Oscar Slotosch, Validas AG

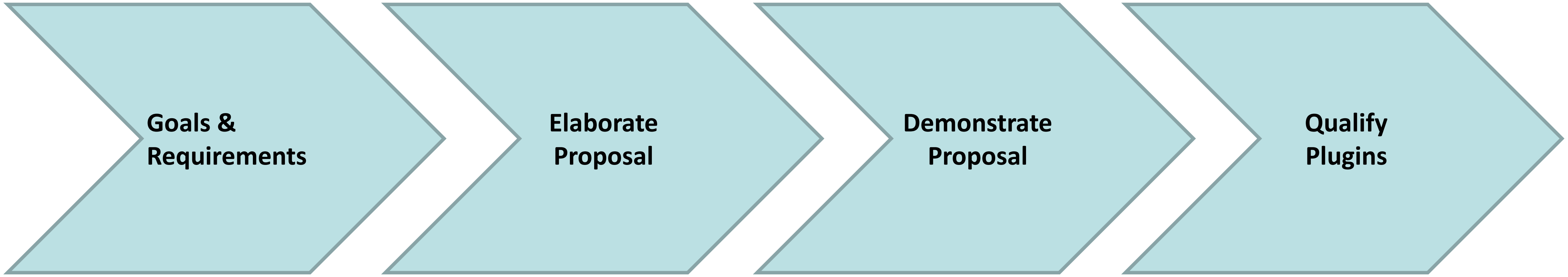
Roadmap towards Development of Qualifyable Eclipse Tools (Eclipse-Project Concept)

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
  - Sixth Steps: Verification, Additional Considerations
- 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
  - Sixth Steps: Verification, Additional Considerations
- 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**

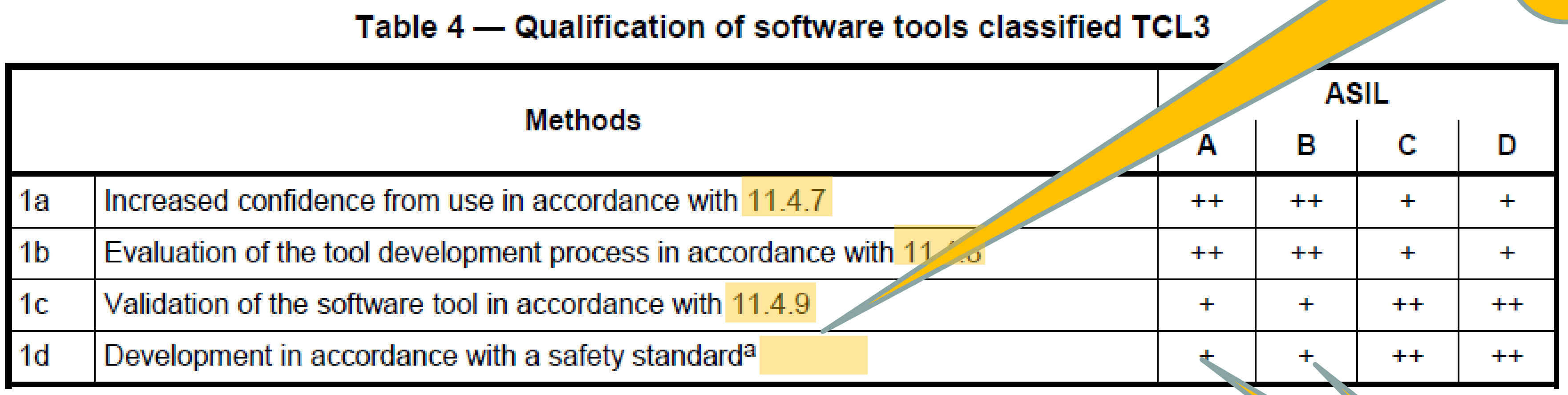


Table 4 — Qualification of software tools classified TCL3

	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]	+	+	++	++

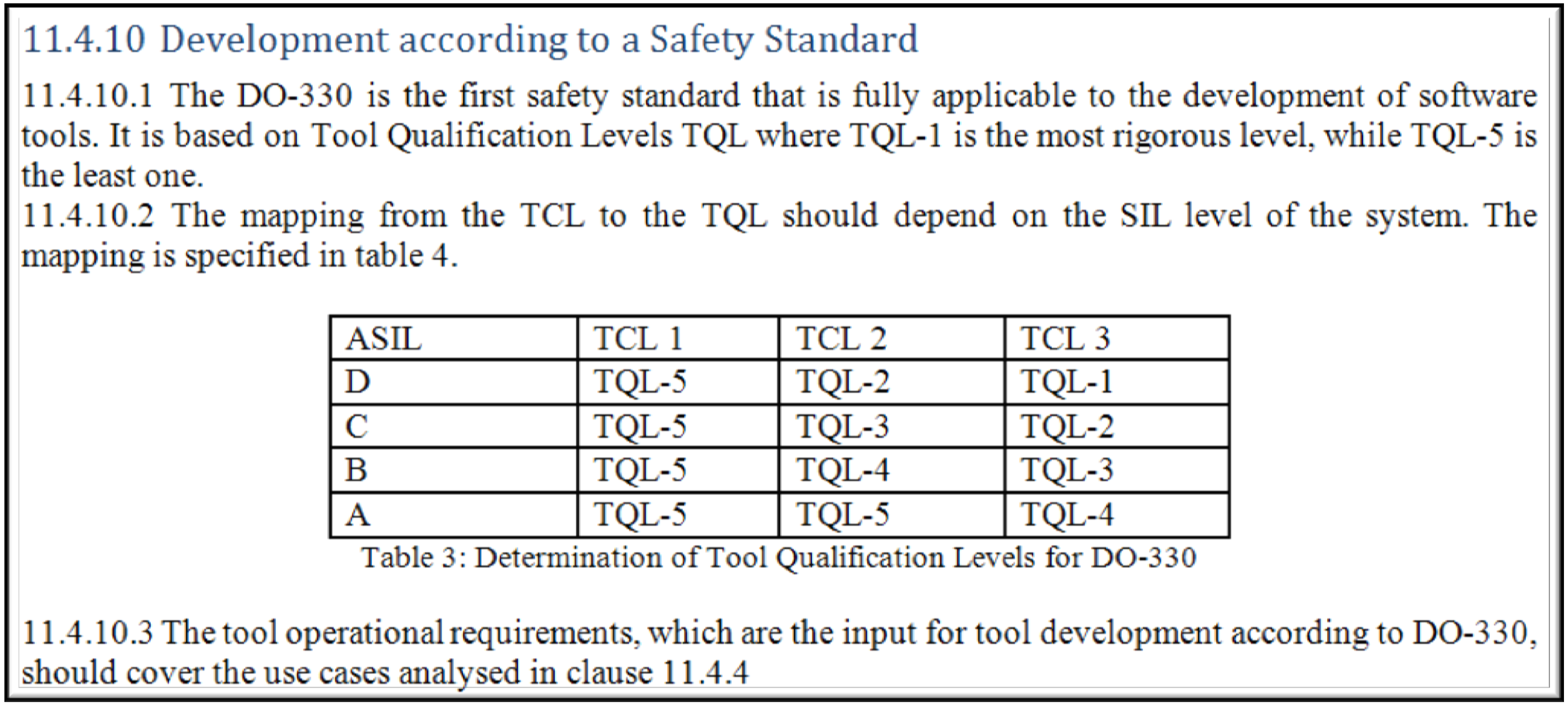
Here is a hole were the new DO-330 standard fits in

- **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

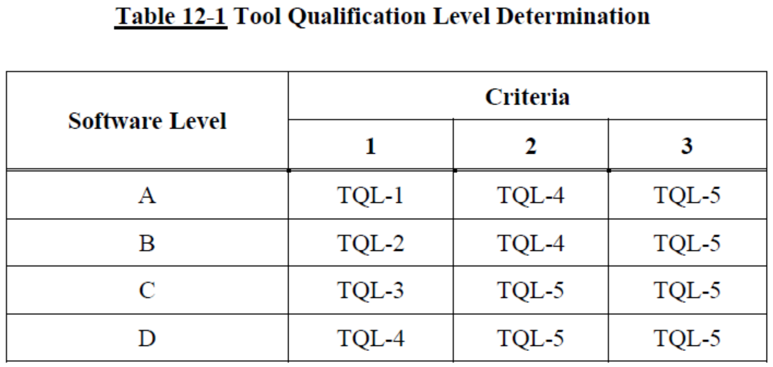


Table 12-1 Tool Qualification Level Determination

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

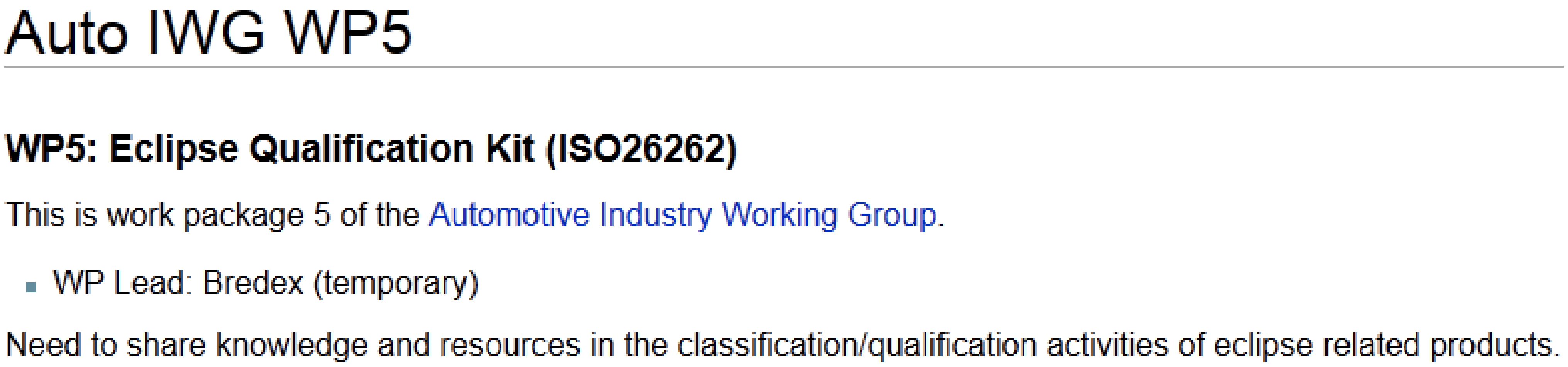
Content

- 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
  - Sixth Steps: Verification, Multi-Function, Coupling, Independency, COTS
- 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

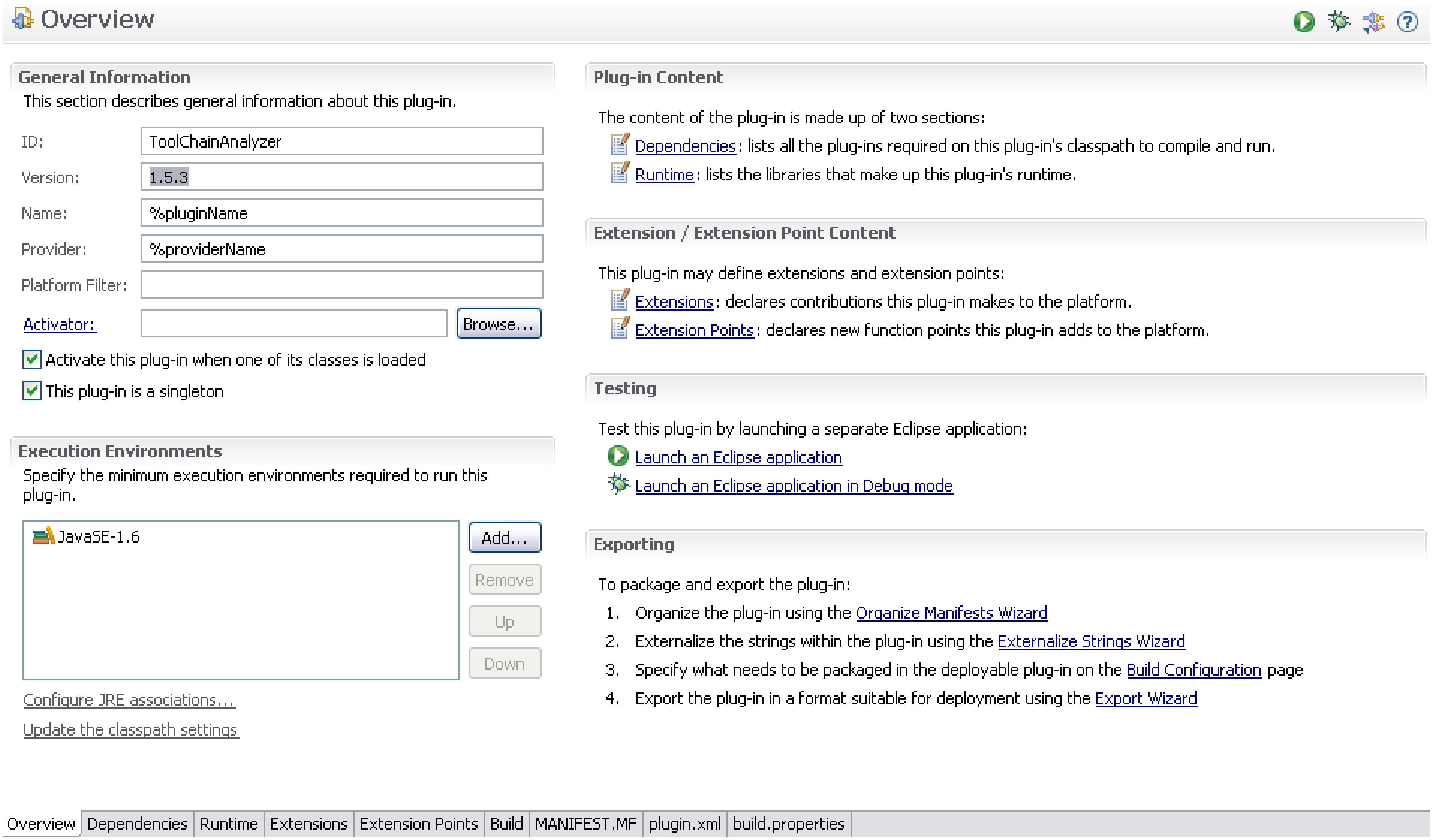
WP5: Eclipse Qualification Kit (ISO26262)

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:

☑ 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

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:

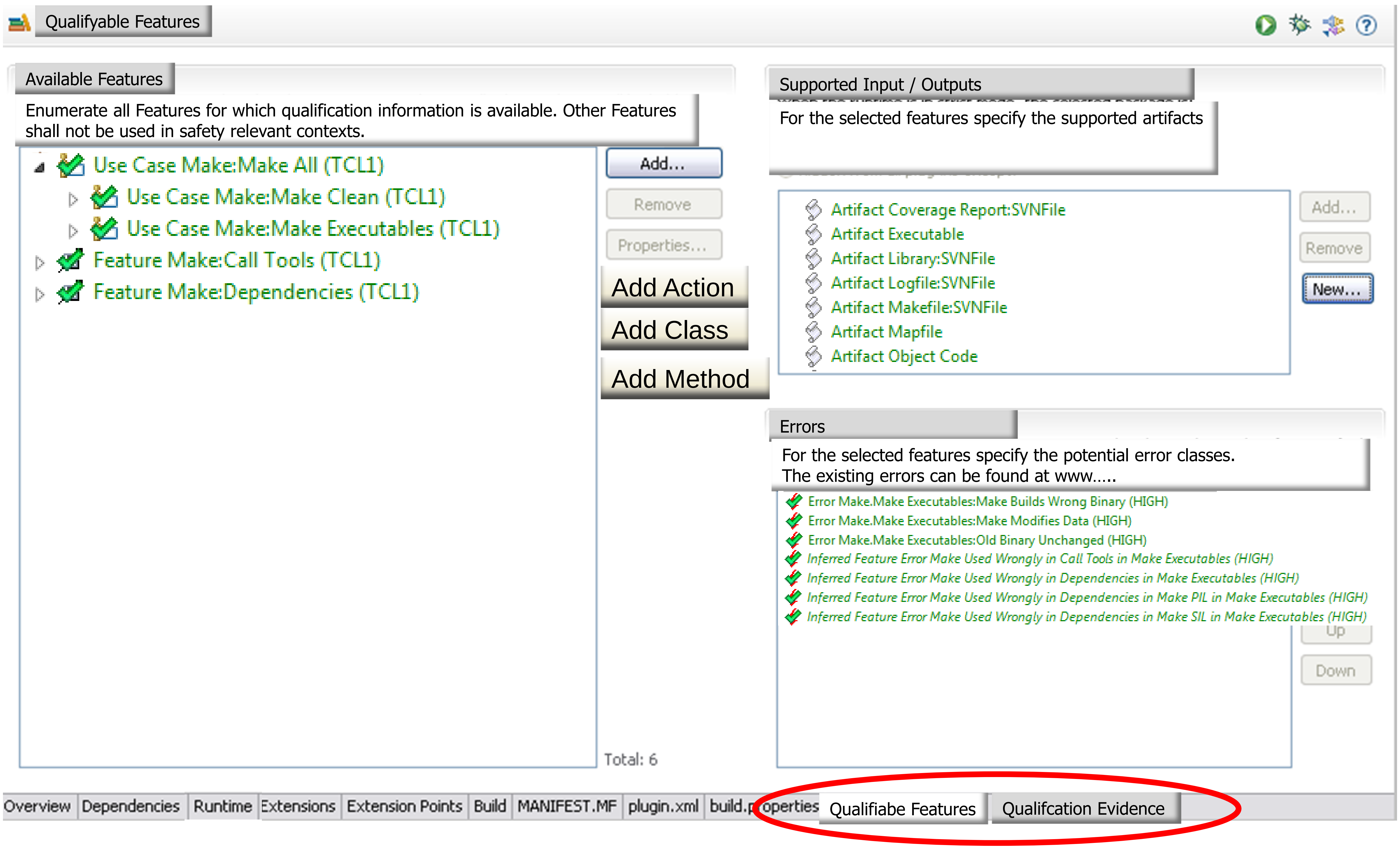
- Launch an Eclipse application
- Launch an Eclipse application in Debug mode

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



Qualifyable 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)

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

Errors

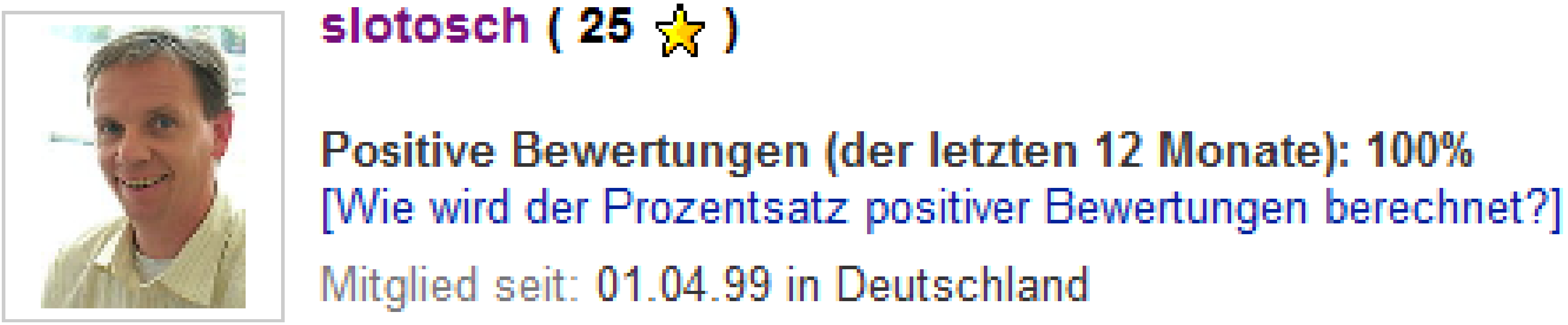
For the selected features specify the potential error classes.
The existing errors can be found at www.....

- 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)*

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:**



slotosch (25 ☆)

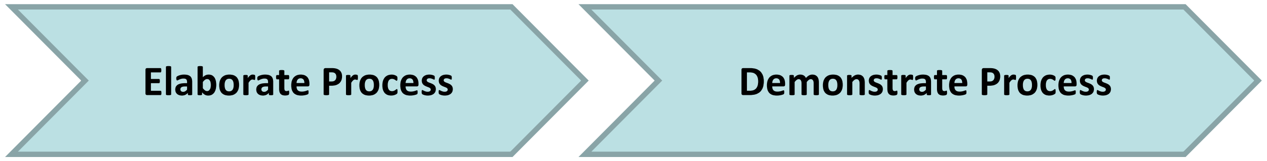
Positive Bewertungen (der letzten 12 Monate): 100%
[Wie wird der Prozentsatz positiver Bewertungen berechnet?]

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
  - Sixth Steps: Verification, Multi-Function, Coupling, Independency, COTS
- Summary

Proposals



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

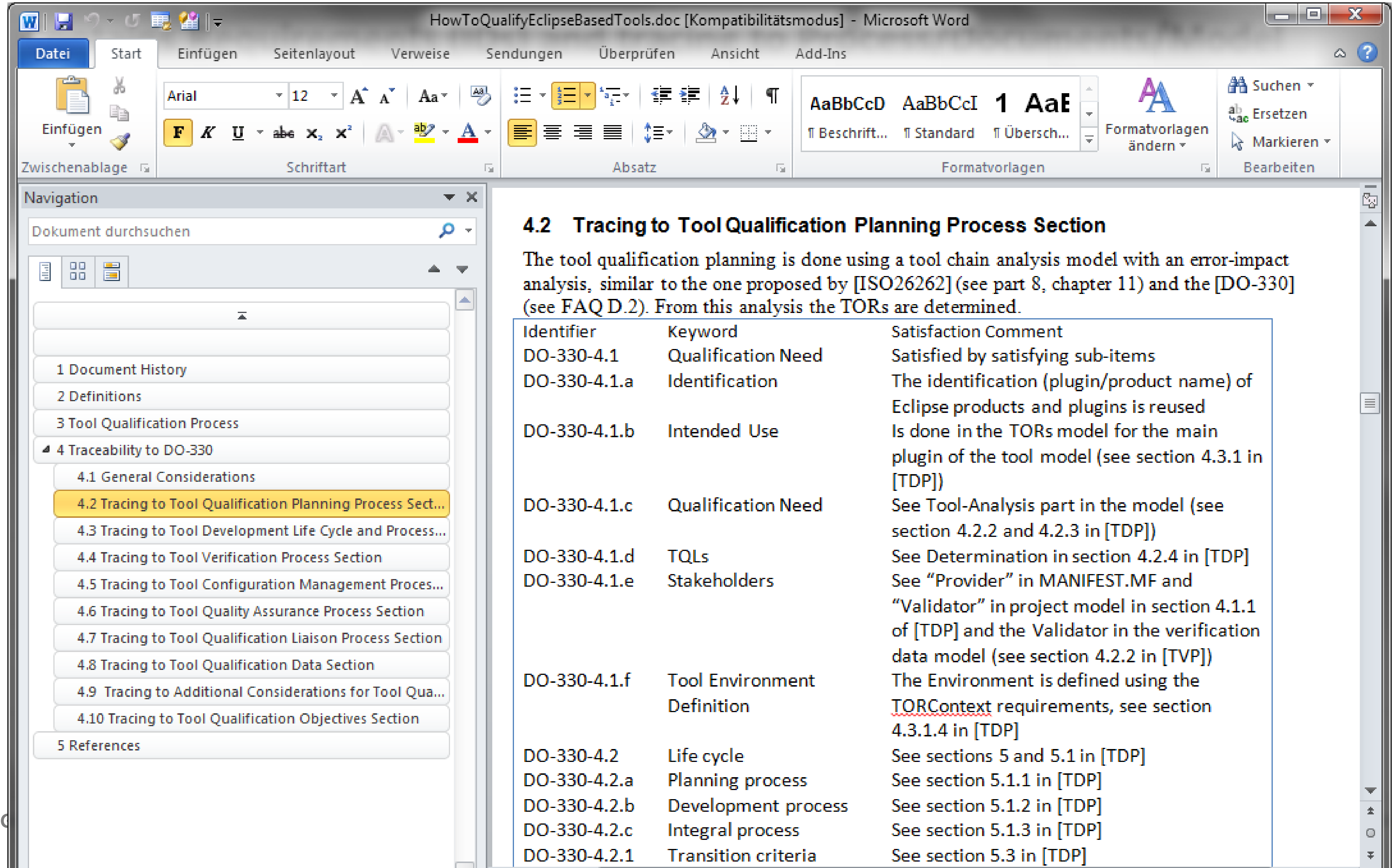
- Roadmap
- Requirements for Tool Qualification (Standards)
- Proposals for Goals for Eclipse
- Proposals for some steps towards Tool Qualification
- Steps on the road
  - **First steps: Requirements handling**
  - Second steps: Design, Coding and Test
  - Third Steps: Planning Tool Analysis and Life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



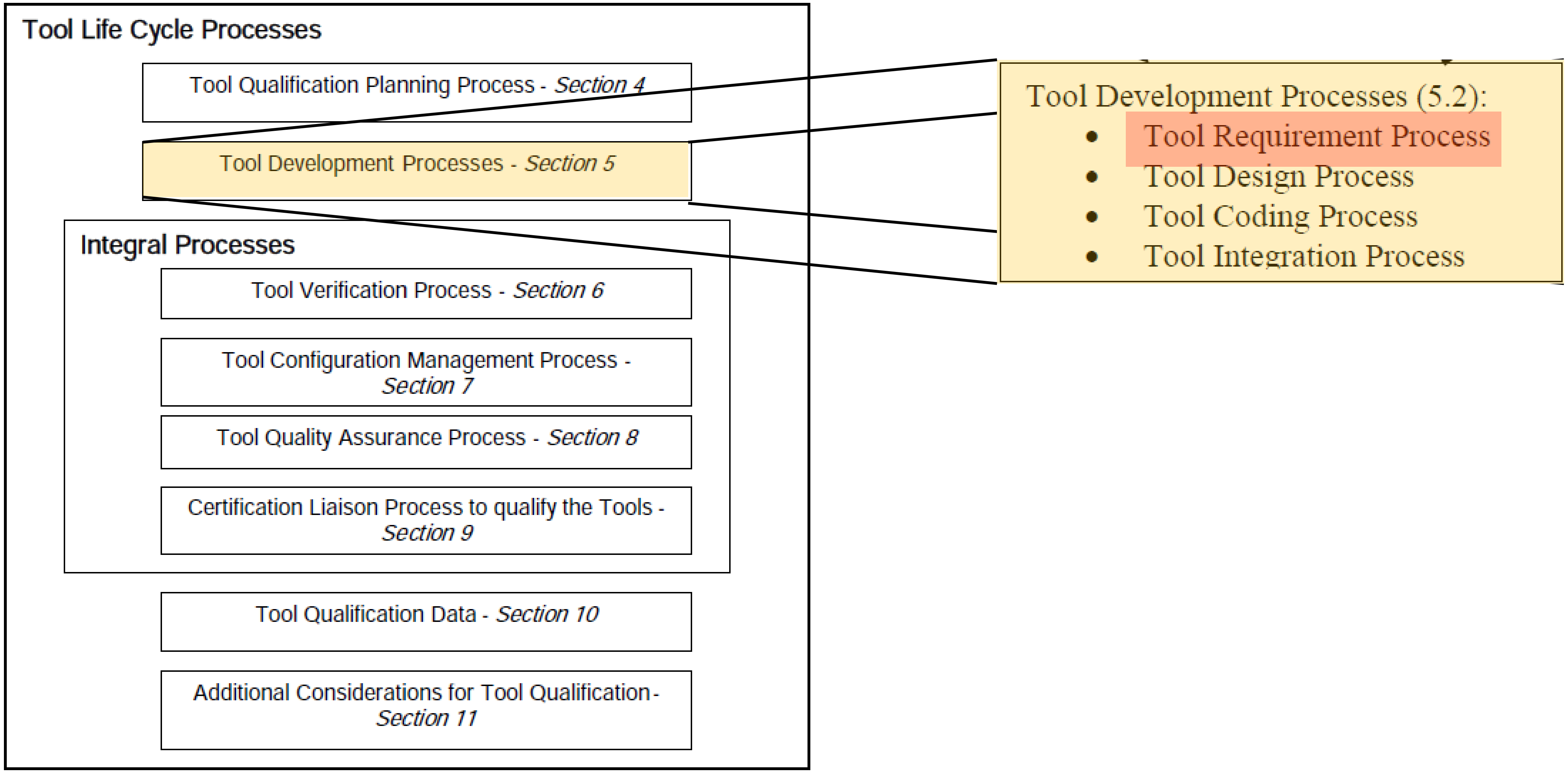
4.2 Tracing to Tool Qualification Planning Process Section

The tool qualification planning is done using a tool chain analysis model with an error-impact analysis, similar to the one proposed by [ISO26262] (see part 8, chapter 11) and the [DO-330] (see FAQ D.2). From this analysis the TORs are determined.

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 TORContext 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**



Tool Life Cycle Processes

Tool Qualification Planning Process - *Section 4*

Tool Development Processes - *Section 5*

Integral Processes

Tool Verification Process - *Section 6*

Tool Configuration Management Process - *Section 7*

Tool Quality Assurance Process - *Section 8*

Certification Liaison Process to qualify the Tools - *Section 9*

Tool Qualification Data - *Section 10*

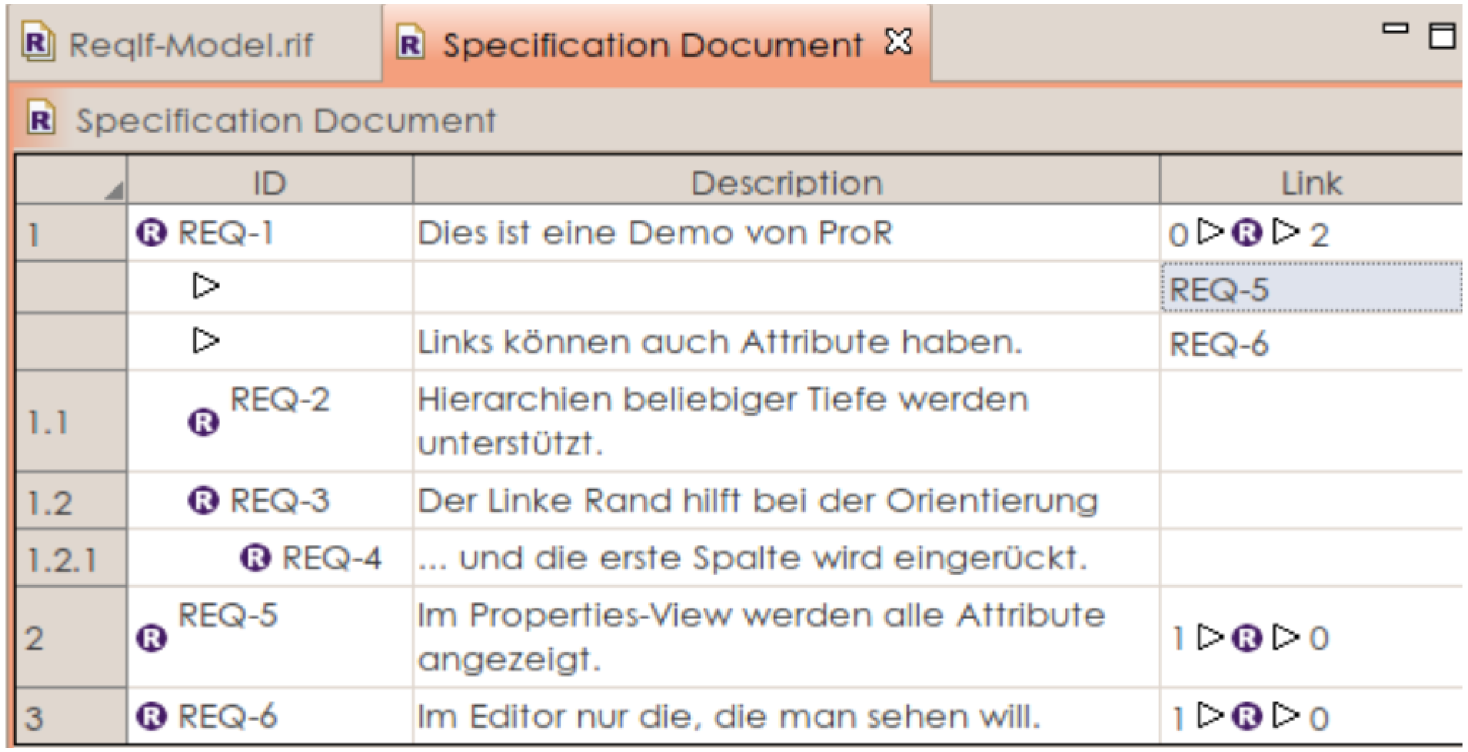
Additional Considerations for Tool Qualification - *Section 11*

Tool Development Processes (5.2):

- Tool Requirement Process
- Tool Design Process
- Tool Coding Process
- Tool Integration Process

Existing Methods: Requirements

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



Specification Document

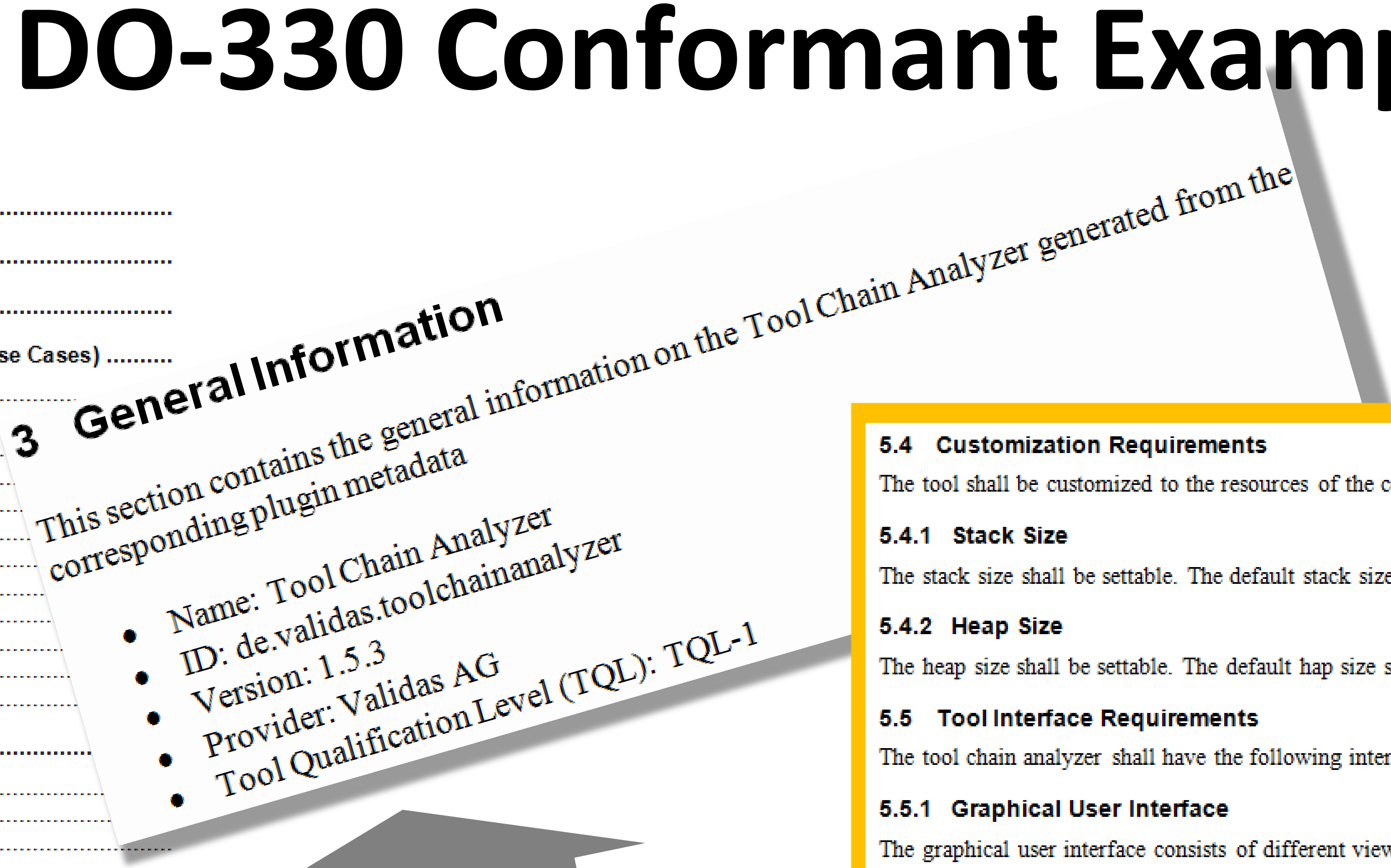
	ID	Description	Link
1	Ⓡ REQ-1	Dies ist eine Demo von ProR	0 ▷ Ⓡ ▷ 2
	▷		REQ-5
	▷	Links können auch Attribute haben.	REQ-6
1.1	Ⓡ REQ-2	Hierarchien beliebiger Tiefe werden unterstützt.	
1.2	Ⓡ REQ-3	Der Linke Rand hilft bei der Orientierung	
1.2.1	Ⓡ REQ-4	... und die erste Spalte wird eingerückt.	
2	Ⓡ REQ-5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

Tool Requirements for Tool Chain Analyzer
Version 0.2

Validas AG

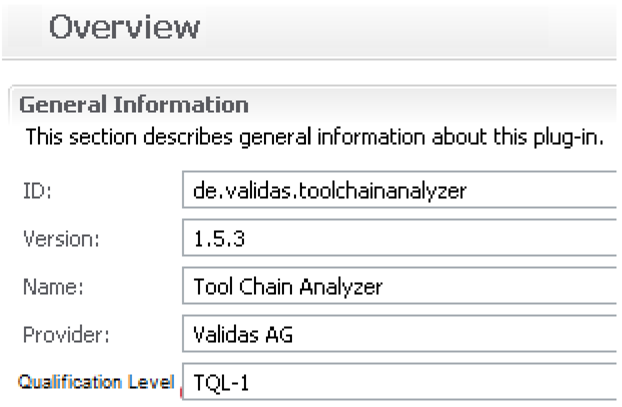


3 General Information

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 were 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 hap 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 created, move and delete elements. Furthermore it can be used to start actions like the im- 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 the 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

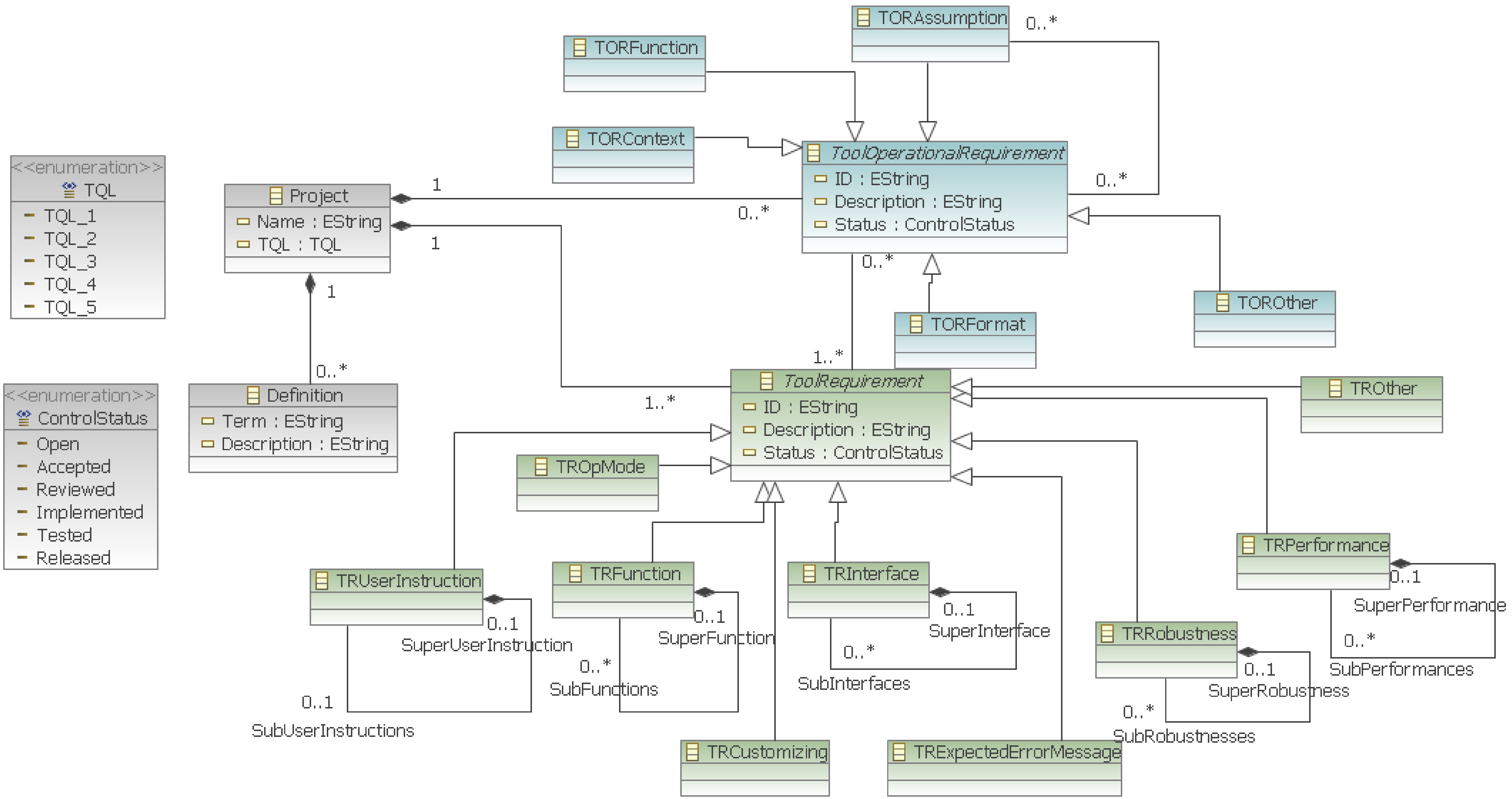
The tool chain models shall be persistent to files. The tool chain analyzer loads models from files and writes the back into files.

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.

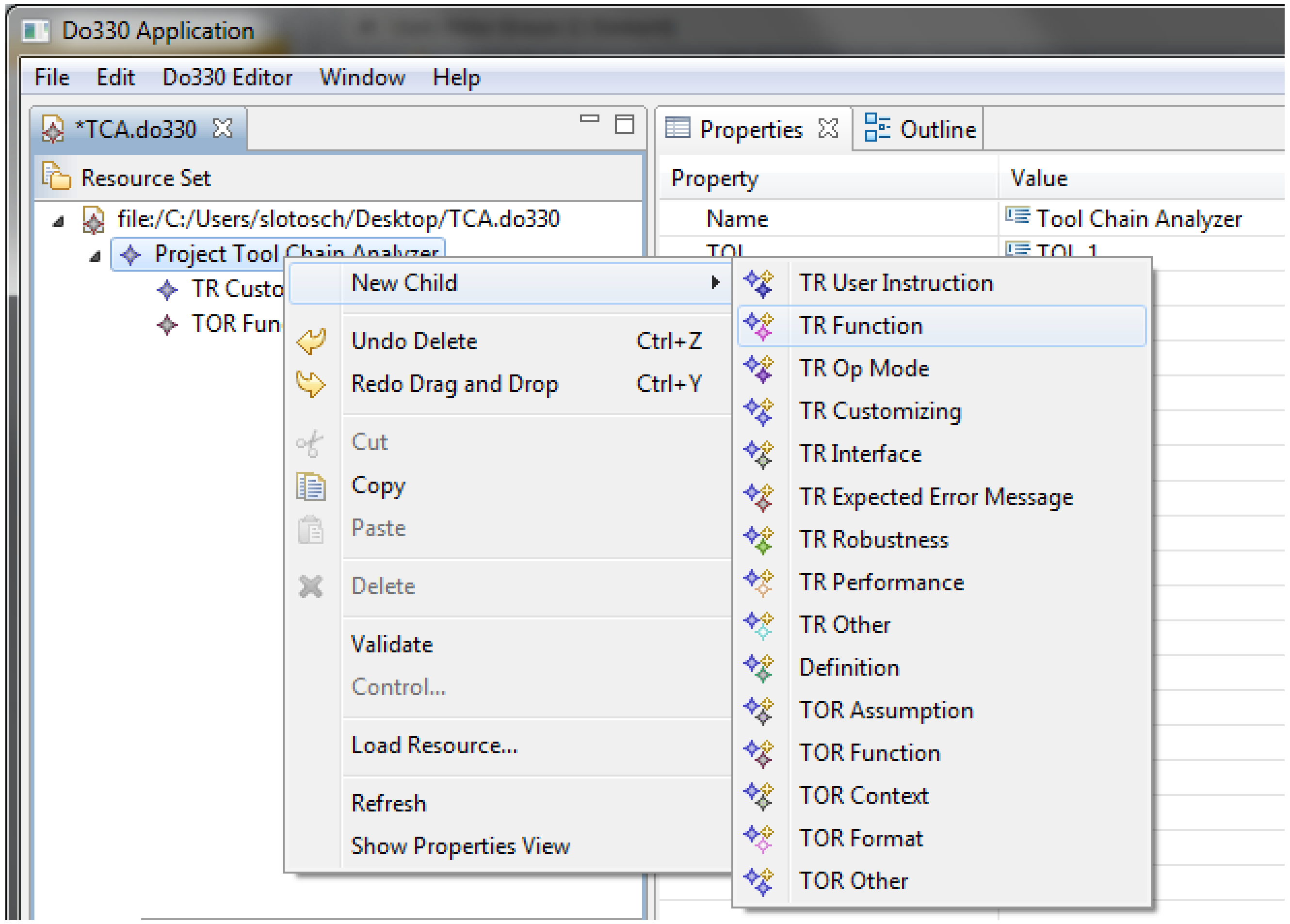
Create Model for Tool-Requirements

- **EMF-Metamodel (Draft) for Tool Requirements**



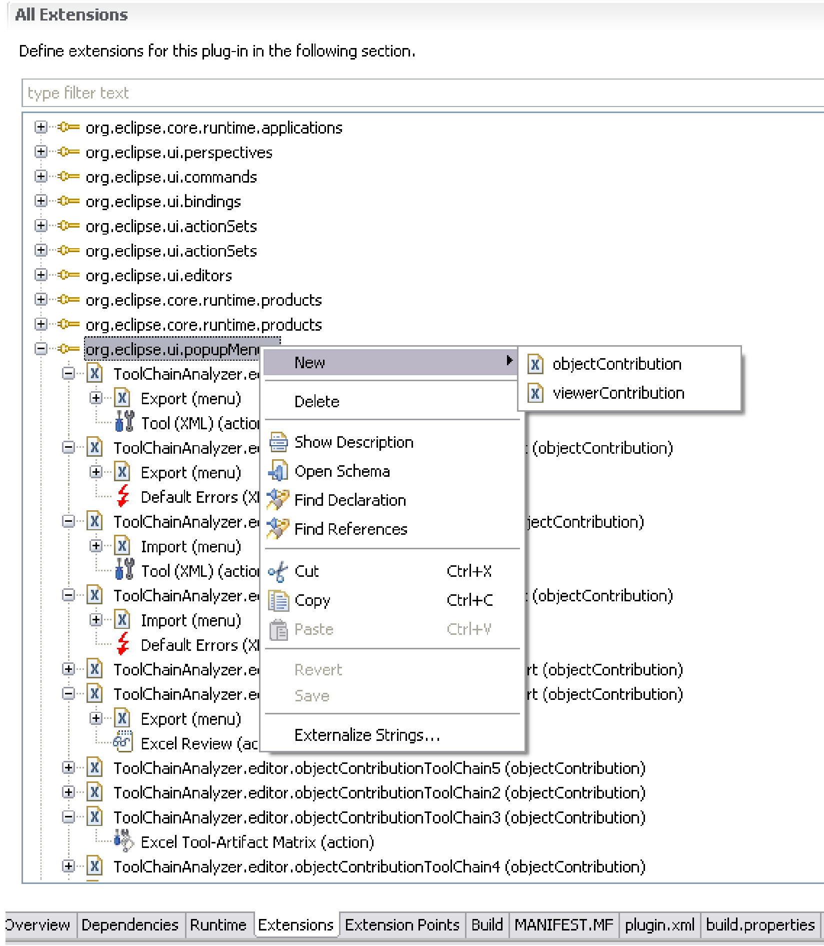
Create Example Model

- **Using the default EMF Editor**



Property	Value
Name	Tool Chain Analyzer
TOL	TOL 1

Comparable: Plugin Extension



All Extensions

Define extensions for this plug-in in the following section.

- **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

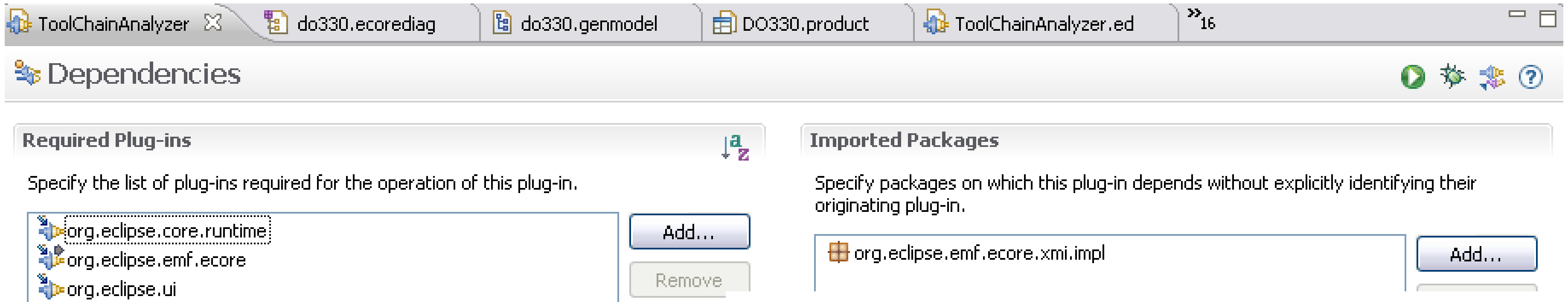
- Roadmap
- Requirements for Tool Qualification (Standards)
- Proposals for Goals for Eclipse
- Proposals for some steps towards Tool Qualification
- Steps on the road
  - First steps: Requirements handling
  - **Second steps: Design, Coding and Test**
  - Third Steps: Planning Tool Analysis and Life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



ToolChainAnalyzer | do330.ecorediag | do330.genmodel | DO330.product | ToolChainAnalyzer.ed | »16

Dependencies

Required Plug-ins

Specify the list of plug-ins required for the operation of this plug-in.

- org.eclipse.core.runtime
- org.eclipse.emf.ecore
- org.eclipse.ui
- ExcelInterface (1.0.0)

Add... | Remove | Up | Down

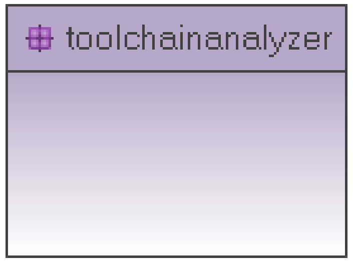
Total: 4

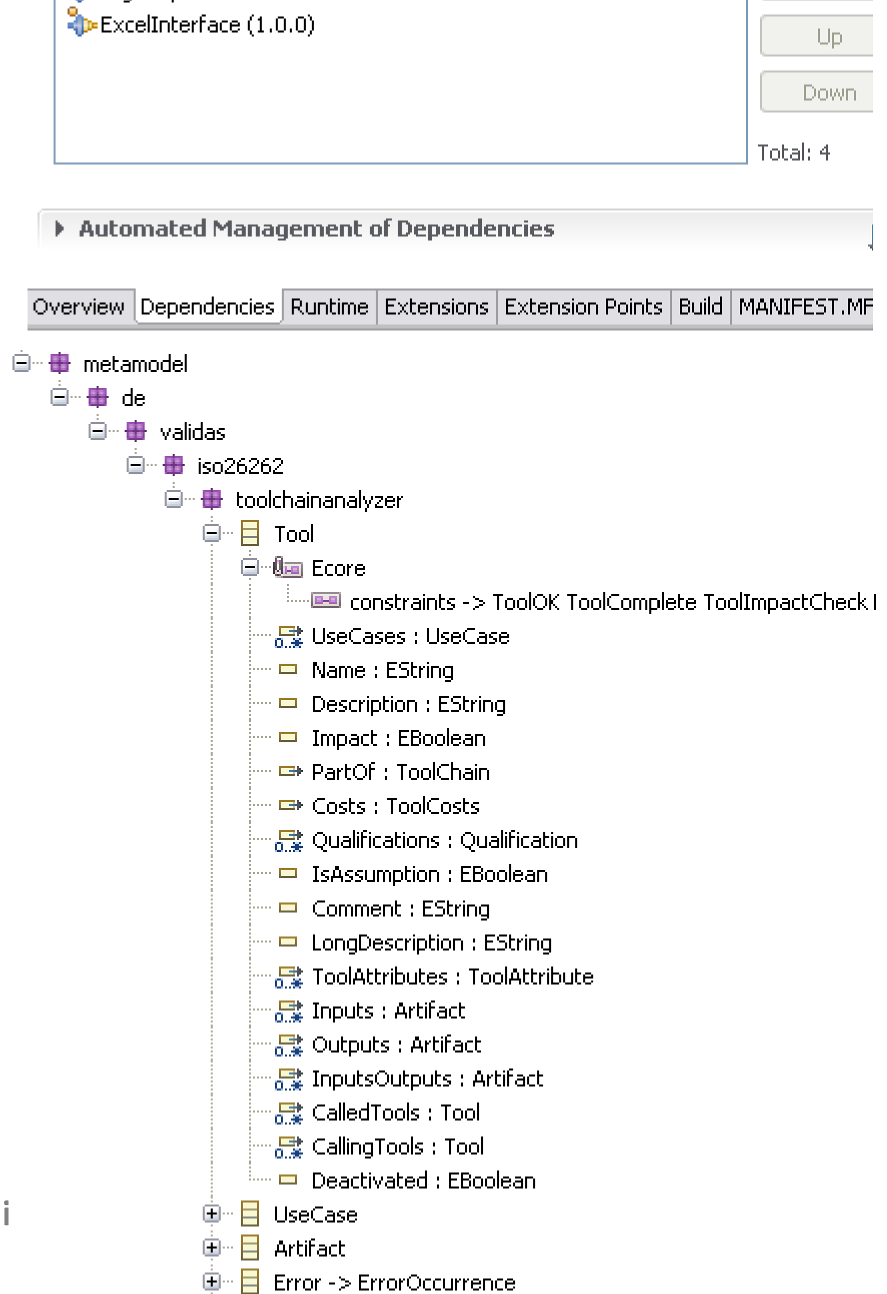
Imported Packages

Specify packages on which this plug-in depends without explicitly identifying their originating plug-in.

- org.eclipse.emf.ecore.xmi.impl

Add...

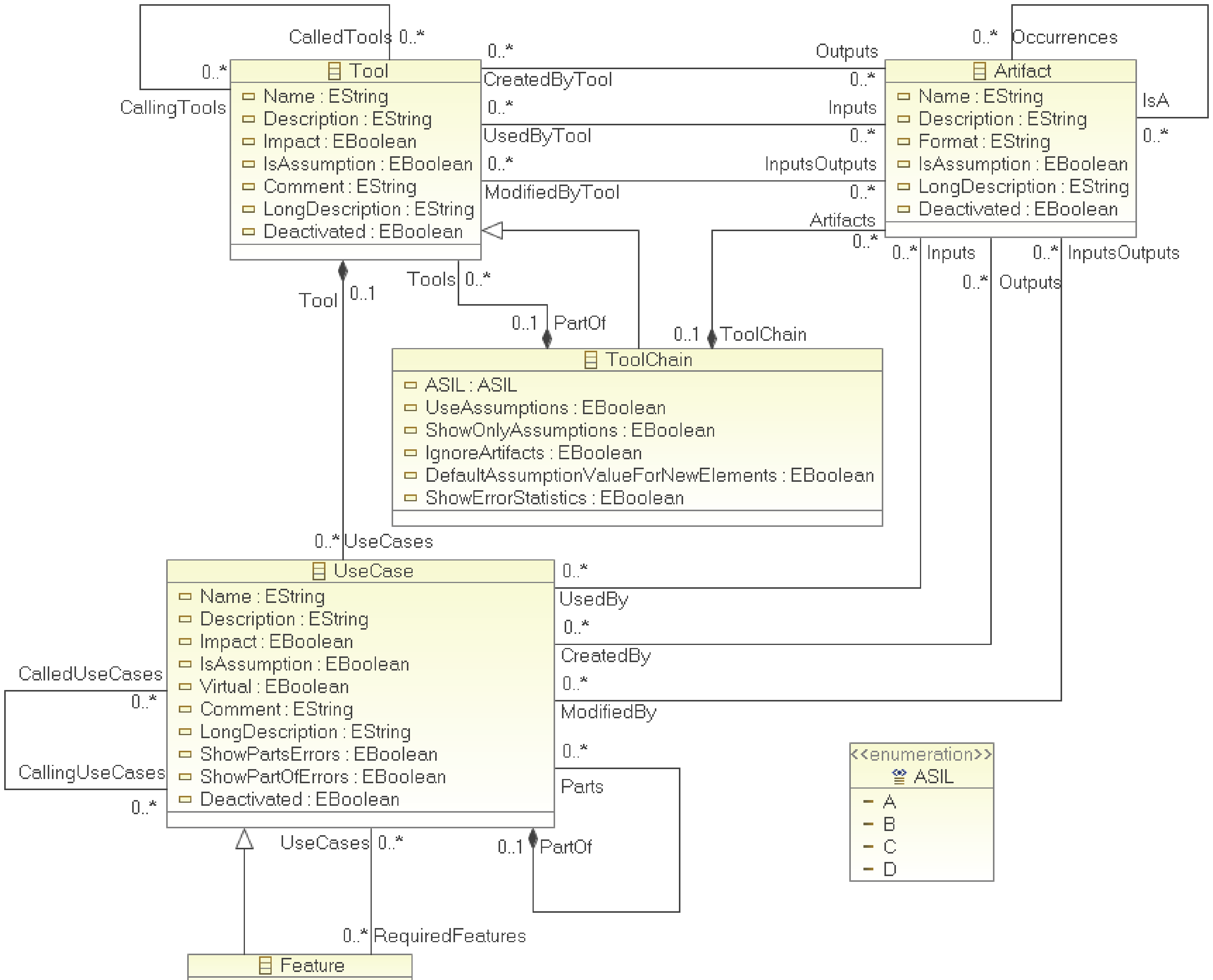




Automated Management of Dependencies

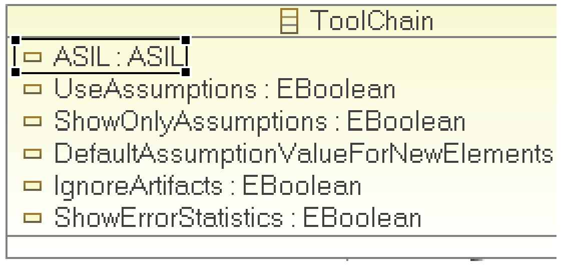
Overview | Dependencies | Runtime | Extensions | Extension Points | Build | MANIFEST.MF

- metamodel
  - de
    - validas
      - iso26262
        - toolchainanalyzer
          - Tool
            - Ecore
              - constraints -> ToolOK ToolComplete ToolImpactCheck I
            - UseCases : UseCase
            - Name : EString
            - Description : EString
            - Impact : EBoolean
            - PartOf : ToolChain
            - Costs : ToolCosts
            - Qualifications : Qualification
            - IsAssumption : EBoolean
            - Comment : EString
            - LongDescription : EString
            - ToolAttributes : ToolAttribute
            - Inputs : Artifact
            - Outputs : Artifact
            - InputsOutputs : Artifact
            - CalledTools : Tool
            - CallingTools : Tool
            - Deactivated : EBoolean
          - UseCase
          - Artifact
          - Error -> ErrorOccurrence

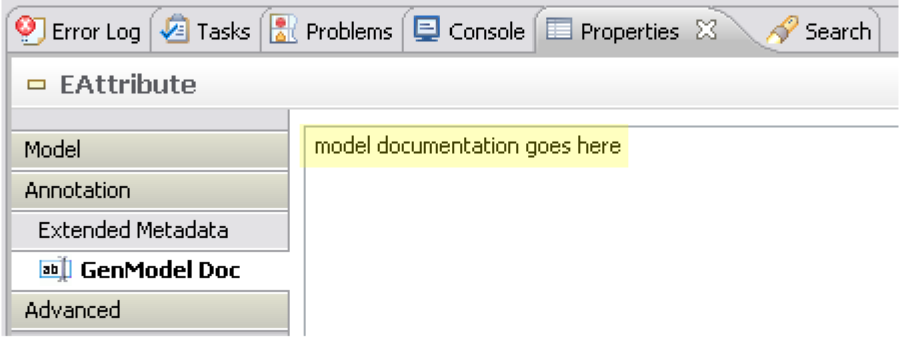


Example EMF Code

- **Model generates code, interface (and description!)**

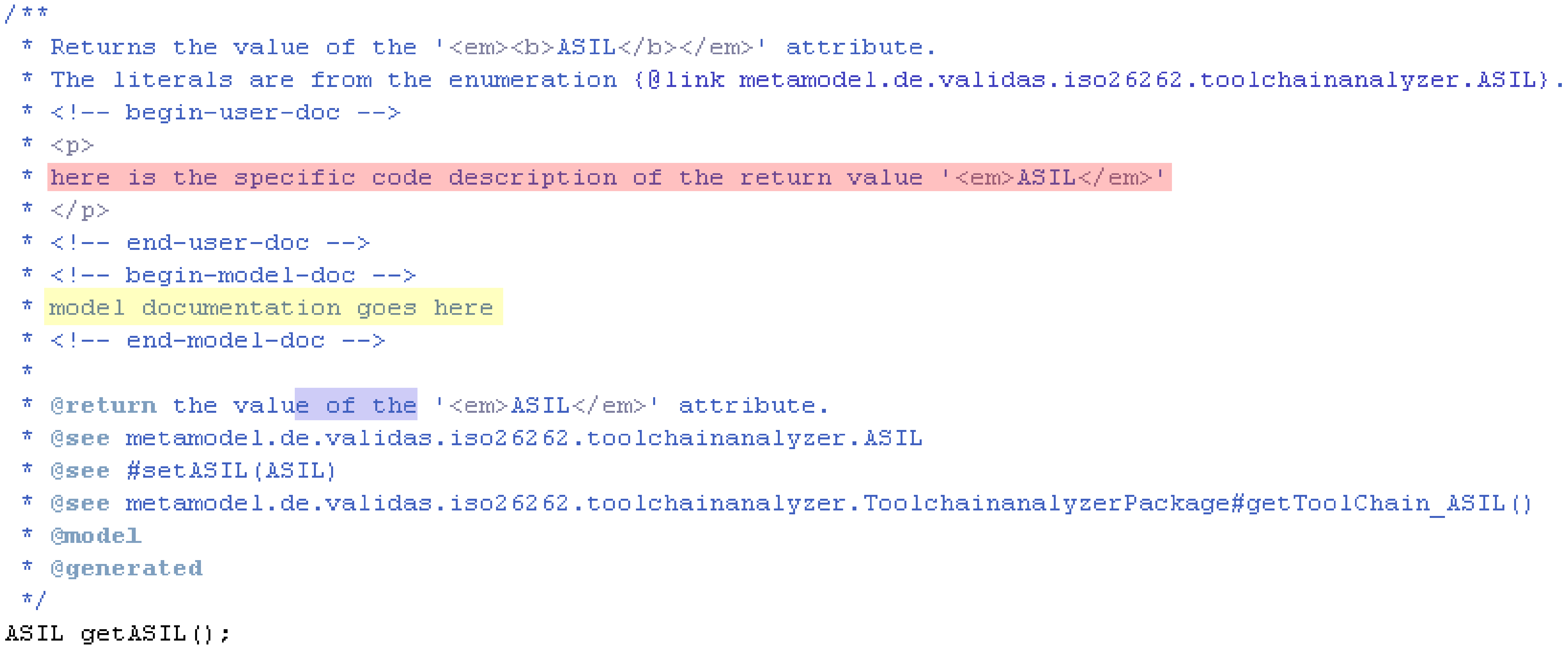


ToolChain
ASIL : ASIL
UseAssumptions : EBoolean
ShowOnlyAssumptions : EBoolean
DefaultAssumptionValueForNewElements
IgnoreArtifacts : EBoolean
ShowErrorStatistics : EBoolean



model documentation goes here

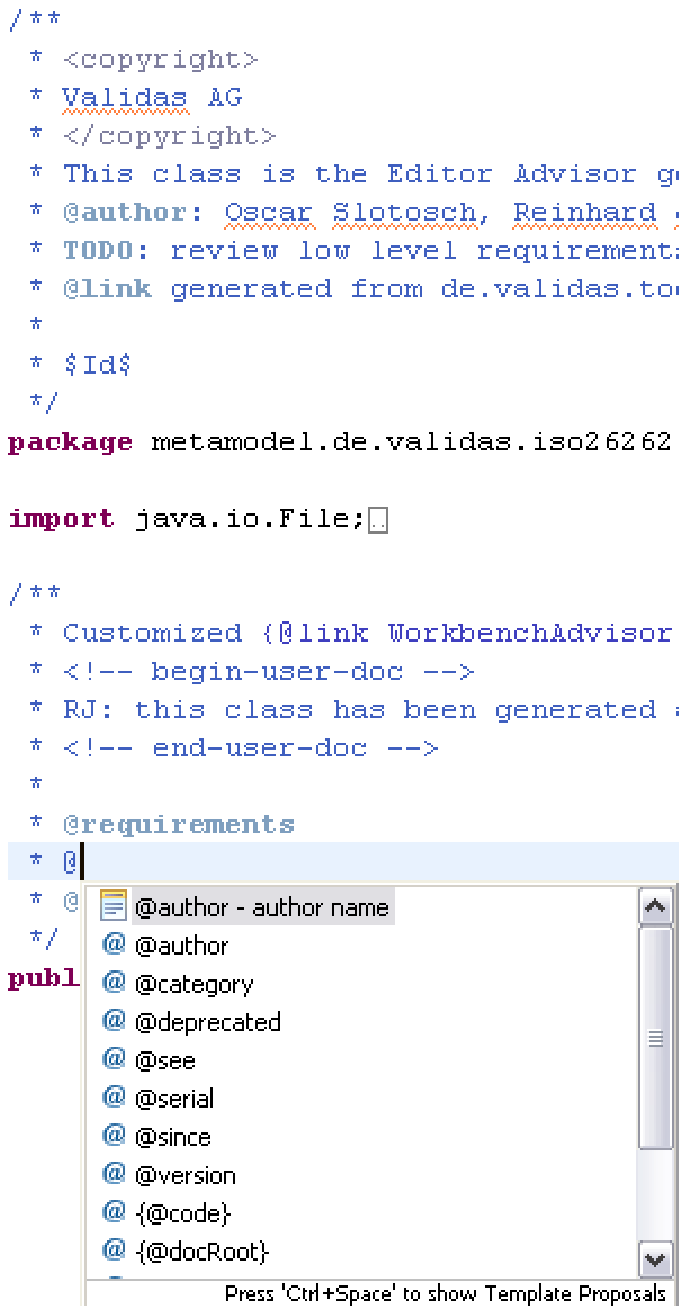
- **Documentation can be inserted into code and model**



```
/**
 * 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?)

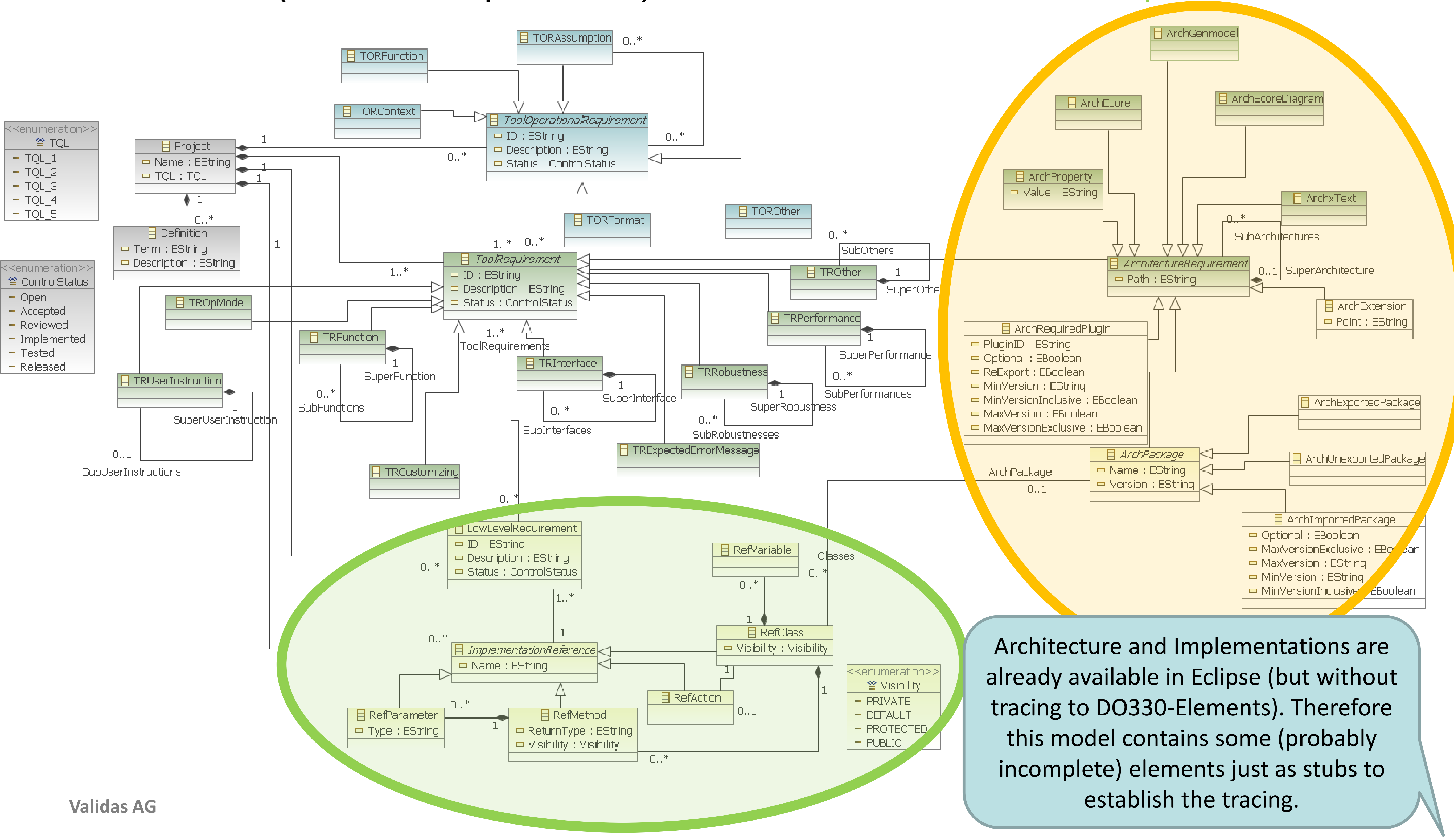


```
/**
 * <copyright>
 * Validas AG
 * </copyright>
 * This class is the Editor Advisor g
 * @author: Oscar Slotosch, Reinhard 
 * TODO: review low level requirement
 * @link generated from de.validas.to
 *
 * $Id$
 */
package metamodel.de.validas.iso26262

import java.io.File;

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

Press 'Ctrl+Space' to show Template Proposals

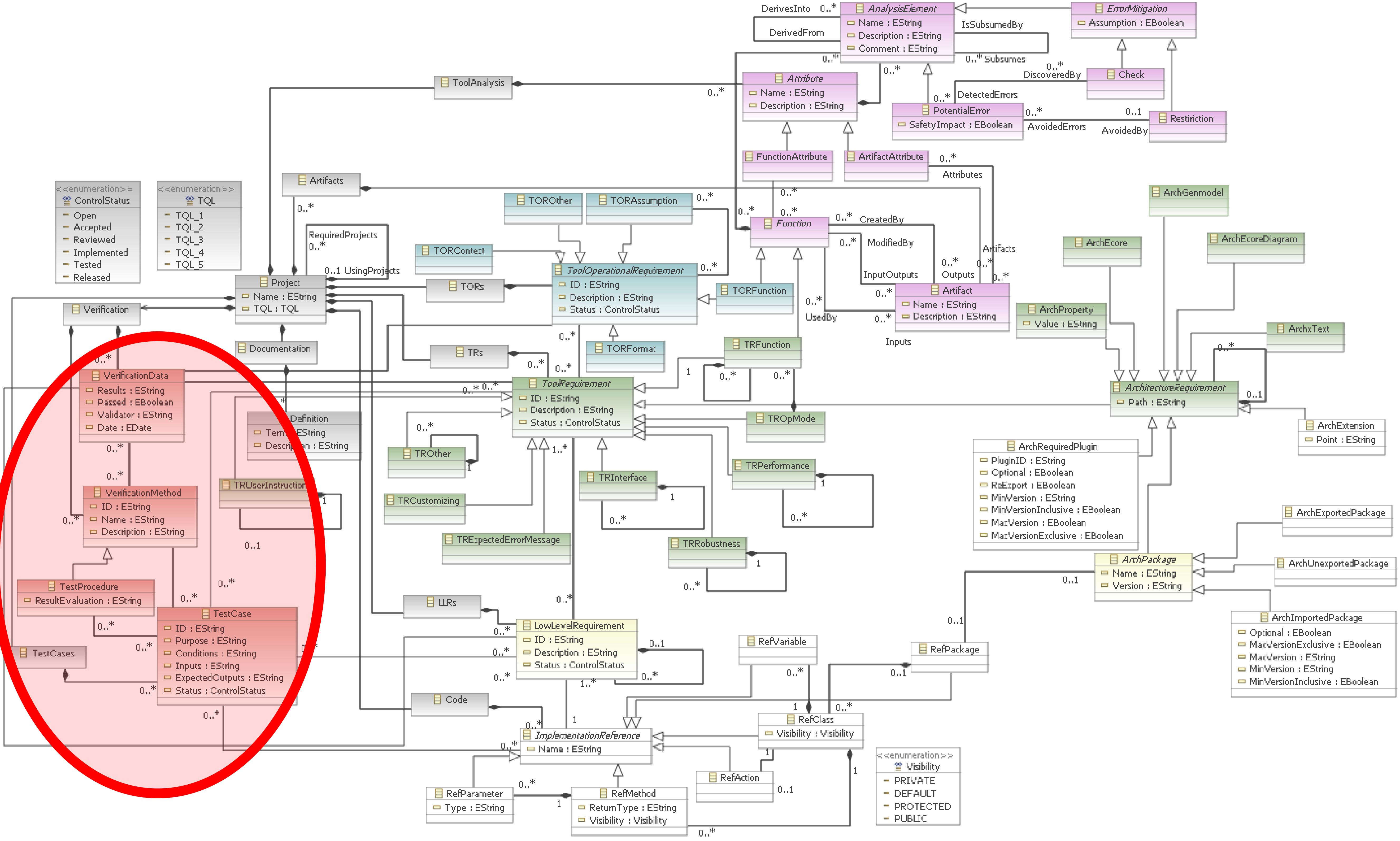
Design Model

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

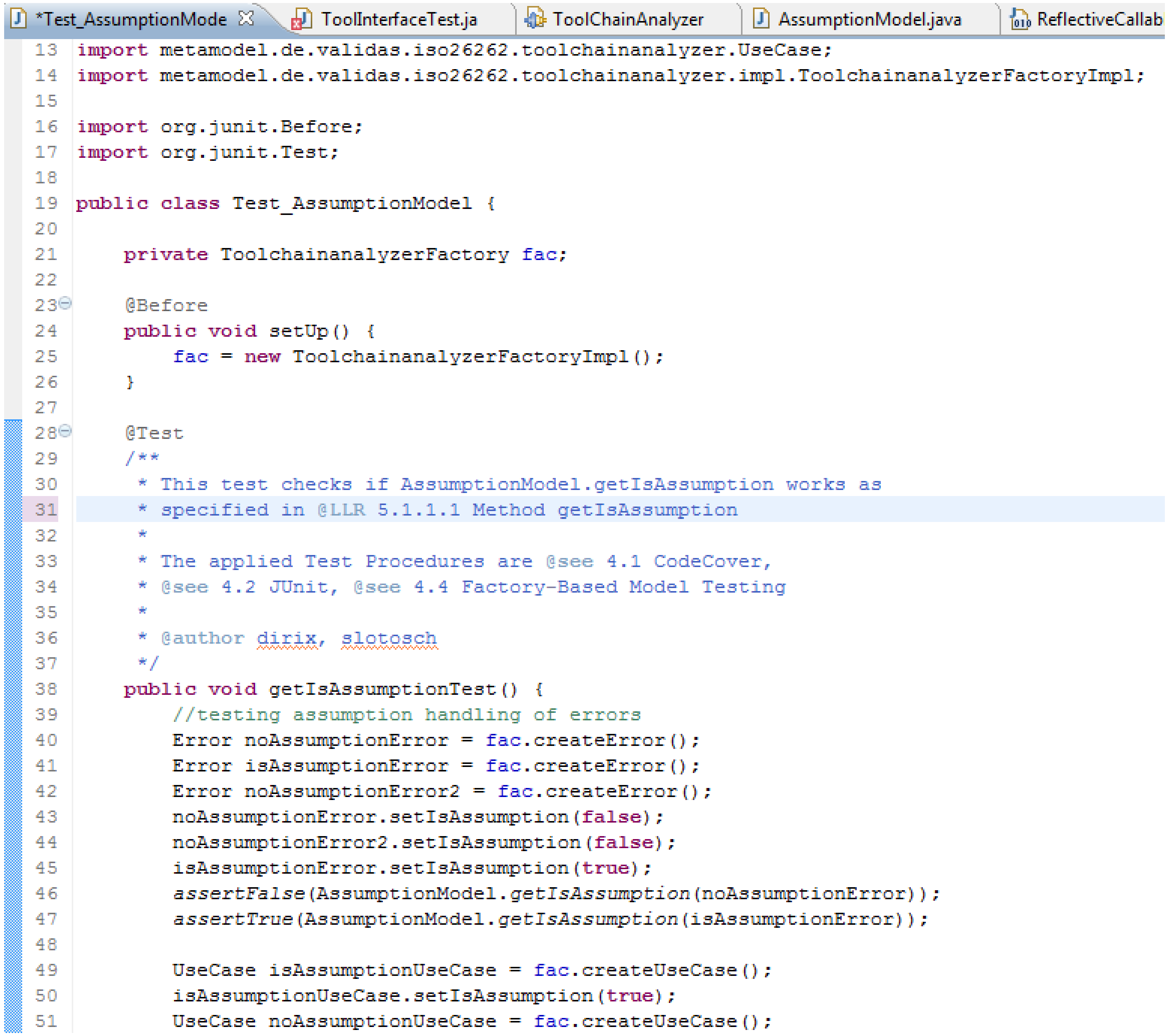


Architecture and Implementations are already available in Eclipse (but without tracing to DO330-Elements). Therefore this model contains some (probably incomplete) elements just as stubs to establish the tracing.

Test Model

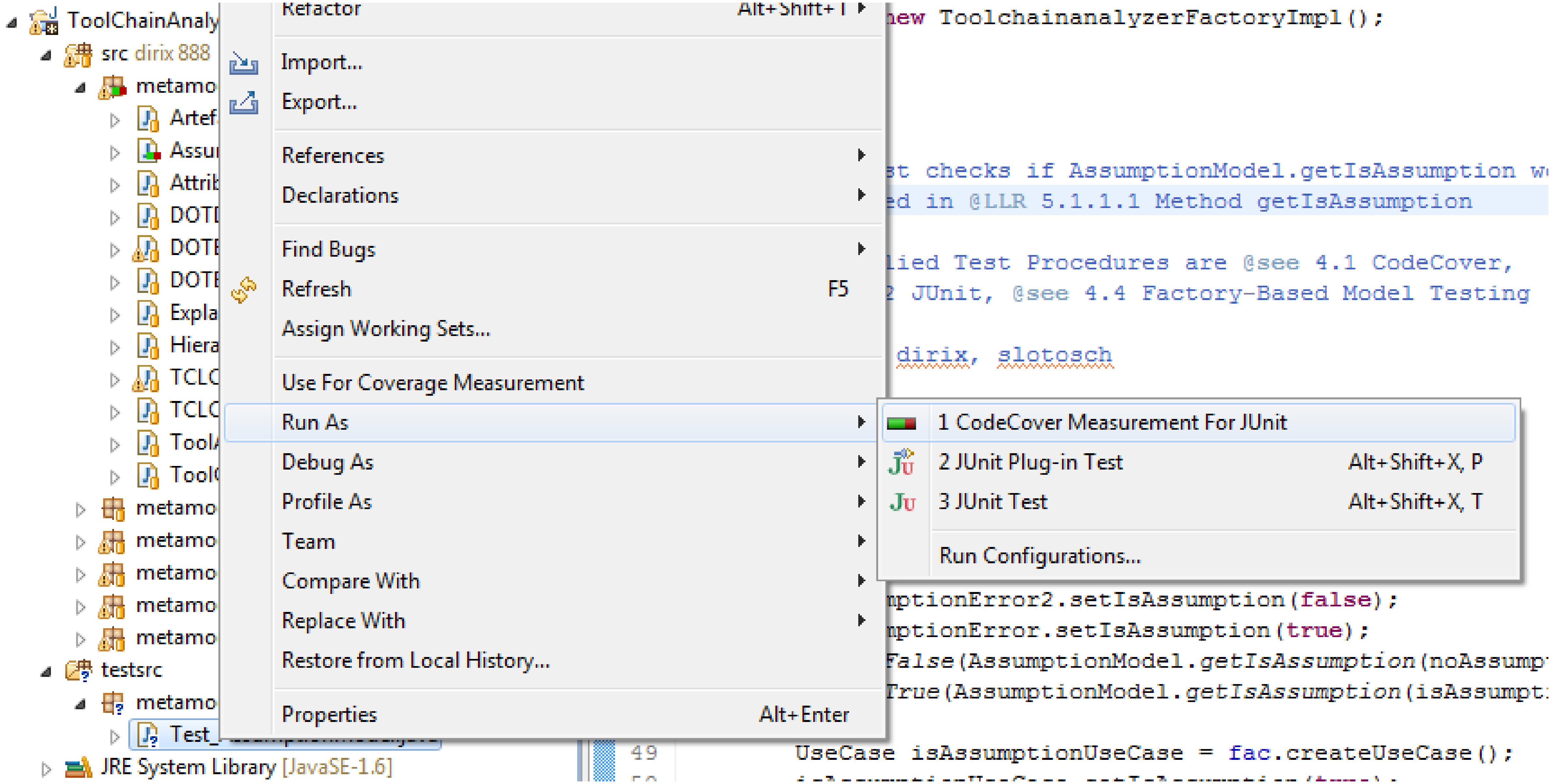

AnalysisElement
Name : EString
Description : EString
Comment : EString
DerivesInto
DerivedFrom
IsSubsumedBy
Subsumes
ErrorMitigation
Assumption : EBoolean
Check
DiscoveredBy
DetectedErrors
PotentialError
SafetyImpact : EBoolean
AvoidedErrors
AvoidedBy
Restriction
ToolAnalysis
Attribute
Name : EString
Description : EString
FunctionAttribute
ArtifactAttribute
Attributes
<<enumeration>>
ControlStatus
Open
Accepted
Reviewed
Implemented
Tested
Released
<<enumeration>>
TQL
TQL_1
TQL_2
TQL_3
TQL_4
TQL_5
Artifacts
TOROther
TORAssumption
Function
CreatedBy
ModifiedBy
InputOutputs
Outputs
Inputs
UsedBy
ArchGenmodel
ArchEcore
ArchEcoreDiagram
TORContext
RequiredProjects
UsingProjects
ToolOperationalRequirement
ID : EString
Description : EString
Status : ControlStatus
Project
Name : EString
TQL : TQL
TORs
TORFunction
Artifact
Name : EString
Description : EString
ArchProperty
Value : EString
ArchxText
Verification
Documentation
TORFormat
TRs
TRFunction
ArchitectureRequirement
Path : EString
ArchExtension
Point : EString
VerificationData
Results : EString
Passed : EBoolean
Validator : EString
Date : EDate
ToolRequirement
ID : EString
Description : EString
Status : ControlStatus
Definition
Term : EString
Description : EString
TROpMode
TROther
ArchRequiredPlugin
PluginID : EString
Optional : EBoolean
ReExport : EBoolean
MinVersion : EString
MinVersionInclusive : EBoolean
MaxVersion : EBoolean
MaxVersionExclusive : EBoolean
TRInterface
TRPerformance
VerificationMethod
ID : EString
Name : EString
Description : EString
TRUserInstruction
TRCustomizing
ArchExportedPackage
TRExpectedErrorMessage
TRRobustness
ArchPackage
Name : EString
Version : EString
ArchUnexportedPackage
TestProcedure
ResultEvaluation : EString
LLRs
TestCase
ID : EString
Purpose : EString
Conditions : EString
Inputs : EString
ExpectedOutputs : EString
Status : ControlStatus
LowLevelRequirement
ID : EString
Description : EString
Status : ControlStatus
ArchImportedPackage
Optional : EBoolean
MaxVersionExclusive : EBoolean
MaxVersion : EString
MinVersion : EString
MinVersionInclusive : EBoolean
TestCases
RefVariable
RefPackage
Code
RefClass
Visibility : Visibility
ImplementationReference
Name : EString
<<enumeration>>
Visibility
PRIVATE
DEFAULT
PROTECTED
PUBLIC
RefAction
RefParameter
Type : EString
RefMethod
ReturnType : EString
Visibility : Visibility
0..*
0..1
1
1..*

Test Implementation



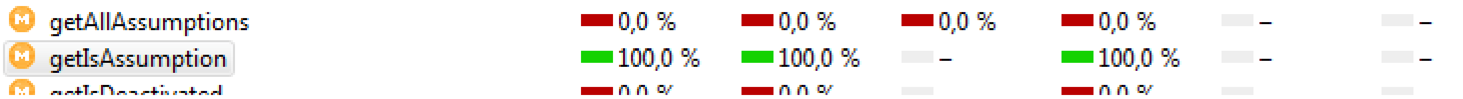
```java
import metamodel.de.validas.iso26262.toolchainanalyzer.UseCase;
import metamodel.de.validas.iso26262.toolchainanalyzer.impl.ToolchainanalyzerFactoryImpl;

import org.junit.Before;
import org.junit.Test;

public class Test_AssumptionModel {

    private ToolchainanalyzerFactory fac;

    @Before
    public void setUp() {
        fac = new ToolchainanalyzerFactoryImpl();
    }

    @Test
    /**
     * This test checks if AssumptionModel.getIsAssumption works as
     * specified in @LLR 5.1.1.1 Method getIsAssumption
     *
     * The applied Test Procedures are @see 4.1 CodeCover,
     * @see 4.2 JUnit, @see 4.4 Factory-Based Model Testing
     *
     * @author dirix, slotosch
     */
    public void getIsAssumptionTest() {
        //testing assumption handling of errors
        Error noAssumptionError = fac.createError();
        Error isAssumptionError = fac.createError();
        Error noAssumptionError2 = fac.createError();
        noAssumptionError.setIsAssumption(false);
        noAssumptionError2.setIsAssumption(false);
        isAssumptionError.setIsAssumption(true);
        assertFalse(AssumptionModel.getIsAssumption(noAssumptionError));
        assertTrue(AssumptionModel.getIsAssumption(isAssumptionError));

        UseCase isAssumptionUseCase = fac.createUseCase();
        isAssumptionUseCase.setIsAssumption(true);
        UseCase noAssumptionUseCase = fac.createUseCase();
```

Test Execution

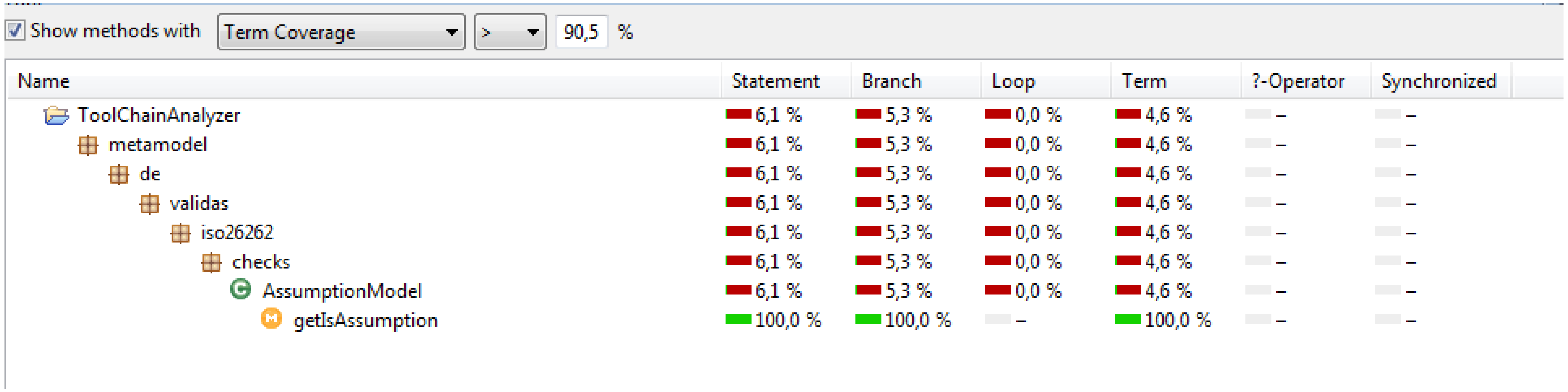


```
new ToolchainanalyzerFactoryImpl();

st 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

mptionError2.setIsAssumption(false);
mptionError.setIsAssumption(true);
False(AssumptionModel.getIsAssumption(noAssump
True(AssumptionModel.getIsAssumption(isAssumpt

      UseCase isAssumptionUseCase = fac.createUseCase();
```

Test Coverage

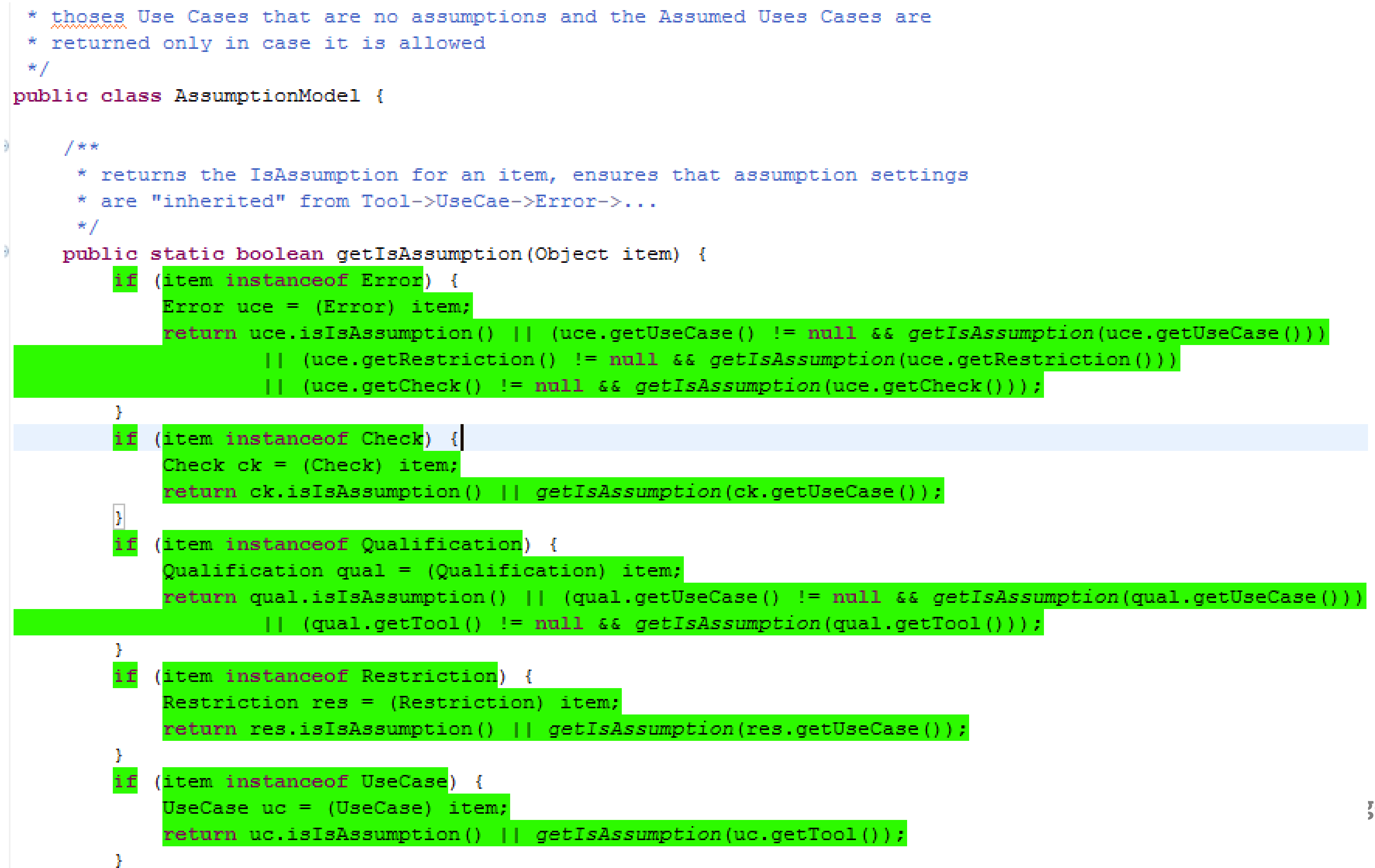


getAllAssumptions	🟥 0,0 %	🟥 0,0 %	🟥 0,0 %	🟥 0,0 %	–	–
getIsAssumption	🟩 100,0 %	🟩 100,0 %	–	🟩 100,0 %	–	–
getIsDeactivated	🟥 0,0 %	🟥 0,0 %		🟥 0,0 %		



☑ 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());
        }
```

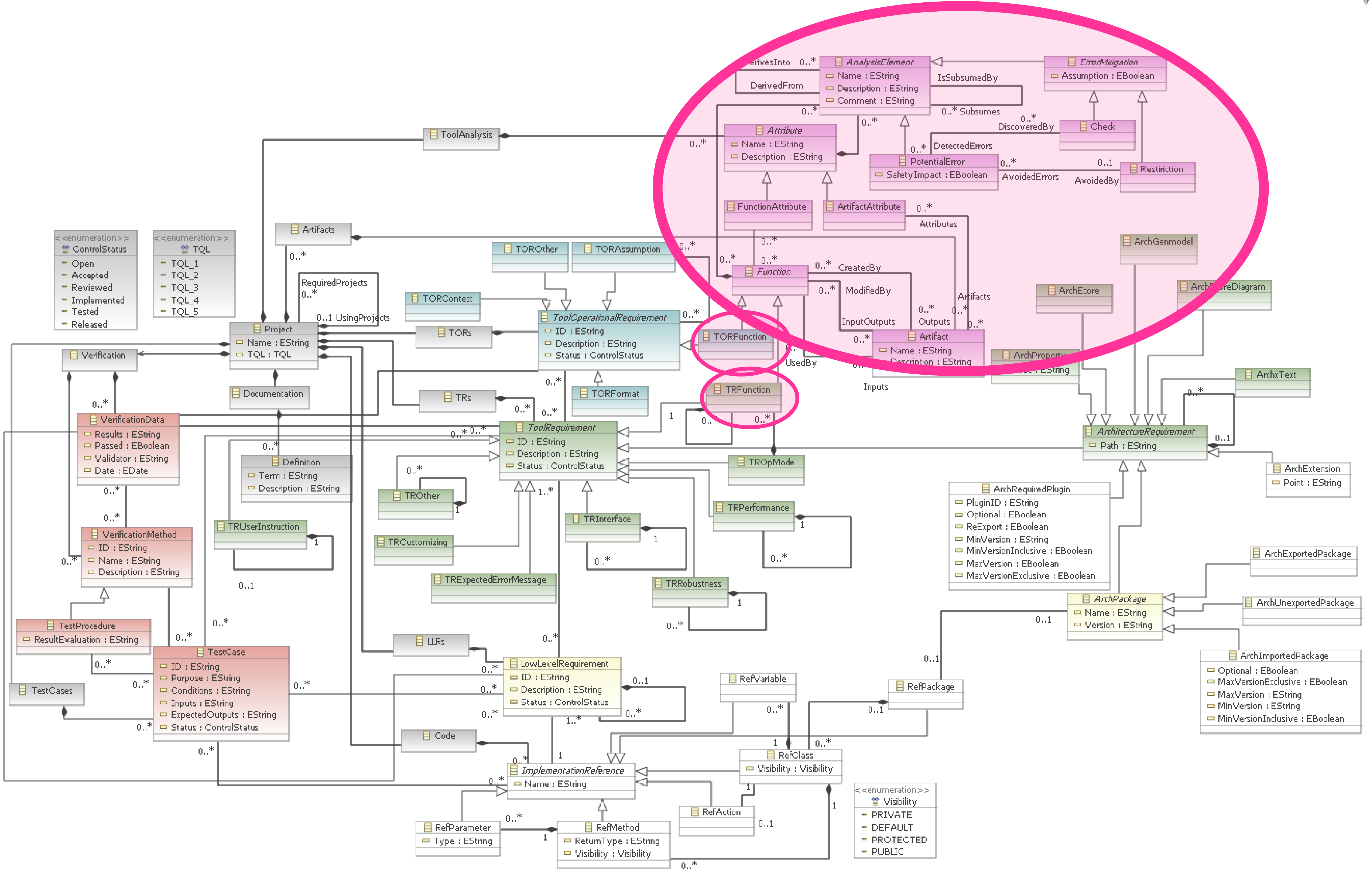
Content

- 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
  - Sixth Steps: Verification, Multi-Function, Coupling, Independency, COTS
- Summary

Planning: Tool Analysis for PSAC

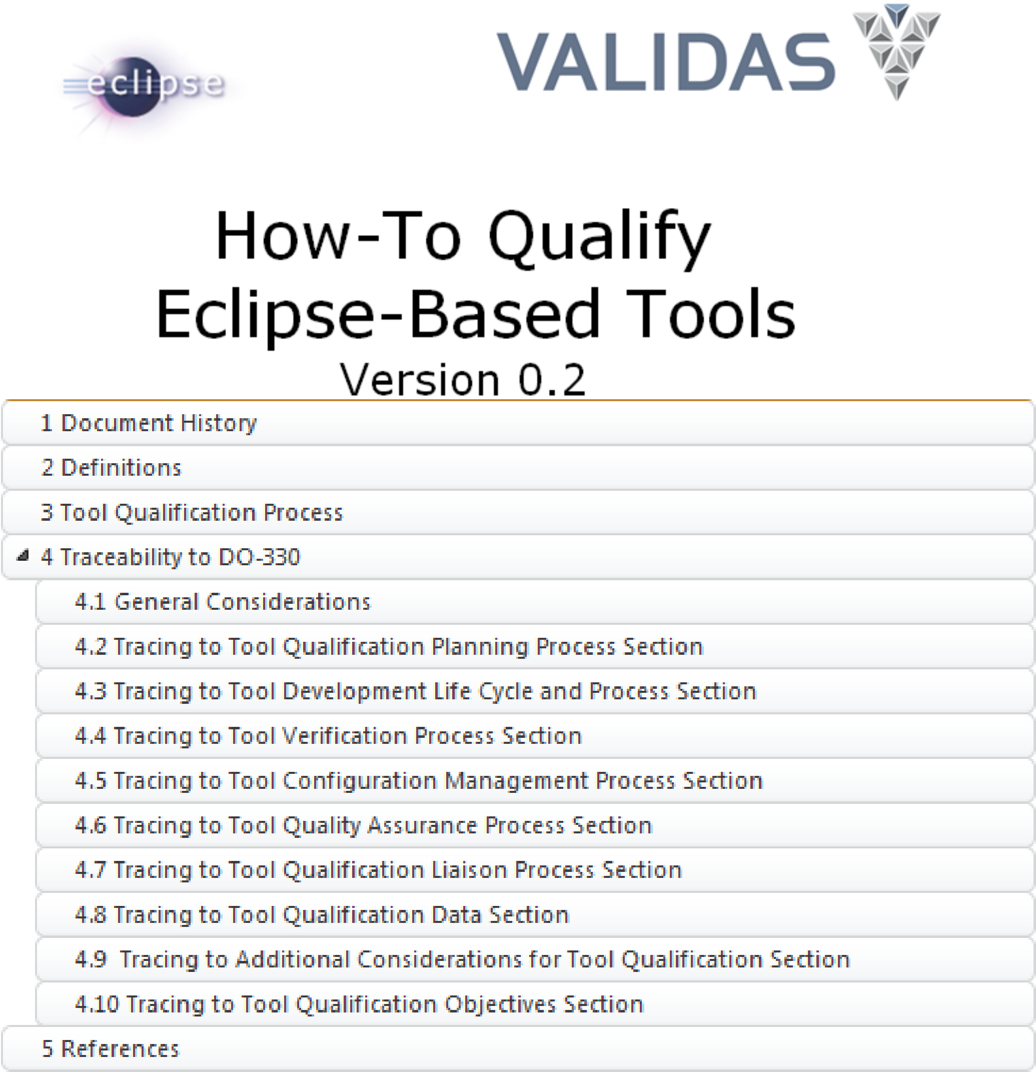
- **Determines "qualification needs" of used tools**
- **Qualification Need: "Required Confidence => Tool Qualification Level (TQL)"**
- **Required Confidence (and mapping to TQL) depends on domains**
  - DO-178C: Criteria 1 – Criteria 3
  - ISO: Tool Confidence Level based on Error Analysis in Use Cases
  - 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



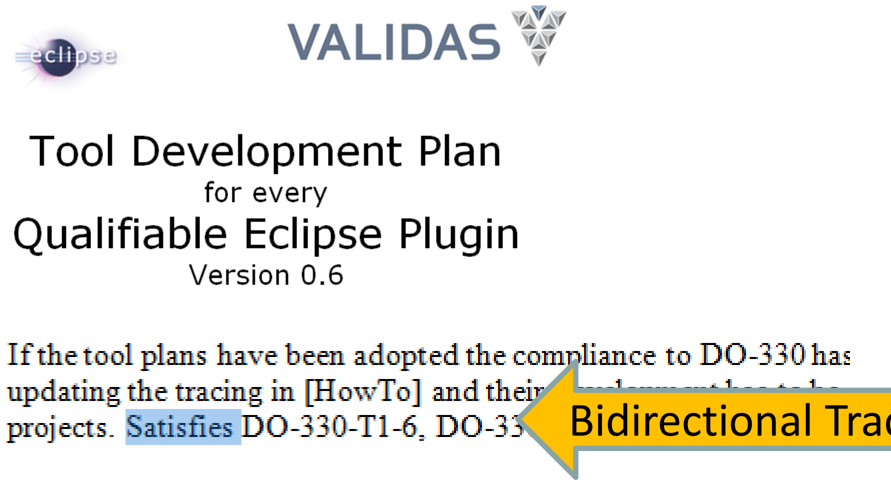
Planning: “How-To Qualify” Document

- **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**



VALIDAS

How-To Qualify
Eclipse-Based Tools
Version 0.2

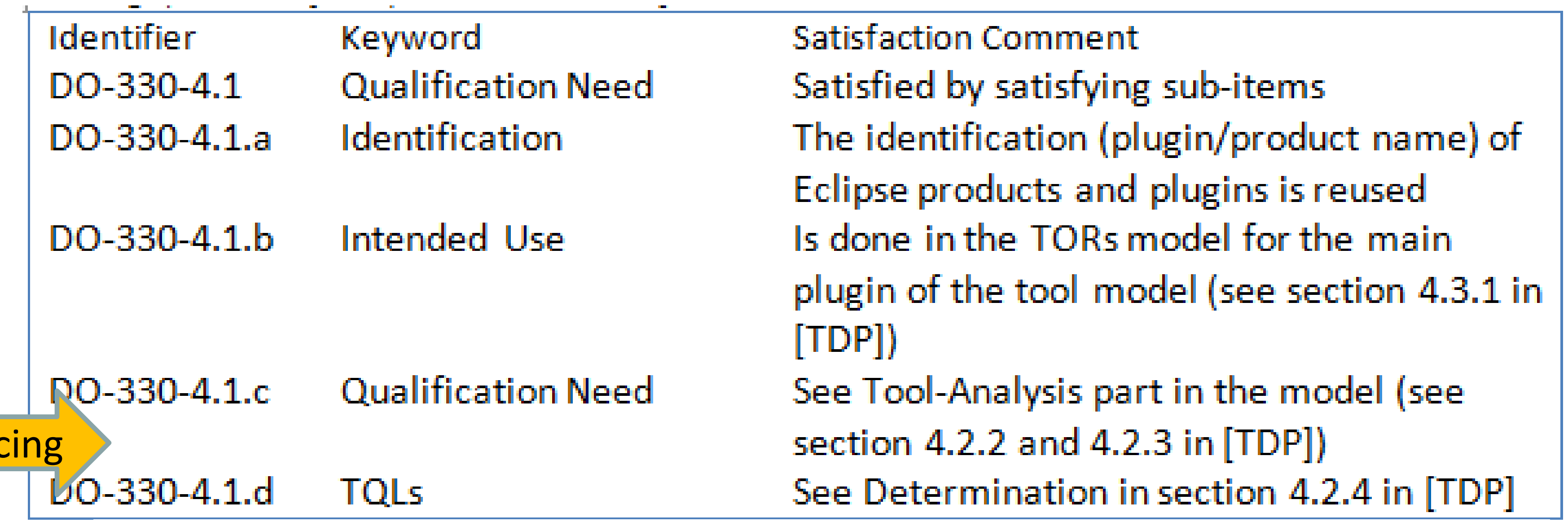


VALIDAS

Tool Development Plan
for every
Qualifiable Eclipse Plugin
Version 0.6

If the tool plans have been adopted the compliance to DO-330 has updating the tracing in [HowTo] and their
projects. Satisfies DO-330-T1-6, DO-33

Bidirectional Tracing

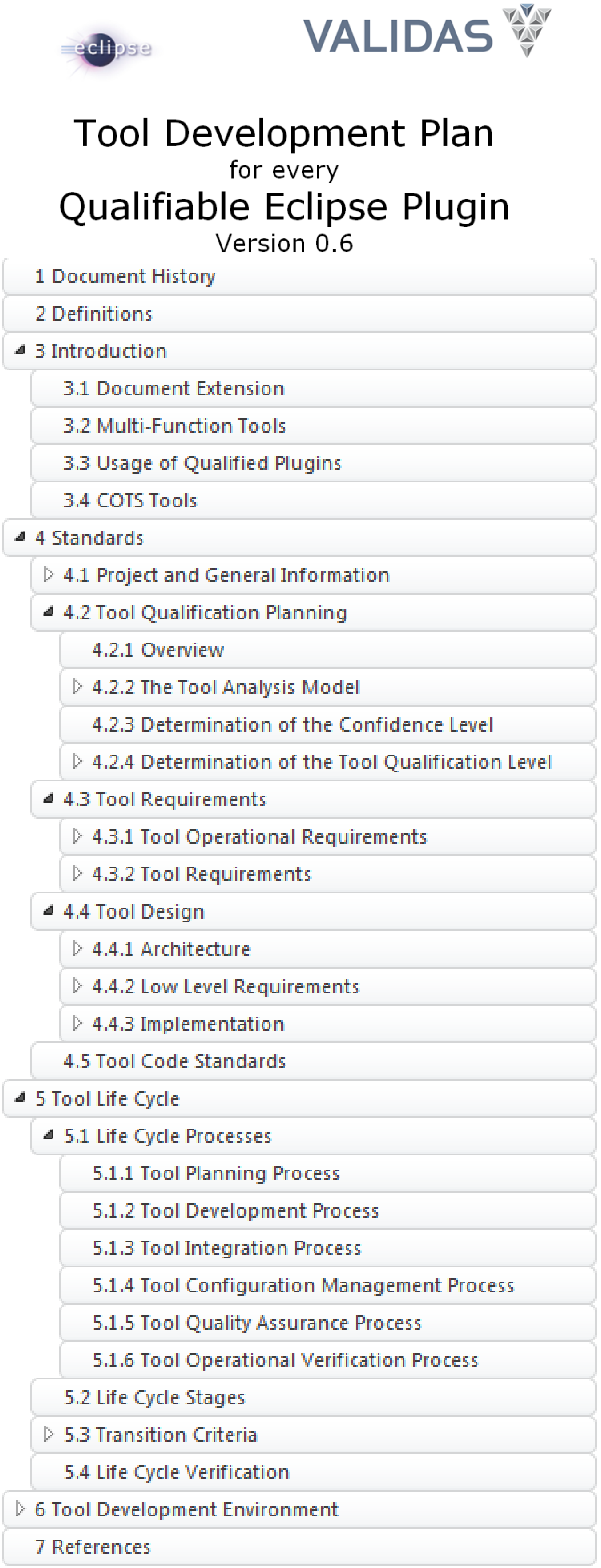


Identifier	Keyword	Satisfaction Comment
DO-330-4.1	Qualification Need	Satisfied by satisfying sub-items
DO-330-4.1.a	Identification	The identification (plugin/product name) of Eclipse products and plugins is reused
DO-330-4.1.b	Intended Use	Is done in the TORs model for the main plugin of the tool model (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

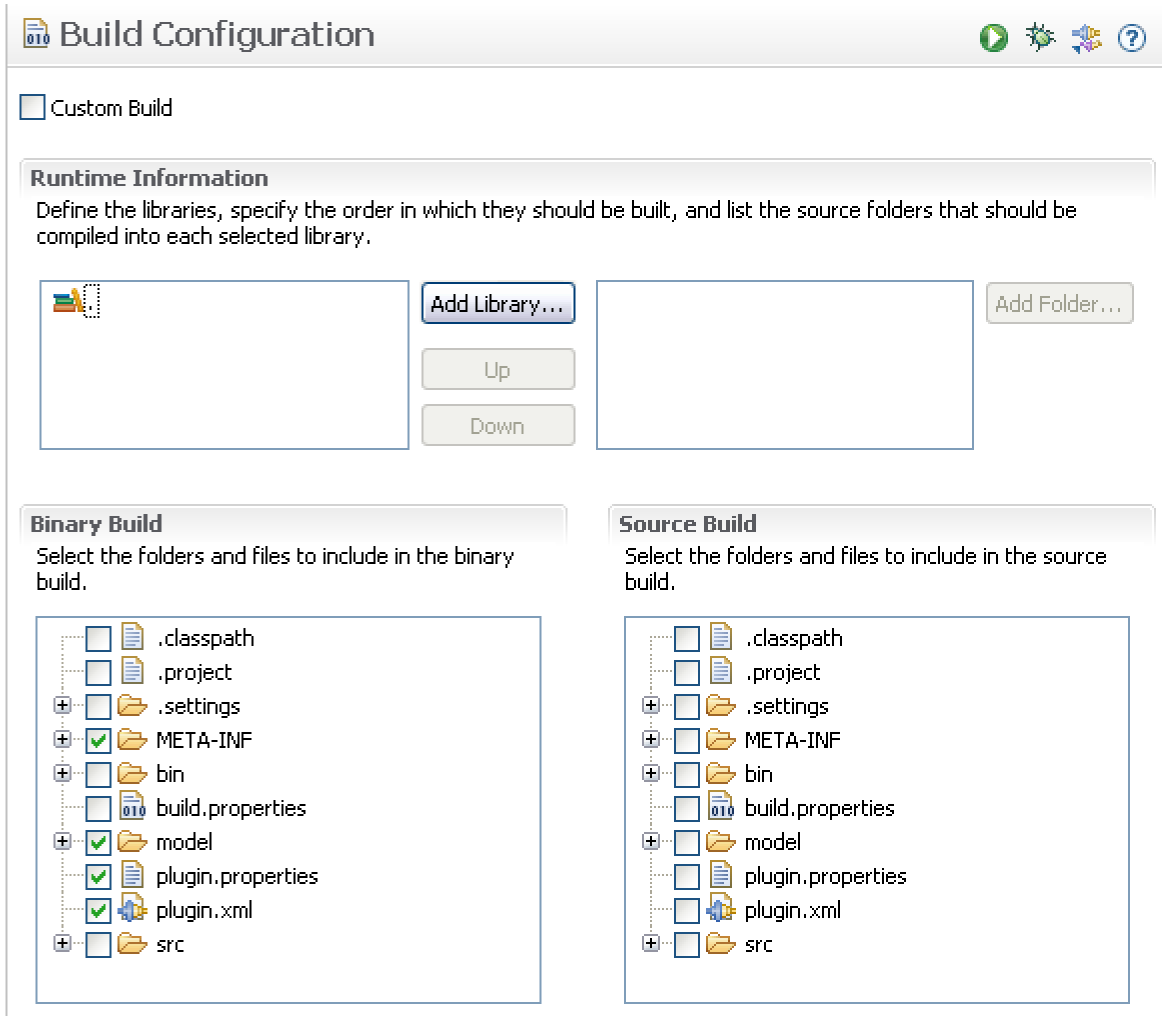
- General Process Description for Qualifiable Eclipse Plugins
- Compliant to DO-330
- Can be adapted by developers (DO-330 compliance!)
- Contains description of how to use the model, i.e. standards for
  - Requirements: TORs, TRs
  - Design, Architecture: TRs, LLRs
  - Implementation
- Specific documents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



Tool Development Plan
for every
Qualifiable Eclipse Plugin
Version 0.6

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
  - …



Build Configuration

☐ Custom Build

Runtime Information

Define the libraries, specify the order in which they should be built, and list the source folders that should be compiled into each selected library.

Binary Build

Select the folders and files to include in the binary build.

- ☐ .classpath
- ☐ .project
- ☐ .settings
- ☑ META-INF
- ☐ bin
- ☐ build.properties
- ☑ model
- ☑ plugin.properties
- ☑ plugin.xml
- ☐ src

Source Build

Select the folders and files to include in the source build.

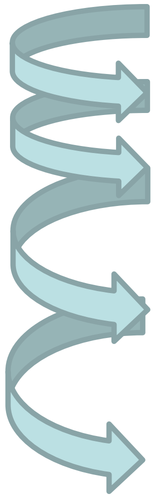
- ☐ .classpath
- ☐ .project
- ☐ .settings
- ☐ META-INF
- ☐ bin
- ☐ build.properties
- ☐ model
- ☐ plugin.properties
- ☐ plugin.xml
- ☐ src

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
  - Sixth Steps: Verification, Multi-Function, Coupling, Independency, COTS
- Summary

Tool Life Cycle for Qualifiable Plugins

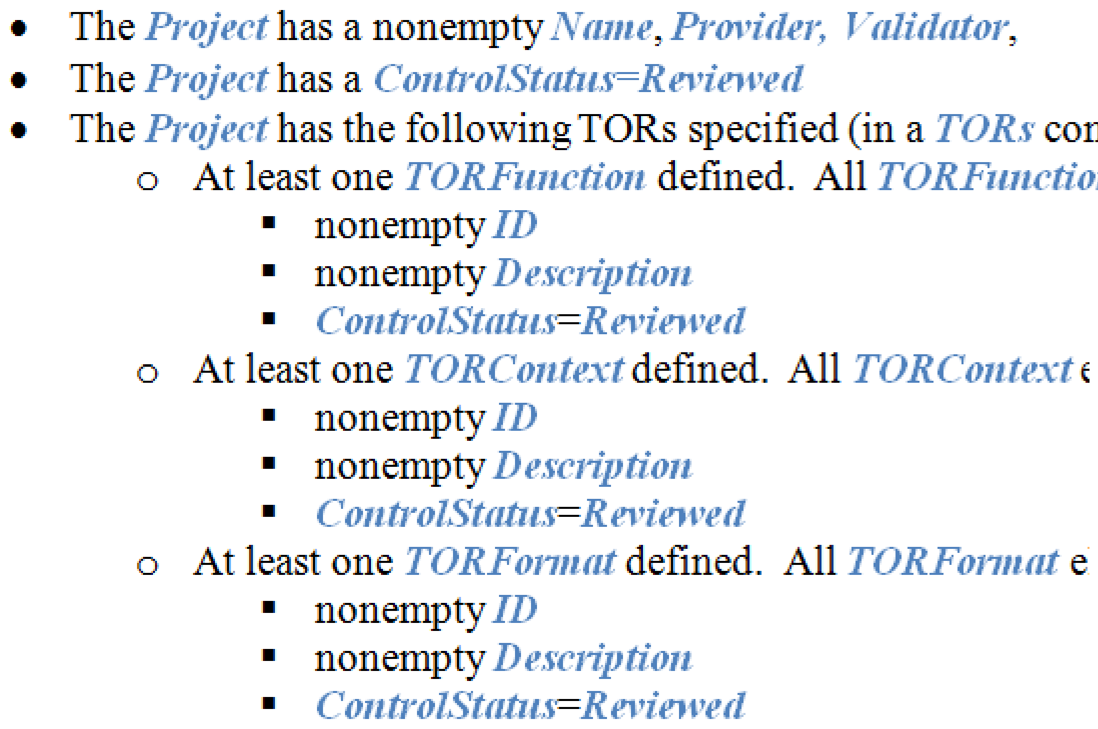
- **Combines the following processes:**
  - Planning (TORs)
  - Development (TR, LLRs)
  - Integration (Verification)
  - Configuration Management
  - Quality Assurance
- **Fits to existing processes (Project process, Release Process) by extending them with a "Qualification Stage"**
- **The following stages are defined (and can be determined automatically from the DO-330 model) such that every release has a well-defined qualification stage**



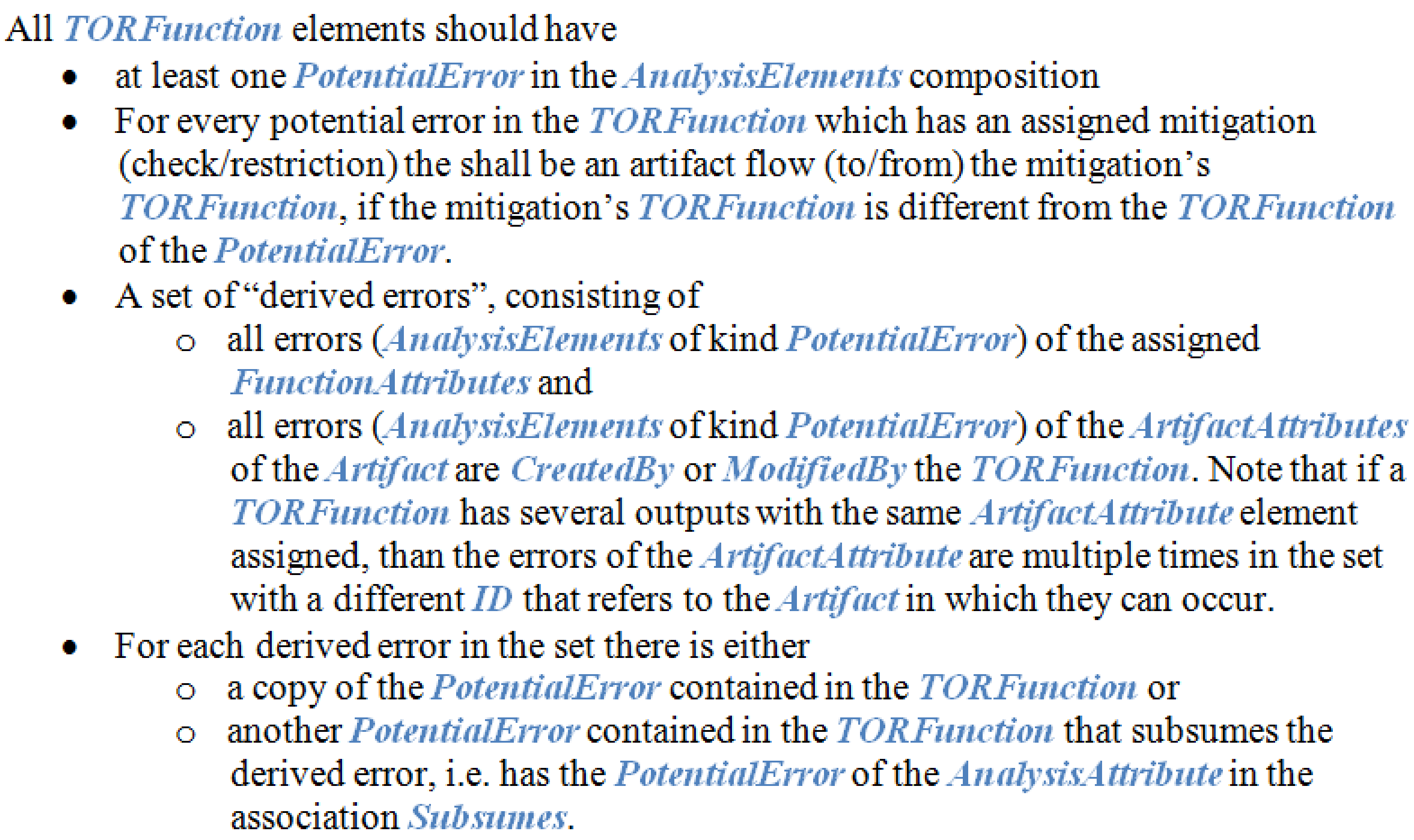
  - **Unqualified-Pre-Alpha Release** ("**Undefined**"): unknown qualification state
  - **Qualification Alpha-Release** ("**Analyzed**"): The TORs are defined and TQL is determined
  - **Qualification Beta-Release** ("**Feature-Complete**"): All requirements (TORs and TRs) are described and have traces 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**

Tool 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* e
    - nonempty *ID*
    - nonempty *Description*
    - *ControlStatus*=*Reviewed*
  - At least one *TORFormat* defined. All *TORFormat* e
    - nonempty *ID*
    - nonempty *Description*
    - *ControlStatus*=*Reviewed*



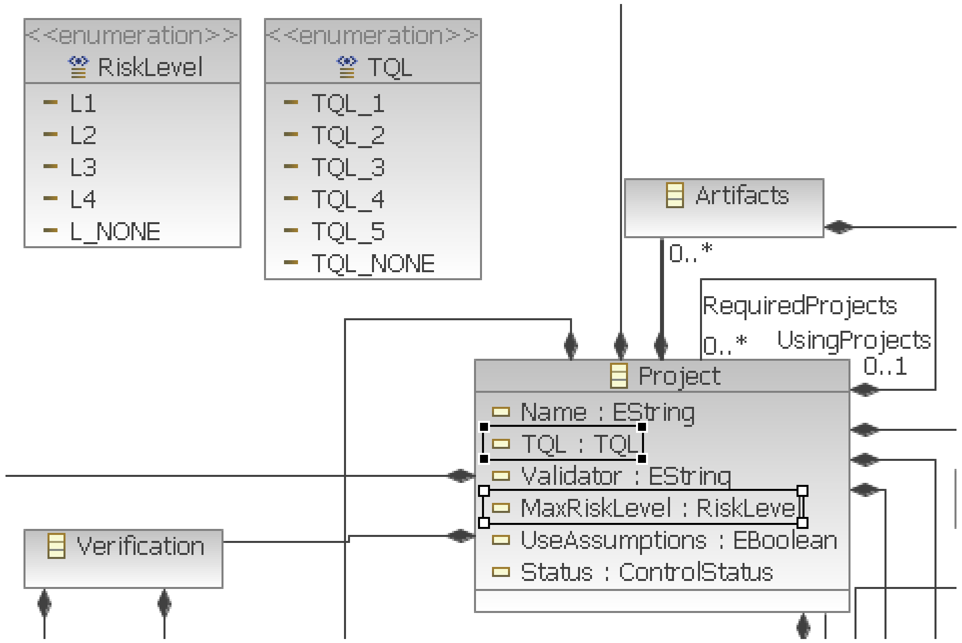
All *TORFunction* elements should have

- 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*.

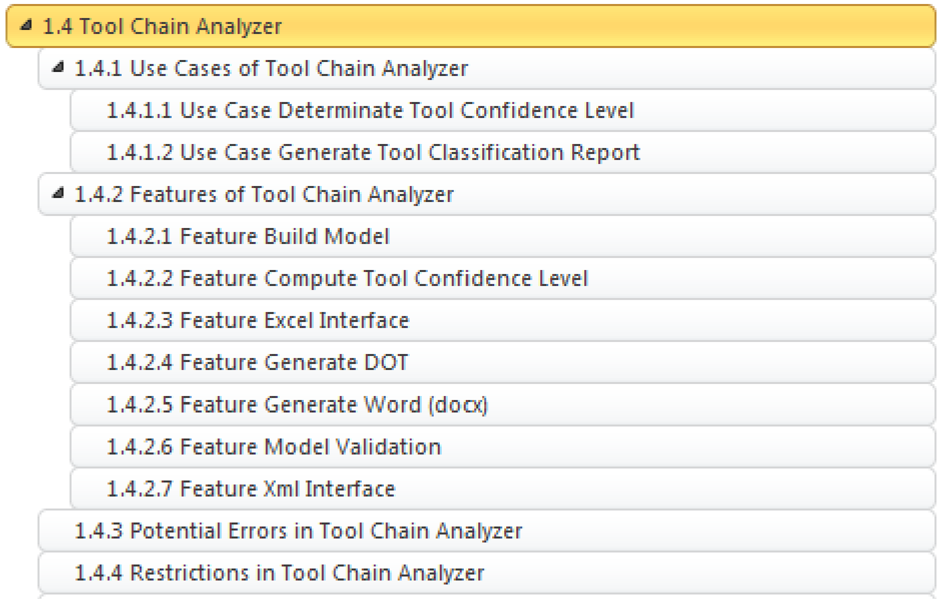
Tool Analysis in PSAC

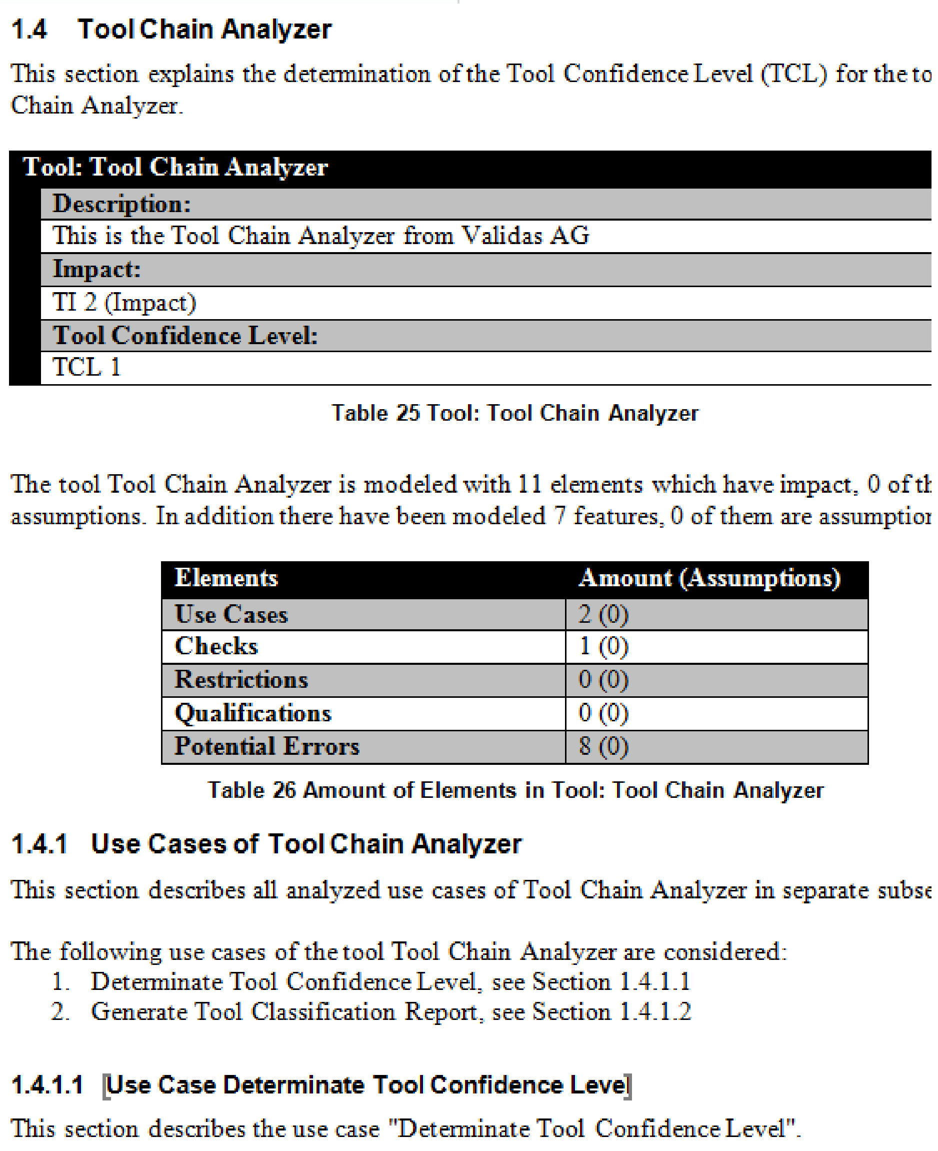
- **From the Qualification Alpha Release (“Analyzed”) of the plugin/tool**
- **Verify the TQL (from qualification needs and Projects max. “RiskLevel”)**
  - L1: Highest Level (ASIL D, Risk Class A,..)
  - L2, L3
  - L4: Lowest Level (ASIL A, Risk Class D,..)
  - L_NONE for uncritical plugins
- **TQL 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

This section explains the determination of the Tool Confidence Level (TCL) for the to 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 th assumptions. In addition there have been modeled 7 features, 0 of them are assumptior

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

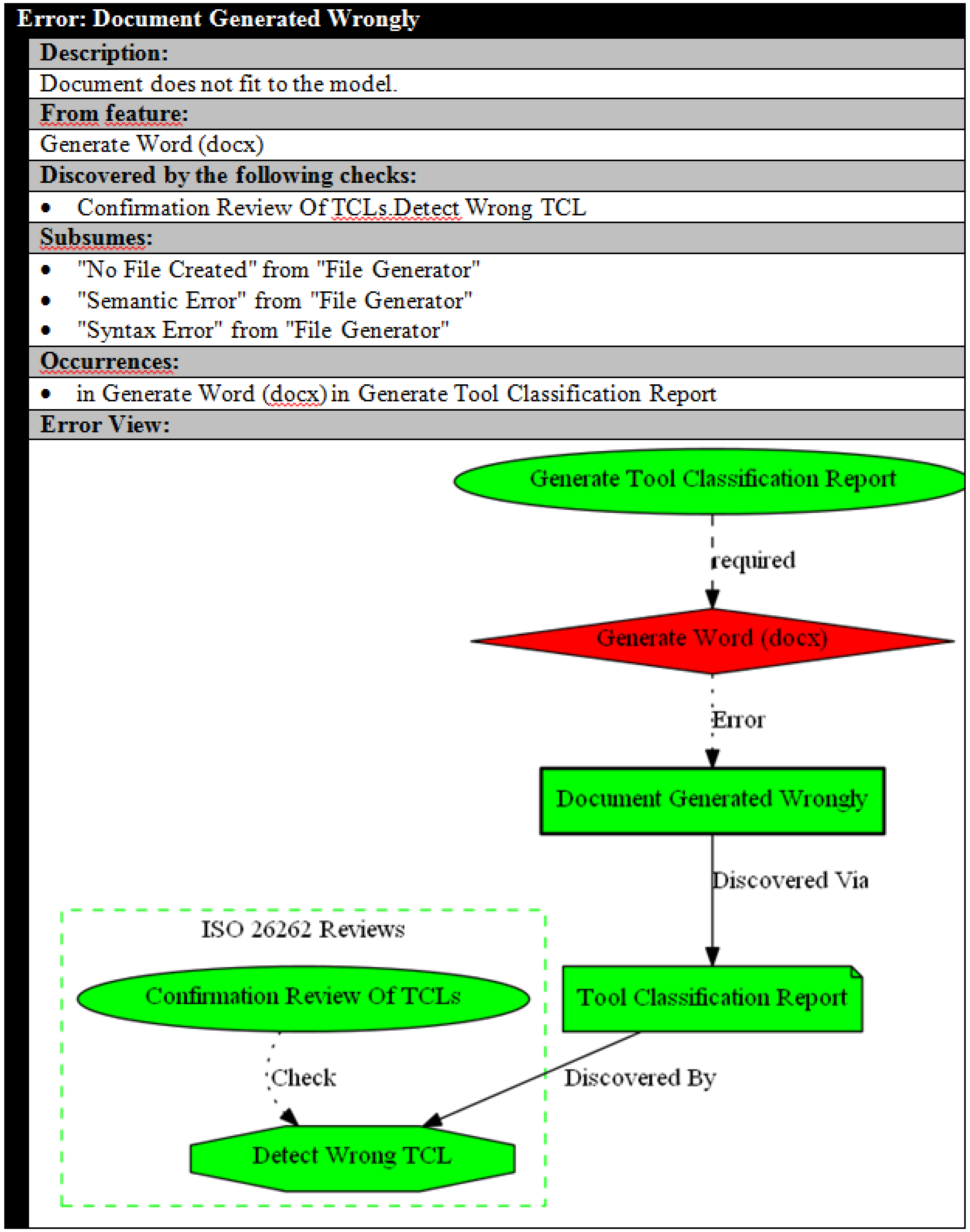
This section describes all analyzed use cases of Tool Chain Analyzer in separate subse

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

1. Determinate 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 Determinate Tool Confidence Leve]

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



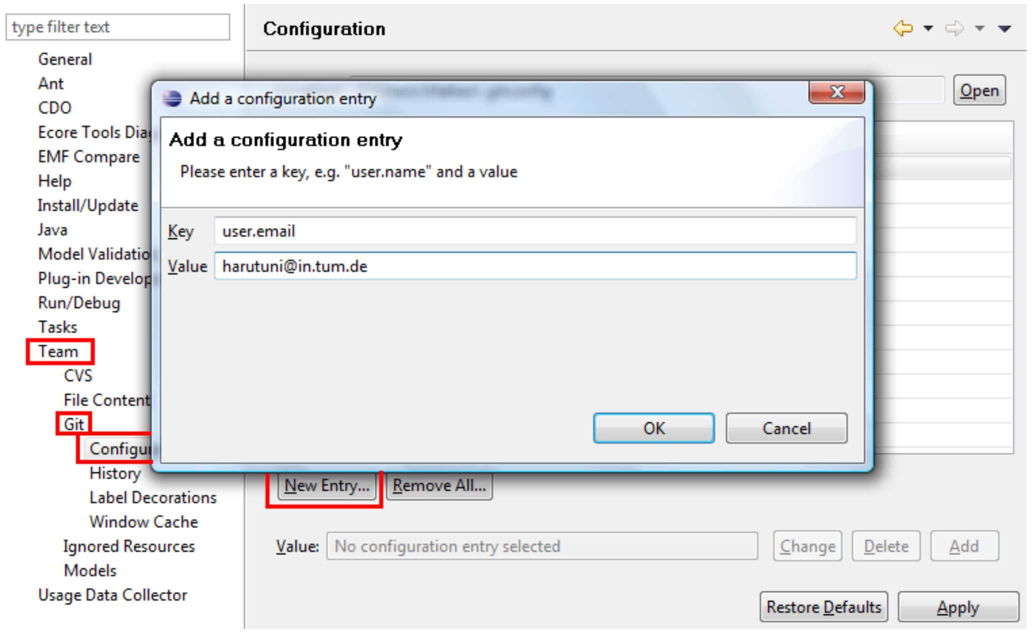
Error: Document Generated Wrongly
Description:
Document does not fit to the model.
From feature:
Generate Word (docx)
Discovered by the following checks:
• Confirmation Review Of TCLs.Detect Wrong TCL
Subsumes:
• "No File Created" from "File Generator" • "Semantic Error" from "File Generator" • "Syntax Error" from "File Generator"
Occurrences:
• in Generate Word (docx) in Generate Tool Classification Report
Error View:

Generate Tool Classification Report
required
Generate Word (docx)
Error
Document Generated Wrongly
Discovered Via
ISO 26262 Reviews
Confirmation Review Of TCLs
Tool Classification Report
Check
Discovered By
Detect Wrong TCL

[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



Add a configuration entry

Add a configuration entry

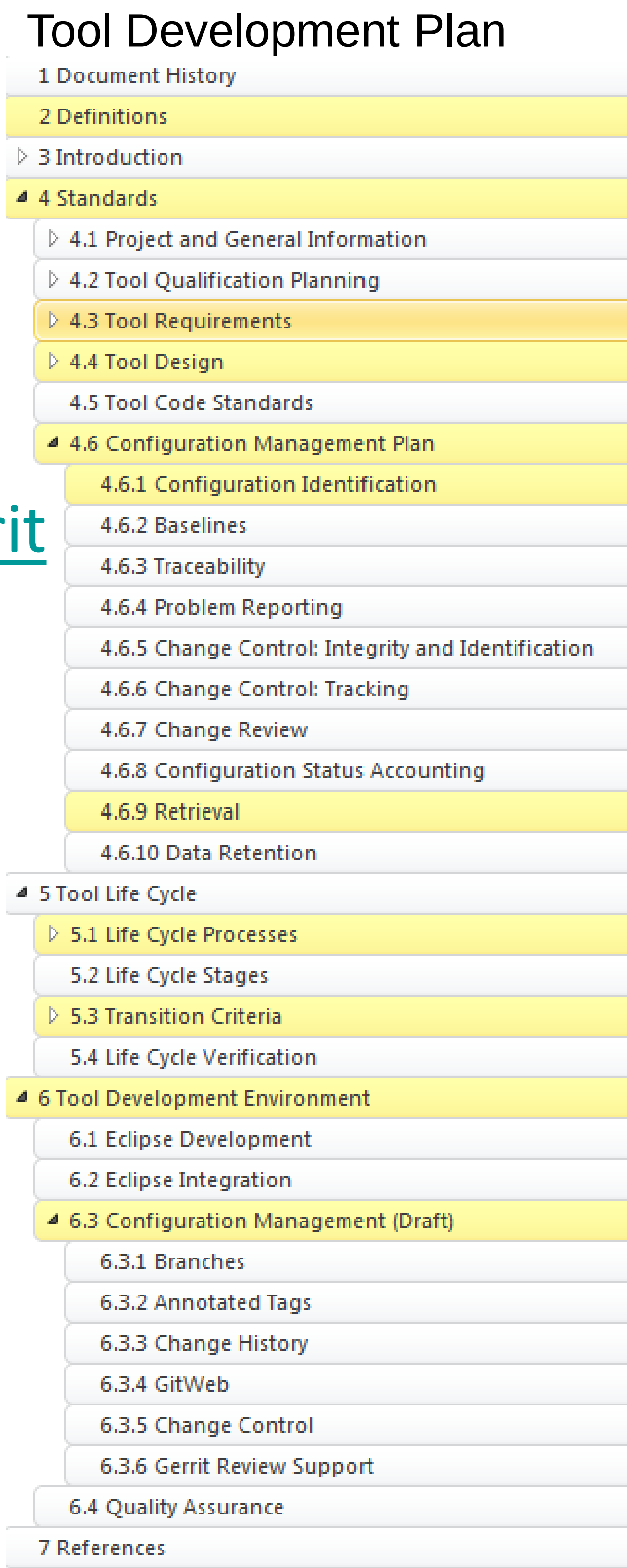
Please enter a key, e.g. "user.name" and a value

Key: user.email

Value: harutuni@in.tum.de

Value: No configuration entry selected

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

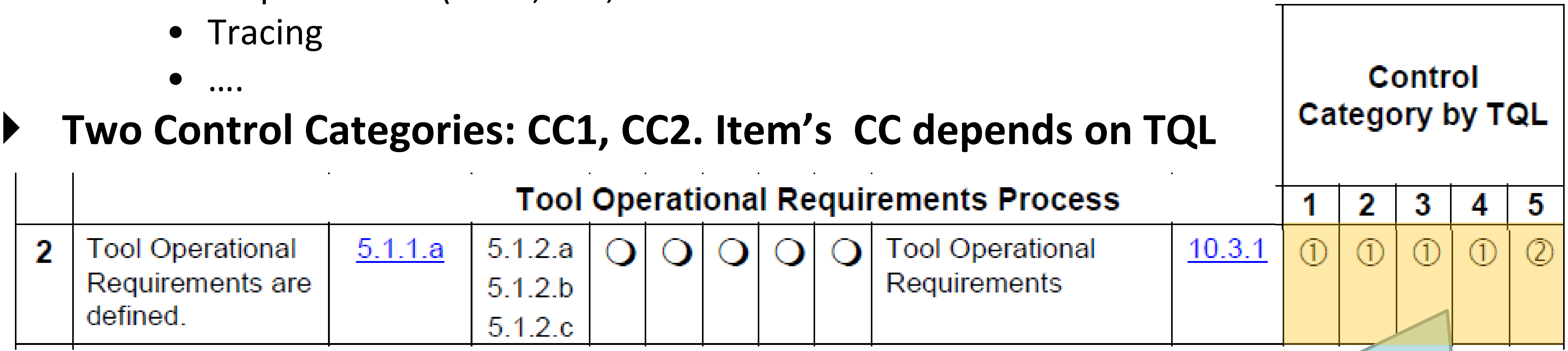


Tool Development Plan

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

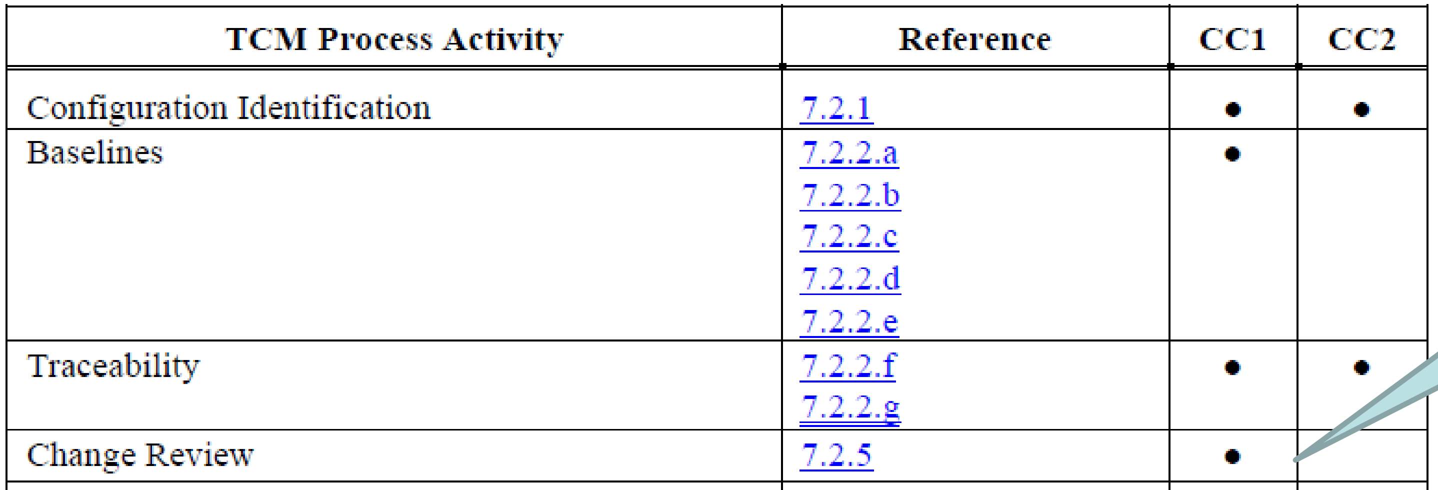
- **Configuration Items are all elements within the Qualifiable Eclipse Project**
  - Sources
  - Architecture
  - DO-330-model
    - Requirements (TORs, TRs,
    - Tracing
    - ….
- **Two Control Categories: CC1, CC2. Item's CC depends on TQL**



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

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

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

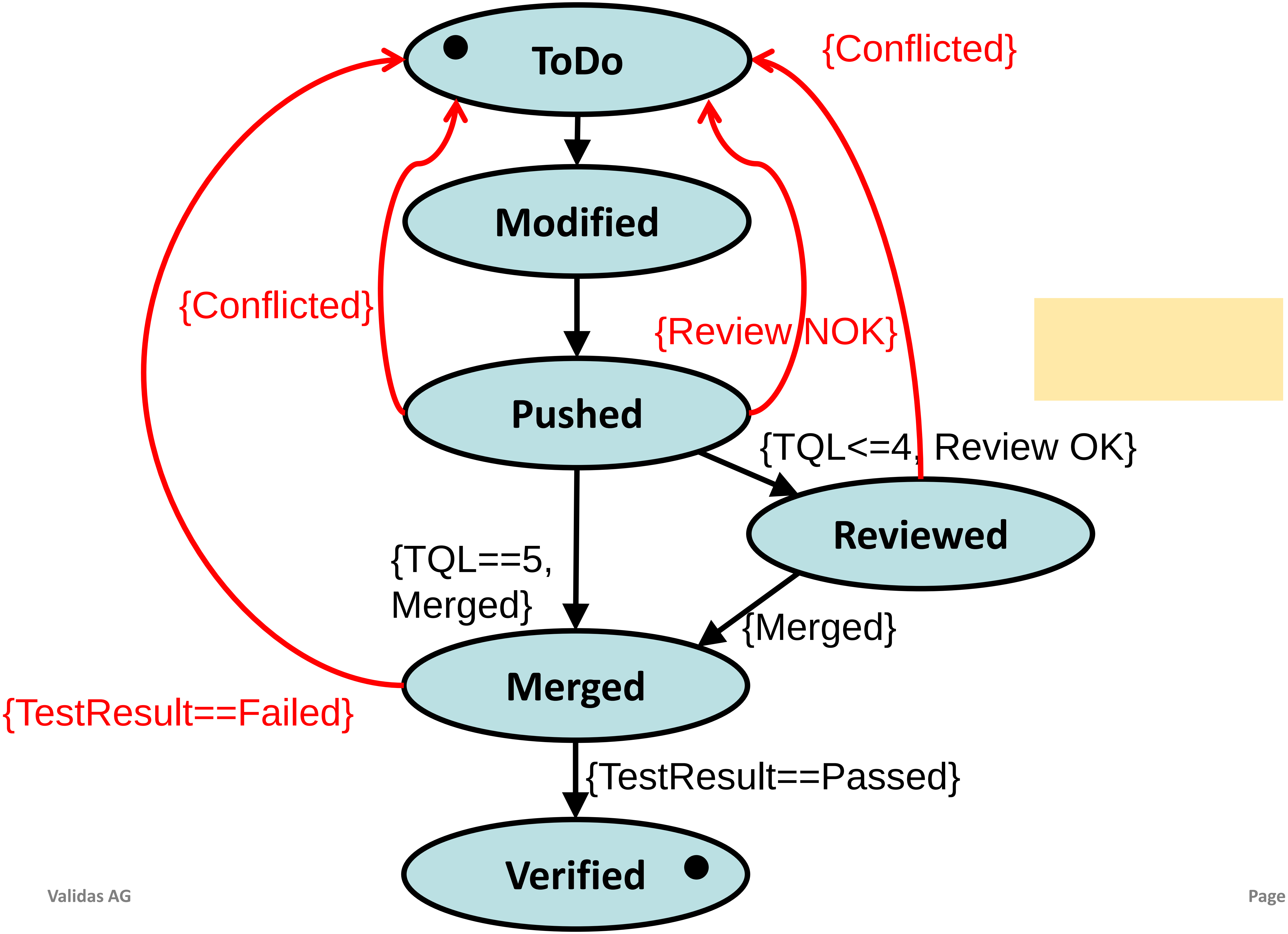


TCM Process Activity	Reference	CC1	CC2
Configuration Identification	7.2.1	●	●
Baselines	7.2.2.a 7.2.2.b 7.2.2.c 7.2.2.d 7.2.2.e	●	
Traceability	7.2.2.f 7.2.2.g	●	●
Change Review	7.2.5	●	

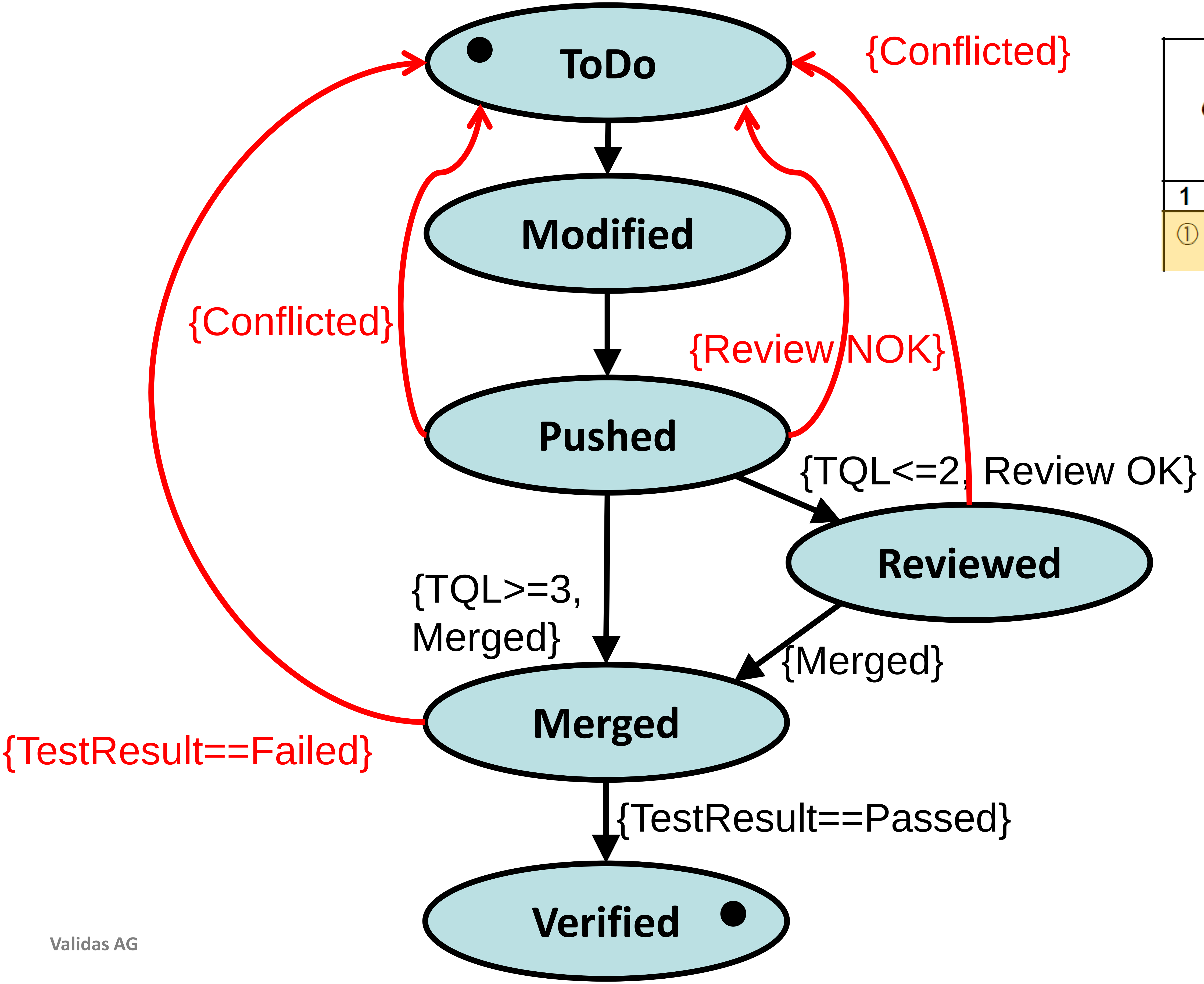
Example: TORs **changes** have to be **reviewed** for TQL-1 to TQL-4 but not for TQL-5

Plugin Extension has to know this (Transition Criteria!)

CM: Control Status of TORs (Proposed)



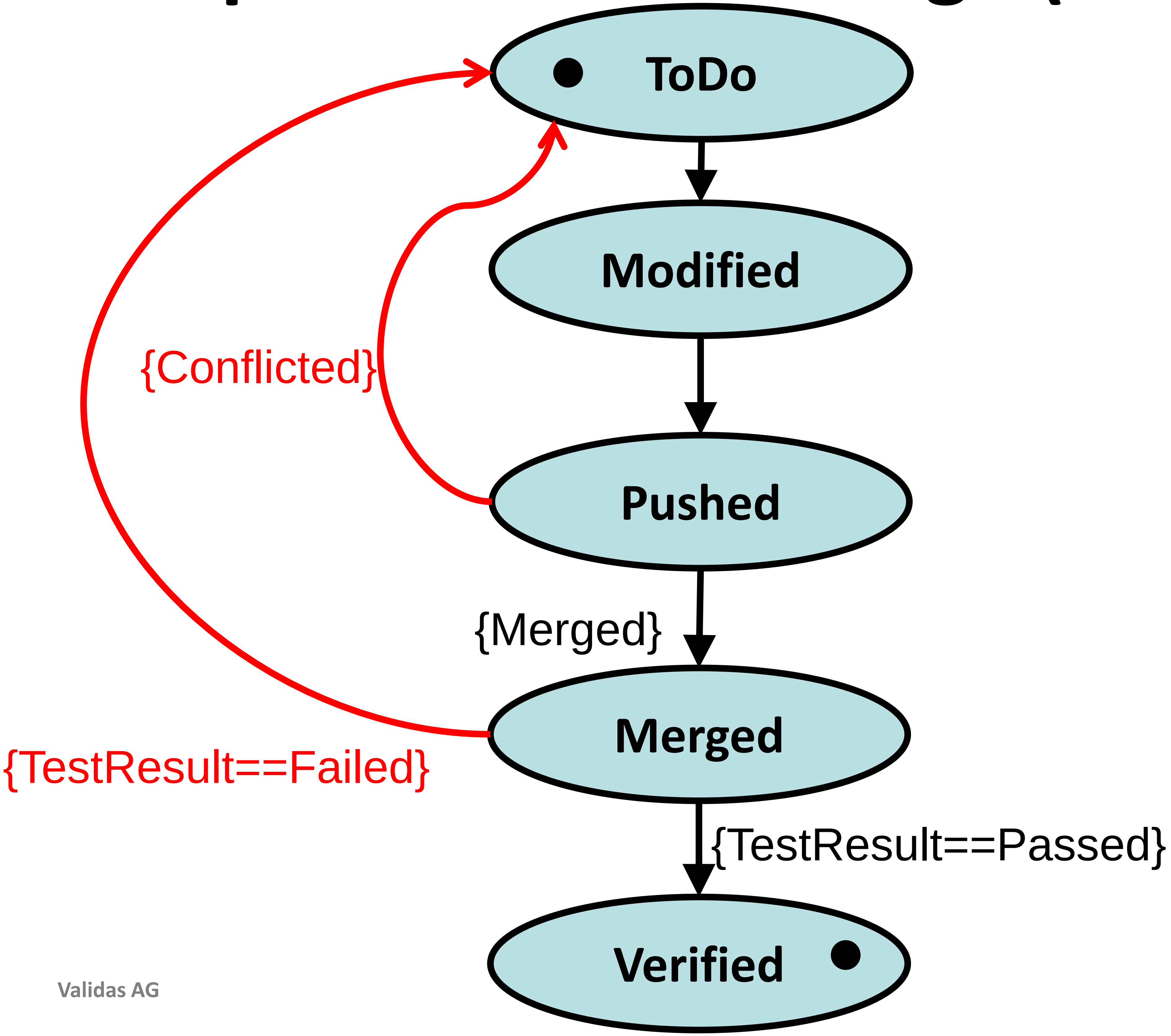
CM: Control Status of Tests (Proposed)



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

TQL-5: Status is Private (unchecked)

CM: Control Status of TRExpectedErrorMessage (Proposed)



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

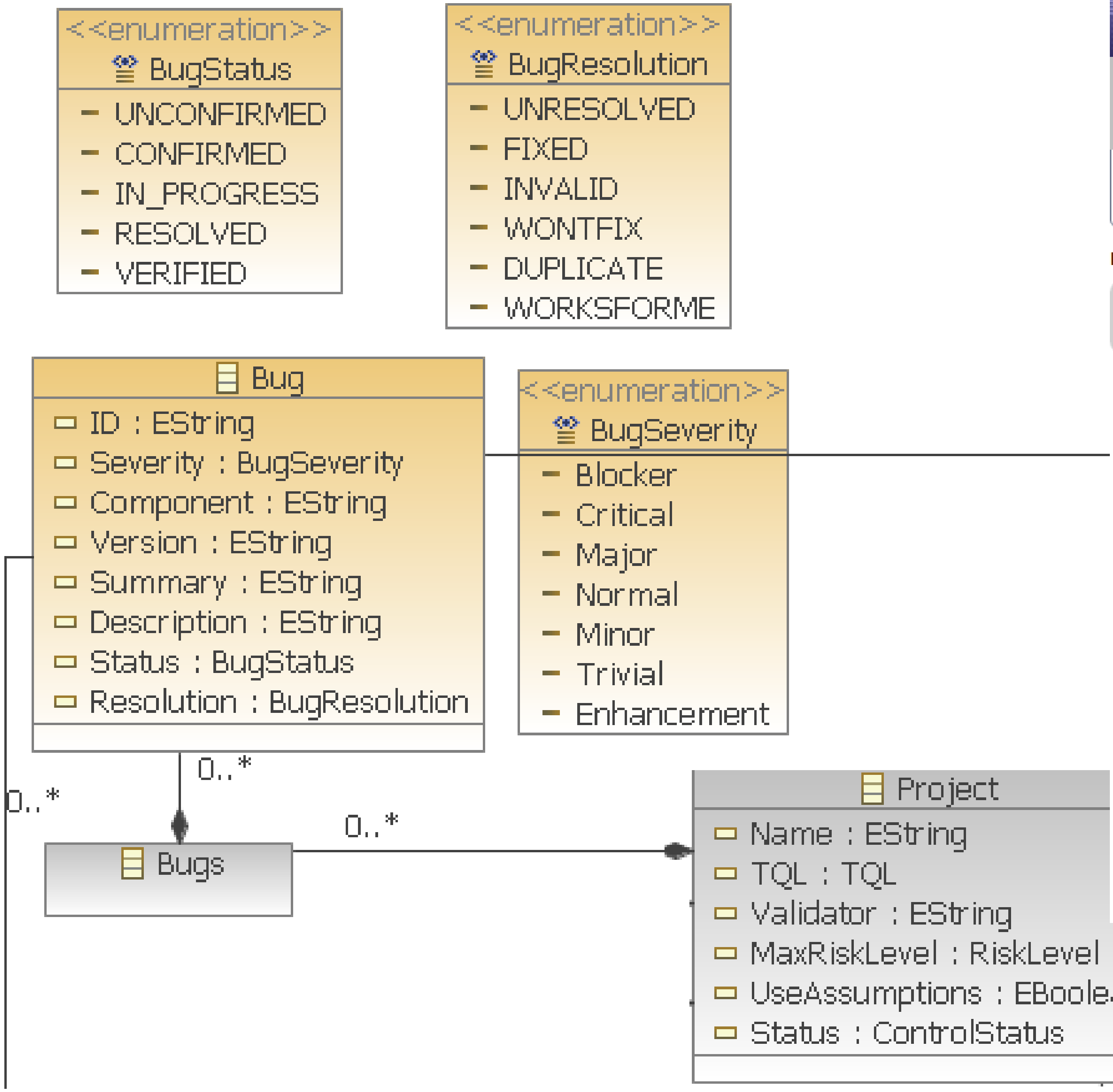
TQL-5:
Status is
Private
(unchecked)

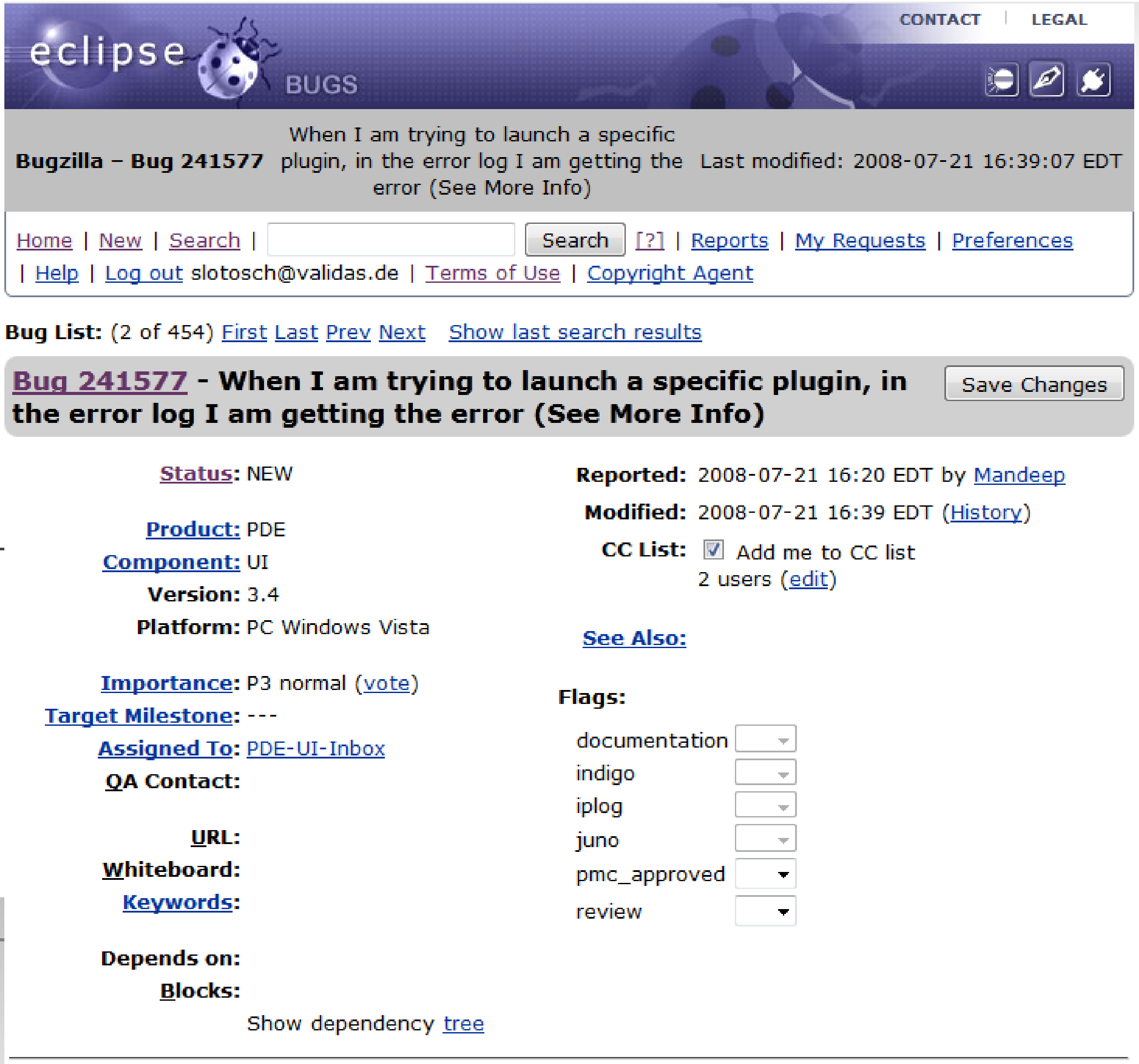
Content

- 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**
  - Sixth Steps: Verification, Multi-Function, Coupling, Independency, COTS
- 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)





Bugzilla – Bug 241577 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

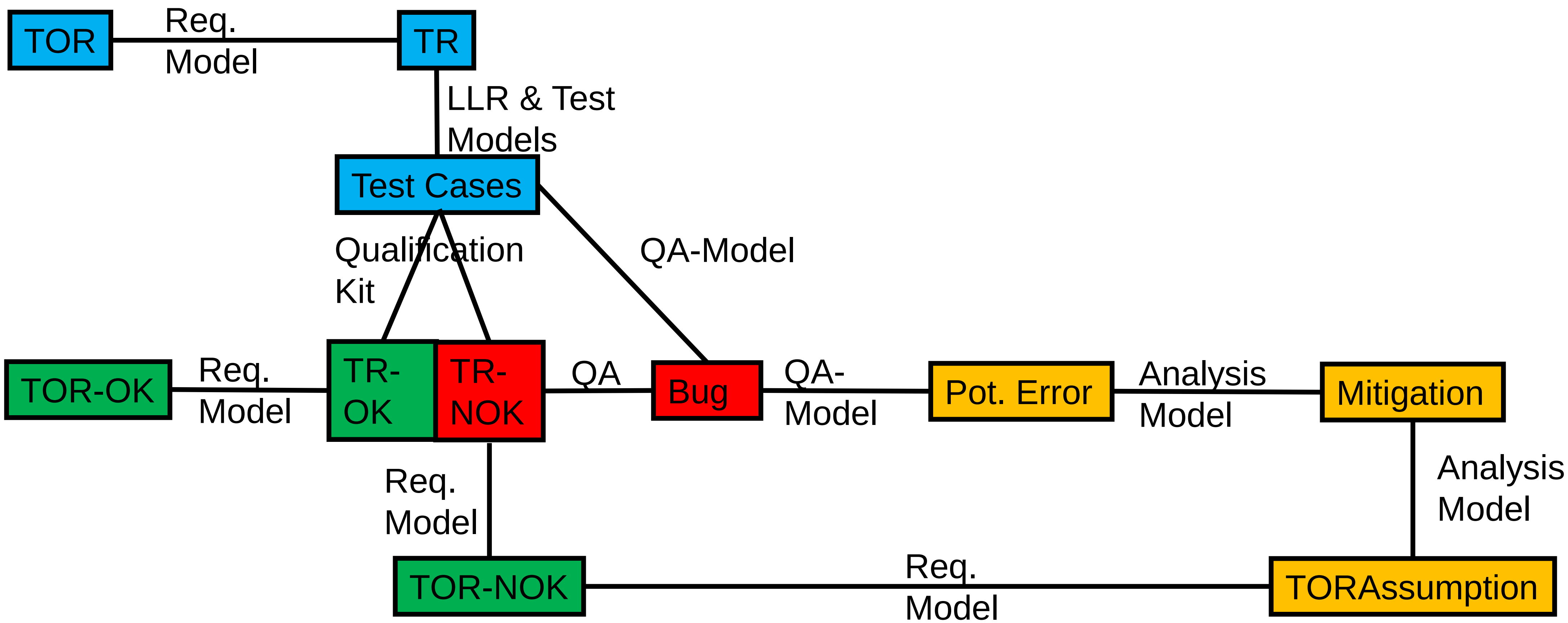
Tool Quality Report

- **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 is 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

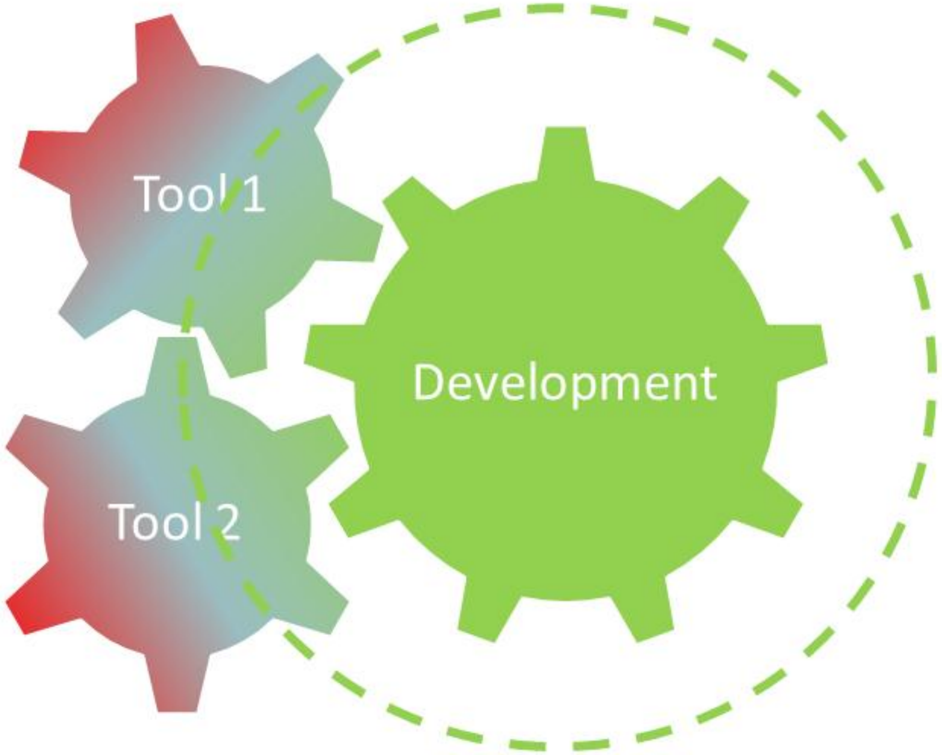
- For all tools with qualification need
- Demonstrate that the tools conform to their requirements (“TOR”), even if qualification shows errors



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

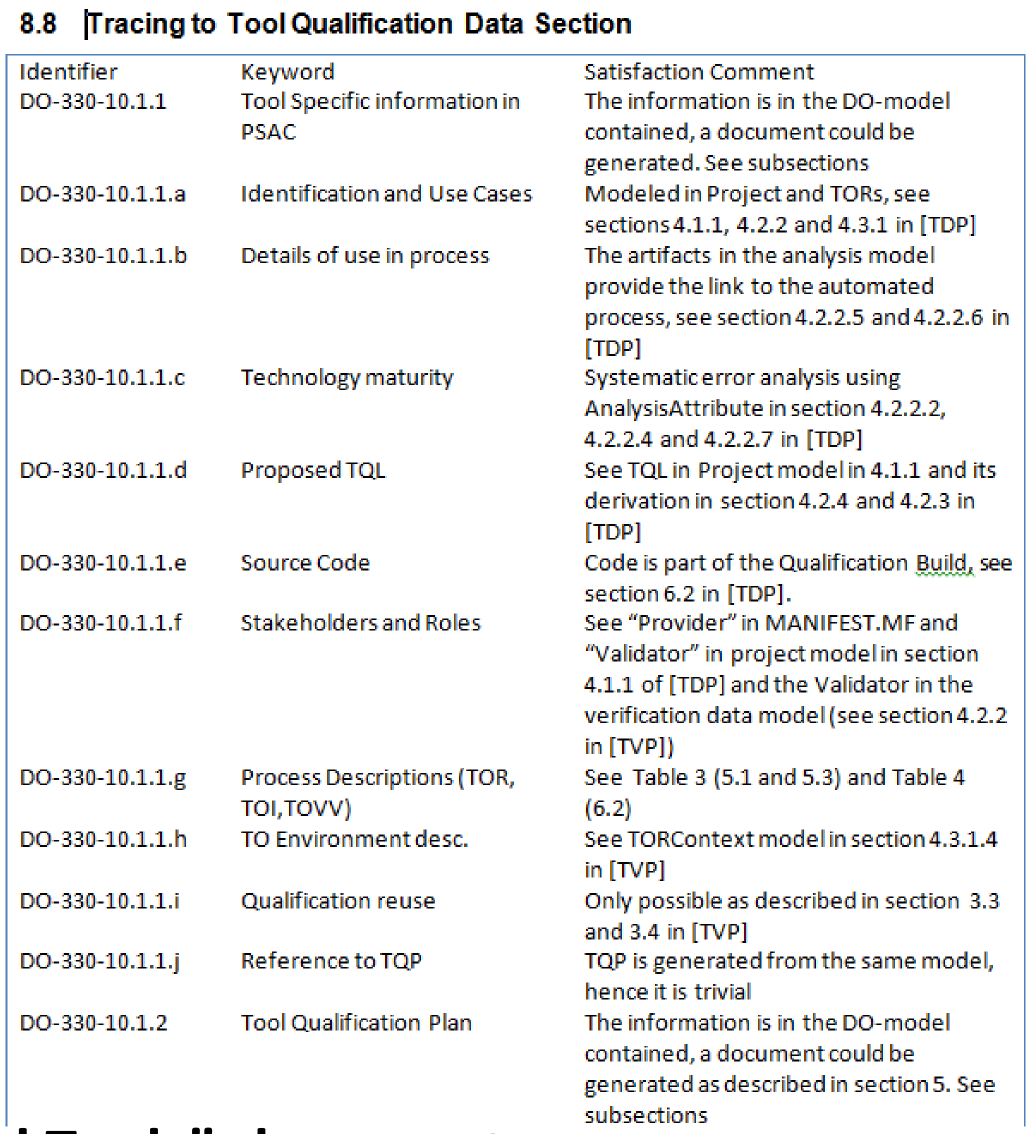
Development Tool Chain



TORs of Tool 1

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

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
  - **Sixth Steps: Verification, Multi-Function, Coupling, Independency, COTS**
- Summary

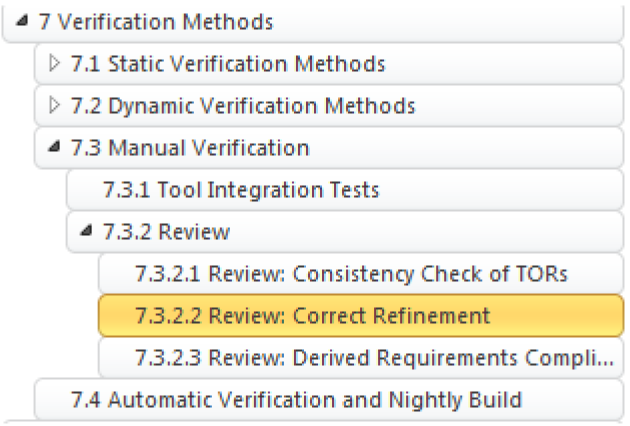
Verification

- **Is documented in the generic document: “Tool Verification Plan”**
- **Includes Tracing to DO-330 (“HowTo-Qualify” document)**
- **For Example DO-330 states in section 6.1.3.1.1.a:**
  Compliance with Tool Operational Requirements: The objective is to ensure that all the Tool Operational Requirements are implemented into the tool requirements and that the derived tool requirements , and the reason for their existence, are correctly defined.

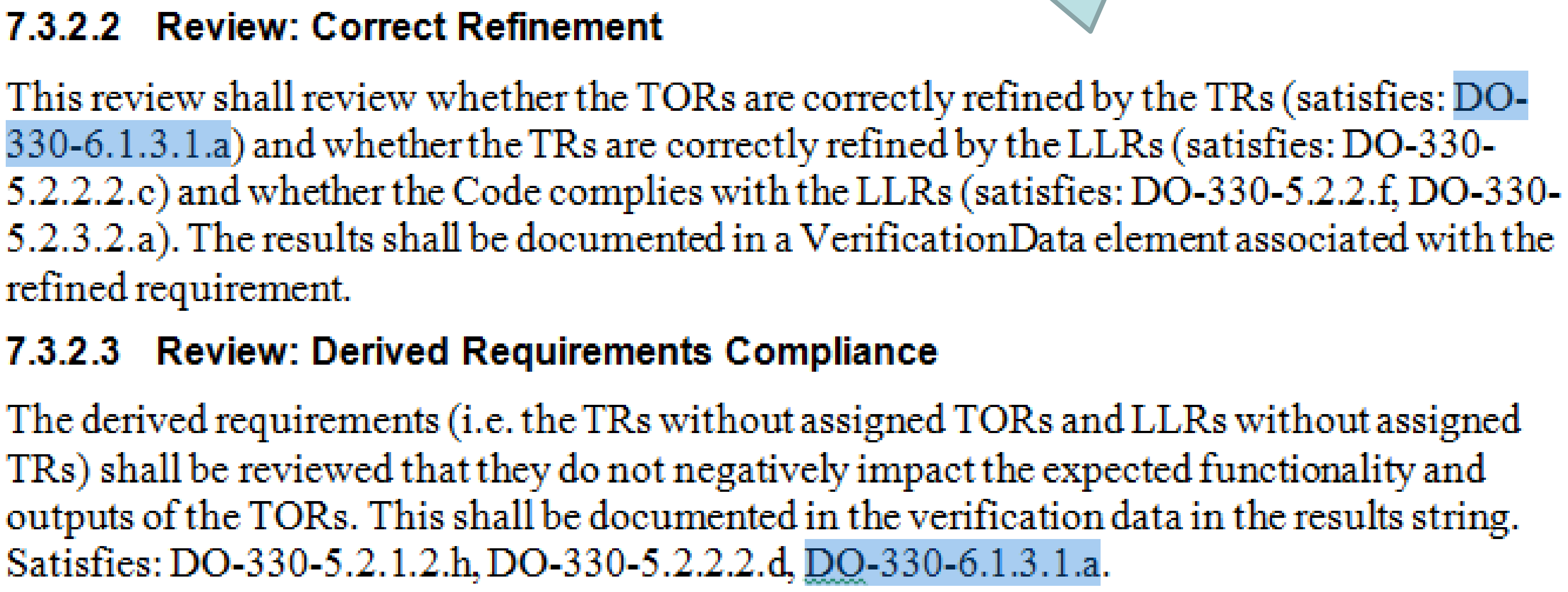
Know-Why: Tracing to DO-330

Efficiency: one review satisfies several DO-330 requirements

Tool Verification Plan



- 7 Verification Methods
  - 7.1 Static Verification Methods
  - 7.2 Dynamic Verification Methods
  - 7.3 Manual Verification
    - 7.3.1 Tool Integration Tests
    - 7.3.2 Review
      - 7.3.2.1 Review: Consistency Check of TORs
      - 7.3.2.2 Review: Correct Refinement
      - 7.3.2.3 Review: Derived Requirements Compli...
  - 7.4 Automatic Verification and Nightly Build



7.3.2.2 Review: Correct Refinement

This review shall review whether the TORs are correctly refined by the TRs (satisfies: DO-330-6.1.3.1.a) and whether the TRs are correctly refined by the LLRs (satisfies: DO-330-5.2.2.2.c) and whether the Code complies with the LLRs (satisfies: DO-330-5.2.2.f, DO-330-5.2.3.2.a). The results shall be documented in a VerificationData element associated with the refined requirement.

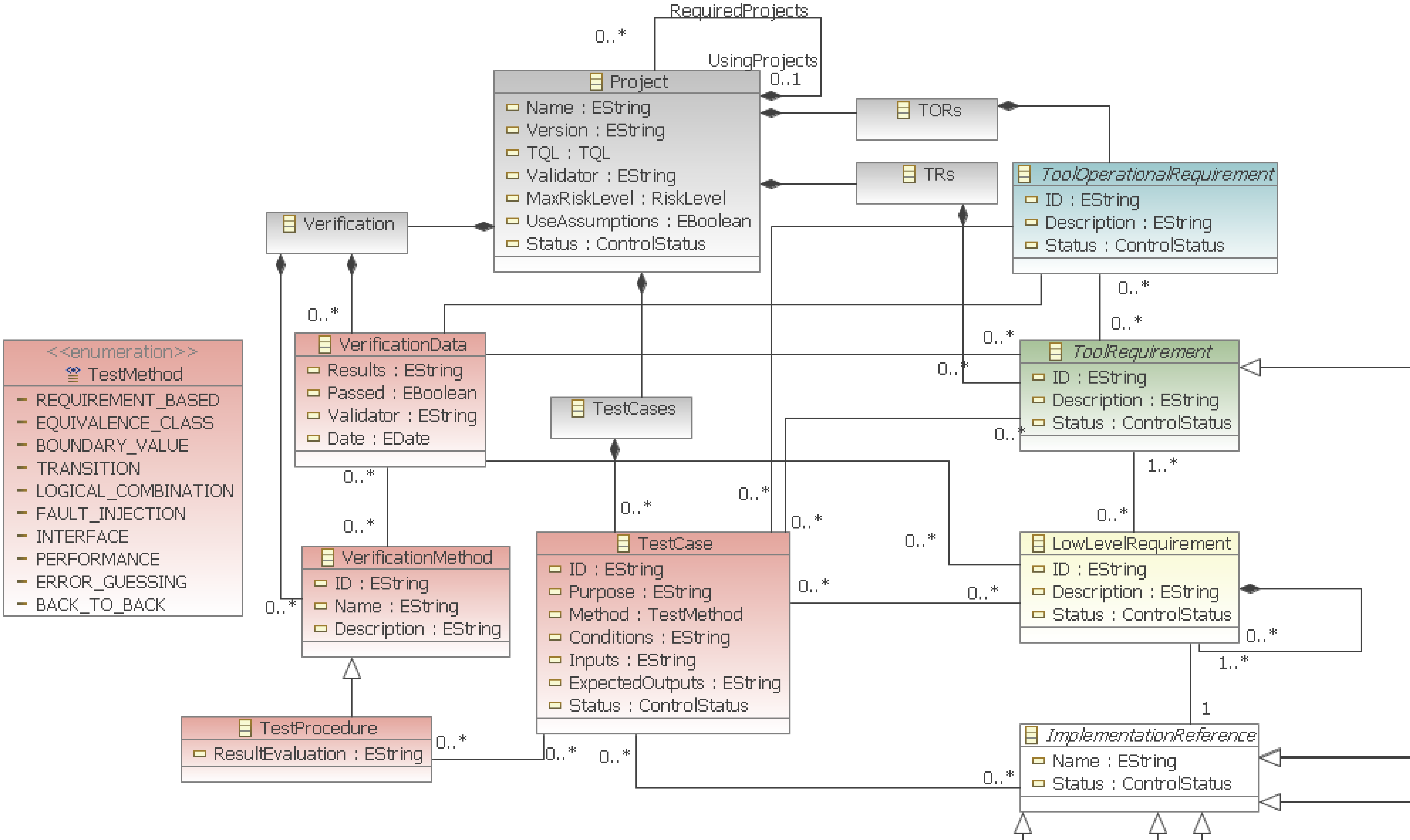
7.3.2.3 Review: Derived Requirements Compliance

The derived requirements (i.e. the TRs without assigned TORs and LLRs without assigned TRs) shall be reviewed that they do not negatively impact the expected functionality and outputs of the TORs. This shall be documented in the verification data in the results string. Satisfies: DO-330-5.2.1.2.h, DO-330-5.2.2.2.d, DO-330-6.1.3.1.a.

- Support?

Test & Verification Model

- **Relates test to requirements (TOR, TR, LLR) & implementation**



Project
- Name : EString
- Version : EString
- TQL : TQL
- Validator : EString
- MaxRiskLevel : RiskLevel
- UseAssumptions : EBoolean
- Status : ControlStatus

TORs

TRs

ToolOperationalRequirement
- ID : EString
- Description : EString
- Status : ControlStatus

Verification

<<enumeration>>
TestMethod
- REQUIREMENT_BASED
- EQUIVALENCE_CLASS
- BOUNDARY_VALUE
- TRANSITION
- LOGICAL_COMBINATION
- FAULT_INJECTION
- INTERFACE
- PERFORMANCE
- ERROR_GUESSING
- BACK_TO_BACK

VerificationData
- Results : EString
- Passed : EBoolean
- Validator : EString
- Date : EDate

TestCases

ToolRequirement
- ID : EString
- Description : EString
- Status : ControlStatus

VerificationMethod
- ID : EString
- Name : EString
- Description : EString

TestCase
- ID : EString
- Purpose : EString
- Method : TestMethod
- Conditions : EString
- Inputs : EString
- ExpectedOutputs : EString
- Status : ControlStatus

LowLevelRequirement
- ID : EString
- Description : EString
- Status : ControlStatus

TestProcedure
- ResultEvaluation : EString

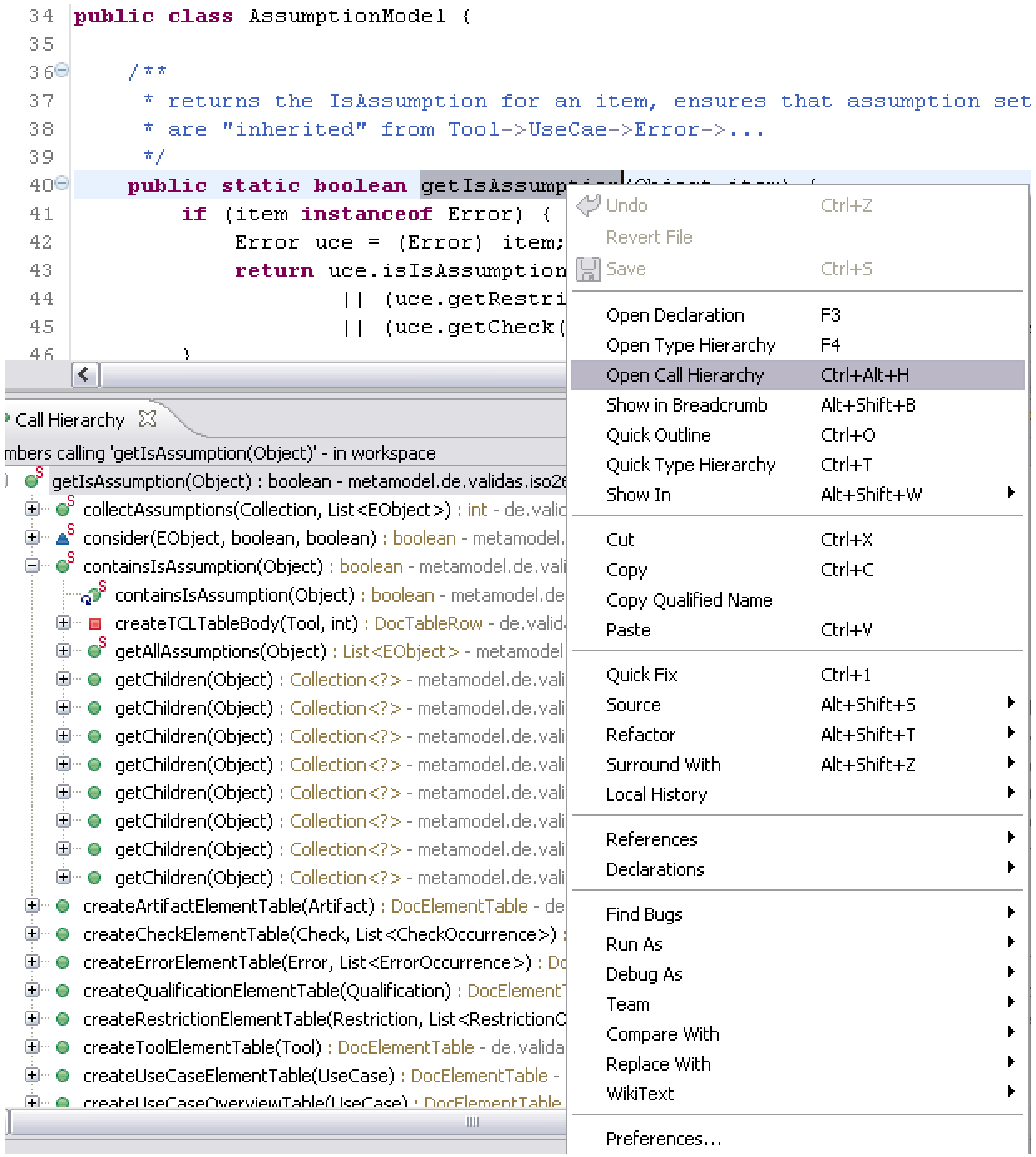
ImplementationReference
- Name : EString
- Status : ControlStatus

Verification Support?

- **The DO-330 model supports verification**
  - Validation: “Every TOR has at least one TR”
  - Validation: “Every TOR has at least on Verification Data”
  - Derived TRs are identified
  - Every Derived TR has a VerificationData
- **Criteria are evaluated from Eclipse within the “tool life cycle transition criteria” that determined the “qualification stage” of a plugin (see slide before)**
- **Gerrit supports reviews (see slide before)**
- **Flexible process:**
  - Do not enforce the waterfall: Requirements -> Design -> Code -> …
  - Allow “Implementability check” to be deferred after implementation

Data & Control Coupling Problem

- **DO-330-6.1.4.3.2.c: "The analysis should confirm that the requirements-based testing has exercised the data coupling and control coupling between the tool code components"**
- **Eclipse supports coupling by "Call Hierarchy" for Variables (Data) and Methods (Control) and "Type Hierarchy" for classes (Data)**
- **It should be shown (integration test) that**
  - Every method is called correctly from parent place in the tree
  - Every variable is read/written correctly (from every parent)
- **Code coverage measurement does not support this**



```
public class AssumptionModel {

    /**
     * returns the IsAssumption for an item, ensures that assumption set
     * are "inherited" from Tool->UseCae->Error->...
     */
    public static boolean getIsAssump
        if (item instanceof Error) {
            Error uce = (Error) item;
            return uce.isIsAssumption
                    || (uce.getRestri
                    || (uce.getCheck(
        }
```

```
Call Hierarchy
mbers calling 'getIsAssumption(Object)' - in workspace
getIsAssumption(Object) : boolean - metamodel.de.validas.iso2
  collectAssumptions(Collection, List<EObject>) : int - de.valic
  consider(EObject, boolean, boolean) : boolean - metamodel.
  containsIsAssumption(Object) : boolean - metamodel.de.vali
    containsIsAssumption(Object) : boolean - metamodel.de
    createTCLTableBody(Tool, int) : DocTableRow - de.valid
    getAllAssumptions(Object) : List<EObject> - metamodel
    getChildren(Object) : Collection<?> - metamodel.de.vali
    getChildren(Object) : Collection<?> - metamodel.de.vali
    getChildren(Object) : Collection<?> - metamodel.de.vali
    getChildren(Object) : Collection<?> - metamodel.de.vali
    getChildren(Object) : Collection<?> - metamodel.de.vali
    getChildren(Object) : Collection<?> - metamodel.de.vali
    getChildren(Object) : Collection<?> - metamodel.de.vali
    getChildren(Object) : Collection<?> - metamodel.de.vali
  createArtifactElementTable(Artifact) : DocElementTable - de
  createCheckElementTable(Check, List<CheckOccurrence>) :
  createErrorElementTable(Error, List<ErrorOccurrence>) : Do
  createQualificationElementTable(Qualification) : DocElement
  createRestrictionElementTable(Restriction, List<RestrictionC
  createToolElementTable(Tool) : DocElementTable - de.valida
  createUseCaseElementTable(UseCase) : DocElementTable -
  createUseCaseOverviewTable(UseCase) : DocElementTable
```

Data & Control Coupling Solutions

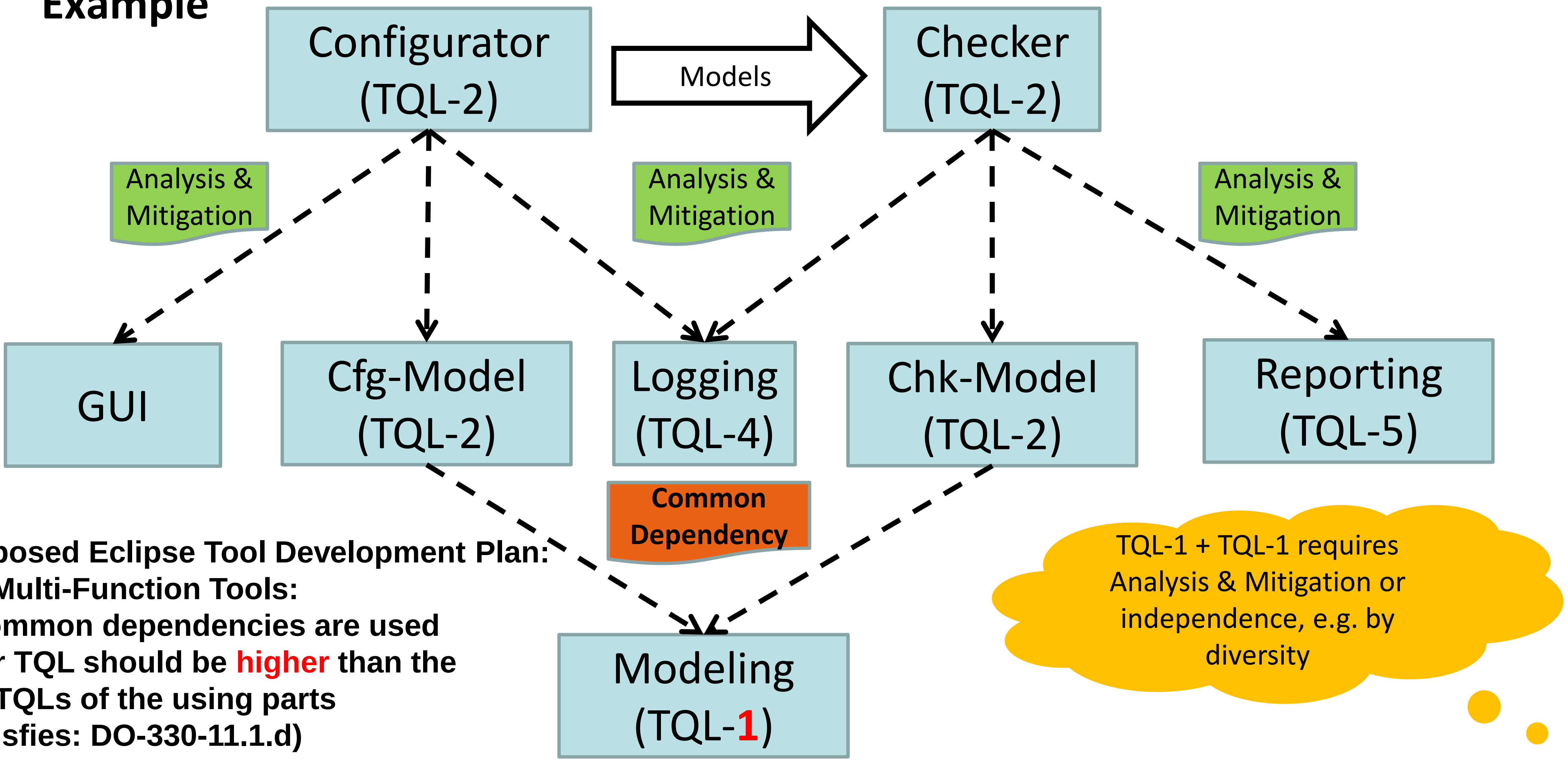
- **Requirements based integration testing including robustness requirements (TOR, TR) with an analysis of data & control coupling**
  - Find a tool that measures coupling or
  - Do it manually:
    - Determine callers for every method / variable
    - run tests for every method document access
    - Analyze & complete uncovered accesses

  Both would lead to a minimization of coupling

- **Since Coupling in Eclipse is implemented from the Eclipse classloader (and parts of the Java runtime environment) it suffices to qualify those elements / parts. OSGI influences the mechanism**

Independence Challenge in MF Tools

- **11.1.d (Additional Considerations for Multi-Function Tools) states: if multi-function tools both produce and verify the same output**
  - Protection shall be used (plugins) AND
  - (for TQL-1 and TQL-2) **Independence**
- **Example**



Configurator (TQL-2)

Models

Checker (TQL-2)

Analysis & Mitigation

Analysis & Mitigation

Analysis & Mitigation

GUI

Cfg-Model (TQL-2)

Logging (TQL-4)

Chk-Model (TQL-2)

Reporting (TQL-5)

Common Dependency

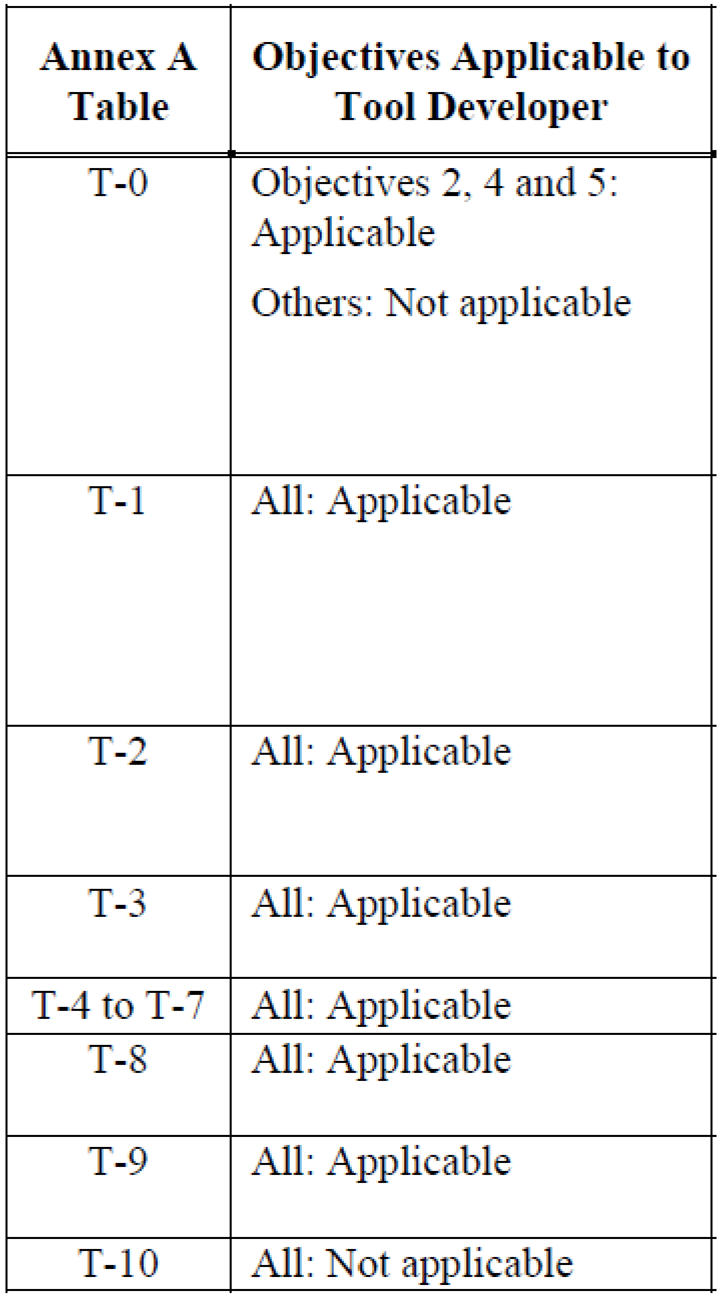
Modeling (TQL-1)

TQL-1 + TQL-1 requires Analysis & Mitigation or independence, e.g. by diversity

Proposed Eclipse Tool Development Plan:
3.2 Multi-Function Tools:
If common dependencies are used their TQL should be higher than the TQLs of the using parts (satisfies: DO-330-11.1.d)

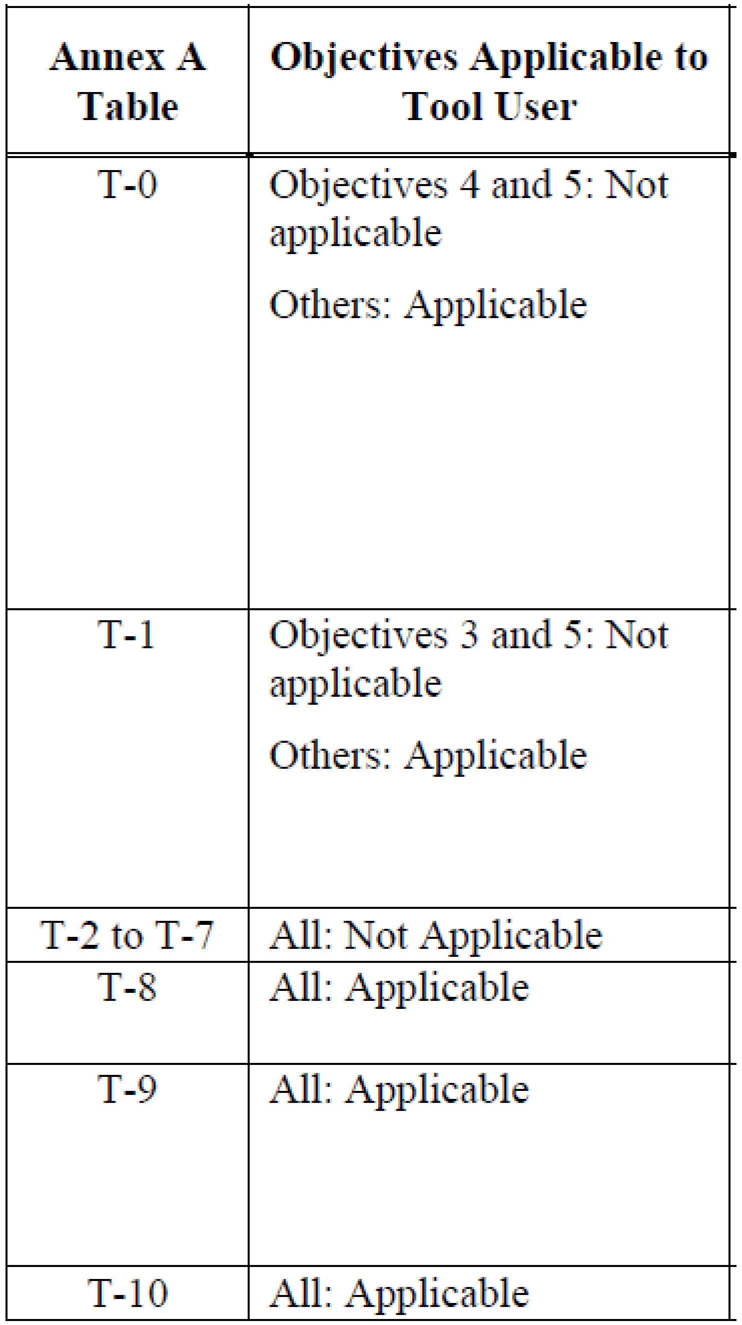
Qualifying COTS Tools

Developer Tasks
(DO-330-11-1)



Annex A Table	Objectives Applicable to Tool Developer
T-0	Objectives 2, 4 and 5: Applicable Others: Not applicable
T-1	All: Applicable
T-2	All: Applicable
T-3	All: Applicable
T-4 to T-7	All: Applicable
T-8	All: Applicable
T-9	All: Applicable
T-10	All: Not applicable

User Tasks
(DO-330-11-2)



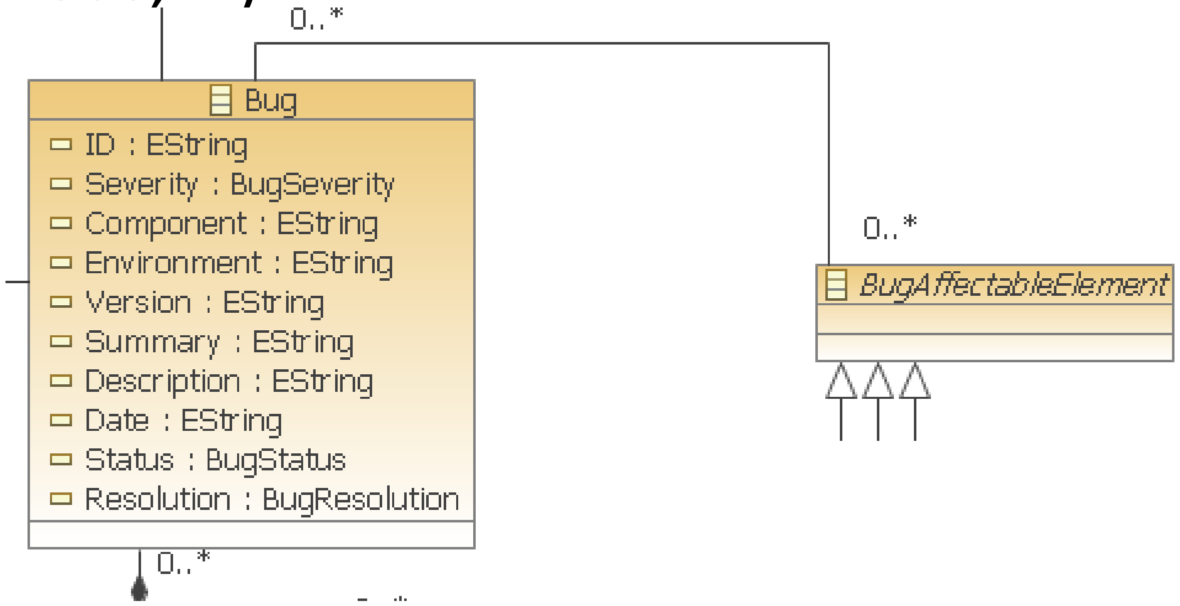
Annex A Table	Objectives Applicable to Tool User
T-0	Objectives 4 and 5: Not applicable Others: Applicable
T-1	Objectives 3 and 5: Not applicable Others: Applicable
T-2 to T-7	All: Not Applicable
T-8	All: Applicable
T-9	All: Applicable
T-10	All: Applicable

Nothing is missing

- **No differences found to the qualifiable plugin approach, where every plugin is treaded like a COTS tool**

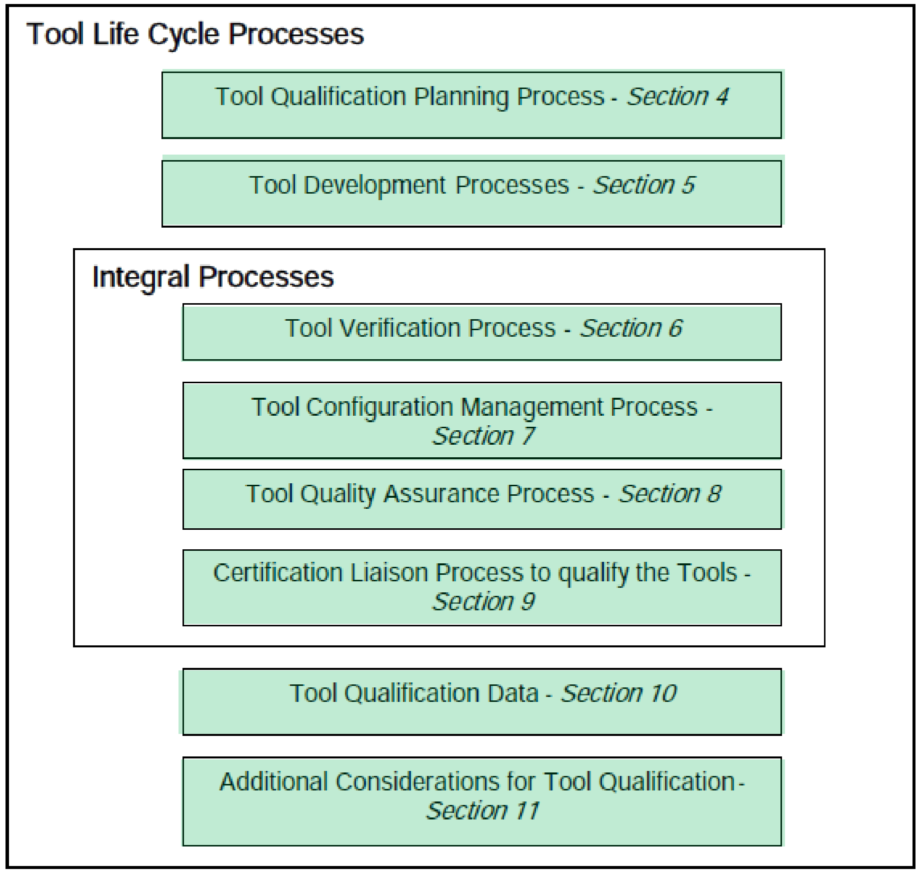
Computation of Maturity

- **Maturity of tool and algorithm is required by: DO-330-10.1.1.c, DO-11.4.1.e**
- **Maturity is determined by the effects of changes of bugs**
  - Immature tools have big changes
  - Mature tools have small changes
- **The effect of a bug is modeled buy it’s affected elements:**
  - Implementation References (Class, Methods,….)
  - LLRs
  - TRs (including architecture)
- **Note that if the architecture changes, many implementation references change**
- **Modeled using BugAffectedElements**
  - TRs (including architecture)
  - LLRs
  - Implementation References
- **Maturity can be computed automatically from**
  - Number of bugs (in a fixed time)
  - Number of affected elements



Roadmap - Status May 2012

1. Goals: DO-330
2. Concept: Eclipse Project QPP
3. Demonstrate & implement QPP
4. Qualify (selected) plugins



Tool Life Cycle Processes

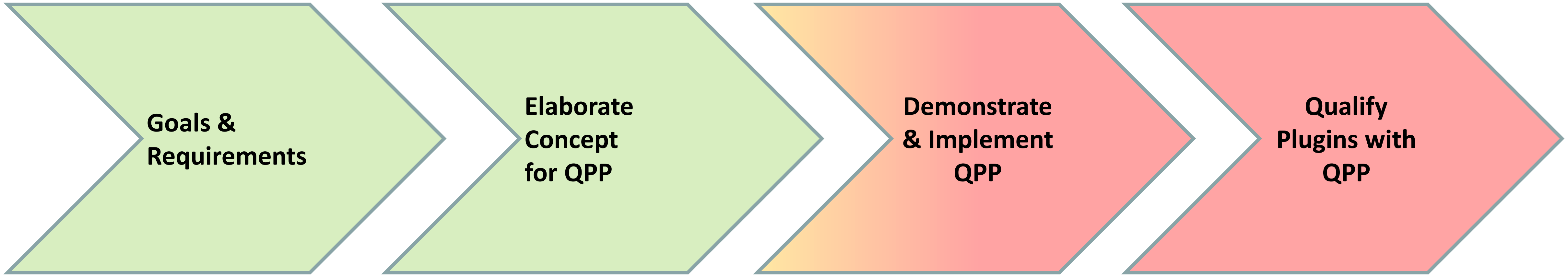
- Tool Qualification Planning Process - *Section 4*
- Tool Development Processes - *Section 5*

Integral Processes

- Tool Verification Process - *Section 6*
- Tool Configuration Management Process - *Section 7*
- Tool Quality Assurance Process - *Section 8*
- Certification Liaison Process to qualify the Tools - *Section 9*

- Tool Qualification Data - *Section 10*
- Additional Considerations for Tool Qualification - *Section 11*

- **Status May 2012**



Goals & Requirements

Elaborate Concept for QPP

Demonstrate & Implement QPP

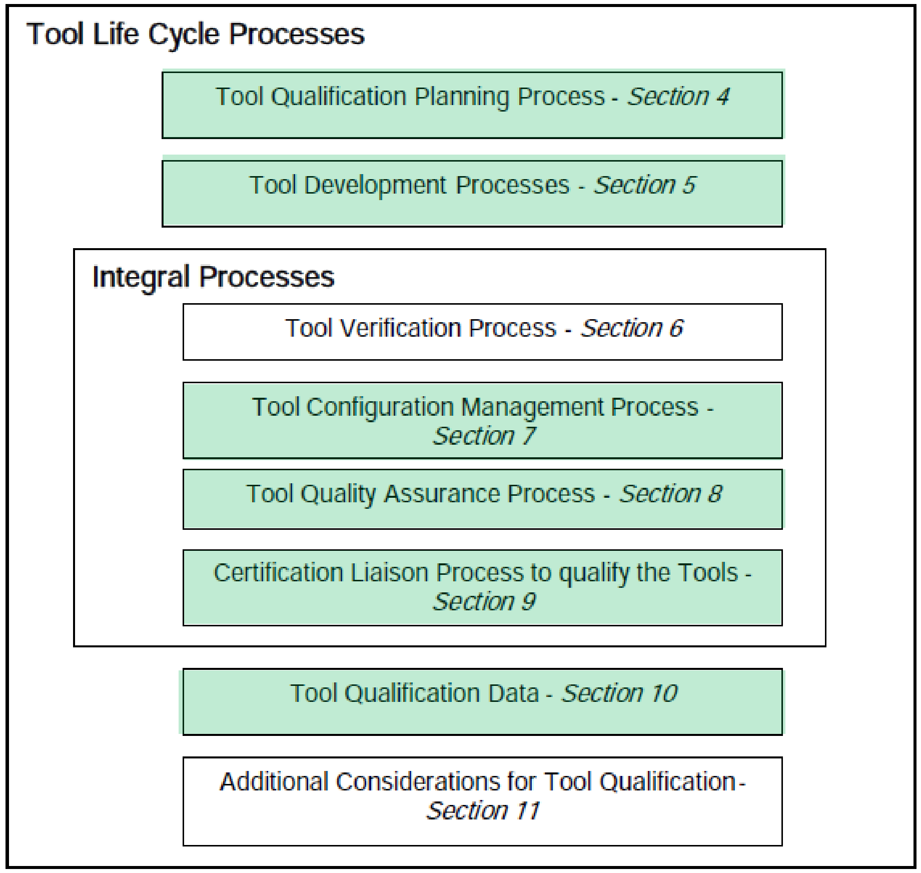
Qualify Plugins with QPP

Goals & Requirements	Elaborate Concept for QPP	Demonstrate & Implement QPP	Qualify Plugins with QPP
Identified	Status: Feasible and 100% of DO-330 covered	**Ready to start**	**Ready to start (Prototyping)**

- **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,..)
- **1st, 2nd, 3rd, 4th, 5th, 6th steps performed**
- **Proposed new role for that work: Eclipse Validator**
- **Validas contributes**


Tool Life Cycle Processes

Tool Qualification Planning Process - *Section 4*

Tool Development Processes - *Section 5*

Integral Processes

Tool Verification Process - *Section 6*

Tool Configuration Management Process - *Section 7*

Tool Quality Assurance Process - *Section 8*

Certification Liaison Process to qualify the Tools - *Section 9*

Tool Qualification Data - *Section 10*

Additional Considerations for Tool Qualification - *Section 11*

Thank You!

VALIDAS

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