STF 476: TDL Phase 2

Status Report

Document History

- 2014-11-26: Document submitted for SG #6
- 2014-09-25: Document submitted for MTS #63
    - long form for SG #5 / Technical Session, short form for MTS #63
- 2014-05-10: Document submitted for MTS #62
    - long form for SG #3 / Technical Session, short form for MTS #62
- 2014-03-19: Document submitted for SG #2

Session and Milestone Planning

Session Planning

- 6 sessions in total
- 2 sessions per milestone
  - 1 preparatory / debriefing
  - 1 finalising
- Homework and remote coordinated work in between

Session Overview

- WK09 Feb 24-28 - Session 1 @ETSI
- WK15 Apr 07-11 - Session 2 @ETSI
- WK23 Jun 02-06 - Session 3 @FOKUS
- WK36 Sep 01-05 - Session 4 @ETSI
- WK42 Oct 13-17 - Session 5 @Siemens
- WK49 Dec 01-05 - Session 6 @ETSI

Milestone 3 Timeline

- WK42 Oct 13-17 - Session 5 @Siemens
  - 5 experts, 25 days, define roadmaps, prepare final drafts
- WK49 Dec 01-05 - Session 6 @ETSI
  - 5 experts, 25 days, finalise final drafts
- WK50 Dec 15-19 - Deliverables ready
- WK05 Jan 27-28 2015 - MTS #64 @Ericsson

Milestone Resources

- ~15 days/expert per milestone
  - assuming roughly equal resource allocation per expert
  - 2x4 days sessions, ~7 days homework

- Milestone 1: ~60 days planned, 44.5 used (4 experts)
- Milestone 2: ~75 days planned, 73/77.5 used* (5 experts)
- Milestone 3: ~75 days planned, 90.5/106 available** (5 experts)

* 4.5 from extension for Part 4
** 50 days allocated for two sessions, ~80 days used after Session 6 (TAM unavailable ATM)

Session Summaries

Session 5 Summary

- Goals
  - Part 1: address outstanding items (time, data), technical review, validation
  - Part 2: align with Part 1, outstanding constructs, update example, notation
  - Part 3: address outstanding items (multiple inheritance, etc.)
  - Part 4: extend integration (time, data), prepare evaluation
  - Plan milestone timeline

Part 1

Fixed, Improved, Consolidated, …

EDITOR NOTE: The document represents the current status of work on the TDL meta-model as follows. Items uncovered in this draft are marked "(open)".

FOUNDATION

ElementImport
distinguish between package import and import of elements from one package

DATA

DataSet
rename to DataType

DataInstance
shall they be packagable? Yes

AnyValue
considered only within a DataInstanceSpecification, not as a special DataInstance

TupleElement
term "tuple" might be not well understood, change to Member
introduce distinction of mandatory/optional elements, isOptional: Boolean = false

TupleElementInstance
change to MemberAssignment
shall refer to a special kind of DataInstanceSpecification (without variables) instead of a DataInstance

StructuredDataType
relax tupleElement[1..*] multiplicity to [*]

DataInstanceSpecification
change to "DataUseSpec"
handle nested structured data instances for constants, proxies and function return values

StaticDataInstanceSpec
add new class to denote static expressions, used when defining structured data instances

EDITOR NOTE: The document represents the current status of work on the TDL meta-model as follows. Items uncovered in this draft are marked "(open)".

FOUNDATION

ElementImport

distinguish between package ir

DATA

DataSet

rename to DataType

DataInsta

shall th

AnyValue

conside

TupleElen

term "t

introdu

TupleElen

change

shall re

Structure

relax tu

DataInsta

change

handle

StaticDat

add ne

TEST BEHAVIOUR

Assignment

componentInstance shall be mandatory, i.e. [1]

TestDescriptionReference

check to include DataInstanceSpec

Assert

TIME

Time, TimeValue

similar to Verdict, treat as predefined instances

--> Time and TimeValue kept in MM to allow specification of time unit

TimeLabel

do not derive from SimpleDataInstance

! --> kept as it is to have this flexibility of data, e.g. defining functions over time labels, passing time labels as parameters of a test description

add a special TimeLabelSpec to use time labels in functions

! --> this suggestion requires to duplicate some parts, which does not seem sensible; the functionality can be provided with the more general DataInstanceSpec

do not make TimeLabel packageable, i.e. keep them local in one test description

! --> if a time label is a simple data instance, it is automatically packageable

re-attach TimeLabel to AtomicBehaviour, remove from Behaviour

TimeConstraint

reference timeLabel needed?

--> considered as derived from time constraint expression

attach TimeConstraint to AtomicBehaviour, remove from Behaviour

TEST CONFIGURATION

ComponentInstance, ComponentType

treat Variable similar to Timer, i.e. Variable is derived from ComponentType

move GateInstance to ComponentType, treat like Timer and Variable

RCP Application

Add new...

- Scratch pad
- Actions.tdlan2
- Behaviour.tdlan2
- Configuration.tdlan2
- Configuration.tdlan2.
- Data.tdlan2
- Foundation.tdlan2
- OpenIssues.tdlan2

```
TDLan Specification Data {

    Type NUMBER ;
    Type BUTTON ;

    Type MESSAGE (optional f1 of type NUMBER, f2 of type BUTTON);
    Type RESPONSE;
    Type PAYLOAD (address of type NUMBER, payload of type MESSAGE);

    NUMBER one;
    NUMBER two;
    NUMBER three;

    BUTTON OK;
    BUTTON CANCEL;

    MESSAGE m1 (f1 = one , f2 = OK);
    MESSAGE m2 (f1 = two , f2 = CANCEL);
    MESSAGE m3 (f1 = any NUMBER , f2 = CANCEL);
    MESSAGE m4 (f1 = any or no NUMBER , f2 = CANCEL);
    MESSAGE m5 (f1 = no NUMBER , f2 = CANCEL);

    PAYLOAD p1 (address = three , payload = m1(f1 = three) );

    Use "data.ttcn3" as TTCN_DATA;

    Map NUMBER to "type restrictedInteger" in TTCN_DATA as NUMBER_MAPPING;

    Map MESSAGE to "type sessionMessage" in TTCN_DATA as MESSAGE_MAPPING with {
        f1 mapped to "id" as F1_MAPPING;
        f2 mapped to "name";
    };
```

RCP Application

Add new...

Scratch pad
Actions.tdlan2
Behaviour.tdlan2
Configuration.tdlan2
Configuration.tdlan2.
Data.tdlan2
Foundation.tdlan2
OpenIssues.tdlan2

```
TDLan Specification Data {

  Type NUMBER ;
```

RCP Application

Add new...

Scratch pad
Actions.tdlan2
Behaviour.tdlan2
Configuration.tdlan2
Configuration.tdlan2.
Data.tdlan2
Foundation.tdlan2
OpenIssues.tdlan2

```
NUMBER two;

Time SECONDS;
SECONDS two_seconds;

Type BUTTON;
BUTTON OK;
BUTTON CANCEL;

Type VERDICT ;
Verdict PASS of type VERDICT ;
Verdict FAIL of type VERDICT ;
VERDICT inconc;


Type MESSAGE (optional f1 of type NUMBER, f2 of type BUTTON);
MESSAGE m1 (f1 = one , f2 = OK);

Action press : "press a button";
Action pressFormally(b1 of type BUTTON) : "press a button formally";

Function init() returns NUMBER ;
Function add(p1 of type NUMBER, optional p2 of type NUMBER) returns NUMBER ;
Function compress(p1 of type NUMBER) returns MESSAGE ;

Use "functions.ttcn3" as TTCN_FUNCTIONS;

Map init to "init()" in TTCN_FUNCTIONS as INIT_MAPPING;
Map add to "add()" in TTCN_FUNCTIONS as ADD_MAPPING with {
    p1 mapped to "par1" ;
    p2 mapped to "par2" ;
};
```

RCP Application

Add new...

Scratch pad
Actions.tdlan2
Behaviour.tdlan2
Configuration.tdlan2
Configuration.tdlan2.
Data.tdlan2
Foundation.tdlan2
OpenIssues.tdlan2

```
TDLan Specification Data {

Component Type ct1 having {
    timer t1;
    timer t2;
    variable v1 of type MESSAGE;
    variable v2 of type NUMBER;
    gate gi1 of type gt1;
    gate gi2 of type gt1;
}

Test Configuration tc1 {
    create SUT sut of type ct1;
    create Tester tester of type ct1;
    connect sut.gi1 to tester.gi1;
    connect sut.gi2 to tester.gi2;
}

Test Description td1 (p1 of type MESSAGE) uses configuration tc1;

Test Description td2 (p2 of type MESSAGE) uses configuration tc1
{
    //execute td1(parameter = one);

    sut.gi1 sends one to tester.gi1;
    sut.gi1 sends one to tester.gi1 where it is assigned to v1;
    sut.gi1 triggers one to tester.gi1;
    sut.gi1 sends call init() to tester.gi1;
    sut.gi1 sends call add(p1 = one, p2 = two) to tester.gi1;
    sut.gi1 sends call compress(p1 = one).f1 to tester.gi1;
    sut.gi1 sends parameter p2.f1 to tester.gi1;
    sut.gi1 sends parameter p2.f1(p1=PASS) to tester.gi1;
    sut.gi1 sends parameter p2.f2 to tester.gi1;
```

RCP Application

Add new...

Scratch pad
Actions.tdlan2
Behaviour.tdlan2
Configuration.tdlan2
Configuration.tdlan2.
Data.tdlan2
Foundation.tdlan2
OpenIssues.tdlan2

```
tester.v1 = one;
tester.v1 = call add(p1 = one, p2 = two);
tester.v1 = call compress(p1 = one).f1;
tester.v2 = tester.v1;
tester.v1 = tester.v2.f1;
tester.v1 = tester.v2.f1(p1=PASS);

gate sut.gi1 is quiet for two_seconds;
gate sut.gi1 waits for two_seconds;
component tester is quiet for two_seconds;
component sut waits for two_seconds;

start t1 for (two_seconds);
t1 times out ;
stop t1;

perform action press;
perform action press on component tester;
perform action pressFormally(OK);
perform action pressFormally(CANCEL) on component sut;
perform action : "press and hold button OK" ;
perform action : "press and hold button CANCEL" on component tester ;

set verdict to PASS;
assert OK ;
assert CANCEL otherwise set verdict to FAIL ;

execute td1(one);

alternatively {
    gate sut.gi1 is quiet for two_seconds;
} or {
```

RCP Application

Add new...

Scratch pad
Actions.tdlan2
Behaviour.tdlan2
Configuration.tdlan2
Configuration.tdlan2.
Data.tdlan2
Foundation.tdlan2
OpenIssues.tdlan2

```
TDLan Specification Data {

    Component Type ct1 having {

        tester.v1 = one;

        execute td1(one);

        alternatively {
            gate sut.gi1 is quiet for two_seconds;
        } or {
            sut.gi1 sends one to tester.gi1;
            tester.gi1 sends two to sut.gi1;
        } or {
            component sut is quiet for two_seconds;
            component tester waits for two_seconds;
        }

        repeat two times {
            sut.gi1 sends one to tester.gi1;
        }

        run {
            repeat two times {
                sut.gi1 sends OK to tester.gi1;
            }
        } in parallel to {
            sut.gi1 sends m1 to tester.gi1;
        }

        every two_seconds {
            sut.gi1 sends m1 to tester.gi1;
        }
    }

}
```

Part 2

Notational Conventions

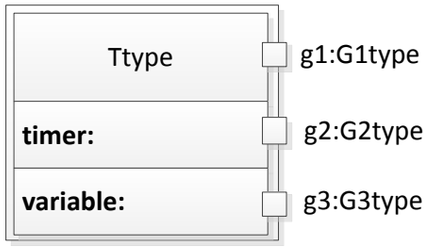
4.3. Notational Conventions

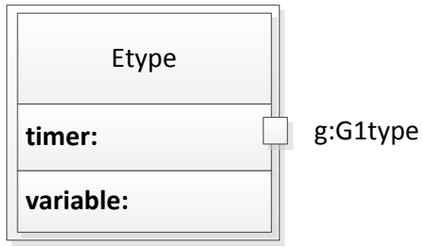
The definition of the TDL Concrete Graphical Notation consists of both shapes and textual labels placed onto this shapes. Textual labels are differentiated into non-terminal textual labels and terminal textual labels. The production rule of a non-terminal textual label are specified by combination of EBNF symbols and OCL-like expressions to navigate over the abstract syntax metamodel of TDL.

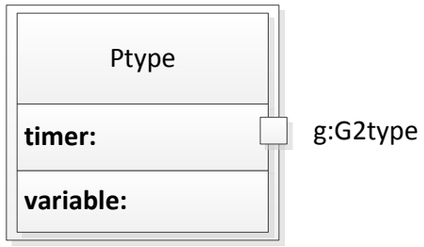
4.3.1 Symbols and meanings for shapes

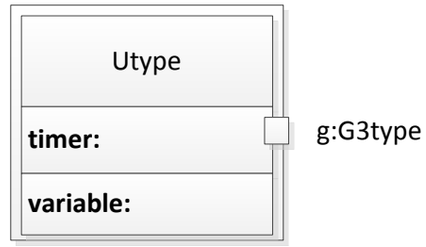
Shapes consist of outermost borders, compartments, and textual labels (i.e., non-terminal textual labels and terminal-textual labels). The conventions for this notation is the following:

- Non-terminal textual labels are typed in small captials (e.g., LABELPRODUCTIONRULE). The name of the label refers to equally named production rule that specifies how the result of the production rule is determined.
- Terminal textual labels are typed in normal characters. If a terminal textual label is typed in **bold**, bold font shall be used in the shape for that terminal textual symbol. (e.g., interaction, **timer**).
- The outermost border of a shape shall not be hidden.
- Compartments and non-terminal textual labels may be hidden to simplify the internal structure of the shape.
- Optional compartments are shaded in a light grey colour while optional non-terminal textual labels are typed in grey colour. Optionality of compartments and non-terminal textual labels is dedicately decided for each shape.
- If a non-terminal textual label is defined to be optional, the non-terminal textual label shall be shown if the surrounding compartment is shown and the corresponding non-terminal textual symbol production rule results in a non-empty string or a non-empty collection of strings.
- References to non-terminal textual procution rules external to the given shape are represented by the name of the referenced production rule enclosed in angle brackets (e.g., <REFERENCEDPROCUTIONRULE>).
- In case textual labels shall be repeated due to an upper value greater 1 of the element of the underyling metamodel, an informal statement in the Constraint section of the respective shape.
- A non-terminal textual label in between hashmarks dentotes a placeholder for a shape.





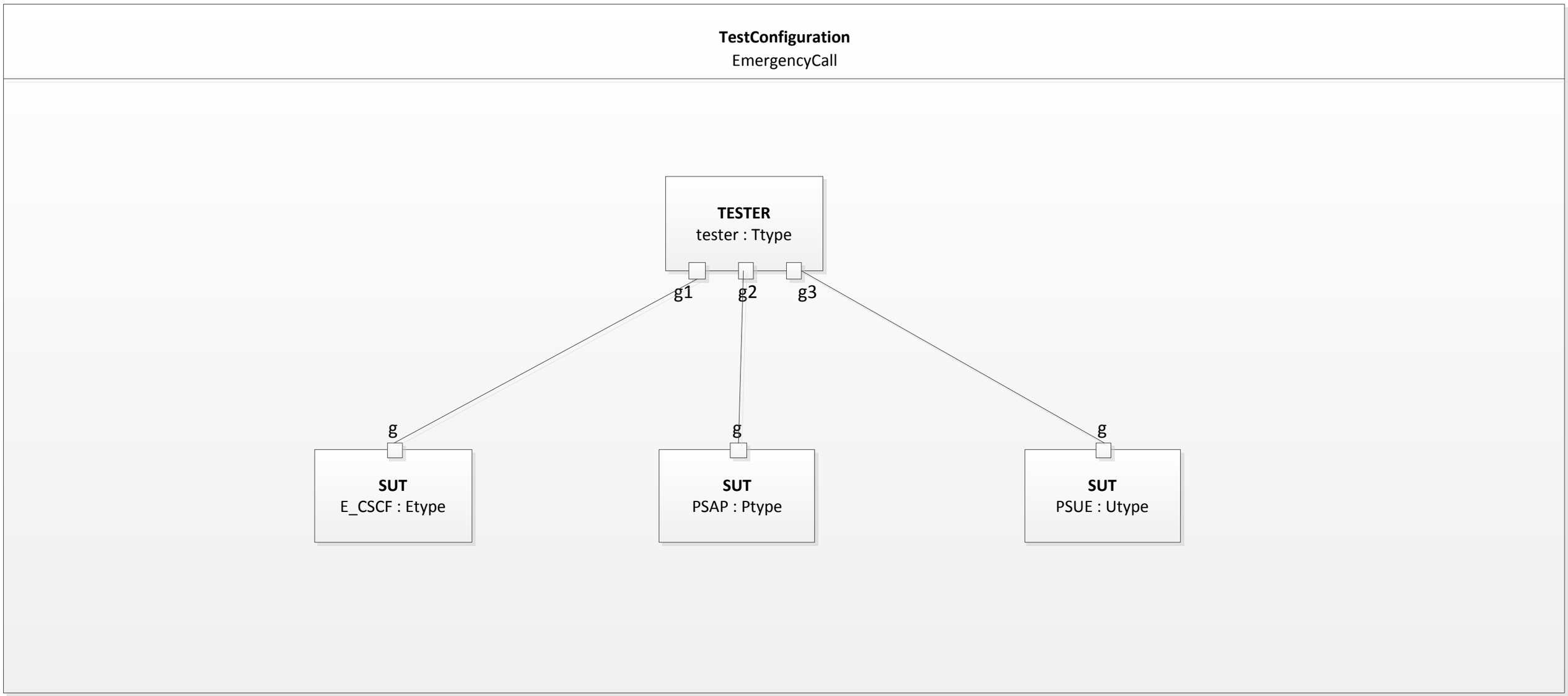




G1type
interaction: 100_TRYING

G2type
interaction: INVITE, RESPONSE

G3type
interaction: MESSAGE, PRACK



TestDescription

UC15

Parameter

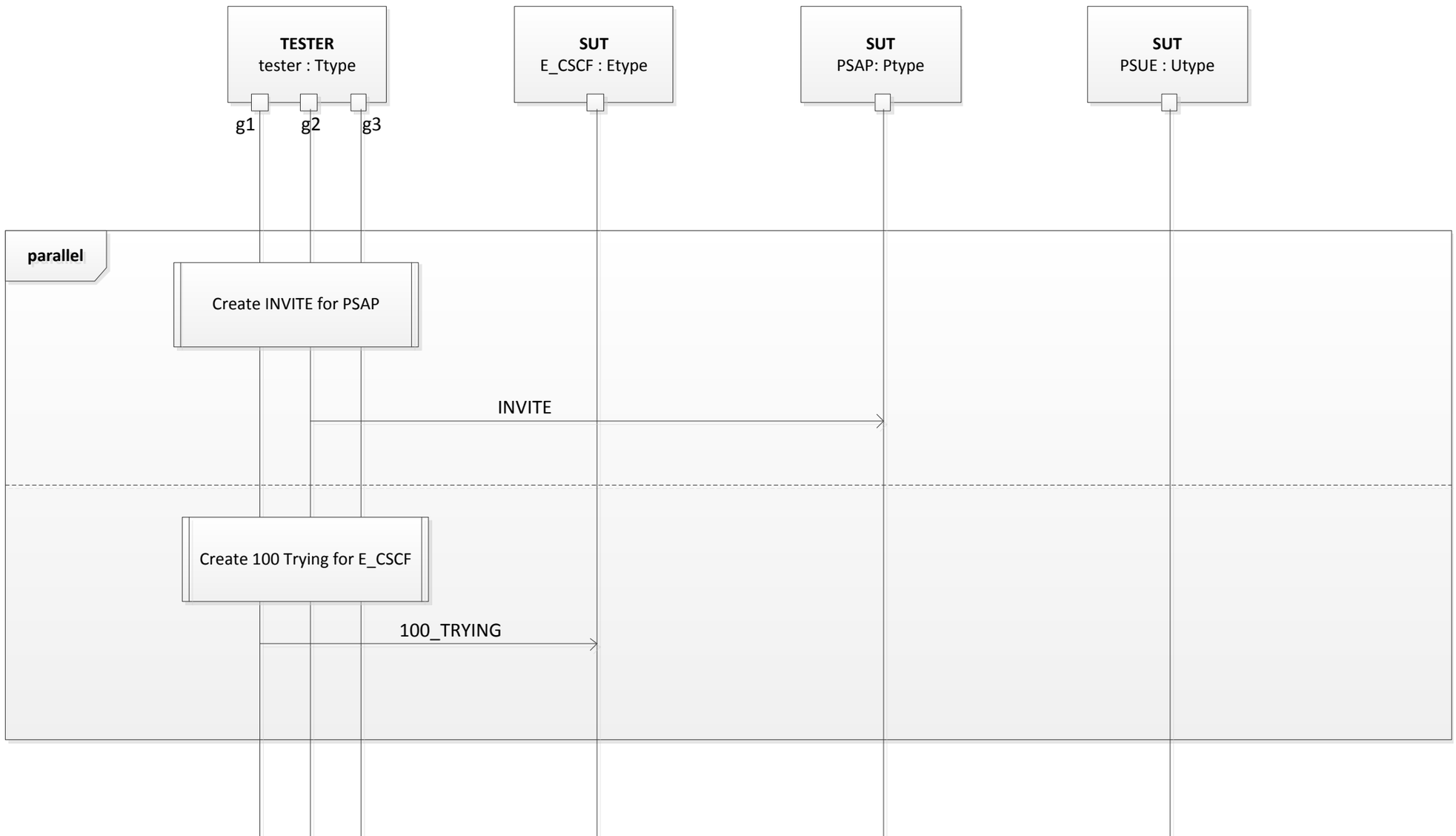
TestObjective

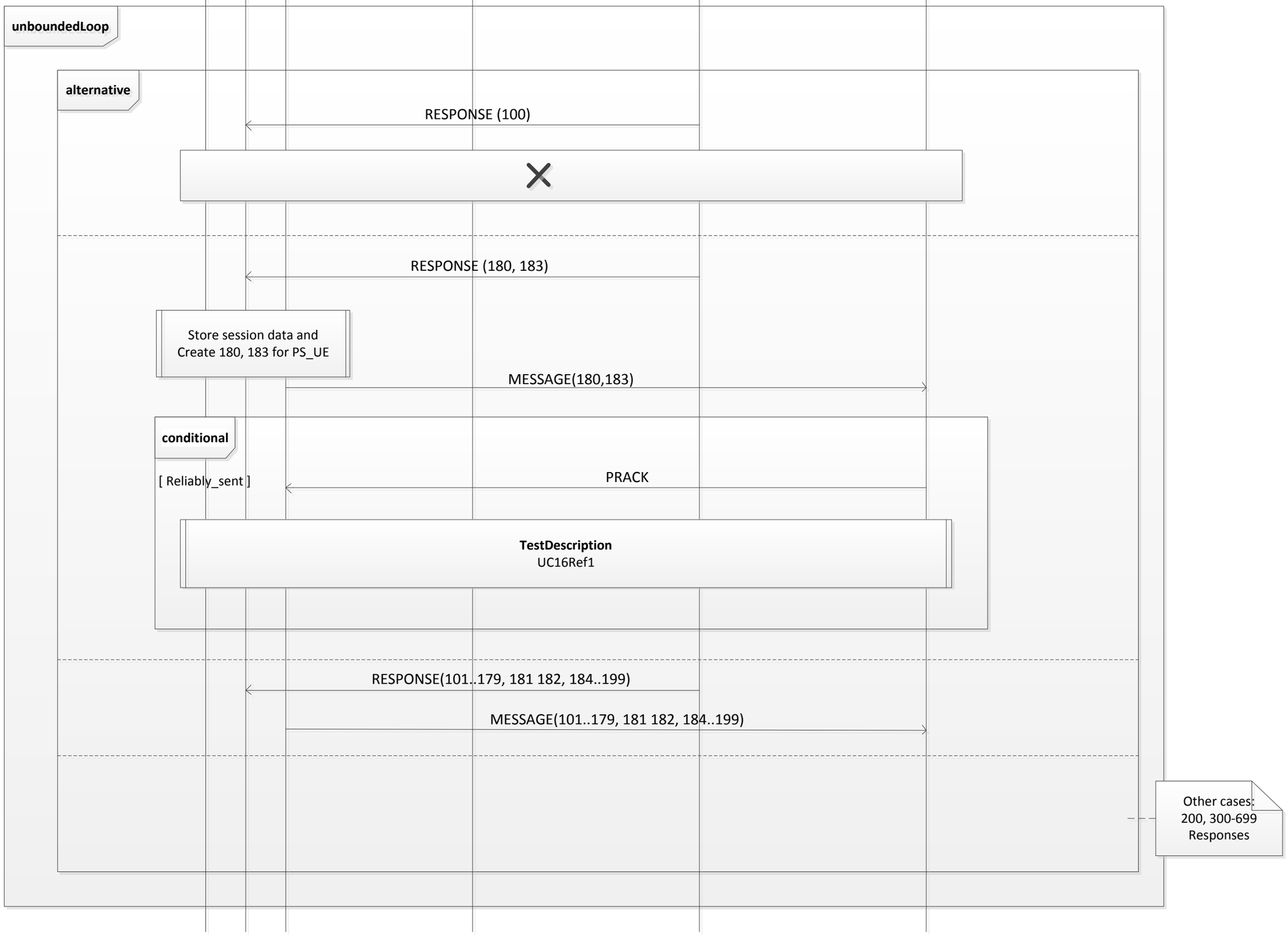
Phase 3 must get accepted

Configuration

EmergencyCall

Behaviour





Part 3

Status

- Solved problems with multiple inheritance
  - multiple inheritance no longer used the meta-model
- Progress on introductory sections
- TDL XSD to be added once the meta-model is finalised

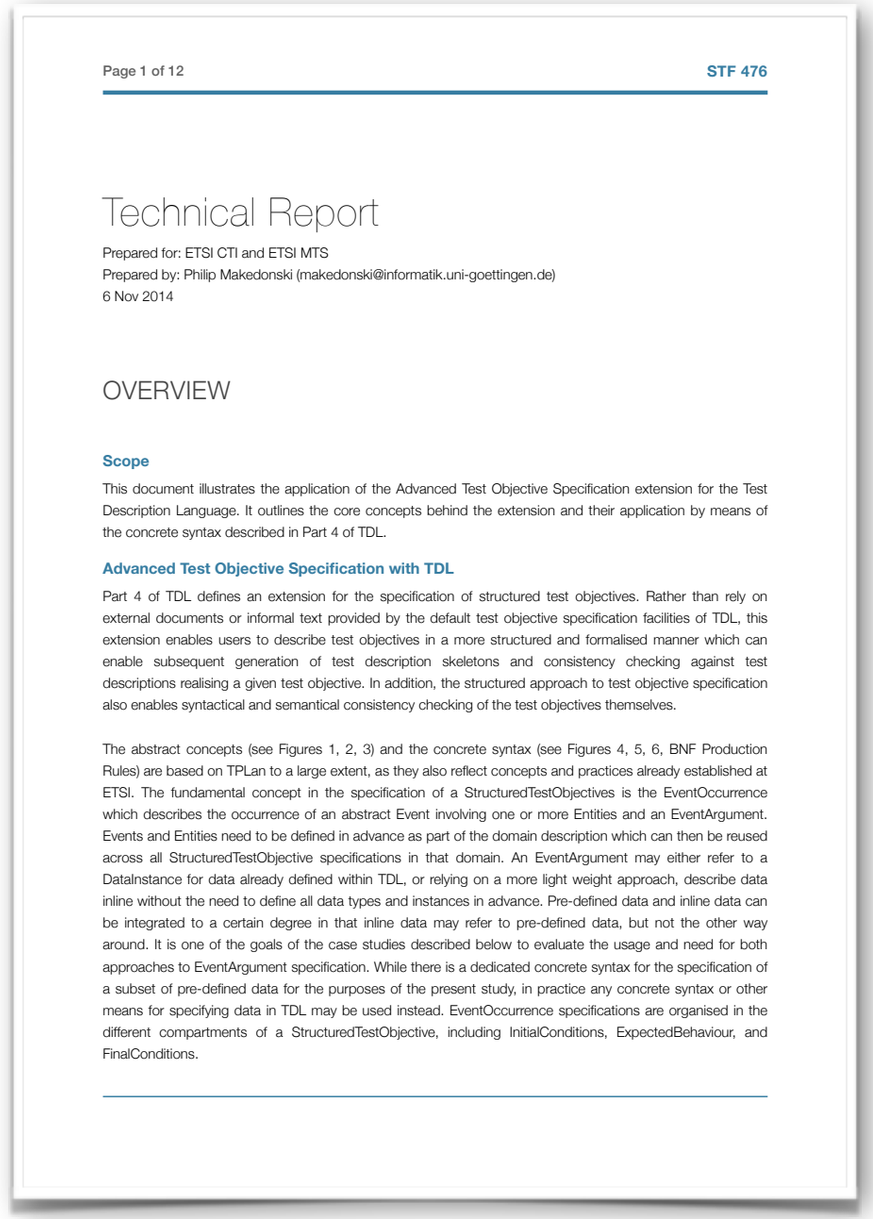
Part 4

```
Data {
    type STRING;
    type NUMBER;
    type FILE containing
        PATH of type STRING,
        CONTENT of type STRING,
        SIZE of type NUMBER;

    STRING "/tmp/data.tplan2x";
    STRING "Package data { ... }";
    STRING data_path;
    STRING data_content;
    STRING workspace_path;

    NUMBER 20;
    NUMBER data_size;

    FILE fileWithLiterals with
        PATH indicating value "/tmp/data.tplan2x",
        CONTENT indicating value "Package data { ... }",
        SIZE indicating value 20
        ;
    FILE fileWithSymbols with
        PATH indicating value data_path,
        CONTENT indicating value data_content,
        SIZE indicating value data_size
        ;
}
```



Page 1 of 12 STF 476

Technical Report

Prepared for: ETSI CTI and ETSI MTS
Prepared by: Philip Makedonski (makedonski@informatik.uni-goettingen.de)
6 Nov 2014

OVERVIEW

Scope

This document illustrates the application of the Advanced Test Objective Specification extension for the Test Description Language. It outlines the core concepts behind the extension and their application by means of the concrete syntax described in Part 4 of TDL.

Advanced Test Objective Specification with TDL

Part 4 of TDL defines an extension for the specification of structured test objectives. Rather than rely on external documents or informal text provided by the default test objective specification facilities of TDL, this extension enables users to describe test objectives in a more structured and formalised manner which can enable subsequent generation of test description skeletons and consistency checking against test descriptions realising a given test objective. In addition, the structured approach to test objective specification also enables syntactical and semantical consistency checking of the test objectives themselves.

The abstract concepts (see Figures 1, 2, 3) and the concrete syntax (see Figures 4, 5, 6, BNF Production Rules) are based on TPLan to a large extent, as they also reflect concepts and practices already established at ETSI. The fundamental concept in the specification of a StructuredTestObjectives is the EventOccurrence which describes the occurrence of an abstract Event involving one or more Entities and an EventArgument. Events and Entities need to be defined in advance as part of the domain description which can then be reused across all StructuredTestObjective specifications in that domain. An EventArgument may either refer to a DataInstance for data already defined within TDL, or relying on a more light weight approach, describe data inline without the need to define all data types and instances in advance. Pre-defined data and inline data can be integrated to a certain degree in that inline data may refer to pre-defined data, but not the other way around. It is one of the goals of the case studies described below to evaluate the usage and need for both approaches to EventArgument specification. While there is a dedicated concrete syntax for the specification of a subset of pre-defined data for the purposes of the present study, in practice any concrete syntax or other means for specifying data in TDL may be used instead. EventOccurrence specifications are organised in the different compartments of a StructuredTestObjective, including InitialConditions, ExpectedBehaviour, and FinalConditions.

```
Data {
    type STRING;
    type NUMBER;
    type FILE containing
        PATH of type STRING,
        CONTENT of type STRING,
        SIZE of type NUMBER;

    STRING "/tmp/data.tplan2x";
    STRING "Package data { ... }";
    STRING data_path;
    STRING data_content;
    STRING workspace_path;

    NUMBER 20;
    NUMBER data_size;

    FILE fileWithLiterals with
        PATH indicating value "/tmp/data.tplan2x",
        CONTENT indicating value "Package data { ... }",
        SIZE indicating value 20
        ;
    FILE fileWithSymbols with
        PATH indicating value data_path,
        CONTENT indicating value data_content,
        SIZE indicating value data_size
        ;
}
```

```
Test Purpose {
    TP Id TP/2/1/1
    Test objective "Check file copy with predefined data"
    Reference "R010"
    PICS Selection
    Initial conditions
    with {
        the Editor opened
    }
    Expected behaviour
    ensure that {
        when {
            the User copies the fileWithLiterals
        }
        then {
            the Editor displays the fileWithSymbols with
                PATH indicating value workspace_path
        }
    }
}

Test Purpose {
    TP Id TP/2/1/2
    Test objective "Check file copy with inline data"
    Reference "R011"
    PICS Selection
    Initial conditions
    with {
        the Editor opened
    }
    Expected behaviour
    ensure that {
        when {
            the User copies the fileWithLiterals
        }
        then {
            the Editor displays a new file
                containing path indicating value "/home/workspace/data.tplan2x" ,
                containing content indicating value "",
                containing size indicating value 0
                ;
        }
    }
}
```

```
Data {
    type STRING;
    type NUMBER;
    type FILE containing
        PATH of type STRING,
        CONTENT of type STRING,
        SIZE of type NUMBER;

    STRING "/tmp/data.tplan2x";
    STRING "Package data { ... }";
    STRING data_path;
    STRING data_content;
    STRING workspace_path;

    NUMBER 20;
    NUMBER data_size;

    FILE fileWithLiterals with
        PATH indicating value "/tmp/data.tplan2x",
        CONTENT indicating value "Package data { ... }",
        SIZE indicating value 20
        ;
    FILE fileWithSymbols with
        PATH indicating value data_path,
        CONTENT indicating value data_content,
        SIZE indicating value data_size
        ;
}
```

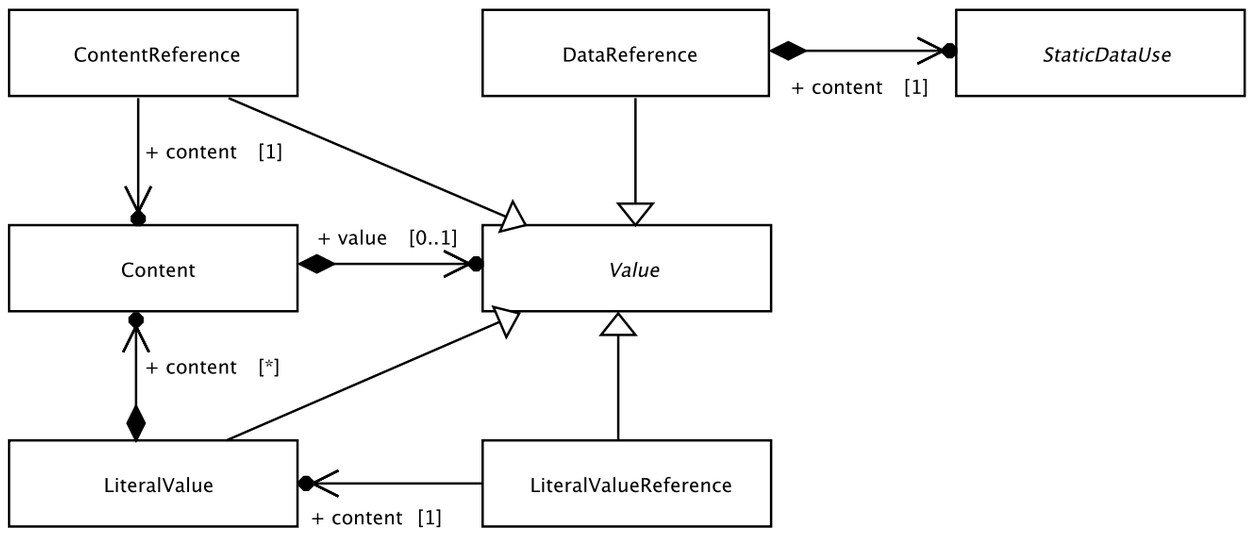
```
Test Purpose {
    TP Id TP/2/1/3
    Test objective "Check file copy with mixed data"
    Reference "R012"
    PICS Selection
    Initial conditions
    with {
        the Editor having opened a new window
    }
    Expected behaviour
    ensure that {
        when {
            the User copies the fileWithLiterals
        }
        then {
            the Editor displays a new file
                containing path indicating value "/home/workspace/data.tplan2x" ,
                containing content corresponding to data_content,
                containing size corresponding to data_size
                ;
        }
    }
    Final conditions
    with {
        the Editor closes the new window
            containing file derived from fileWithLiterals with
                PATH indicating value data_path
            ;
    }
}
```

```
type STRING;
type NUMBER;
type FILE containing
    PATH of type STRING,
    CONTENT of type STRING,
    SIZE of type NUMBER;

STRING "/tmp/data.tplan2x";
STRING "Package data { ... }";
STRING data_path;
STRING data_content;
```

- [StructuredTestObjective] TP/2/1/3
  - name = TP/2/1/3
  - objectiveURI (1) =
  - description = "Check file copy with mixed data"
  - initialConditions (1)
  - expectedBehaviour (1)
    - [ExpectedBehaviour] Expected behaviour
      - name = Expected behaviour
      - annotation (1)
      - whenClause (1)
      - thenClause (1)
        - [EventSequence]
          - annotation (1)
          - events (1)
            - [EventOccurrence]
              - entityReference (1)
              - eventReference (1)
              - eventArgument (1)
                - [LiteralValue] file
                  - name = file
                  - comment (2)
                  - content (3)
                    - [Content] path
                      - name = path
                      - value (1)
                        - [LiteralValue] "/home/workspace/data.tplan2x"
                    - [Content] content
                      - name = content
                      - value (1)
                        - [DataReference]
                          - comment (1)
                          - content (1)
                            - [DataInstanceUse]
                              - dataInstance (1)
                                - [StructuredDataInstance] data_content
                                  - name = data_content
                                  - dataType (1)
                                    - [StructuredDataType] STRING
                    - [Content] size
  - finalConditions (1)

```
Test Purpose {
    TP Id TP/2/1/3
    Test objective "Check file copy with mixed data"
    Reference "R012"
    PICS Selection
    Initial conditions
    with {
        the Editor having opened a new window
    }
    Expected behaviour
    ensure that {
        when {
            the User copies the fileWithLiterals
        }
        then {
            the Editor displays a new file
                containing path indicating value "/home/workspace/data.tplan2x" ,
                containing content corresponding to data_content,
                containing size corresponding to data_size
                ;
        }
    }
    Final conditions
    with {
        the Editor closes the new window
            containing file derived from fileWithLiterals with
                PATH indicating value data_path
            ;
    }
}
```



Session 6 Planning

- Goals
  - Review all documents thoroughly
  - Discuss and address any outstanding minor issues
  - Identify and delegate any open issues for Phase 3
  - Update, align, and extend examples for all parts where applicable
  - Discuss feedback from Part 4 evaluation, assist with setup if needed
  - Catch up with Mantis?

Change Logs…

Change Logs…

Change Logs…

Continued Discussion on MM changes, Session #5, Munich
==========

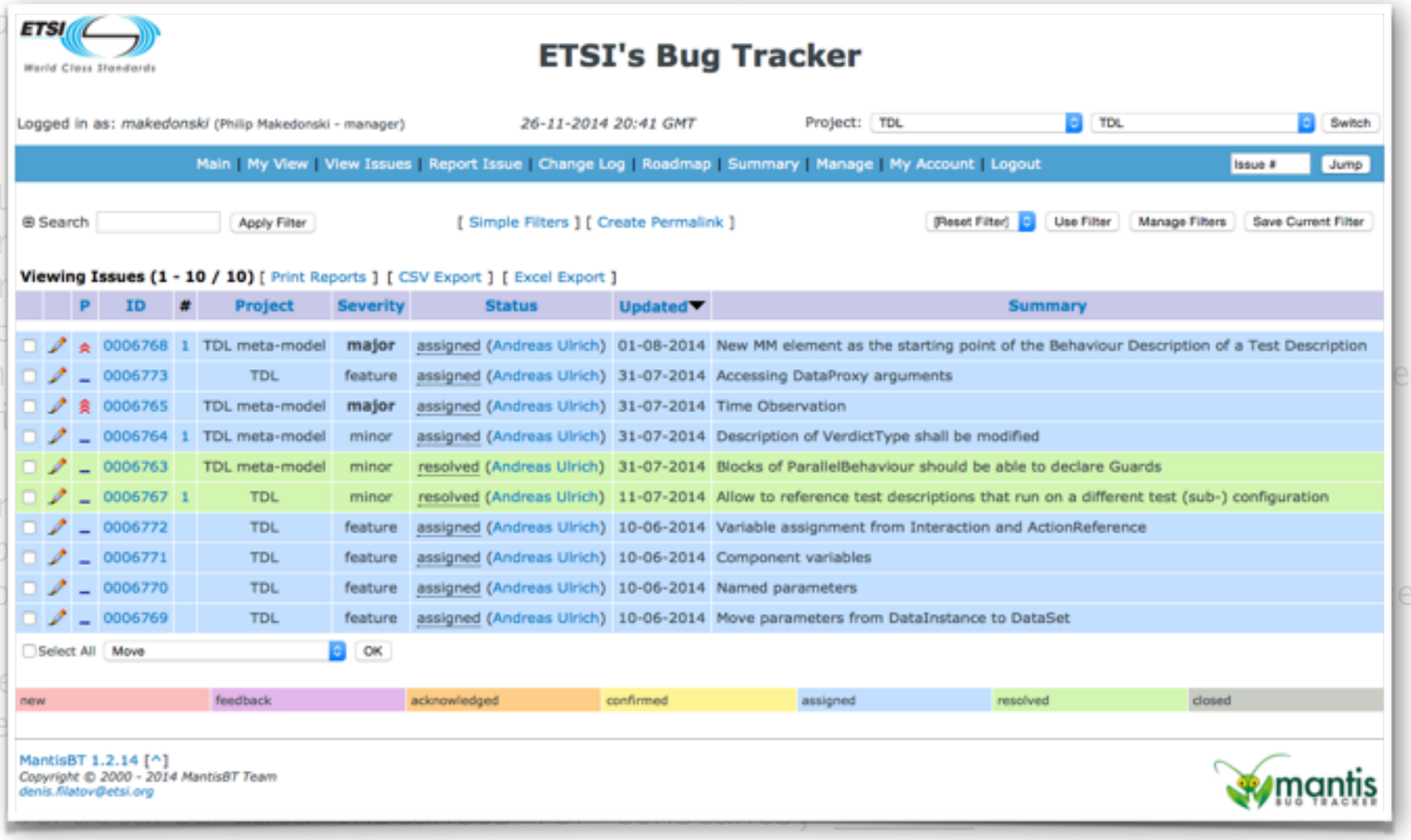
x... changes covered in an updated MM
o... changes partially implemented in MM and/or MM document
!... suggestion not implemented, conflict identified

FOUNDATION
ElementImport
    distinguish between package import and import of elements from one package

DATA
DataSet
    rename to DataType
DataInstance
    shall they be packageable? Yes
AnyValue
    considered only within a DataInstanceSpecification, not as a special DataInstance
TupleElement (open)
    term "tuple" might be not well understood, change to Member
    introduce distinction of mandatory/optional elements, isOptional: Boolean = false
!   --> optional elements stir complexity (requiring functions with data type parameters like isPresent() and distinction of * vs. ?); use case for this feature is not clear; data description shall largely remain in a type system outside of TDL
TupleElementInstance
    change to MemberAssignment
    shall refer to a special kind of DataInstanceSpecification (without variables) instead of a DataInstance
StructuredDataType
    relax tupleElement[1..*] multiplicity to [*]
DataInstanceSpec
    handle nested structured data instances for constants, proxies and function return values
StaticDataInstanceSpec
    add new class to denote static expressions, used when defining structured data instances

Change Logs…

- Determine scope and granularity of changes within current STF
- More diligence and consistent use recommended for the future



The Future of TDL

- ToR for Phase 3 approved!
  - Preparatory meeting, CfE, etc. : TBD
  - Date for additional meeting to discuss Phase 3 : TBD
- Launch event
- TDL logo
  - Provide a list of keywords and ideas to Em by 05.12.2014

Any Other Business?