



Oscar Slotosch, Validas AG

Roadmap towards Development of Qualifyable Eclipse Tools (Eclipse-Project Concept)

Content



- ▶ **Roadmap**
- ▶ Requirements for Tool Qualification (Standards)
- ▶ Proposals for Goals for Eclipse
- ▶ Proposals for some steps towards Tool Qualification
- ▶ Steps on the road
 - First steps: Requirements handling
 - Second steps: Design, Coding and Test
 - Third Steps: Planning Tool Analysis and Life Cycle
 - Fourth Steps: Life Cycle Refined, PSAC Generation, CM
 - **Fifth Steps: Quality Assurance, Qualification Liaison Process, Data**
- ▶ Summary

Roadmap



- ▶ **Identify goals & requirements for tool qualification in Eclipse**
- ▶ **Propose process / project**
- ▶ **Demonstrate tool qualification & improve proposal**
- ▶ **Establish proposal: Qualify (selected) plugins**



- ▶ **Is this a Eclipse project? Not a typical 😊**
- ▶ **Is this an Industrial Working Group process?**

Content



- ▶ Roadmap
- ▶ **Requirements for Tool Qualification (Standards)**
- ▶ Proposals for Goals for Eclipse
- ▶ Proposals for some steps towards Tool Qualification
- ▶ Steps on the road
 - First steps: Requirements handling
 - Second steps: Design, Coding and Test
 - Third Steps: Planning Tool Analysis and Life Cycle
 - Fourth Steps: Life Cycle Refined, PSAC Generation, CM
 - **Fifth Steps: Quality Assurance, Qualification Liaison Process, Data**
- ▶ Summary

Tool Qualification (Summary)



► Standards require tool qualification: ISO 26262, IEC 61508, DO, EN 50128

► Process:

- Classify **all** used tools (Impact, Use-Cases, Artifacts)
- Qualify critical tools
- Use tools

► Qualification Methods ISO 26262

Table 4 — Qualification of software tools classified TCL3

Methods		ASIL			
		A	B	C	D
1a	Increased confidence from use in accordance with 11.4.7	++	++	+	+
1b	Evaluation of the tool development process in accordance with 11.4.8	++	++	+	+
1c	Validation of the software tool in accordance with 11.4.9	+	+	++	++
1d	Development in accordance with a safety standard ^a	+	+	++	++

Here is a hole
were the new
DO-330
standard fits in

► Some tools provide qualification kits for confidence with evidence into

- Correctness of functions by testing them “validation.”
- Development process by documentation
-

Since DO-330 is
scalable, here
could also be a
++

Extension of the ISO 26262?



- **Possible** extension / integration of DO-330 into ISO 26262 could look like:

11.4.10 Development according to a Safety Standard

11.4.10.1 The DO-330 is the first safety standard that is fully applicable to the development of software tools. It is based on Tool Qualification Levels TQL where TQL-1 is the most rigorous level, while TQL-5 is the least one.

11.4.10.2 The mapping from the TCL to the TQL should depend on the SIL level of the system. The mapping is specified in table 4.

ASIL	TCL 1	TCL 2	TCL 3
D	TQL-5	TQL-2	TQL-1
C	TQL-5	TQL-3	TQL-2
B	TQL-5	TQL-4	TQL-3
A	TQL-5	TQL-5	TQL-4

Table 3: Determination of Tool Qualification Levels for DO-330

11.4.10.3 The tool operational requirements, which are the input for tool development according to DO-330, should cover the use cases analysed in clause 11.4.4

- **Similar chapters exist in DO-178C and DO-254**

Table 12-1 Tool Qualification Level Determination

Software Level	Criteria		
	1	2	3
A	TQL-1	TQL-4	TQL-5
B	TQL-2	TQL-4	TQL-5
C	TQL-3	TQL-5	TQL-5
D	TQL-4	TQL-5	TQL-5

- **Extension is not necessary to apply DO-330 in ISO 26262 but could clarify**

Content



- ▶ Roadmap
- ▶ Requirements for Tool Qualification (Standards)
- ▶ **Proposals for Goals for Eclipse**
- ▶ Proposals for some steps towards Tool Qualification
- ▶ Steps on the road
 - First steps: Requirements handling
 - Second steps: Design, Coding and Test
 - Third Steps: Planning Tool Analysis and Life Cycle
 - Fourth Steps: Life Cycle Refined, PSAC Generation, CM
 - **Fifth Steps: Quality Assurance, Qualification Liaison Process, Data**
- ▶ Summary

Goals for Eclipse IWG



- ▶ **Exchange & share knowledge**
 - Motivate developers & community to provide qualifyable plugins
- ▶ **Provide classification support to users of Eclipse tools**
- ▶ **Support the development of qualifyable tools (“Qualification Kits”)**
 - Validation
 - Safety-Standard (DO-330)
- ▶ **Apply this to reference tools ARTOP, EMF,... ?**
- ▶ **Current status (web-page):**

Auto IWG WP5

WP5: Eclipse Qualification Kit (ISO26262)

This is work package 5 of the [Automotive Industry Working Group](#).

- WP Lead: Bredex (temporary)

Need to share knowledge and resources in the classification/qualification activities of eclipse related products.

Current Eclipse Metadata



Overview



General Information

This section describes general information about this plug-in.

ID:	ToolChainAnalyzer
Version:	1.5.3
Name:	%pluginName
Provider:	%providerName
Platform Filter:	
Activator:	Browse...

- ☒ Activate this plug-in when one of its classes is loaded
- ☒ This plug-in is a singleton

Execution Environments

Specify the minimum execution environments required to run this plug-in.

JavaSE-1.6	Add...
	Remove
	Up
	Down

[Configure JRE associations...](#)
[Update the classpath settings](#)

Plug-in Content

The content of the plug-in is made up of two sections:

- [Dependencies](#): lists all the plug-ins required on this plug-in's classpath to compile and run.
- [Runtime](#): lists the libraries that make up this plug-in's runtime.

Extension / Extension Point Content

This plug-in may define extensions and extension points:

- [Extensions](#): declares contributions this plug-in makes to the platform.
- [Extension Points](#): declares new function points this plug-in adds to the platform.

Testing

Test this plug-in by launching a separate Eclipse application:

- [Launch an Eclipse application](#)
- [Launch an Eclipse application in Debug mode](#)

Exporting

To package and export the plug-in:

1. Organize the plug-in using the [Organize Manifests Wizard](#)
2. Externalize the strings within the plug-in using the [Externalize Strings Wizard](#)
3. Specify what needs to be packaged in the deployable plug-in on the [Build Configuration](#) page
4. Export the plug-in in a format suitable for deployment using the [Export Wizard](#)

Vision: Eclipse Classification Data



Qualifiable Features

Available Features

Enumerate all Features for which qualification information is available. Other Features shall not be used in safety relevant contexts.

Use Case Make:Make All (TCL1)

Use Case Make:Make Clean (TCL1)

Use Case Make:Make Executables (TCL1)

Feature Make:Call Tools (TCL1)

Feature Make:Dependencies (TCL1)

Add...

Remove

Properties...

Add Action

Add Class

Add Method

Total: 6

Supported Input / Outputs

For the selected features specify the supported artifacts

Artifact Coverage Report:SVNFile

Artifact Executable

Artifact Library:SVNFile

Artifact Logfile:SVNFile

Artifact Makefile:SVNFile

Artifact Mapfile

Artifact Object Code

Add...

Remove

New...

Errors

For the selected features specify the potential error classes.
The existing errors can be found at [www....](#)

Error Make.Make Executables:Make Builds Wrong Binary (HIGH)

Error Make.Make Executables:Make Modifies Data (HIGH)

Error Make.Make Executables:Old Binary Unchanged (HIGH)

Inferred Feature Error Make Used Wrongly in Call Tools in Make Executables (HIGH)

Inferred Feature Error Make Used Wrongly in Dependencies in Make Executables (HIGH)

Inferred Feature Error Make Used Wrongly in Dependencies in Make PIL in Make Executables (HIGH)

Inferred Feature Error Make Used Wrongly in Dependencies in Make SIL in Make Executables (HIGH)

Up

Down

Overview

Dependencies

Runtime

Extensions

Extension Points

Build

MANIFEST.MF

plugin.xml

build.properties

Qualifiable Features

Qualification Evidence

Validas AG

Page 10

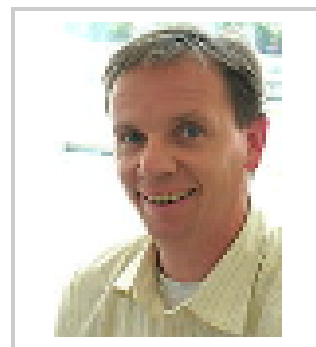
Proposed Role: Eclipse Validator



There is much (different) work to do such that we need a new kind of worker: The Validator

- ▶ Should provide confidence
- ▶ Should be more formalized than a committer
- ▶ Should have qualifications e.g. by filling out questionnaires on
 - Eclipse qualification process
 - DO-330
- ▶ Should have responsibilities (answer to questions)
- ▶ Should earn “credits” for each successful validation action
 - Executed reviews
 - Formulated requirements
 - Created use/test cases
 - Feedback
 - ...

- ▶ **Comparable:**
Confidence in ebay:



slotsch (25 ★)

Positive Bewertungen (der letzten 12 Monate): 100%
[Wie wird der Prozentsatz positiver Bewertungen berechnet?]

Mitglied seit: 01.04.99 in Deutschland

Content



- ▶ Roadmap
- ▶ Requirements for Tool Qualification (Standards)
- ▶ Proposals for Goals for Eclipse
- ▶ **Proposals for some steps towards Tool Qualification**
- ▶ Steps on the road
 - First steps: Requirements handling
 - Second steps: Design, Coding and Test
 - Third Steps: Planning Tool Analysis and Life Cycle
 - Fourth Steps: Life Cycle Refined, PSAC Generation, CM
 - **Fifth Steps: Quality Assurance, Qualification Liaison Process, Data**
- ▶ Summary

Proposals

Elaborate Process

Demonstrate Process



Following activities are necessary to achieve goals:

- ▶ **Agree on focus, e.g. “Metadata extension for qualification information”**
- ▶ **Provide classification support to users of plugins**
 - Use case
 - Potential errors
 - Possible mitigations for errors
 - TCL inference
- ▶ **Provide qualification support**
 - Create checklist for DO-330 requirements (depending on the TQL)
 - Qualification data (general, plugin specific, user adaptable)
 - Requirements (general, development, operational)
 - Check Eclipse against the checklist, create
 - Mapping of Eclipse -> DO-330
 - Identify gaps: missing data/requirements
 - Provide model (EMF?) for the missing data
- ▶ **Demonstrate it: Small example e.g. EclipseCon**
- ▶ **Validate it: bigger example**

Content



- ▶ Roadmap
- ▶ Requirements for Tool Qualification (Standards)
- ▶ Proposals for Goals for Eclipse
- ▶ Proposals for some steps towards Tool Qualification
- ▶ Steps on the road
 - **First steps: Requirements handling**
 - Second steps: Design, Coding and Test
 - Third Steps: Planning Tool Analysis and Life Cycle
 - Fourth Steps: Life Cycle Refined, PSAC Generation, CM
 - **Fifth Steps: Quality Assurance, Qualification Liaison Process, Data**
- ▶ Summary

First Steps on the Road



- ▶ **Create a checklist to show the DO-330 compliance**
- ▶ **Make a/some simple example tool(s) that shall comply with DO-330**
- ▶ **Work on selected topics: Requirements, Test, Code, ...**
 - Analyze existing Eclipse process
 - Analyze possibilities for the topic e.g. RIF, tracing, tests,...
 - Create example document (eventually based on existing methods)
 - Check DO-330 compliance
 - Create model (for creation of document)
 - Review/Validate for:
 - Expressiveness
 - practicability
 - possible improvements
 - Make proposal for Eclipse integration (part)
- ▶ **Until DO-330 is completely satisfiable**
- ▶ **Make integrated proposal for Eclipse Extension (EMF,..)**

Checklist for DO-330 compliance (refined)



- ▶ Document created: “How-To Qualify Eclipse-based Tools”
- ▶ Contains Requirements (IDs) and tracing to Process/Documents/Model

The screenshot shows a Microsoft Word document titled "HowToQualifyEclipseBasedTools.doc" in Kompatibilitätsmodus. The document is structured with a table of contents on the left and a detailed table of requirements and their satisfaction comments on the right.

Table of Contents (Left):

- 1 Document History
- 2 Definitions
- 3 Tool Qualification Process
- 4 Traceability to DO-330
 - 4.1 General Considerations
 - 4.2 Tracing to Tool Qualification Planning Process Sect...
 - 4.3 Tracing to Tool Development Life Cycle and Process...
 - 4.4 Tracing to Tool Verification Process Section
 - 4.5 Tracing to Tool Configuration Management Proces...
 - 4.6 Tracing to Tool Quality Assurance Process Section
 - 4.7 Tracing to Tool Qualification Liaison Process Section
 - 4.8 Tracing to Tool Qualification Data Section
 - 4.9 Tracing to Additional Considerations for Tool Qua...
 - 4.10 Tracing to Tool Qualification Objectives Section
- 5 References

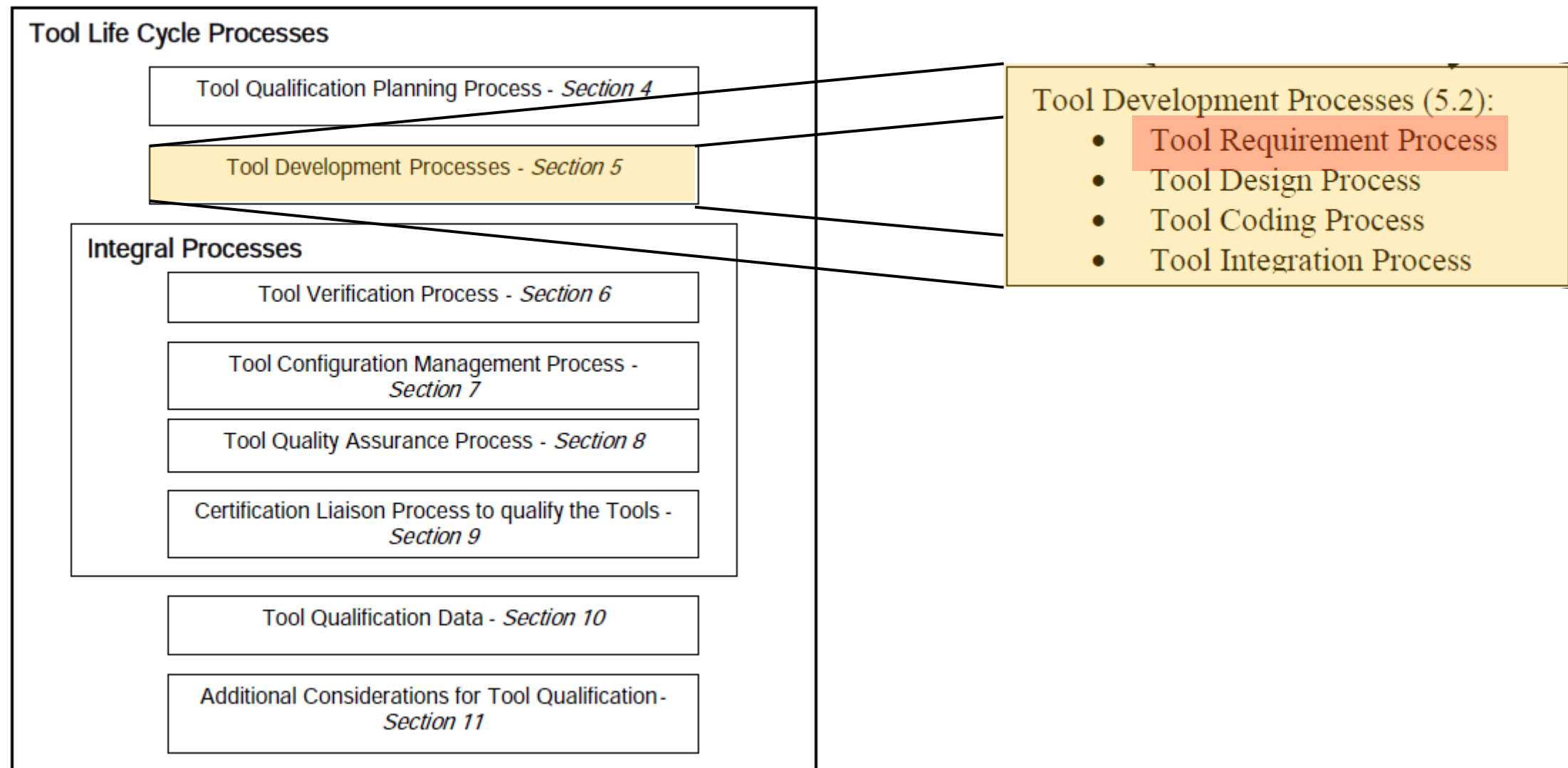
Table of Requirements (Right):

Identifier	Keyword	Satisfaction Comment
DO-330-4.1	Qualification Need	Satisfied by satisfying sub-items
DO-330-4.1.a	Identification	The identification (plugin/product name) of Eclipse products and plugins is reused
DO-330-4.1.b	Intended Use	Is done in the TORs model for the main plugin of the tool model (see section 4.3.1 in [TDP])
DO-330-4.1.c	Qualification Need	See Tool-Analysis part in the model (see section 4.2.2 and 4.2.3 in [TDP])
DO-330-4.1.d	TQLs	See Determination in section 4.2.4 in [TDP]
DO-330-4.1.e	Stakeholders	See "Provider" in MANIFEST.MF and "Validator" in project model in section 4.1.1 of [TDP] and the Validator in the verification data model (see section 4.2.2 in [TVP])
DO-330-4.1.f	Tool Environment Definition	The Environment is defined using the <u>TORContext</u> requirements, see section 4.3.1.4 in [TDP]
DO-330-4.2	Life cycle	See sections 5 and 5.1 in [TDP]
DO-330-4.2.a	Planning process	See section 5.1.1 in [TDP]
DO-330-4.2.b	Development process	See section 5.1.2 in [TDP]
DO-330-4.2.c	Integral process	See section 5.1.3 in [TDP]
DO-330-4.2.1	Transition criteria	See section 5.3 in [TDP]

DO-330 Topics



► Structure of DO-330



Existing Methods: Requirements



- ▶ Currently not practiced in Eclipse
- ▶ RMF / ProR (Incubation):



Specification Document			
	ID	Description	Link
1	REQ-1	Dies ist eine Demo von ProR	0 ▷ REQ ▷ 2
	▷		REQ-5
	▷	Links können auch Attribute haben.	REQ-6
1.1	REQ-2	Hierarchien beliebiger Tiefe werden unterstützt.	
1.2	REQ-3	Der Linke Rand hilft bei der Orientierung	
1.2.1	REQ-4	... und die erste Spalte wird eingerückt.	
2	REQ-5	Im Properties-View werden alle Attribute angezeigt.	1 ▷ REQ ▷ 0
3	REQ-6	Im Editor nur die, die man sehen will.	1 ▷ REQ ▷ 0

- ▶ general approach, not tailored for tool requirements
- ▶ Adoptable to tool requirements by creating corresponding requirements types
- ▶ First Investigation
 - Nice usability e.g. for creating new requirements
 - Polymorphic links (any requirement can be linked)
 - Extensible to design / test / ...?
 - Do we need RIF within Eclipse?

Create DO-330 Conformant Example



1	Document History
2	Definitions
3	General Information
4	Tool Operational Requirements (Use Cases)
4.1	Functional Requirements
4.1.1	Tool Chain Analysis
4.1.2	Tool Analysis
4.1.3	Report Generation
4.2	Context Requirements
4.2.1	Environment
4.3	Format Requirements
4.3.1	Models
4.3.2	Reports
4.4	Assumptions
4.4.1	Model Validation
4.4.2	Report Review
5	Tool Requirements (Features)
5.1	User Instructions
5.1.1	User Manual
5.2	Operation Modes
5.2.1	Single-User Mode
5.2.2	Restricted Multi-User Mode
5.3	Tool Functions
5.3.1	Modeling of Tool Chains
5.3.2	Computation of the TCLs
5.3.3	Generic Error Model
5.3.4	Report Generation
5.3.5	Model Validation
5.4	Customization Requirements
5.4.1	Stack Size
5.4.2	Heap Size
5.5	Tool Interface Requirements
5.5.1	Graphical User Interface
5.5.2	File Interface
5.5.3	DOT Interface
5.5.4	Excel Interface
5.6	Expected Error Message
5.6.1	Syntactical Inconsistent Models
5.6.2	Internal Error Messages
5.6.3	Log-Files
5.7	Robustness Requirements
5.7.1	Operating Systems
5.7.2	Model Size
5.8	Performance Requirements

Tool Requirements for Tool Chain Analyzer
Version 0.2

Validas AG

3 General Information

This section contains the general information on the Tool Chain Analyzer generated from the corresponding plugin metadata

- Name: Tool Chain Analyzer
- ID: de.validas.toolchainanalyzer
- Version: 1.5.3
- Provider: Validas AG
- Tool Qualification Level (TQL): TQL-1

from
MANIFEST.MF

Overview

General Information

This section describes general information about this plug-in.

ID:	de.validas.toolchainanalyzer
Version:	1.5.3
Name:	Tool Chain Analyzer
Provider:	Validas AG
Qualification Level	TQL-1



From Tool
Requirements
Model

5.4 Customization Requirements

The tool shall be customized to the resources of the computer where it is executed.

5.4.1 Stack Size

The stack size shall be settable. The default stack size should be 400 MB

5.4.2 Heap Size

The heap size shall be settable. The default heap size should be 1000 MB.

5.5 Tool Interface Requirements

The tool chain analyzer shall have the following interfaces.

5.5.1 Graphical User Interface

The graphical user interface consists of different views and property dialogs.

5.5.1.1 Structure View

The structure view represents the tool chain models in a tree view with the structure how the elements are modeled. The structure views also contains the actions to create, move and delete elements. Furthermore it can be used to start actions like the import and export of tool models.

5.5.1.2 Property View

The property view shows the properties (attributes and relations) of the elements selected in the tree view. They can be edited either directly in the view or in property dialogs that start when the elements are double-clicked.

5.5.1.3 Property Dialogs

The property dialogs are used to edit long text fields or complex relations in the modeled elements. They are started from the property view.

5.5.1.4 Flow View

The flow view shows the information flows within the model, e.g. from one tool to another via the artifact that is written and read. Furthermore the error derivation flow from the general error model to the features and use cases.

5.5.2 File Interface

The tool chain models shall be persistent to files. The tool chain analyzer loads models from files and writes the back into files.

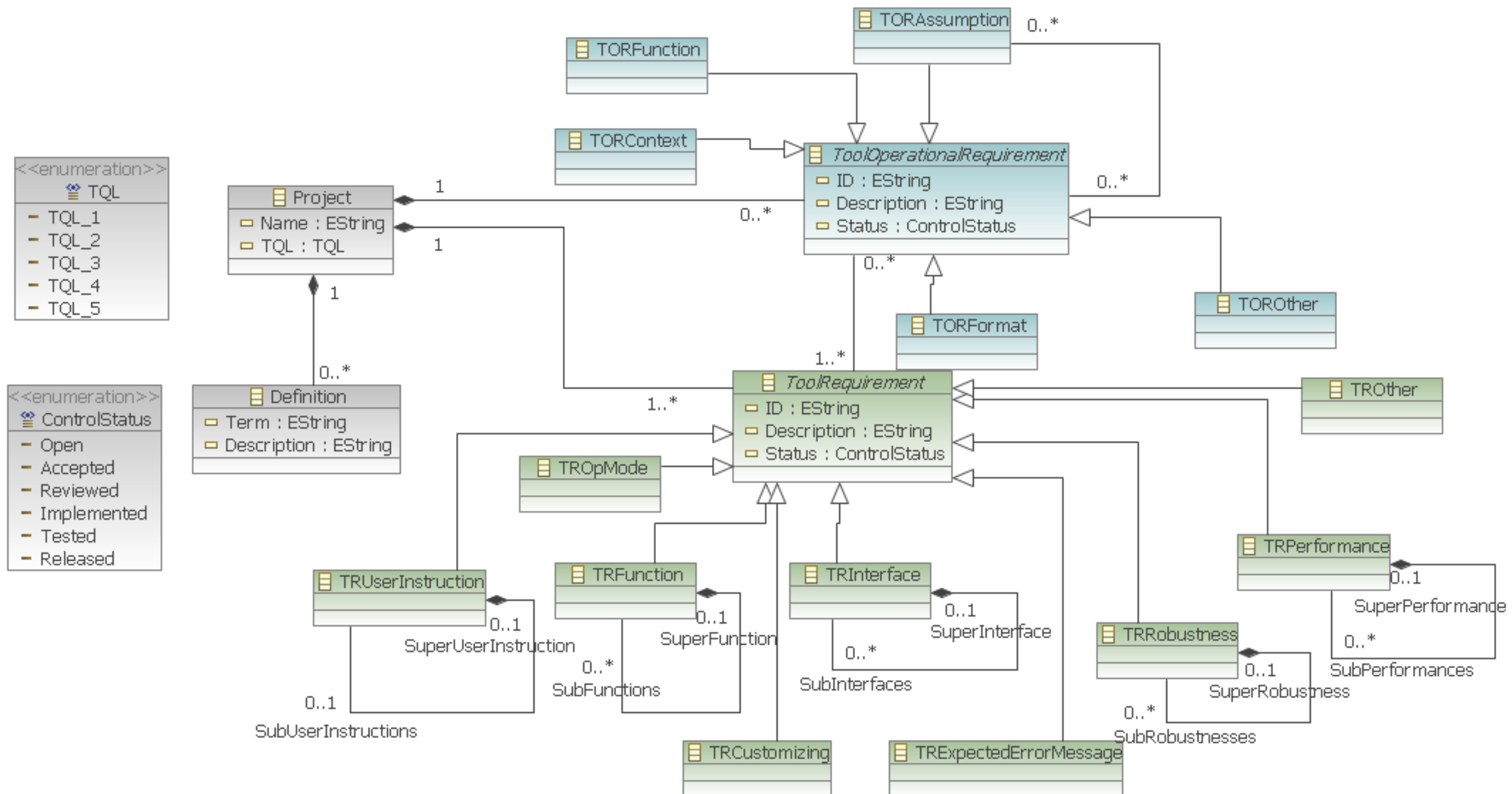
5.5.3 DOT Interface

For drawing the images to explain the error flow in the model the graphviz tool with the DOT language. The intermediate files are accessible and can be modified or integrated into other images.

Create Model for Tool-Requirements



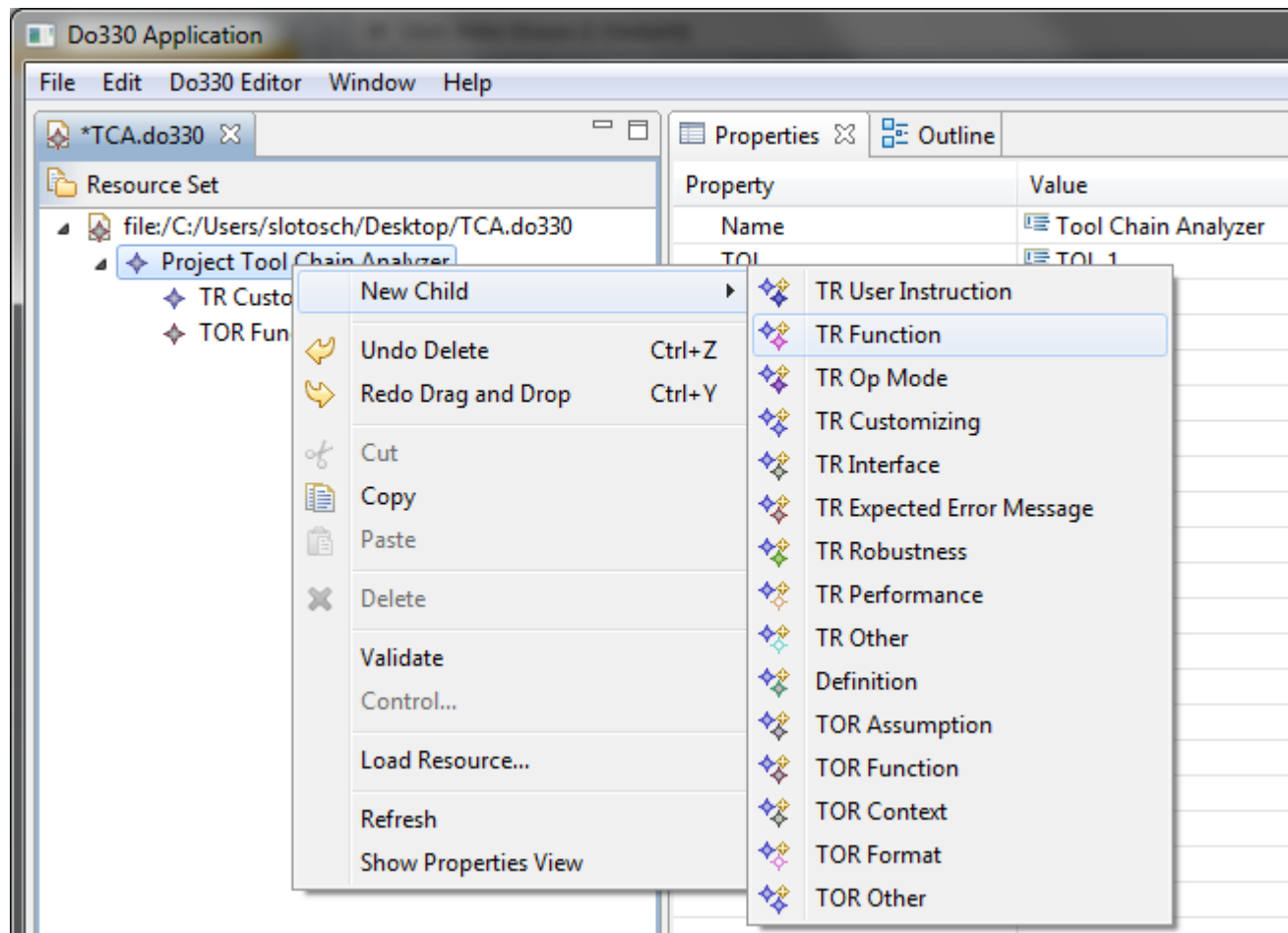
► EMF-Metamodel (Draft) for Tool Requirements



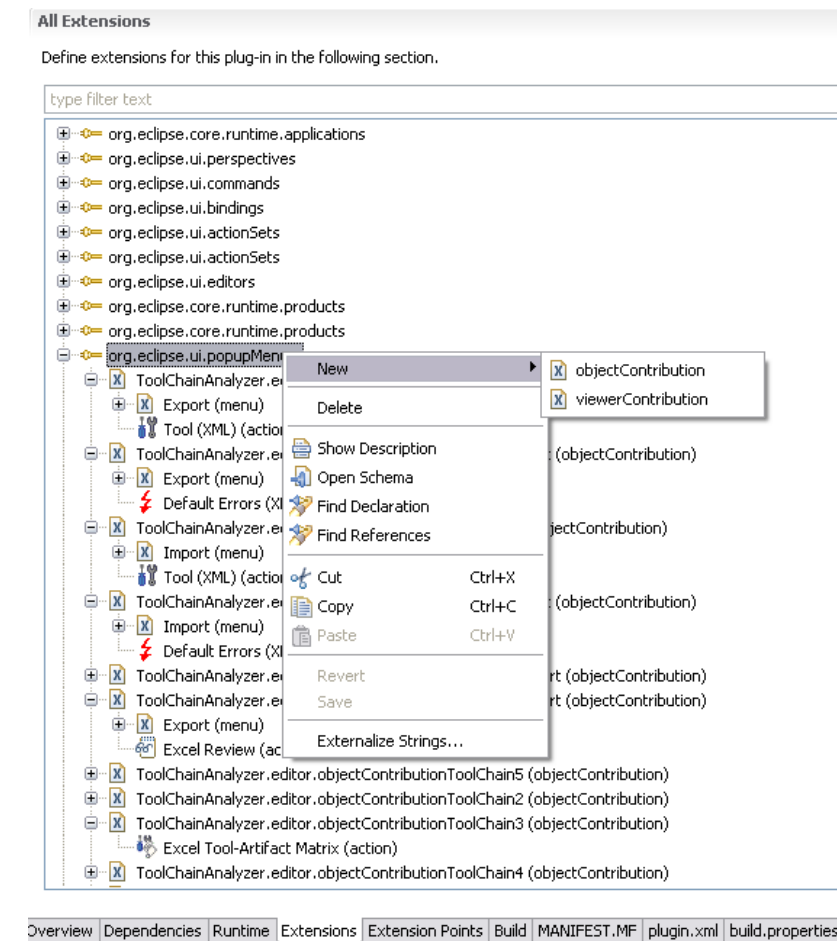
Create Example Model



► Using the default EMF Editor



Comparable: Plugin Extension



- Shows how simple requirements could be created with Eclipse
- The example (DO-330 conforming) document can be generated completely from the model
- Tracing: TOR <-> TR is done using Eclipse association editors

Content



- ▶ Roadmap
- ▶ Requirements for Tool Qualification (Standards)
- ▶ Proposals for Goals for Eclipse
- ▶ Proposals for some steps towards Tool Qualification
- ▶ Steps on the road
 - First steps: Requirements handling
 - **Second steps: Design, Coding and Test**
 - Third Steps: Planning Tool Analysis and Life Cycle
 - Fourth Steps: Life Cycle Refined, PSAC Generation, CM
 - **Fifth Steps: Quality Assurance, Qualification Liaison Process, Data**
- ▶ Summary

Design and Coding (5.2.2. and 10.2.2)

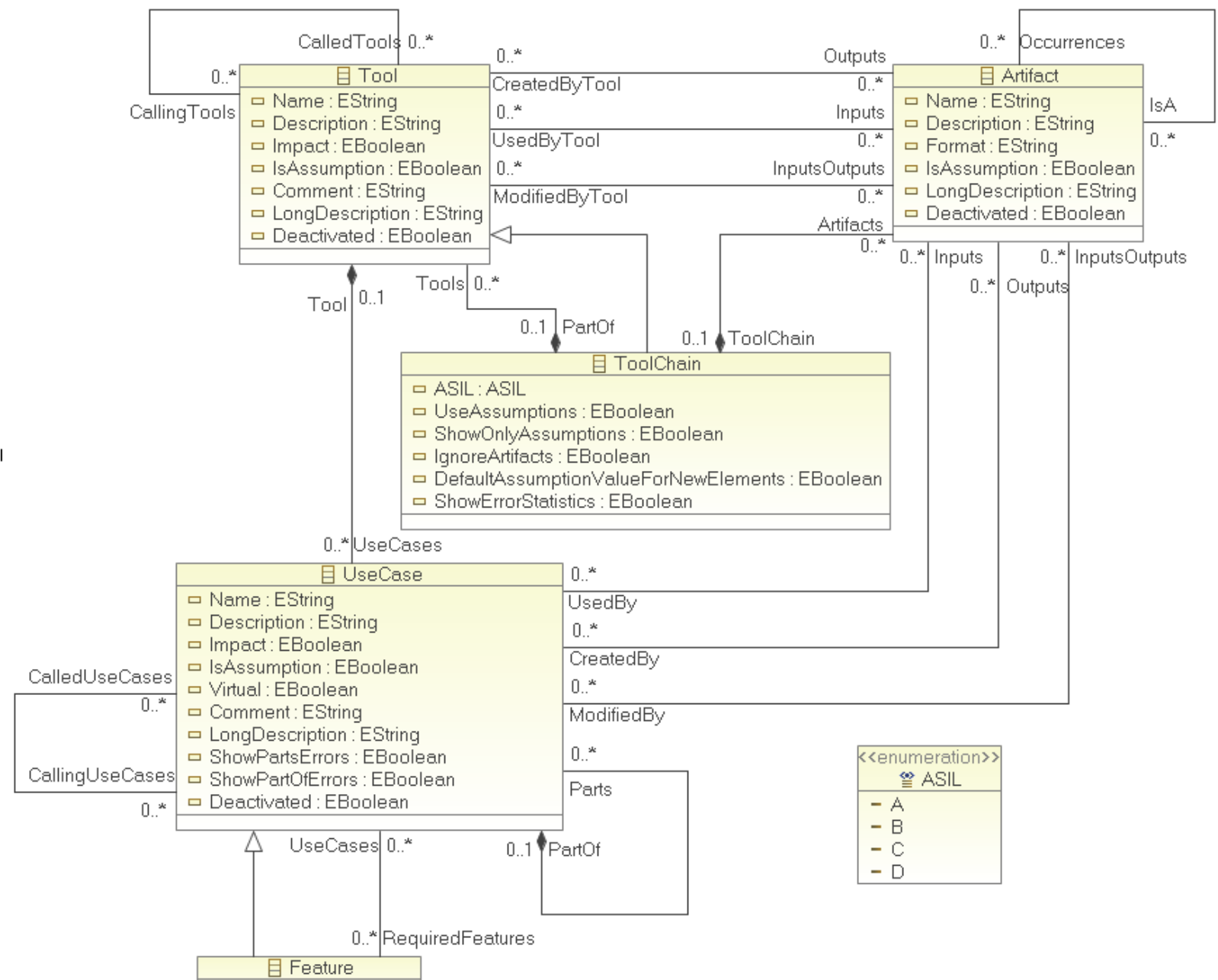
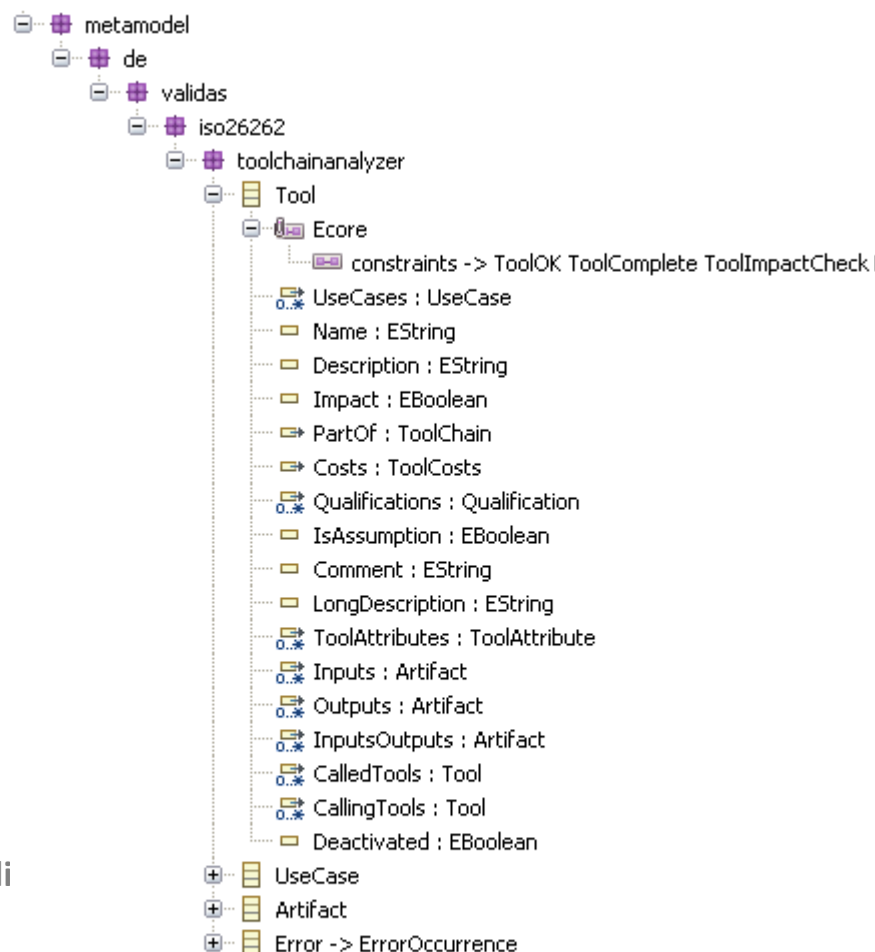
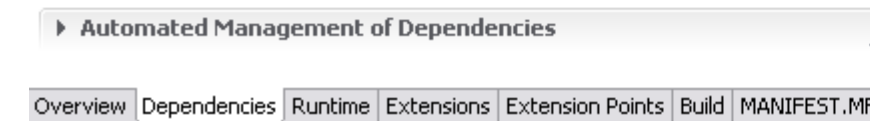
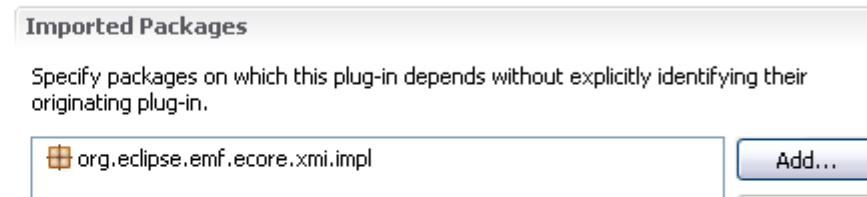
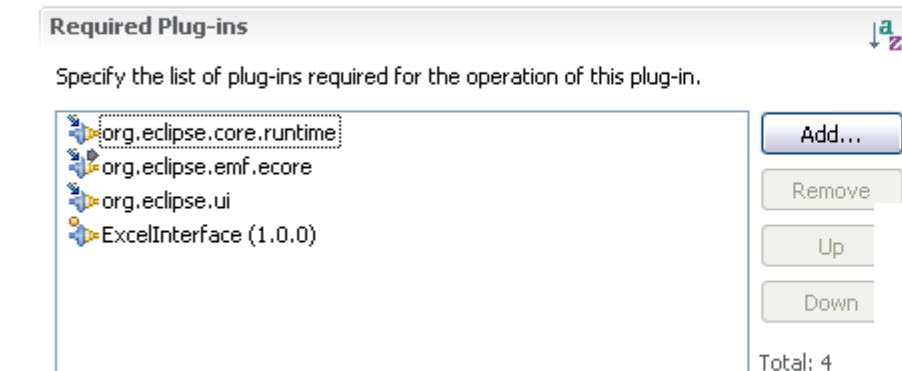
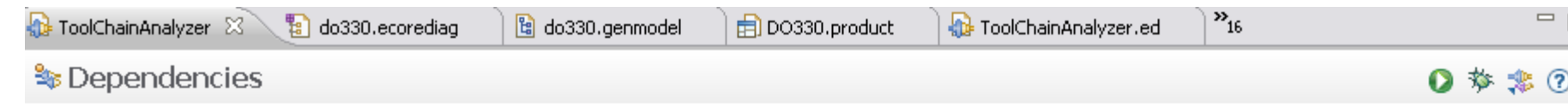


- ▶ **Design = Architecture + Low Level Requirements (LLRs)**
- ▶ **Design Description:**
 - Tool architecture: tool structure to implement TR
 - Detailed Description how TRs are allocated in architecture
 - Input/output description of architecture elements
 - Data & control flow
 - Scheduling procedures
 - Protection (if used)
 - Used components (incl. baselines)
 - LLRs including tracing to TR
 - Derived LLRs (not traceable to TRs)
 - Justification required including
 - No negative impact to TORs and TRs
- ▶ **Verifiable and consistent**
- ▶ **Compliant to standards**

Architecture Examples in Eclipse



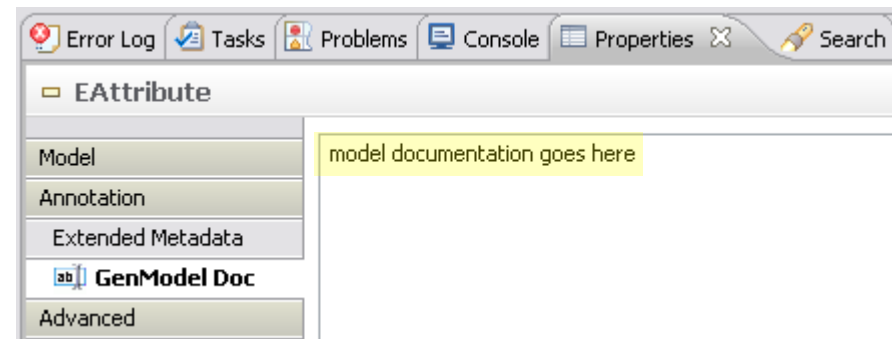
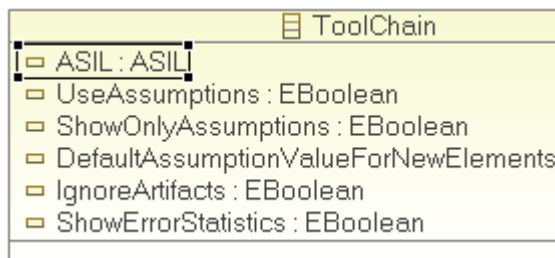
► Architecture: Plugins & packages, EMF models, xText grammars



Example EMF Code



- Model generates code, interface (and description!)



- Documentation can be inserted into code and model

```
/**
 * Returns the value of the '<em><b>ASIL</b></em>' attribute.
 * The literals are from the enumeration {@link metamodel.de.validas.iso26262.toolchainanalyzer.ASIL}.
 * <!-- begin-user-doc -->
 * <p>
 * here is the specific code description of the return value '<em>ASIL</em>'
 * </p>
 * <!-- end-user-doc -->
 * <!-- begin-model-doc -->
 * model documentation goes here
 * <!-- end-model-doc -->
 *
 * @return the value of the '<em>ASIL</em>' attribute.
 * @see metamodel.de.validas.iso26262.toolchainanalyzer.ASIL
 * @see #setASIL(ASIL)
 * @see metamodel.de.validas.iso26262.toolchainanalyzer.ToolchainanalyzerPackage#getToolChain_ASIL()
 * @model
 * @generated
 */
ASIL getASIL();
```

Low Level Requirements



- ▶ Can be directly implemented
- ▶ Not the code but it's detailed descriptions
 - Class: Name, super classes, visibility, interfaces, exceptions, purpose
 - Methods: Name, parameters, types, exceptions, visibility, purpose
 - Variables: Name, type, visibility, purpose
 - Contributions:
 - Actions
 - Menus
 - ShortCuts
 - Separators
 - ...
- ▶ Currently:
 - tags & templates in the code
 - No tracing to requirements possible (due to missing requirements?)

```
/**
 * <copyright>
 * Validas AG
 * </copyright>
 * This class is the Editor Advisor g
 * @author: Oscar Slotosch, Reinhard
 * TODO: review low level requirement:
 * @link generated from de.validas.to
 *
 * $Id$
 */
```

```
package metamodel.de.validas.iso26262
```

```
import java.io.File;
```

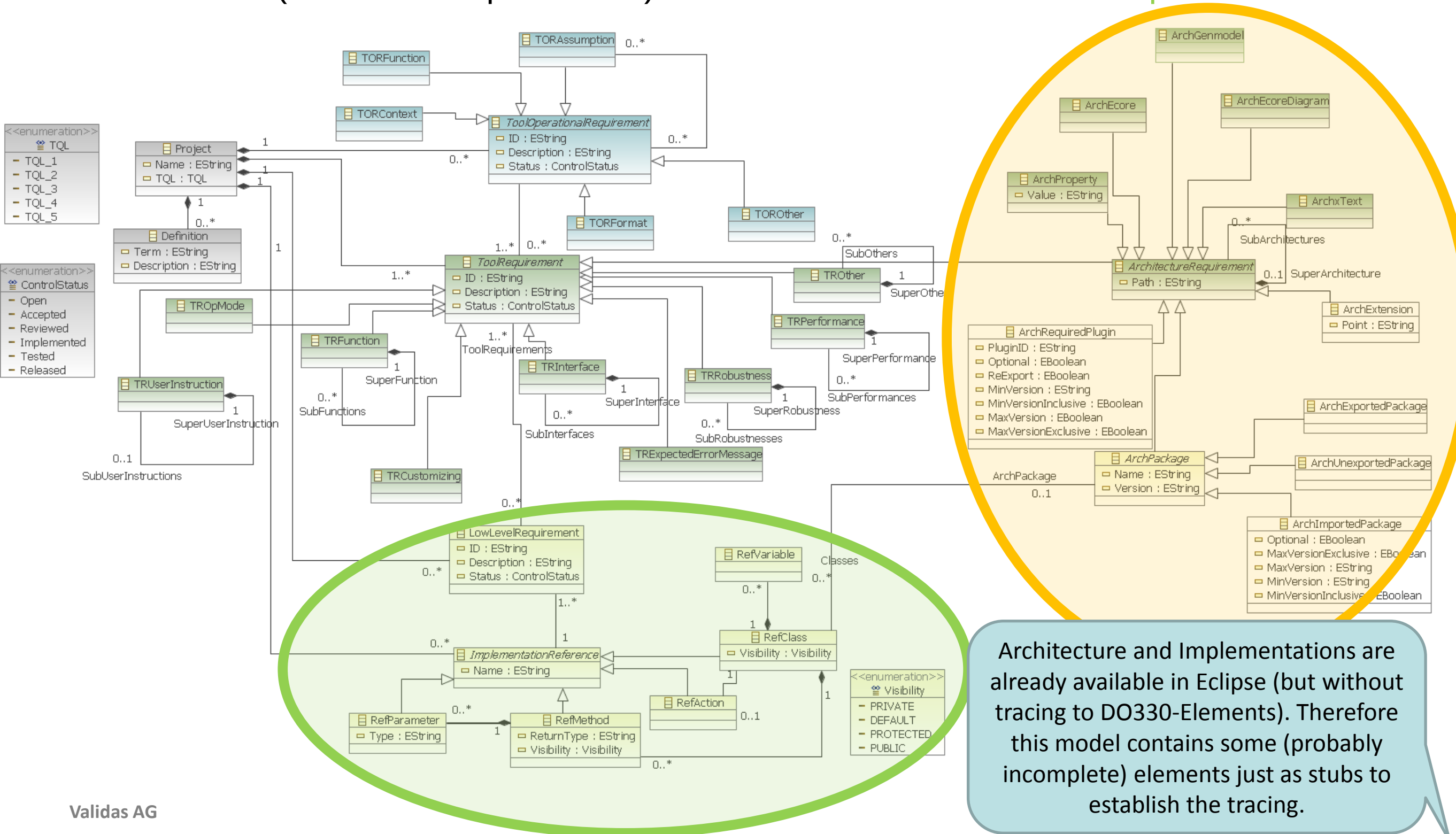
```
/**
 * Customized {@link WorkbenchAdvisor
 * <!-- begin-user-doc -->
 * RJ: this class has been generated
 * <!-- end-user-doc -->
 *
```

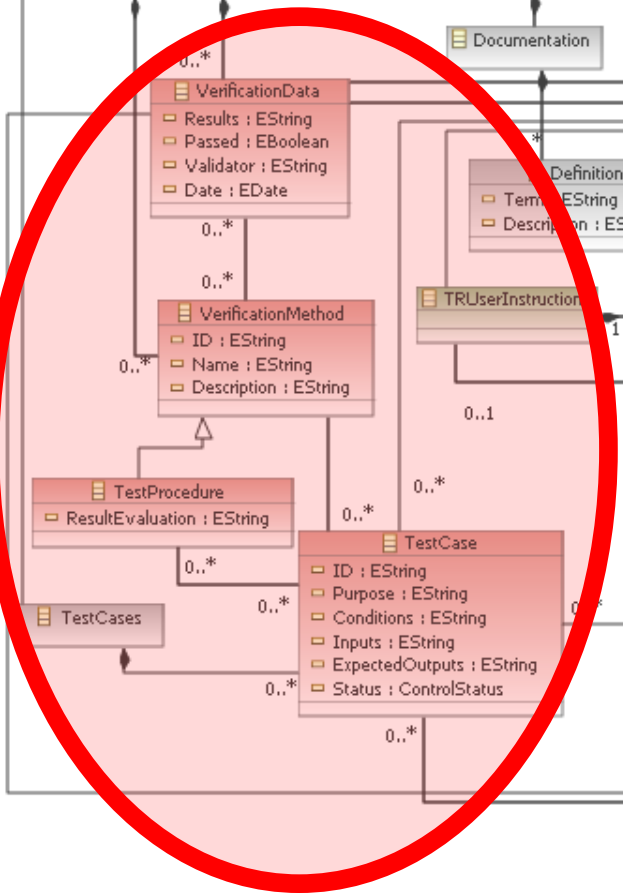
```
 * @requirements
```

```
 * @
 * @
 * @author - author name
 * @author
 * @category
 * @deprecated
 * @see
 * @serial
 * @since
 * @version
 * {@code}
 * {@docRoot}
```

```
publ
Press 'Ctrl+Space' to show Template Proposals
```

The design model extends the requirements model by **Architecture** (also Tool Requirements) and **LLRs** with references to **Implementation**





Test Implementation



```
*Test_AssumptionModel X ToolInterfaceTest.java ToolChainAnalyzer AssumptionModel.java ReflectiveCallab
13 import metamodel.de.validas.iso26262.toolchainanalyzer.UseCase;
14 import metamodel.de.validas.iso26262.toolchainanalyzer.impl.ToolchainanalyzerFactoryImpl;
15
16 import org.junit.Before;
17 import org.junit.Test;
18
19 public class Test_AssumptionModel {
20
21     private ToolchainanalyzerFactory fac;
22
23     @Before
24     public void setUp() {
25         fac = new ToolchainanalyzerFactoryImpl();
26     }
27
28     @Test
29     /**
30      * This test checks if AssumptionModel.getIsAssumption works as
31      * specified in @LLR 5.1.1.1 Method getIsAssumption
32      *
33      * The applied Test Procedures are @see 4.1 CodeCover,
34      * @see 4.2 JUnit, @see 4.4 Factory-Based Model Testing
35      *
36      * @author dirix, slotosch
37      */
38     public void getIsAssumptionTest() {
39         //testing assumption handling of errors
40         Error noAssumptionError = fac.createError();
41         Error isAssumptionError = fac.createError();
42         Error noAssumptionError2 = fac.createError();
43         noAssumptionError.setIsAssumption(false);
44         noAssumptionError2.setIsAssumption(false);
45         isAssumptionError.setIsAssumption(true);
46         assertFalse(AssumptionModel.getIsAssumption(noAssumptionError));
47         assertTrue(AssumptionModel.getIsAssumption(isAssumptionError));
48
49         UseCase isAssumptionUseCase = fac.createUseCase();
50         isAssumptionUseCase.setIsAssumption(true);
51         UseCase noAssumptionUseCase = fac.createUseCase();
```


Test Execution



ToolChainAnaly

src dirix 888

metamo

Artef

Assu

Attrib

DOT

DOT

DOT

Expla

Hiera

TCLC

TCLC

Tool

Tool

metamo

metamo

metamo

metamo

metamo

testsrc

metamo

Test

JRE System Library [JavaSE-1.6]

Refactor Alt+Shift+I

Import...

Export...

References

Declarations

Find Bugs

Refresh F5

Assign Working Sets...

Use For Coverage Measurement

Run As

Debug As

Profile As

Team

Compare With

Replace With

Restore from Local History...

Properties Alt+Enter

new ToolchainAnalyzerFactoryImpl();

est checks if AssumptionModel.getIsAssumption w

ed in @LLR 5.1.1.1 Method getIsAssumption

lied Test Procedures are @see 4.1 CodeCover,

2 JUnit, @see 4.4 Factory-Based Model Testing

dirix, slotosch

1 CodeCover Measurement For JUnit

2 JUnit Plug-in Test Alt+Shift+X, P

3 JUnit Test Alt+Shift+X, T

Run Configurations...

ptionError2.setIsAssumption(false);

ptionError.setIsAssumption(true);

False(AssumptionModel.getIsAssumption(noAssump

True(AssumptionModel.getIsAssumption(isAssumpt

UseCase isAssumptionUseCase = fac.createUseCase();

isAssumptionUseCase.setIsAssumption(true);

Test Coverage



getAllAssumptions	0,0 %	0,0 %	0,0 %	0,0 %	-	-
getIsAssumption	100,0 %	100,0 %	-	100,0 %	-	-
getIsRestricted	0,0 %	0,0 %	-	0,0 %	-	-

☒ Show methods with

Term Coverage

>

 90,5 %

Name	Statement	Branch	Loop	Term	?-Operator	Synchronized
ToolChainAnalyzer	6,1 %	5,3 %	0,0 %	4,6 %	-	-
metamodel	6,1 %	5,3 %	0,0 %	4,6 %	-	-
de	6,1 %	5,3 %	0,0 %	4,6 %	-	-
validas	6,1 %	5,3 %	0,0 %	4,6 %	-	-
iso26262	6,1 %	5,3 %	0,0 %	4,6 %	-	-
checks	6,1 %	5,3 %	0,0 %	4,6 %	-	-
AssumptionModel	6,1 %	5,3 %	0,0 %	4,6 %	-	-
getIsAssumption	100,0 %	100,0 %	-	100,0 %	-	-

```

* thoses Use Cases that are no assumptions and the Assumed Uses Cases are
* returned only in case it is allowed
*/
public class AssumptionModel {

    /**
     * returns the IsAssumption for an item, ensures that assumption settings
     * are "inherited" from Tool->UseCae->Error->...
     */
    public static boolean getIsAssumption(Object item) {
        if (item instanceof Error) {
            Error uce = (Error) item;
            return uce.isIsAssumption() || (uce.getUseCase() != null && getIsAssumption(uce.getUseCase()))
                || (uce.getRestriction() != null && getIsAssumption(uce.getRestriction()))
                || (uce.getCheck() != null && getIsAssumption(uce.getCheck()));
        }
        if (item instanceof Check) {
            Check ck = (Check) item;
            return ck.isIsAssumption() || getIsAssumption(ck.getUseCase());
        }
        if (item instanceof Qualification) {
            Qualification qual = (Qualification) item;
            return qual.isIsAssumption() || (qual.getUseCase() != null && getIsAssumption(qual.getUseCase()))
                || (qual.getTool() != null && getIsAssumption(qual.getTool()));
        }
        if (item instanceof Restriction) {
            Restriction res = (Restriction) item;
            return res.isIsAssumption() || getIsAssumption(res.getUseCase());
        }
        if (item instanceof UseCase) {
            UseCase uc = (UseCase) item;
            return uc.isIsAssumption() || getIsAssumption(uc.getTool());
        }
    }
}

```

Content



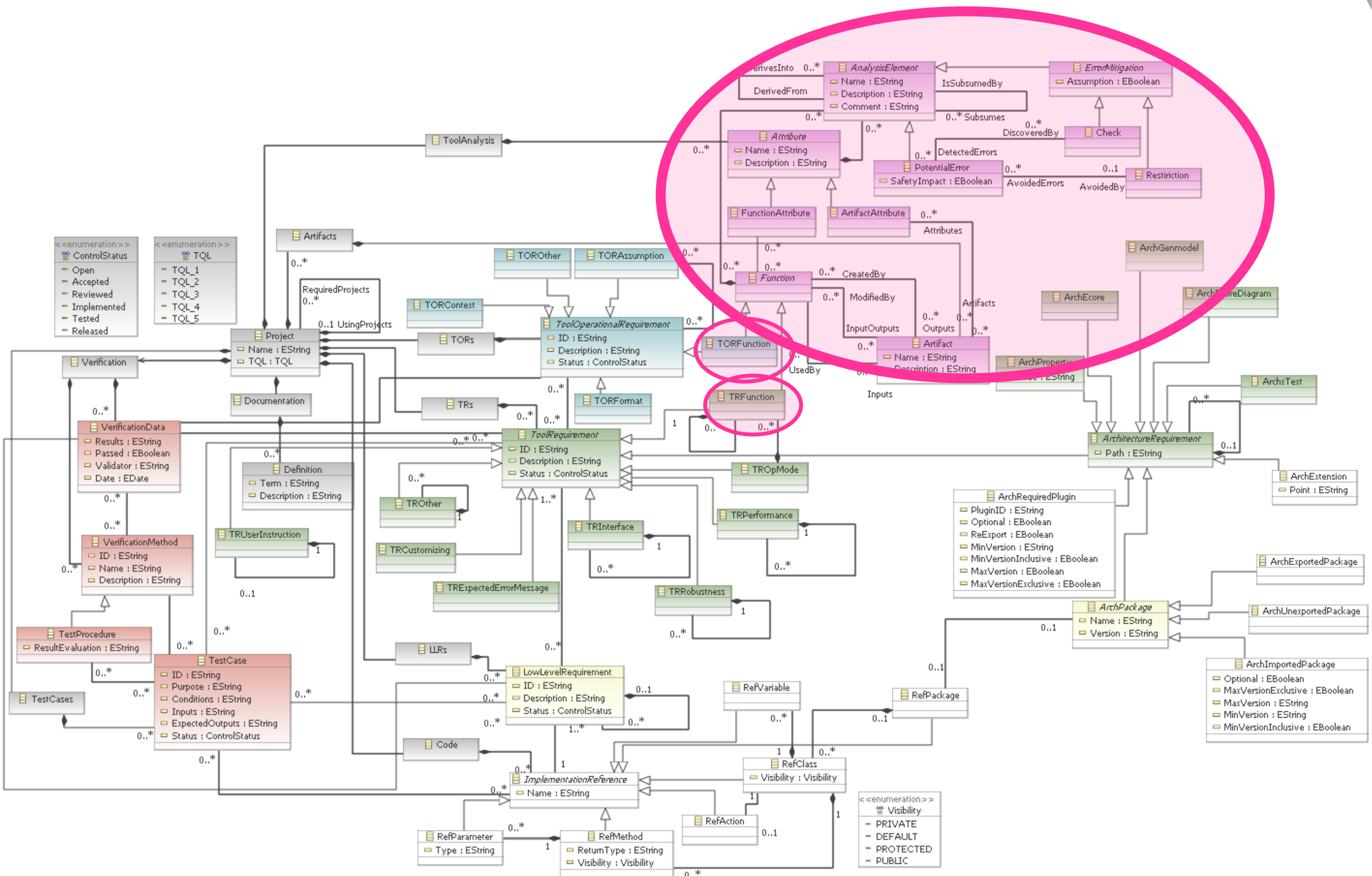
- ▶ Roadmap
- ▶ Requirements for Tool Qualification (Standards)
- ▶ Proposals for Goals for Eclipse
- ▶ Proposals for some steps towards Tool Qualification
- ▶ Steps on the road
 - First steps: Requirements handling
 - Second steps: Design, Coding and Test
 - **Third Steps: Planning Tool Analysis and Life Cycle**
 - Fourth Steps: Life Cycle Refined, PSAC Generation, CM
 - **Fifth Steps: Quality Assurance, Qualification Liaison Process, Data**
- ▶ Summary

Planning: Tool Analysis for PSAC



- ▶ **Determines “qualification needs” of used tools**
- ▶ **Qualification Need: “Required Confidence => Tool Qualification Level (TQL)”**
- ▶ **Required Confidence (and mapping to TQL) depends on domains**
 - DO-178C: Criteria 1 – Criteria 3
 - ISO: Tool Confidence Level based on Error Analysis in Use Cases
 - IEC 61508: Tool Classification: T1 – T3
- ▶ **Tool Chain Analysis Method (RECOMP) can be applied in all domains to determine the Required Confidence**
- ▶ **Simple Tool Chain Analysis Model has been added to DO-330 meta model**
- ▶ **Connections to existing model (“TORFunction”, “TRFunction”)**

Planning: Analysis Model for PSAC





Version 0.2

- 1 Document History
- 2 Definitions
- 3 Tool Qualification Process
- 4 Traceability to DO-330
 - 4.1 General Considerations
 - 4.2 Tracing to Tool Qualification Planning Process Section
 - 4.3 Tracing to Tool Development Life Cycle and Process Section
 - 4.4 Tracing to Tool Verification Process Section
 - 4.5 Tracing to Tool Configuration Management Process Section
 - 4.6 Tracing to Tool Quality Assurance Process Section
 - 4.7 Tracing to Tool Qualification Liaison Process Section
 - 4.8 Tracing to Tool Qualification Data Section
 - 4.9 Tracing to Additional Considerations for Tool Qualification Section
 - 4.10 Tracing to Tool Qualification Objectives Section
- 5 References

- ▶ **General explanations**
- ▶ **Conformance to DO-330 (bidirectional Tracing)**
 - Structures according to DO-330
 - Identification of Requirements
 - Tables for Tracing
 - Tracing against IDs is also contained in other Documents like TDP, TVP,...
- ▶ **Bidirectional tracing ensures that not too much is models/requested within Eclipse qualification process**



Tool Development Plan

for every

Qualifiable Eclipse Plugin

Version 0.6

If the tool plans have been adopted the compliance to DO-330 has been achieved by updating the tracing in [HowTo] and their associated test cases for the tool projects. Satisfies DO-330-T1-6, DO-330-T1-7, DO-330-T1-8, DO-330-T1-9, DO-330-T1-10, DO-330-T1-11, DO-330-T1-12, DO-330-T1-13, DO-330-T1-14, DO-330-T1-15, DO-330-T1-16, DO-330-T1-17, DO-330-T1-18, DO-330-T1-19, DO-330-T1-20, DO-330-T1-21, DO-330-T1-22, DO-330-T1-23, DO-330-T1-24, DO-330-T1-25, DO-330-T1-26, DO-330-T1-27, DO-330-T1-28, DO-330-T1-29, DO-330-T1-30, DO-330-T1-31, DO-330-T1-32, DO-330-T1-33, DO-330-T1-34, DO-330-T1-35, DO-330-T1-36, DO-330-T1-37, DO-330-T1-38, DO-330-T1-39, DO-330-T1-40, DO-330-T1-41, DO-330-T1-42, DO-330-T1-43, DO-330-T1-44, DO-330-T1-45, DO-330-T1-46, DO-330-T1-47, DO-330-T1-48, DO-330-T1-49, DO-330-T1-50, DO-330-T1-51, DO-330-T1-52, DO-330-T1-53, DO-330-T1-54, DO-330-T1-55, DO-330-T1-56, DO-330-T1-57, DO-330-T1-58, DO-330-T1-59, DO-330-T1-60, DO-330-T1-61, DO-330-T1-62, DO-330-T1-63, DO-330-T1-64, DO-330-T1-65, DO-330-T1-66, DO-330-T1-67, DO-330-T1-68, DO-330-T1-69, DO-330-T1-70, DO-330-T1-71, DO-330-T1-72, DO-330-T1-73, DO-330-T1-74, DO-330-T1-75, DO-330-T1-76, DO-330-T1-77, DO-330-T1-78, DO-330-T1-79, DO-330-T1-80, DO-330-T1-81, DO-330-T1-82, DO-330-T1-83, DO-330-T1-84, DO-330-T1-85, DO-330-T1-86, DO-330-T1-87, DO-330-T1-88, DO-330-T1-89, DO-330-T1-90, DO-330-T1-91, DO-330-T1-92, DO-330-T1-93, DO-330-T1-94, DO-330-T1-95, DO-330-T1-96, DO-330-T1-97, DO-330-T1-98, DO-330-T1-99, DO-330-T1-100, DO-330-T1-101, DO-330-T1-102, DO-330-T1-103, DO-330-T1-104, DO-330-T1-105, DO-330-T1-106, DO-330-T1-107, DO-330-T1-108, DO-330-T1-109, DO-330-T1-110, DO-330-T1-111, DO-330-T1-112, DO-330-T1-113, DO-330-T1-114, DO-330-T1-115, DO-330-T1-116, DO-330-T1-117, DO-330-T1-118, DO-330-T1-119, DO-330-T1-120, DO-330-T1-121, DO-330-T1-122, DO-330-T1-123, DO-330-T1-124, DO-330-T1-125, DO-330-T1-126, DO-330-T1-127, DO-330-T1-128, DO-330-T1-129, DO-330-T1-130, DO-330-T1-131, DO-330-T1-132, DO-330-T1-133, DO-330-T1-134, DO-330-T1-135, DO-330-T1-136, DO-330-T1-137, DO-330-T1-138, DO-330-T1-139, DO-330-T1-140, DO-330-T1-141, DO-330-T1-142, DO-330-T1-143, DO-330-T1-144, DO-330-T1-145, DO-330-T1-146, DO-330-T1-147, DO-330-T1-148, DO-330-T1-149, DO-330-T1-150, DO-330-T1-151, DO-330-T1-152, DO-330-T1-153, DO-330-T1-154, DO-330-T1-155, DO-330-T1-156, DO-330-T1-157, DO-330-T1-158, DO-330-T1-159, DO-330-T1-160, DO-330-T1-161, DO-330-T1-162, DO-330-T1-163, DO-330-T1-164, DO-330-T1-165, DO-330-T1-166, DO-330-T1-167, DO-330-T1-168, DO-330-T1-169, DO-330-T1-170, DO-330-T1-171, DO-330-T1-172, DO-330-T1-173, DO-330-T1-174, DO-330-T1-175, DO-330-T1-176, DO-330-T1-177, DO-330-T1-178, DO-330-T1-179, DO-330-T1-180, DO-330-T1-181, DO-330-T1-182, DO-330-T1-183, DO-330-T1-184, DO-330-T1-185, DO-330-T1-186, DO-330-T1-187, DO-330-T1-188, DO-330-T1-189, DO-330-T1-190, DO-330-T1-191, DO-330-T1-192, DO-330-T1-193, DO-330-T1-194, DO-330-T1-195, DO-330-T1-196, DO-330-T1-197, DO-330-T1-198, DO-330-T1-199, DO-330-T1-200, DO-330-T1-201, DO-330-T1-202, DO-330-T1-203, DO-330-T1-204, DO-330-T1-205, DO-330-T1-206, DO-330-T1-207, DO-330-T1-208, DO-330-T1-209, DO-330-T1-210, DO-330-T1-211, DO-330-T1-212, DO-330-T1-213, DO-330-T1-214, DO-330-T1-215, DO-330-T1-216, DO-330-T1-217, DO-330-T1-218, DO-330-T1-219, DO-330-T1-220, DO-330-T1-221, DO-330-T1-222, DO-330-T1-223, DO-330-T1-224, DO-330-T1-225, DO-330-T1-226, DO-330-T1-227, DO-330-T1-228, DO-330-T1-229, DO-330-T1-230, DO-330-T1-231, DO-330-T1-232, DO-330-T1-233, DO-330-T1-234, DO-330-T1-235, DO-330-T1-236, DO-330-T1-237, DO-330-T1-238, DO-330-T1-239, DO-330-T1-240, DO-330-T1-241, DO-330-T1-242, DO-330-T1-243, DO-330-T1-244, DO-330-T1-245, DO-330-T1-246, DO-330-T1-247, DO-330-T1-248, DO-330-T1-249, DO-330-T1-250, DO-330-T1-251, DO-330-T1-252, DO-330-T1-253, DO-330-T1-254, DO-330-T1-255, DO-330-T1-256, DO-330-T1-257, DO-330-T1-258, DO-330-T1-259, DO-330-T1-260, DO-330-T1-261, DO-330-T1-262, DO-330-T1-263, DO-330-T1-264, DO-330-T1-265, DO-330-T1-266, DO-330-T1-267, DO-330-T1-268, DO-330-T1-269, DO-330-T1-270, DO-330-T1-271, DO-330-T1-272, DO-330-T1-273, DO-330-T1-274, DO-330-T1-275, DO-330-T1-276, DO-330-T1-277, DO-330-T1-278, DO-330-T1-279, DO-330-T1-280, DO-330-T1-281, DO-330-T1-282, DO-330-T1-283, DO-330-T1-284, DO-330-T1-285, DO-330-T1-286, DO-330-T1-287, DO-330-T1-288, DO-330-T1-289, DO-330-T1-290, DO-330-T1-291, DO-330-T1-292, DO-330-T1-293, DO-330-T1-294, DO-330-T1-295, DO-330-T1-296, DO-330-T1-297, DO-330-T1-298, DO-330-T1-299, DO-330-T1-300, DO-330-T1-301, DO-330-T1-302, DO-330-T1-303, DO-330-T1-304, DO-330-T1-305, DO-330-T1-306, DO-330-T1-307, DO-330-T1-308, DO-330-T1-309, DO-330-T1-310, DO-330-T1-311, DO-330-T1-312, DO-330-T1-313, DO-330-T1-314, DO-330-T1-315, DO-330-T1-316, DO-330-T1-317, DO-330-T1-318, DO-330-T1-319, DO-330-T1-320, DO-330-T1-321, DO-330-T1-322, DO-330-T1-323, DO-330-T1-324, DO-330-T1-325, DO-330-T1-326, DO-330-T1-327, DO-330-T1-328, DO-330-T1-329, DO-330-T1-330, DO-330-T1-331, DO-330-T1-332, DO-330-T1-333, DO-330-T1-334, DO-330-T1-335, DO-330-T1-336, DO-330-T1-337, DO-330-T1-338, DO-330-T1-339, DO-330-T1-340, DO-330-T1-341, DO-330-T1-342, DO-330-T1-343, DO-330-T1-344, DO-330-T1-345, DO-330-T1-346, DO-330-T1-347, DO-330-T1-348, DO-330-T1-349, DO-330-T1-350, DO-330-T1-351, DO

Validas AG

Identifier	Keyword	Satisfaction Comment
DO-330-4.1	Qualification Need	Satisfied by satisfying sub-items
DO-330-4.1.a	Identification	The identification (plugin/product name) of Eclipse products and plugins is reused
DO-330-4.1.b	Intended Use	Is done in the TORs model for the main plugin of the tool model (see section 4.3.1 in [TDP])
DO-330-4.1.c	Qualification Need	See Tool-Analysis part in the model (see section 4.2.2 and 4.2.3 in [TDP])
DO-330-4.1.d	TQLs	See Determination in section 4.2.4 in [TDP]

Table 2: Tracing Table to Tool Qualification Planning Process

Tool Development Plan



VALIDAS

Tool Development Plan
for every
Qualifiable Eclipse Plugin
Version 0.6

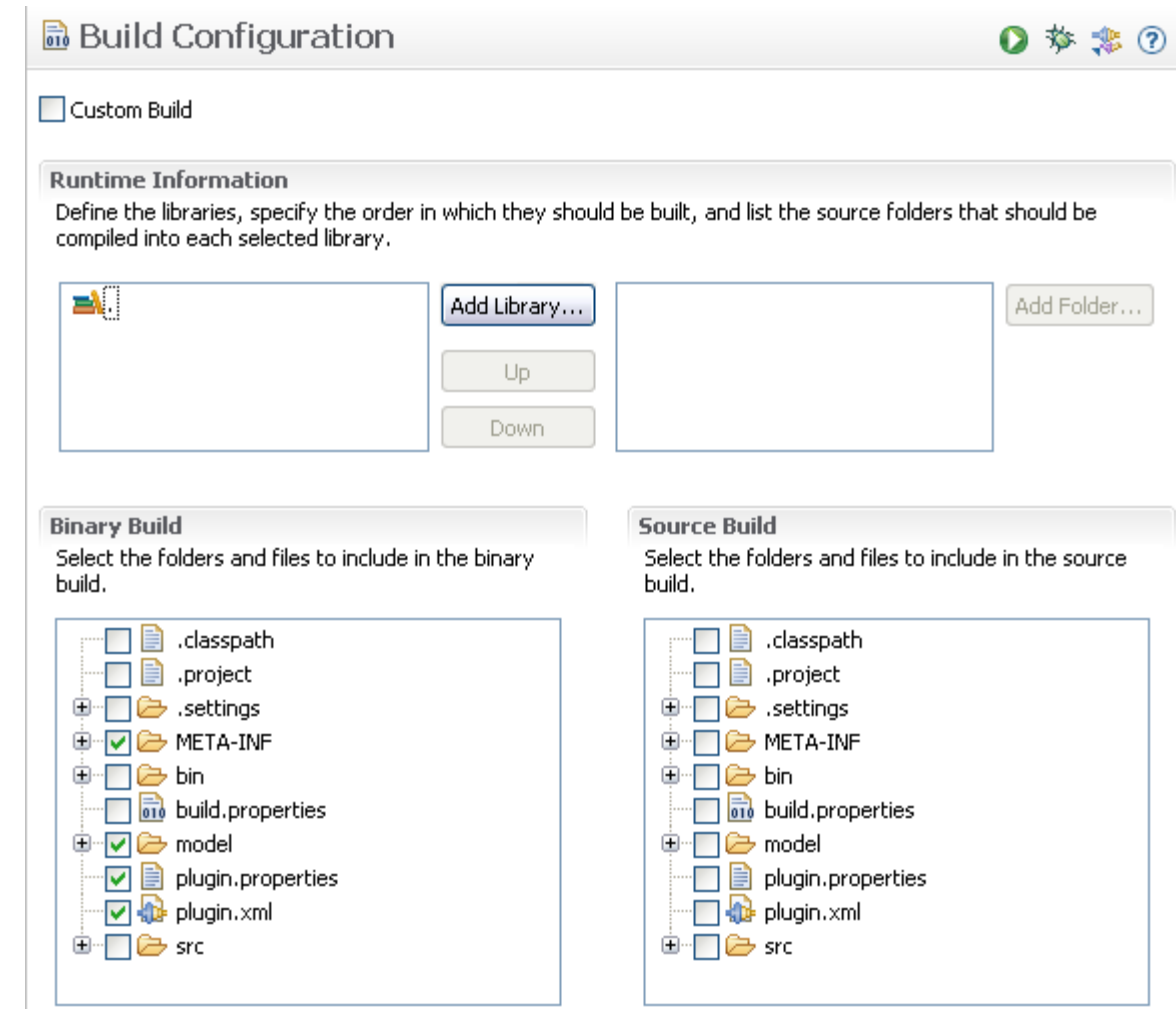
- ▶ **General Process Description for Qualifiable Eclipse Plugins**
- ▶ **Compliant to DO-330**
- ▶ **Can be adapted by developers (DO-330 compliance!)**
- ▶ **Contains description of how to use the model, i.e. standards for**
 - Requirements: TORs, TRs
 - Design, Architecture: TRs, LLRs
 - Implementation
- ▶ **Specific documents can be generated from the DO-330 model, the architecture and the (enriched) implementation**
 - Requirements for <Tool Name>
 - Design for <Tool Name>
 - ...
- ▶ **Examples for some specific document exists**
- ▶ **Similar document for verification: Tool Verification Plan**

1 Document History
2 Definitions
3 Introduction
3.1 Document Extension
3.2 Multi-Function Tools
3.3 Usage of Qualified Plugins
3.4 COTS Tools
4 Standards
4.1 Project and General Information
4.2 Tool Qualification Planning
4.2.1 Overview
4.2.2 The Tool Analysis Model
4.2.3 Determination of the Confidence Level
4.2.4 Determination of the Tool Qualification Level
4.3 Tool Requirements
4.3.1 Tool Operational Requirements
4.3.2 Tool Requirements
4.4 Tool Design
4.4.1 Architecture
4.4.2 Low Level Requirements
4.4.3 Implementation
4.5 Tool Code Standards
5 Tool Life Cycle
5.1 Life Cycle Processes
5.1.1 Tool Planning Process
5.1.2 Tool Development Process
5.1.3 Tool Integration Process
5.1.4 Tool Configuration Management Process
5.1.5 Tool Quality Assurance Process
5.1.6 Tool Operational Verification Process
5.2 Life Cycle Stages
5.3 Transition Criteria
5.4 Life Cycle Verification
6 Tool Development Environment
7 References

Build Qualification Kit



- ▶ **Currently: 2 Builds available in Eclipse**
 - Source Build
 - Binary Build
- ▶ **Missing: Qualifiable Build Configuration with plugin specific**
 - Qualification information (DO-330 Model)
 - Test Cases / Coverage
 - Verification results
 - Documents
 - ...



Content



- ▶ Roadmap
- ▶ Requirements for Tool Qualification (Standards)
- ▶ Proposals for Goals for Eclipse
- ▶ Proposals for some steps towards Tool Qualification
- ▶ Steps on the road
 - First steps: Requirements handling
 - Second steps: Design, Coding and Test
 - Third Steps: Planning Tool Analysis and Life Cycle
 - **Fourth Steps: Life Cycle Refined, PSAC Generation, CM**
 - **Fifth Steps: Quality Assurance, Qualification Liaison Process, Data**
- ▶ Summary

Tool Life Cycle for Qualifiable Plugins



- ▶ **Combines the following processes:**
 - Planning (TORs)
 - Development (TR, LLRs)
 - Integration (Verification)
 - Configuration Management
 - Quality Assurance
- ▶ **Fits to existing processes (Project process, Release Process) by extending them with a “Qualification Stage”**
- ▶ **The following stages are defined (and can be determined automatically from the DO-330 model) such that every release has a well-defined qualification stage**
 - **Unqualified-Pre-Alpha Release (“Undefined”)**: unknown qualification state
 - **Qualification Alpha-Release (“Analyzed”)**: The TORs are defined and TQL is determined
 - **Qualification Beta-Release (“Feature-Complete”)**: All requirements (TORs and TRs) are described and have traces to LLRs and Code
 - **Qualification Release Candidate (“Verification Defined”)**: All required verification steps are defined. No open bugs of the category “Blocker” are available.
 - **Qualification Release: (“Successfully Verified”)** Verification has been successfully executed and are documented within the qualification kit
- ▶ **Transition Criteria are formally defined, based on the DO-330 model**

Life Cycle Transition Criteria



- ▶ Defined in the “Tool Development Plan”
- ▶ Required by DO-330-4.2.1, DO-330-4.2.2, DO-330-4.3.b
- ▶ Quite formal definition (can be checked automatically) based on the DO-330 model of the tool
- ▶ Example (truncated): Transition to Qualification Alpha State (“Analyzed”)

- The *Project* has a nonempty *Name*, *Provider*, *Validator*,
- The *Project* has a *ControlStatus=Reviewed*
- The *Project* has the following TORs specified (in a *TORs* container):

- At least one *TORFunction* defined. All *TORFunction* elements have
 - nonempty *ID*
 - nonempty *Description*
 - *ControlStatus=Reviewed*
- At least one *TORContext* defined. All *TORContext* elements have
 - nonempty *ID*
 - nonempty *Description*
 - *ControlStatus=Reviewed*
- At least one *TORFormat* defined. All *TORFormat* elements have
 - nonempty *ID*
 - nonempty *Description*
 - *ControlStatus=Reviewed*

All *TORFunction* elements should have

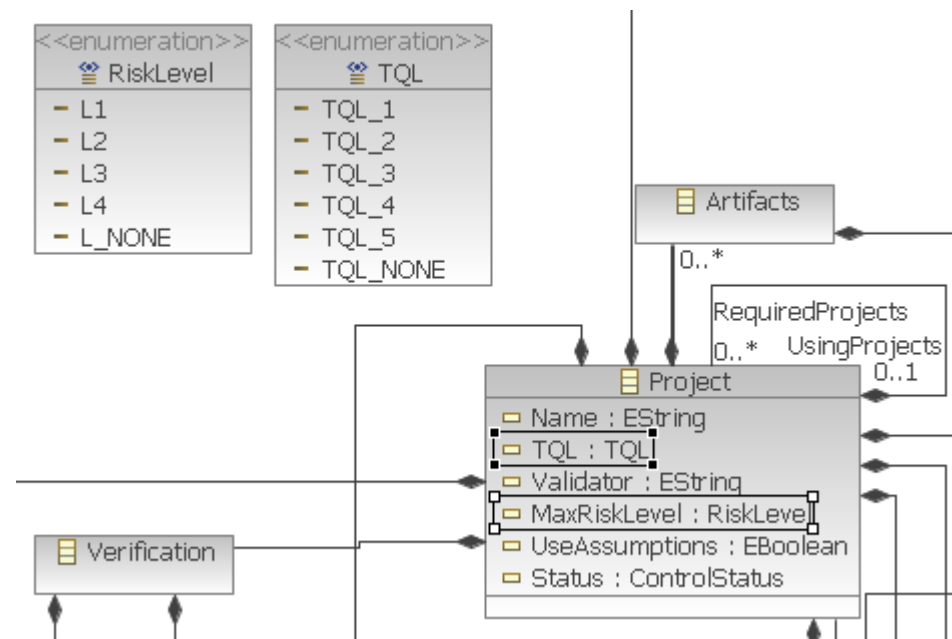
- at least one *PotentialError* in the *AnalysisElements* composition
- For every potential error in the *TORFunction* which has an assigned mitigation (check/restriction) the shall be an artifact flow (to/from) the mitigation’s *TORFunction*, if the mitigation’s *TORFunction* is different from the *TORFunction* of the *PotentialError*.
- A set of “derived errors”, consisting of
 - all errors (*AnalysisElements* of kind *PotentialError*) of the assigned *FunctionAttributes* and
 - all errors (*AnalysisElements* of kind *PotentialError*) of the *ArtifactAttributes* of the *Artifact* are *CreatedBy* or *ModifiedBy* the *TORFunction*. Note that if a *TORFunction* has several outputs with the same *ArtifactAttribute* element assigned, than the errors of the *ArtifactAttribute* are multiple times in the set with a different *ID* that refers to the *Artifact* in which they can occur.
- For each derived error in the set there is either
 - a copy of the *PotentialError* contained in the *TORFunction* or
 - another *PotentialError* contained in the *TORFunction* that subsumes the derived error, i.e. has the *PotentialError* of the *AnalysisAttribute* in the association *Subsumes*.

Tool Analysis in PSAC



- ▶ From the **Qualification Alpha Release** (“**Analyzed**”) of the plugin/tool
- ▶ Verify the TQL (from qualification needs and Projects max. “RiskLevel”)
 - L1: Highest Level (ASIL D, Risk Class A,..)
 - L2, L3
 - L4: Lowest Level (ASIL A, Risk Class D,..)
 - L_NONE for uncritical plugins
- ▶ TQL can also be TQL_NONE for L_NONE or no qualification need
- ▶ List the assumptions of the analysis
- ▶ Generate Documentation for the PSAC that justifies the TQL

Similar to Validas
Tool Chain Analyzer:
Checks &
Computations



Generated PSAC Example from TCA



1.4 Tool Chain Analyzer

1.4.1 Use Cases of Tool Chain Analyzer

- 1.4.1.1 Use Case Determine Tool Confidence Level
- 1.4.1.2 Use Case Generate Tool Classification Report

1.4.2 Features of Tool Chain Analyzer

- 1.4.2.1 Feature Build Model
- 1.4.2.2 Feature Compute Tool Confidence Level
- 1.4.2.3 Feature Excel Interface
- 1.4.2.4 Feature Generate DOT
- 1.4.2.5 Feature Generate Word (docx)
- 1.4.2.6 Feature Model Validation
- 1.4.2.7 Feature Xml Interface

1.4.3 Potential Errors in Tool Chain Analyzer

1.4.4 Restrictions in Tool Chain Analyzer

1.4.5 Checks in Tool Chain Analyzer

1.4.6 Assumptions

1.4.7 TCL Determination

- 1.4.7.1 TCL Determination for Use C
- 1.4.7.2 TCL Determination for Use C

1.4 Tool Chain Analyzer

This section explains the determination of the Tool Confidence Level (TCL) for the Tool Chain Analyzer.

Tool: Tool Chain Analyzer

Description:

This is the Tool Chain Analyzer from Validas AG

Impact:

TI 2 (Impact)

Tool Confidence Level:

TCL 1

Table 25 Tool: Tool Chain Analyzer

The tool Tool Chain Analyzer is modeled with 11 elements which have impact, 0 of them are assumptions. In addition there have been modeled 7 features, 0 of them are assumption

Elements	Amount (Assumptions)
Use Cases	2 (0)
Checks	1 (0)
Restrictions	0 (0)
Qualifications	0 (0)
Potential Errors	8 (0)

Table 26 Amount of Elements in Tool: Tool Chain Analyzer

1.4.1 Use Cases of Tool Chain Analyzer

This section describes all analyzed use cases of Tool Chain Analyzer in separate subse

The following use cases of the tool Tool Chain Analyzer are considered:

1. Determine Tool Confidence Level, see Section 1.4.1.1
2. Generate Tool Classification Report, see Section 1.4.1.2

1.4.1.1 [Use Case Determine Tool Confidence Level]

This section describes the use case "Determine Tool Confidence Level".

Error: Document Generated Wrongly

Description:

Document does not fit to the model.

From feature:

Generate Word (docx)

Discovered by the following checks:

- Confirmation Review Of TCLs Detect Wrong TCL

Subsumes:

- "No File Created" from "File Generator"
- "Semantic Error" from "File Generator"
- "Syntax Error" from "File Generator"

Occurrences:

- in Generate Word (docx) in Generate Tool Classification Report

Error View:

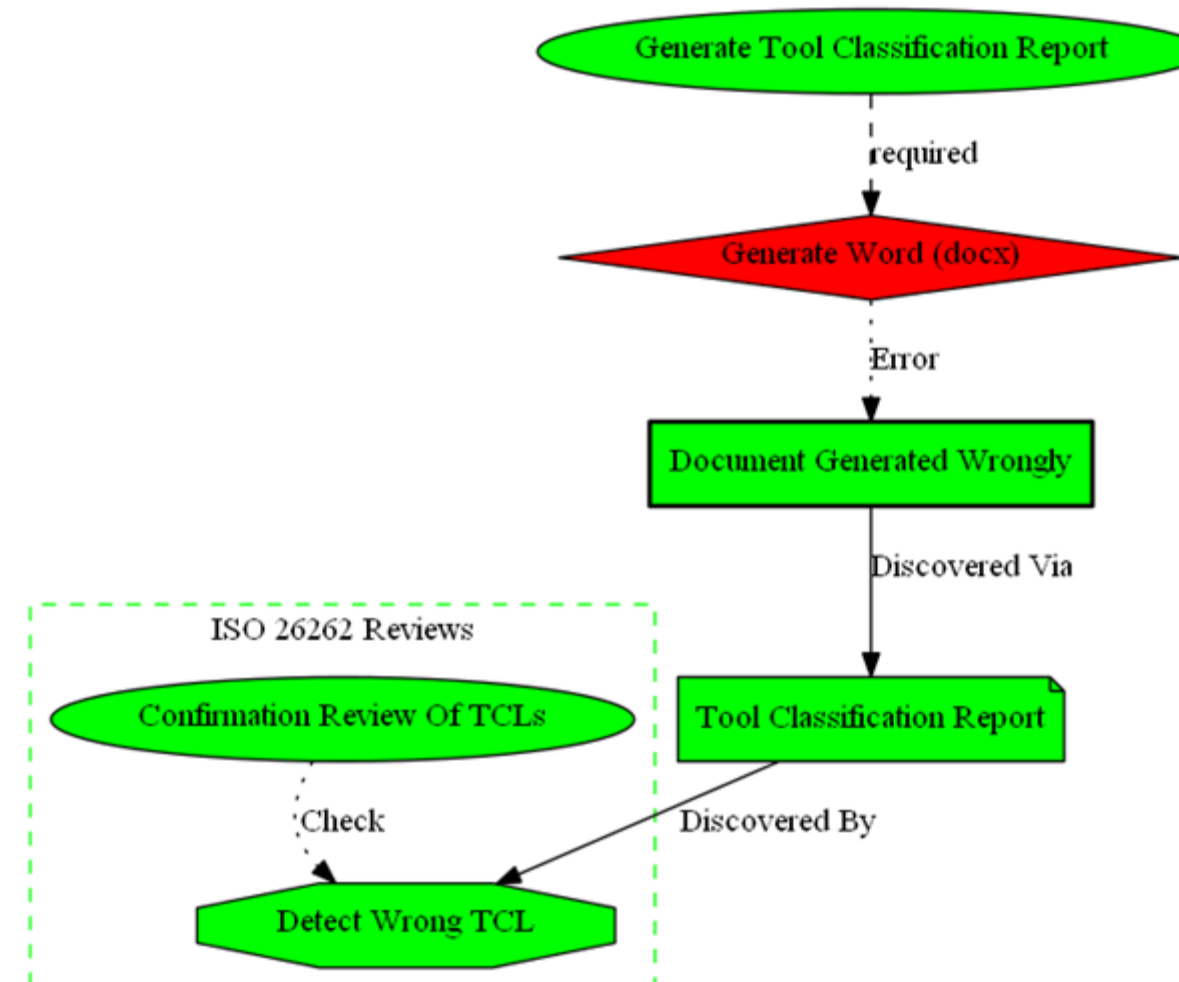


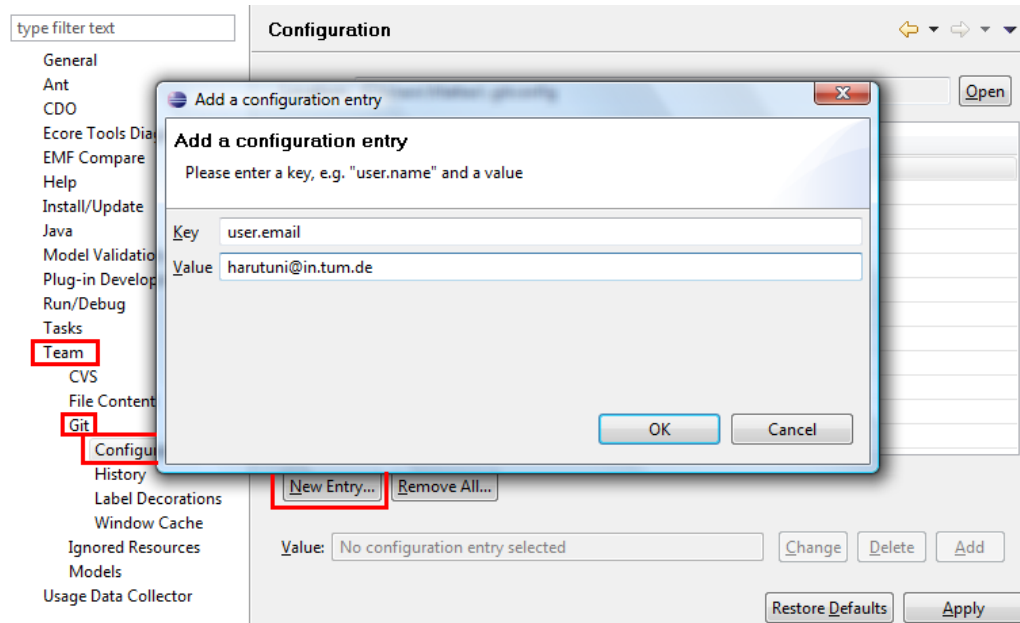
Table 49 Error: Document Generated Wrongly

Configuration Management



- ▶ We require eGit and Gerrit to be used for qualifiable Eclipse plugins
- ▶ Some details (branch-names, configuration..) to be discussed (currently with BMW-CarIT)
- ▶ Informations

- <http://www.eclipse.org/egit/>
- <http://progit.org/book/>
- <http://www.slideshare.net/stefanlay/eclipse-git-und-gerrit>



- ▶ Described in Tool Development Plan (Version 0.8)
- ▶ Traced against “How-To-Qualify” Document (DO-330)

Tool Development Plan

- 1 Document History
- 2 Definitions
- 3 Introduction
- 4 Standards
 - 4.1 Project and General Information
 - 4.2 Tool Qualification Planning
 - 4.3 Tool Requirements
 - 4.4 Tool Design
 - 4.5 Tool Code Standards
 - 4.6 Configuration Management Plan
 - 4.6.1 Configuration Identification
 - 4.6.2 Baselines
 - 4.6.3 Traceability
 - 4.6.4 Problem Reporting
 - 4.6.5 Change Control: Integrity and Identification
 - 4.6.6 Change Control: Tracking
 - 4.6.7 Change Review
 - 4.6.8 Configuration Status Accounting
 - 4.6.9 Retrieval
 - 4.6.10 Data Retention
- 5 Tool Life Cycle
 - 5.1 Life Cycle Processes
 - 5.2 Life Cycle Stages
 - 5.3 Transition Criteria
 - 5.4 Life Cycle Verification
- 6 Tool Development Environment
 - 6.1 Eclipse Development
 - 6.2 Eclipse Integration
 - 6.3 Configuration Management (Draft)
 - 6.3.1 Branches
 - 6.3.2 Annotated Tags
 - 6.3.3 Change History
 - 6.3.4 GitWeb
 - 6.3.5 Change Control
 - 6.3.6 Gerrit Review Support
 - 6.4 Quality Assurance
- 7 References

Configuration Management



- ▶ **Configuration Items are all elements within the Qualifiable Eclipse Project**
 - Sources
 - Architecture
 - DO-330-model
 - Requirements (TORs, TRs,
 - Tracing
 -

- ▶ **Two Control Categories: CC1, CC2. Item's CC depends on TQL**

Tool Operational Requirements Process											Control Category by TQL				
											1	2	3	4	5
2	Tool Operational Requirements are defined.	5.1.1.a	5.1.2.a	○	○	○	○	○	Tool Operational Requirements	10.3.1	①	①	①	①	②
			5.1.2.b												
			5.1.2.c												

- ▶ **Definition of Control Categories (DO-330):**

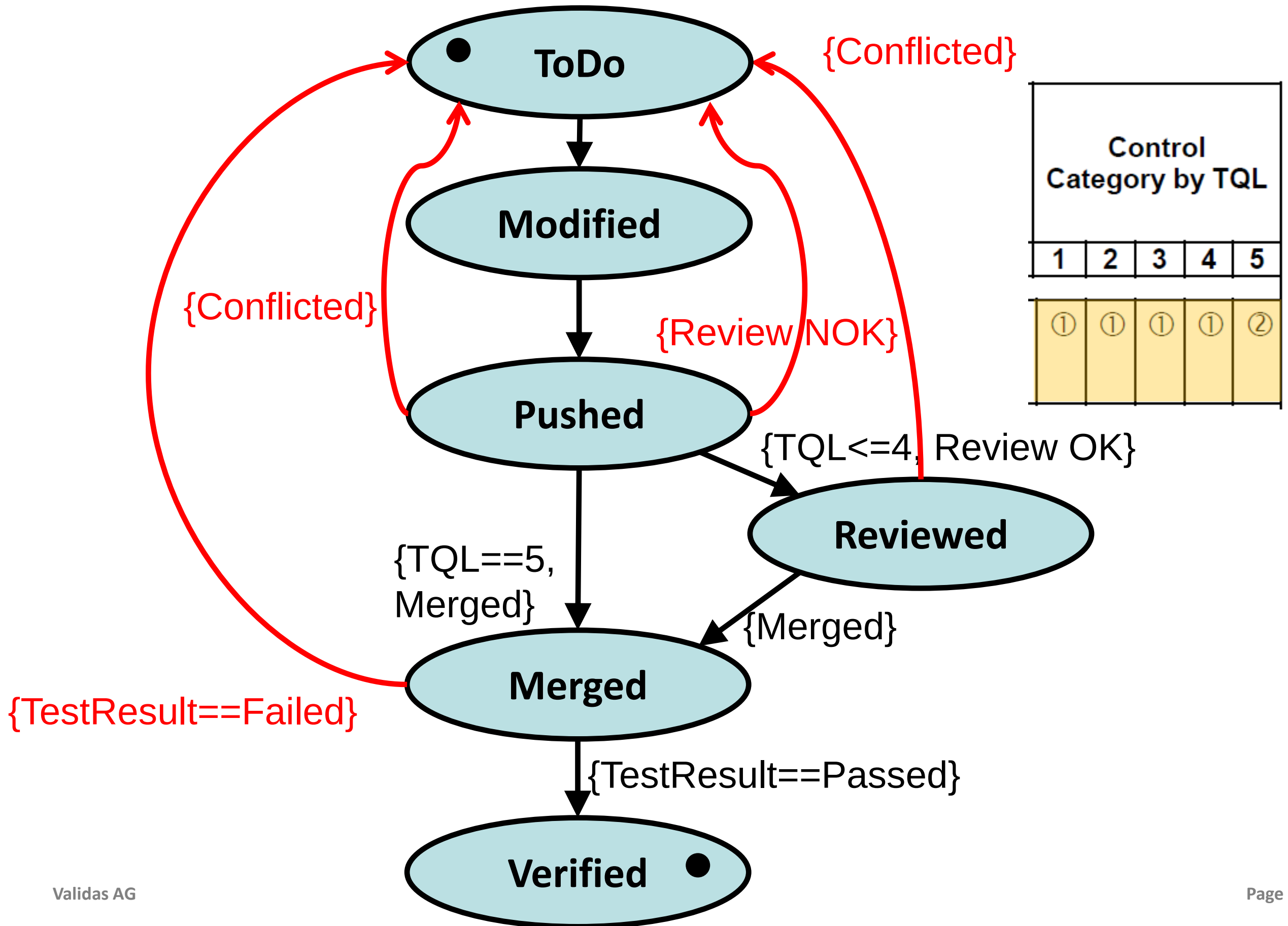
Table 7-1 TCM Process Activities Associated with CC1 and CC2 Data

TCM Process Activity	Reference	CC1	CC2
Configuration Identification	7.2.1	•	•
Baselines	7.2.2.a 7.2.2.b 7.2.2.c 7.2.2.d 7.2.2.e	•	
Traceability	7.2.2.f 7.2.2.g	•	•
Change Review	7.2.5	•	

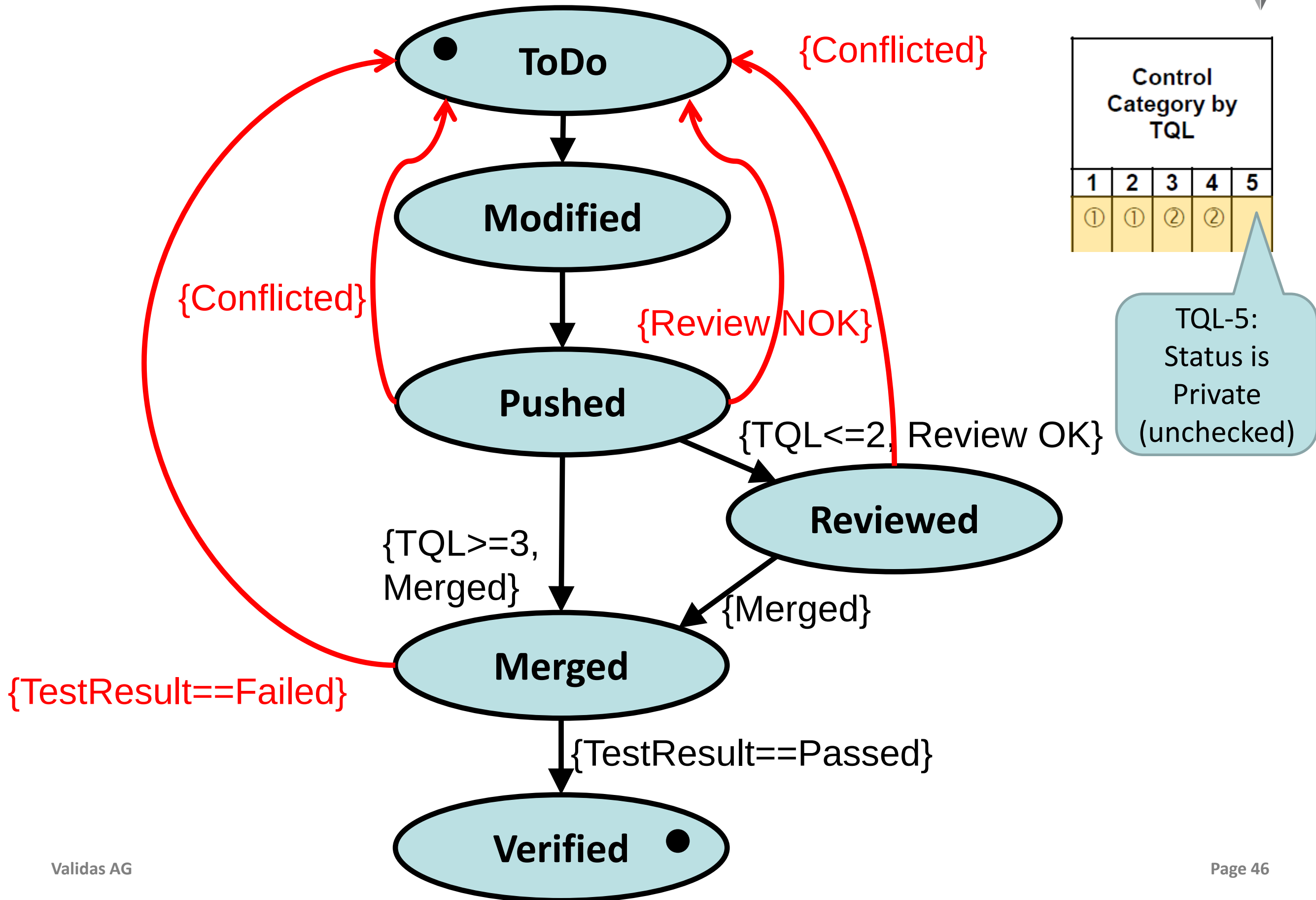
Example: TORs **changes** have to be **reviewed** for TQL-1 to TQL-4 but not for TQL-5

Plugin Extension has to know this (Transition Criteria!)

CM: Control Status of TORs (Proposed)

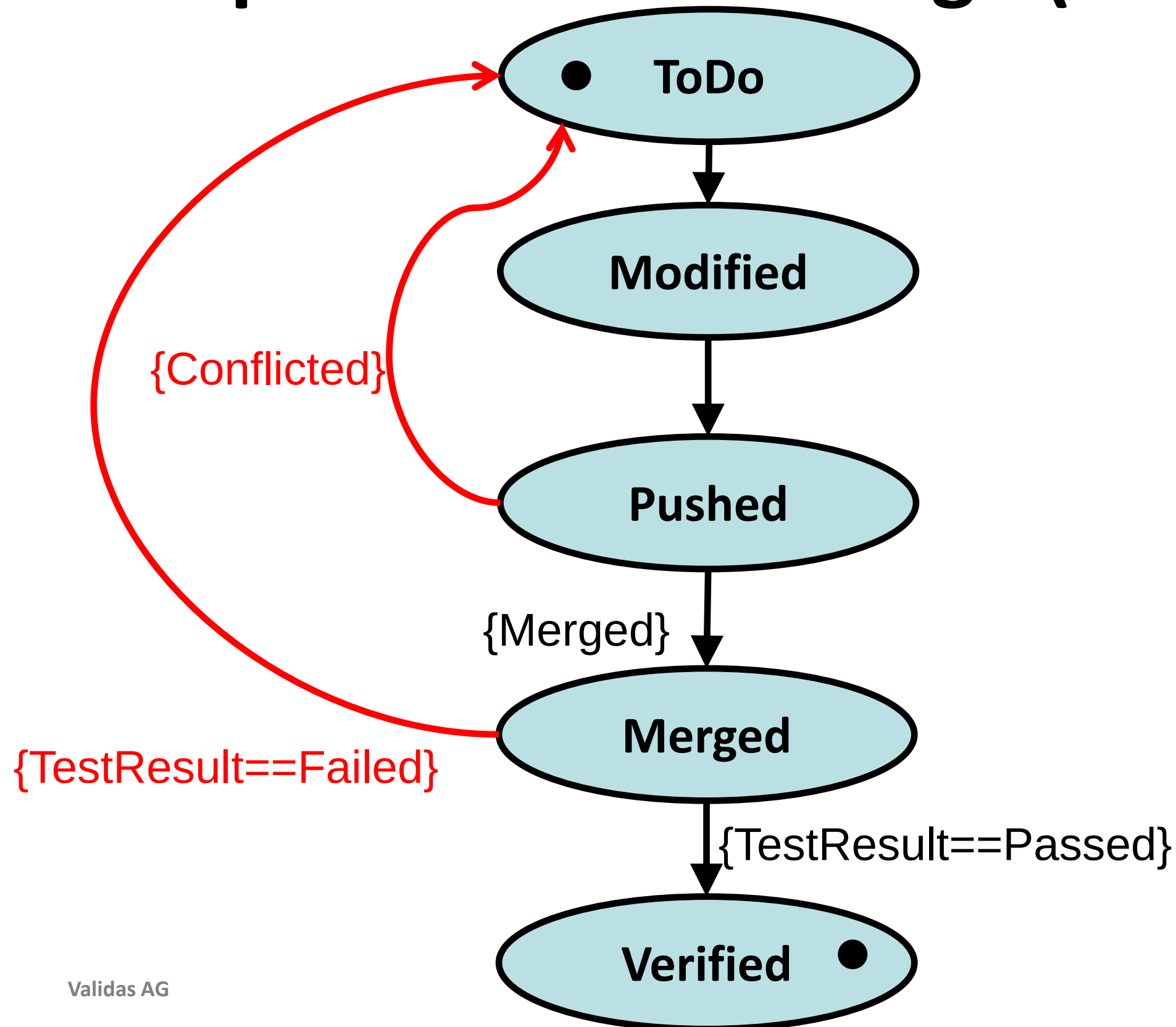


CM: Control Status of Tests (Proposed)





CM: Control Status of TRExpectedErrorMessage (Proposed)



Control Category by TQL				
1	2	3	4	5
②	②	②	②	

TQL-5:
Status is
Private
(unchecked)

Content

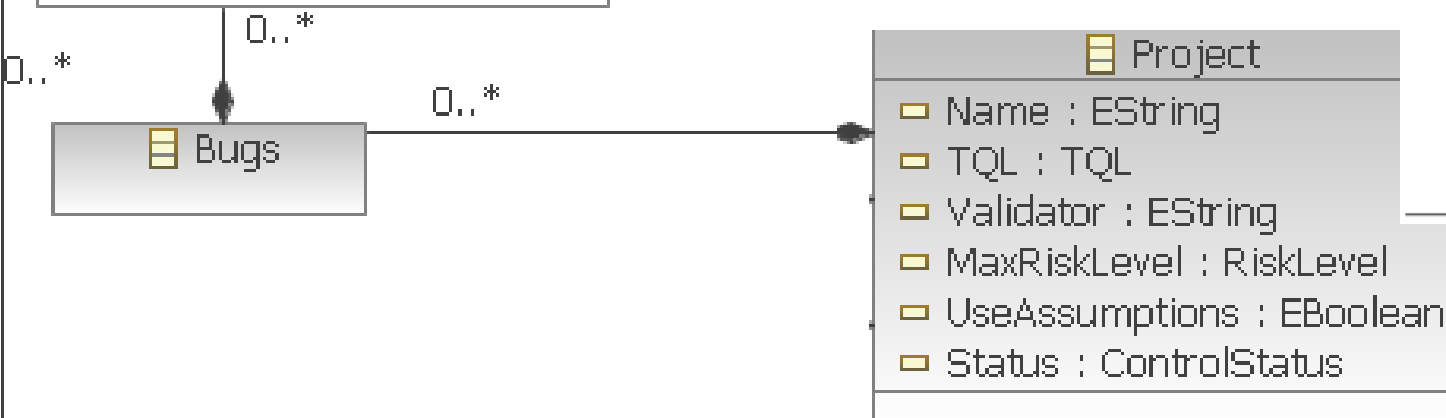
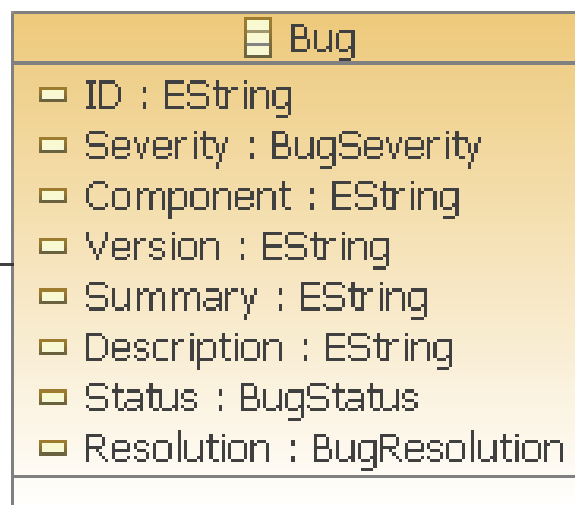
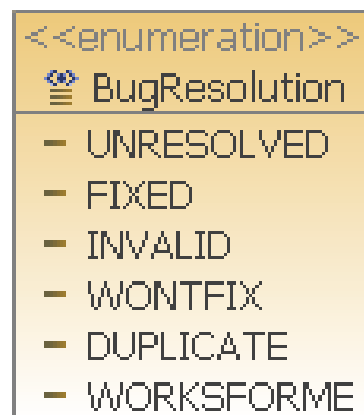
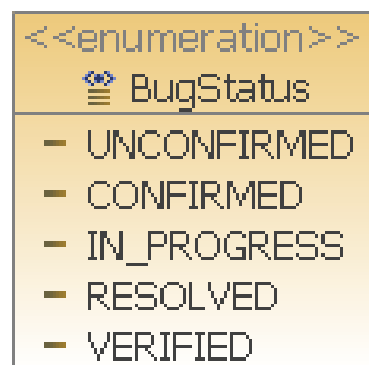


- ▶ Roadmap
- ▶ Requirements for Tool Qualification (Standards)
- ▶ Proposals for Goals for Eclipse
- ▶ Proposals for some steps towards Tool Qualification
- ▶ Steps on the road
 - First steps: Requirements handling
 - Second steps: Design, Coding and Test
 - Third Steps: Planning Tool Analysis and Life Cycle
 - Fourth Steps: Life Cycle Refined, PSAC Generation, CM
 - **Fifth Steps: Quality Assurance, Qualification Liaison Process, Data**
- ▶ Summary

Quality Assurance



- ▶ Interface model to bugzilla (or other bug tracking systems)
- ▶ Contains references to test cases (that can be verified) and potential errors of the analysis (together with possible mitigations/work arounds)



eclipse BUGS

When I am trying to launch a specific plugin, in the error log I am getting the error (See More Info) Last modified: 2008-07-21 16:39:07 EDT

Home | New | Search | Search | [?] | Reports | My Requests | Preferences
| Help | Log out | slotosch@validas.de | Terms of Use | Copyright Agent

Bug List: (2 of 454) First Last Prev Next Show last search results

Bug 241577 - When I am trying to launch a specific plugin, in the error log I am getting the error (See More Info) Save Changes

Status: NEW

Product: PDE

Component: UI

Version: 3.4

Platform: PC Windows Vista

Importance: P3 normal (vote)

Target Milestone: ---

Assigned To: PDE-UI-Inbox

QA Contact:

URL:

Whiteboard:

Keywords:

Depends on:

Blocks: Show dependency tree

Reported: 2008-07-21 16:20 EDT by Mandeep

Modified: 2008-07-21 16:39 EDT (History)

CC List: ☒ Add me to CC list
2 users (edit)

See Also:

Flags:

documentation

indigo

iplog

juno

pmc_approved

review

Tool Quality Report



- ▶ **For every qualified Plugin there will be a tool quality report**
- ▶ **Contains required documentation of the audit & review of the plugin**
 - Pointer to used tool development plan
 - Verification that it has been reviewed for DO-330 consistency
 - Verification of the used Eclipse development environment
 - **Known-Bug** analysis (mapping to potential errors)
 - Verification method for the qualification stage of the do-330 model (manual/new Eclipse support)
 - Tag and successful nightly build report verification
 - Checks of required plugins quality reports
- ▶ **A checklist in the tool development plan and a template eases the creation of the tool quality report**

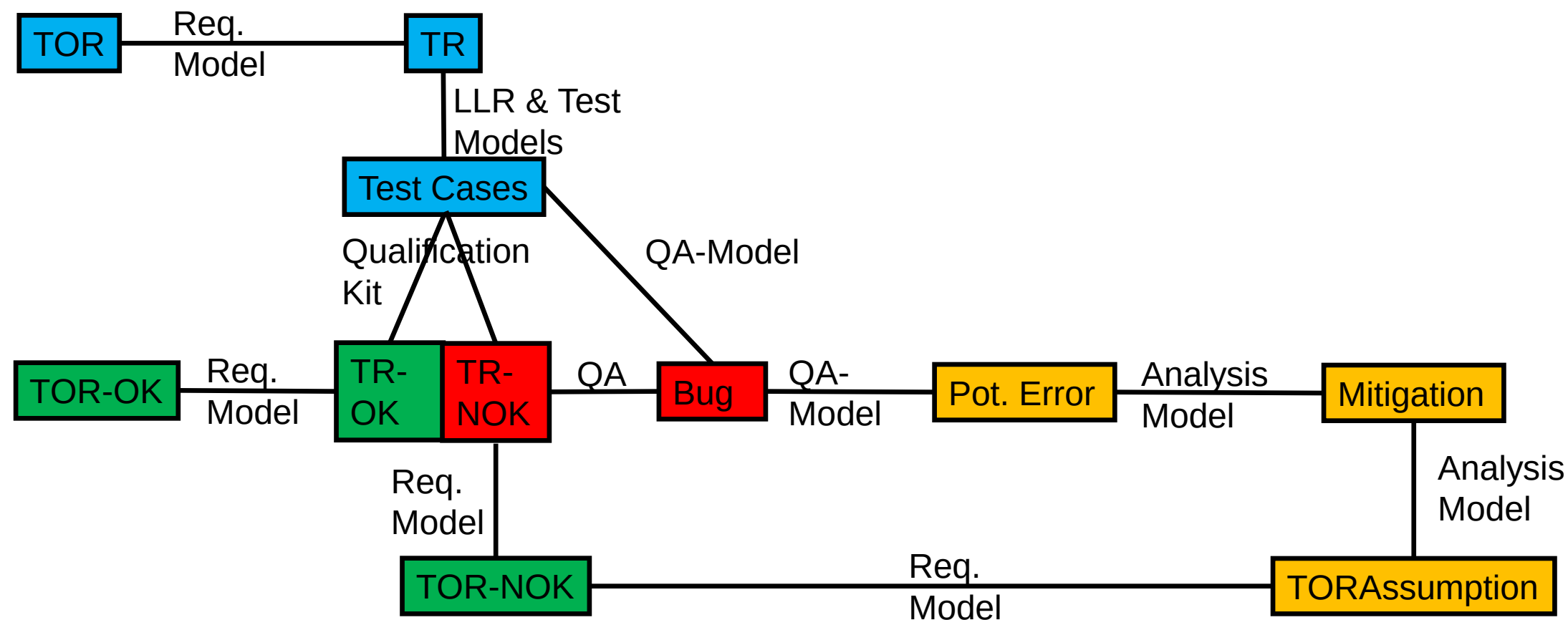
Creation of this report is the only manual step that cannot be computed automatically from the DO-330 model.

“last check before release”

Qualification Liaison Process



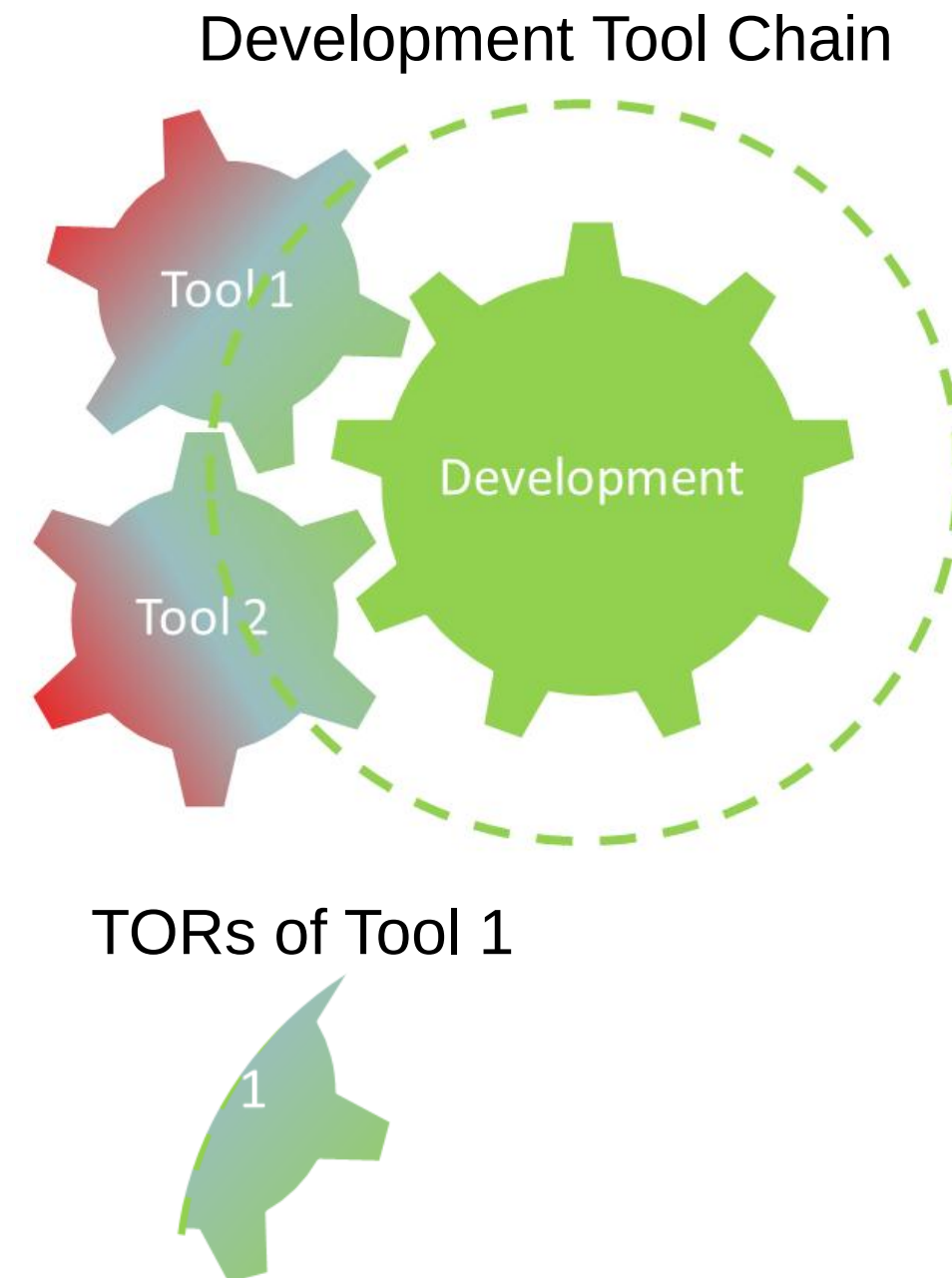
- ▶ For all tools with qualification need
- ▶ Demonstrate that the tools conform to their requirements (“TOR”), even if qualification shows errors



Tool Aspects in Software



- ▶ The interface to the application domains / developed systems
- ▶ Depends on the processes (eliminated/supported) by the tool
- ▶ Consider the complete tool chain for development of SW
- ▶ **Planning**
 - ISO 26262-8-11.5.1: Software tool criteria evaluation report
 - DO-178C-11.1: Plan for Software Aspects in Certification (PSAC)
 - g: Additional Considerations ... **tool qualification**
 - DO-330-10.1.1: Tool-Specific Information in PSAC
- ▶ **Accomplishment**
 - ISO:26262-8-11.5.2: Software tool qualification report
 - DO-178C-11.20: Software Accomplishment Summary (SAS)
 - DO-330-10.1.16: Tool-Specific Information in SAS



Qualification Data



- ▶ **All data produced during development and verification process**
 - From tool analysis, TORs
 - to tool installation report
- ▶ **Satisfies all elements in section 10 of DO-330**
- ▶ **Tool specific data**
 - DO-330-model of Eclipse
 - Generated into specific documents
 - Requirements
 - Verification Plan
 - Verification results
 - Problem reports, ...
- ▶ **Process (tool independent)**
 - Tool development plan
 - Tool verification plan
- ▶ **Meta-Data: contained in “HowTo-Qualify Eclipse-based Tools”-document**
 - Concept, Liaison Process
 - Tracing to DO-330
 - Qualification Planning & Qualification Report

8.8 [Tracing to Tool Qualification Data Section

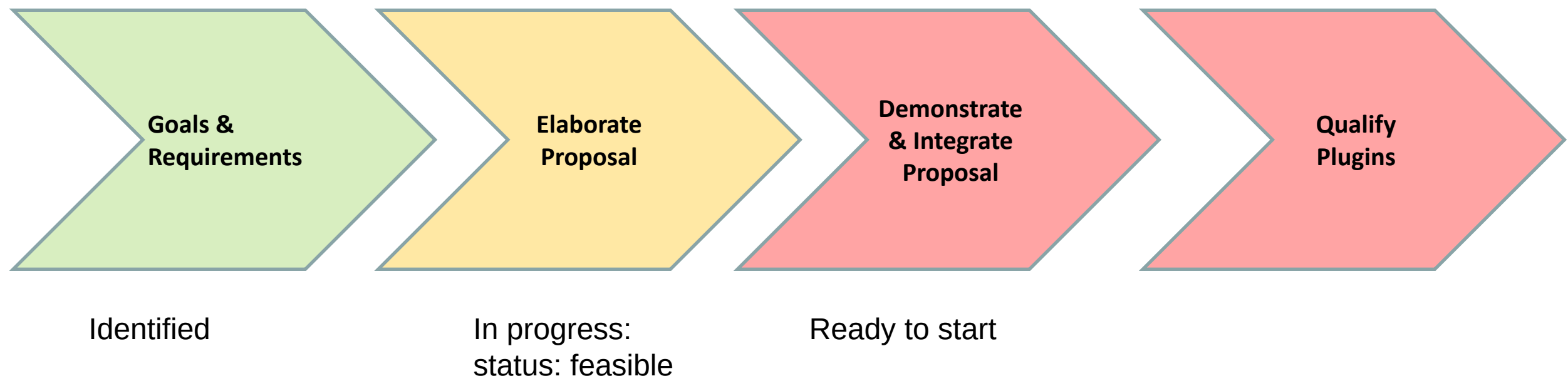
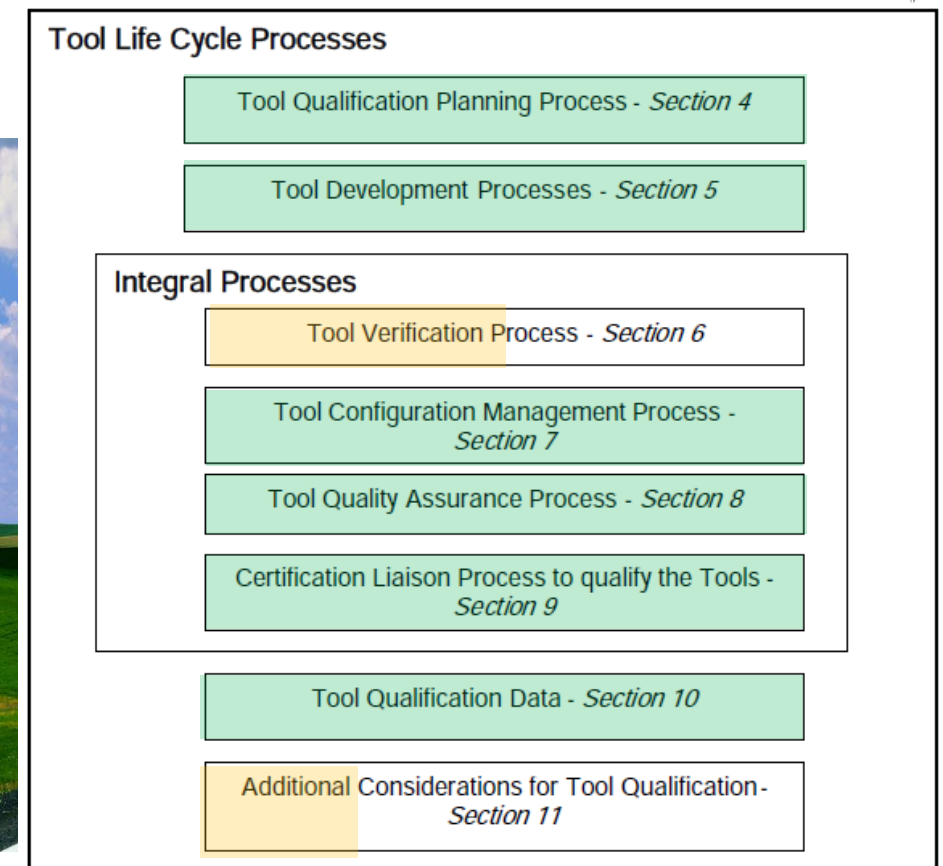
Identifier	Keyword	Satisfaction Comment
DO-330-10.1.1	Tool Specific information in PSAC	The information is in the DO-model contained, a document could be generated. See subsections
DO-330-10.1.1.a	Identification and Use Cases	Modeled in Project and TORs, see sections 4.1.1, 4.2.2 and 4.3.1 in [TDP]
DO-330-10.1.1.b	Details of use in process	The artifacts in the analysis model provide the link to the automated process, see section 4.2.2.5 and 4.2.2.6 in [TDP]
DO-330-10.1.1.c	Technology maturity	Systematic error analysis using AnalysisAttribute in section 4.2.2.2, 4.2.2.4 and 4.2.2.7 in [TDP]
DO-330-10.1.1.d	Proposed TQL	See TQL in Project model in 4.1.1 and its derivation in section 4.2.4 and 4.2.3 in [TDP]
DO-330-10.1.1.e	Source Code	Code is part of the Qualification Build, see section 6.2 in [TDP].
DO-330-10.1.1.f	Stakeholders and Roles	See “Provider” in MANIFEST.MF and “Validator” in project model in section 4.1.1 of [TDP] and the Validator in the verification data model (see section 4.2.2 in [TVP])
DO-330-10.1.1.g	Process Descriptions (TOR, TOI, TOVV)	See Table 3 (5.1 and 5.3) and Table 4 (6.2)
DO-330-10.1.1.h	TO Environment desc.	See TORContext model in section 4.3.1.4 in [TVP]
DO-330-10.1.1.i	Qualification reuse	Only possible as described in section 3.3 and 3.4 in [TVP]
DO-330-10.1.1.j	Reference to TQP	TQP is generated from the same model, hence it is trivial
DO-330-10.1.2	Tool Qualification Plan	The information is in the DO-model contained, a document could be generated as described in section 5. See subsections

Roadmap - Status May 2012



1. Identify goals & requirements for tool qualification in Eclipse
2. Propose process / project
3. Demonstrate tool qualification & integrate proposal into Eclipse Plugin Framework
4. Establish proposal: Qualify (selected) plugins

► Status May 2012

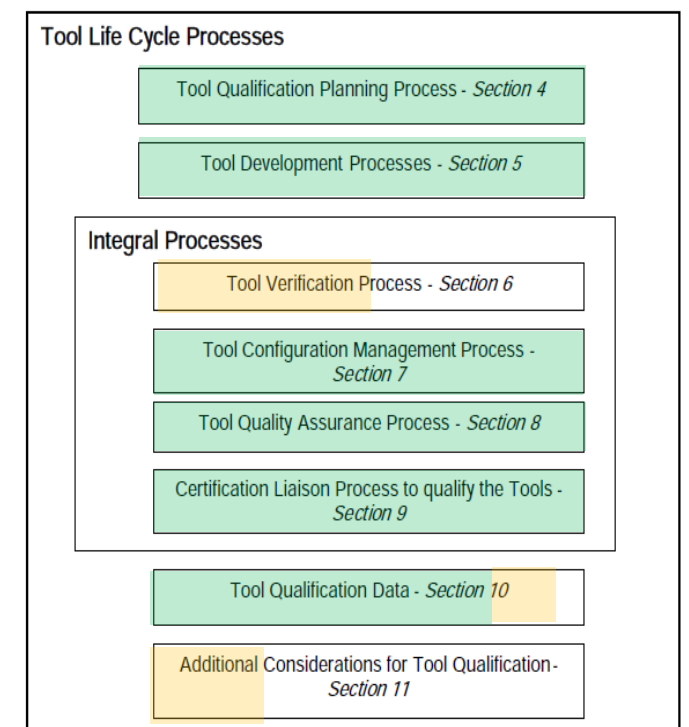


- **Summary: Qualification is feasible and qualification (based on current prototype) could be started now**

Summary



- ▶ **Roadmap towards development of qualifiable Eclipse tools & plugins**
 - Classification: Tool Analysis -> Planning Process
 - Qualification: Process & Model for Qualifiable Plugins
 - Usage: Fulfill Assumptions and apply qualification kits
- ▶ **Applicable to all relevant standards (ISO 26262, IEC 61508, DO-178C, EN 50128,..)**
- ▶ **Metadata extension for qualification information of plugins: DO-330 model**
- ▶ **Much work in progress**
 - Tracing to How-To-Qualify Document
 - Modeling: gaps to current meta-information
 - Create documentations (TDP,TVP,..)
- ▶ **First, second, thirds, fourth, fifth steps performed**
- ▶ **Proposed new role for that work: Eclipse Validator**
- ▶ **Validas contributes**



Thank You!



VALIDAS 

Arnulfstraße 27
80335 München
www.validas.de
info@validas.de