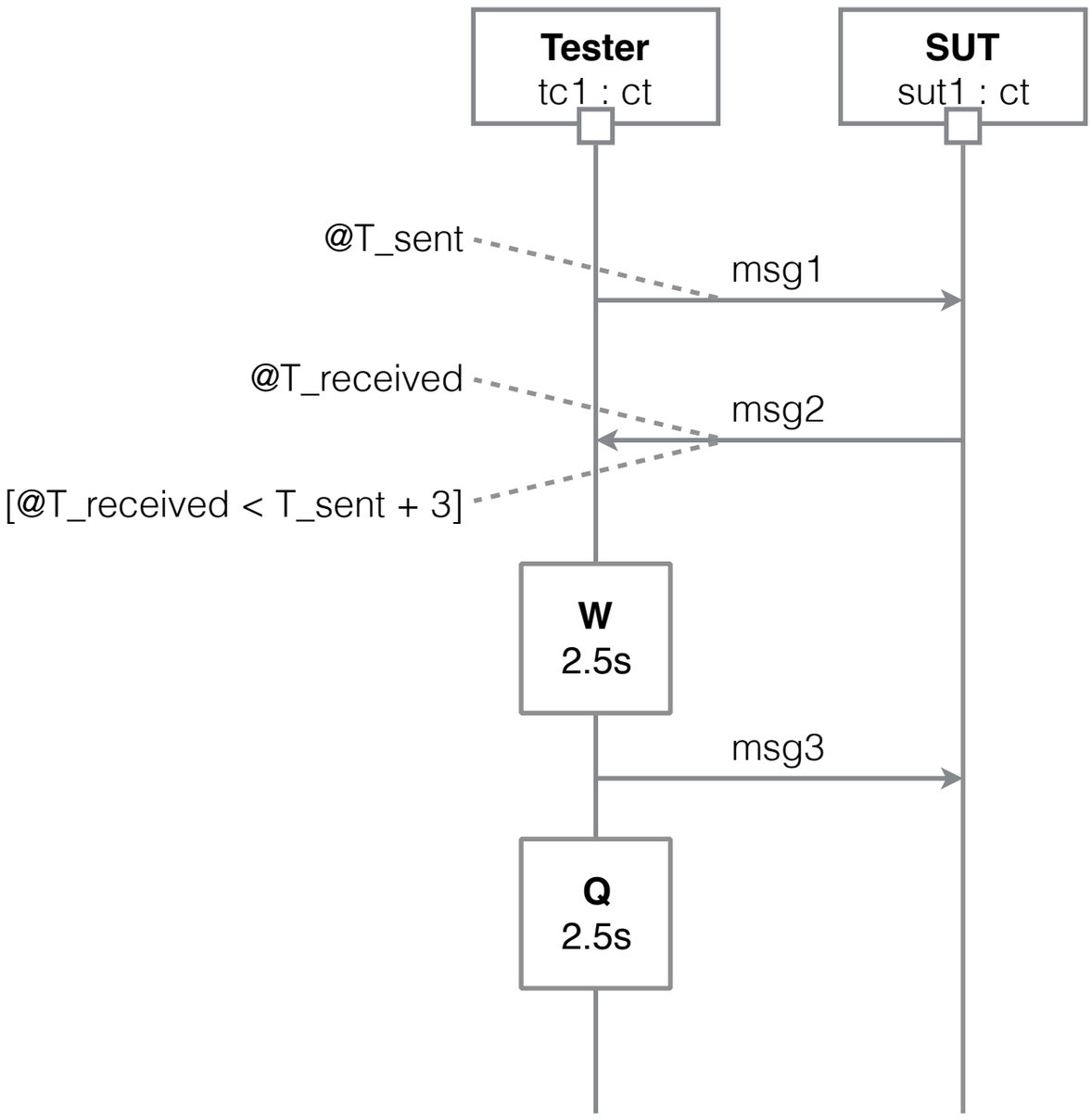
- Mapping: Time
  - all concepts expressed by timers
  - local time keeping per component
  - time constraints challenging


Draft ETSI ES 203 119-6 V1.1.1 (2018-04)

ETSI STANDARD

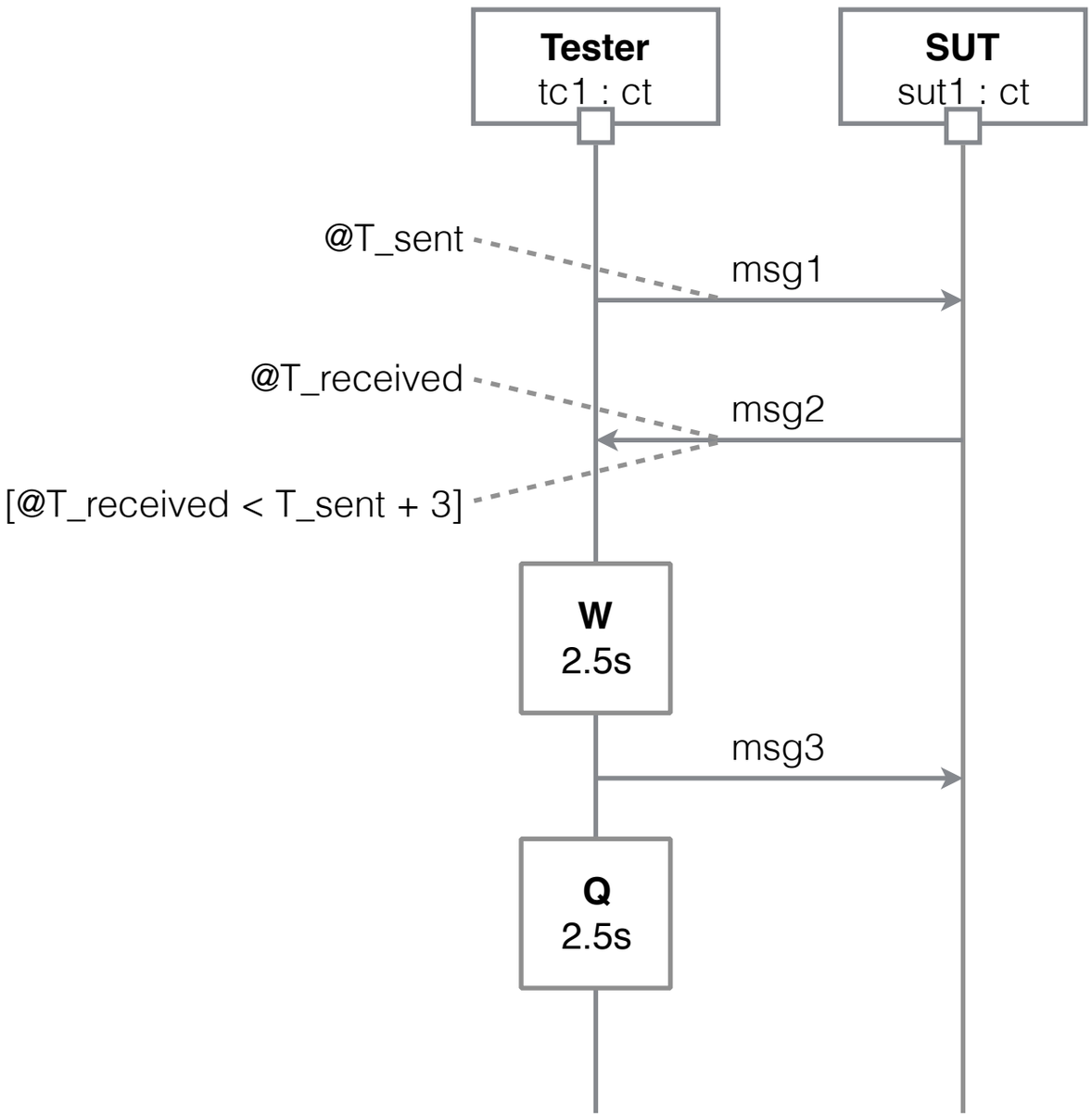
Methods for Testing and Specification (MTS);
The Test Description Language (TDL);
Part 6: Mapping to TTCN-3

Mapping TDL to TTCN-3: Time




```
function behaviourOfTESTER_tc1() runs on ct {
    timeKeeper.start(forever)

    g.send(msg1);
    //Time label
    var float T_sent := timeKeeper.read;

    g.receive(msg2);
    var float T_received := timeKeeper.read;
    //Time constraint
    if (T_received > T_sent + 3) {
        setverdict(fail);
        mtc.stop;
    }
    //...
}
```

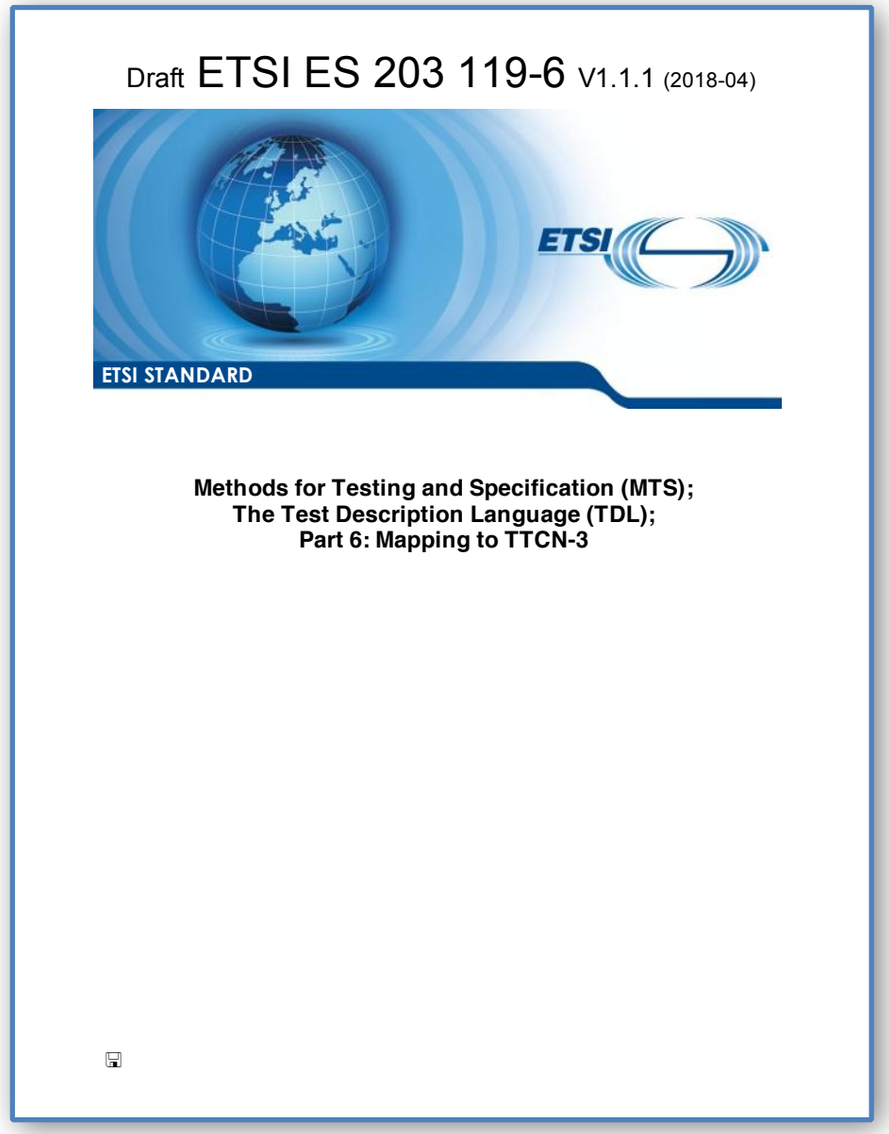
Mapping TDL to TTCN-3: Time




```
function behaviourOfTESTER_tc1() runs on ct {
    //...

    //Wait
    timer T1_wait_1;
    var default wh := activate(Wait_handler_AS());
    T1_wait_1.start(2.5);
    T1_wait_1.timeout;
    deactivate(wh);

    g.send(msg3);

    //Quiescence
    timer T1_quiescence_1;
    T1_quiescence_1.start(2.5);
    alt {
        [] T1_quiescence_1.timeout {setverdict (pass);}
        [] any port.check(receive) {setverdict (fail);}
    }
}

altstep Wait_handler_AS() {
    //for suppressing handling of unexpected behaviour
    [] any port.check(receive) {repeat;}
}
```

Mapping TDL to TTCN-3

- Everything else
  - packages -> modules
  - element imports -> imports
  - annotations ->
    - comments
    - special instructions
    - code (TTCN3Code)
  - test objectives -> comments
  - comments -> comments


Draft ETSI ES 203 119-6 V1.1.1 (2018-04)

ETSI STANDARD

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

Concluding remarks

- New technology, growing rapidly
- Open-source project for essential tool support
  - lower barrier to entry, accelerate adoption
  - commercial tool support not yet available
- Custom tools can be put together in a matter of hours
  - basic yet capable
  - make early adoption easier
- Advanced solutions still require additional effort
  - not immediately necessary to get started with using TDL

Concluding remarks

- Mapping may seem straightforward at first
  - but things can get very complicated the closer one looks
  - both languages have evolved to become rather complex
- Identify assumptions and semantic gaps
  - some restrictions may not be immediately obvious
  - some concepts may not be mappable at all in a useful way
  - adaptations to both languages can make mappings easier
  - some assumptions may need to be challenged
- A standardised mapping defines baseline expectations
  - tool- and user-specific can be optionally applied on top

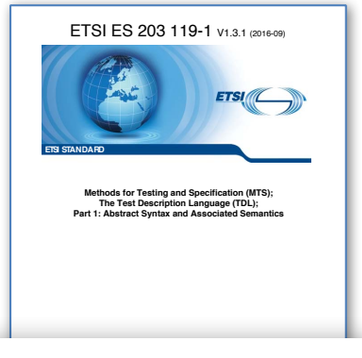
Concluding remarks

- ES 203 119-6 currently being finalised
  - further refinements and examples
  - locally ordered test descriptions only
  - some mapping-related restrictions
  - ready for approval: January 2018
  - publishing date: April 2018
- Prototypical implementation for validation
  - high-level model-to-model transformation
  - available under the TDL open source project: March 2018

Summary

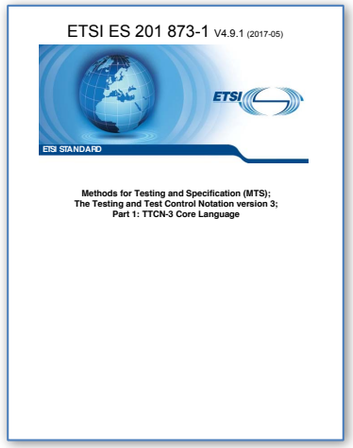
What is TDL?

- 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)



What is TTCN-3?

- Testing and Test Control Notation
  - Specification and implementation of all kinds of black-box tests
  - Platform independent link between modelling and execution
  - Component-based approach

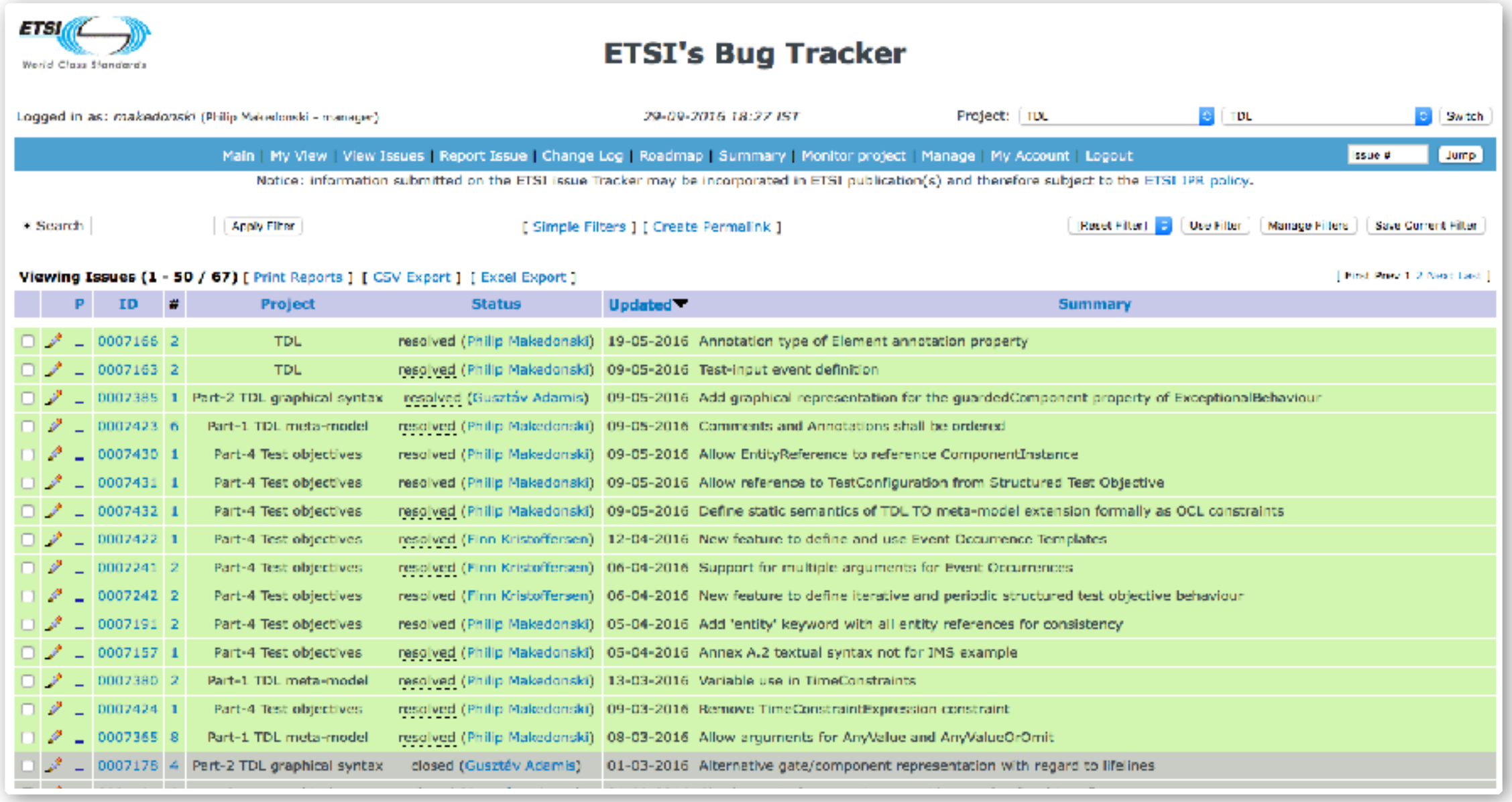


Mapping TDL to TTCN-3

- 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)

tdl.etsi.org

What would you want to see in TDL?



ETSI's Bug Tracker

Logged in as: makedonski (Philip Makedonski - manager) 29-09-2016 18:27 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.

+ Search | Apply Filter [Simple Filters] [Create Permalink] Reset Filter Use Filter Manage Filters Save Current Filter

Viewing Issues (1 - 50 / 67) [Print Reports] [CSV Export] [Excel Export] [First Prev 1 2 Next Last]

P	ID	#	Project	Status	Updated	Summary
_	0007166	2	TDL	resolved (Philip Makedonski)	19-05-2016	Annotation type of Element annotation property
_	0007163	2	TDL	resolved (Philip Makedonski)	09-05-2016	Test-input event definition
_	0007385	1	Part-2 TDL graphical syntax	resolved (Gusztáv Adamis)	09-05-2016	Add graphical representation for the guardedComponent property of ExceptionalBehaviour
_	0007423	6	Part-1 TDL meta-model	resolved (Philip Makedonski)	09-05-2016	Comments and Annotations shall be ordered
_	0007430	1	Part-4 Test objectives	resolved (Philip Makedonski)	09-05-2016	Allow EntityReference to reference ComponentInstance
_	0007431	1	Part-4 Test objectives	resolved (Philip Makedonski)	09-05-2016	Allow reference to TestConfiguration from Structured Test Objective
_	0007432	1	Part-4 Test objectives	resolved (Philip Makedonski)	09-05-2016	Define static semantics of TDL TO meta-model extension formally as OCL constraints
_	0007422	1	Part-4 Test objectives	resolved (Finn Kristoffersen)	12-04-2016	New feature to define and use Event Occurrence Templates
_	0007241	2	Part-4 Test objectives	resolved (Finn Kristoffersen)	06-04-2016	Support for multiple arguments for Event Occurrences
_	0007242	2	Part-4 Test objectives	resolved (Finn Kristoffersen)	06-04-2016	New feature to define iterative and periodic structured test objective behaviour
_	0007191	2	Part-4 Test objectives	resolved (Philip Makedonski)	05-04-2016	Add 'entity' keyword with all entity references for consistency
_	0007157	1	Part-4 Test objectives	resolved (Philip Makedonski)	05-04-2016	Annex A.2 textual syntax not for IMS example
_	0007380	2	Part-1 TDL meta-model	resolved (Philip Makedonski)	13-03-2016	Variable use in TimeConstraints
_	0007424	1	Part-4 Test objectives	resolved (Philip Makedonski)	09-03-2016	Remove TimeConstraintExpression constraint
_	0007365	8	Part-1 TDL meta-model	resolved (Philip Makedonski)	08-03-2016	Allow arguments for AnyValue and AnyValueOrOmit
_	0007175	4	Part-2 TDL graphical syntax	closed (Gusztáv Adamis)	01-03-2016	Alternative gate/component representation with regard to lifelines

From TDL to TTCN-3:
A Step By Step Tutorial

Philip Makedonski, Gusztav Adamis, Martti Käärik,
Finn Kristoffersen, Gyorgy Rethy

tdl.etsi.org