



Oscar Slotosch, Validas AG

# **Model-Based Tool Qualification**

## **The Roadmap of Eclipse towards Tool Qualification**

# Content



- ▶ **Introduction: Validas & Eclipse Automotive IWG WP5**
- ▶ The roadmap of Eclipse towards Tool Qualification
- ▶ Model-Based Tool Qualification
  - Overview
  - Model & Automatization
  - Documentation generation
- ▶ Example: Ongoing demonstrator
- ▶ Business model (proposal): Pay per qualification
- ▶ Summary

# Validas AG About Us



- ▶ We are a technology consultancy for quality assurance of embedded systems
- ▶ Our core competences are model based development, model based testing, test automation, **tool qualification**
- ▶ We develop innovative software engineering methods, implement them in form of tools and processes and we support our customers in their application.
- ▶ We are members of AUTOSAR and Eclipse Foundation







# 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,... ?**

# Content

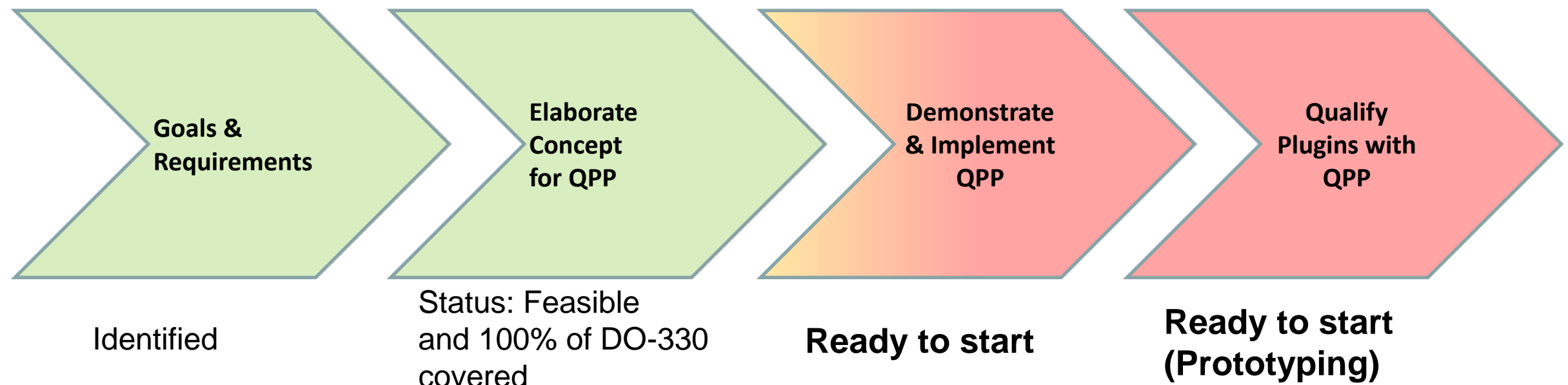


- ▶ Introduction: Validas & Eclipse Automotive IWG WP5
- ▶ **The roadmap of Eclipse towards Tool Qualification**
- ▶ Model-Based Tool Qualification
  - Overview
  - Model & Automatization
  - Documentation generation
- ▶ Example: Ongoing demonstrator
- ▶ Business model (proposal): Pay per qualification
- ▶ Summary

# Roadmap - Status June 2012



1. **Goals: DO-330**
2. **Concept: model-based tool qualification**
3. **Demonstrate & implement with an Eclipse Project: QPP (Qualifiable Plugin Projects)**
4. **Qualify (selected) plugins**



# Content



- ▶ Introduction: Validas & Eclipse Automotive IWG WP5
- ▶ The roadmap of Eclipse towards Tool Qualification
- ▶ **Model-Based Tool Qualification**
  - **Overview**
  - **Model & Automatization**
  - **Documentation generation**
- ▶ Example: Ongoing demonstrator
- ▶ Business model (proposal): Pay per qualification
- ▶ Summary



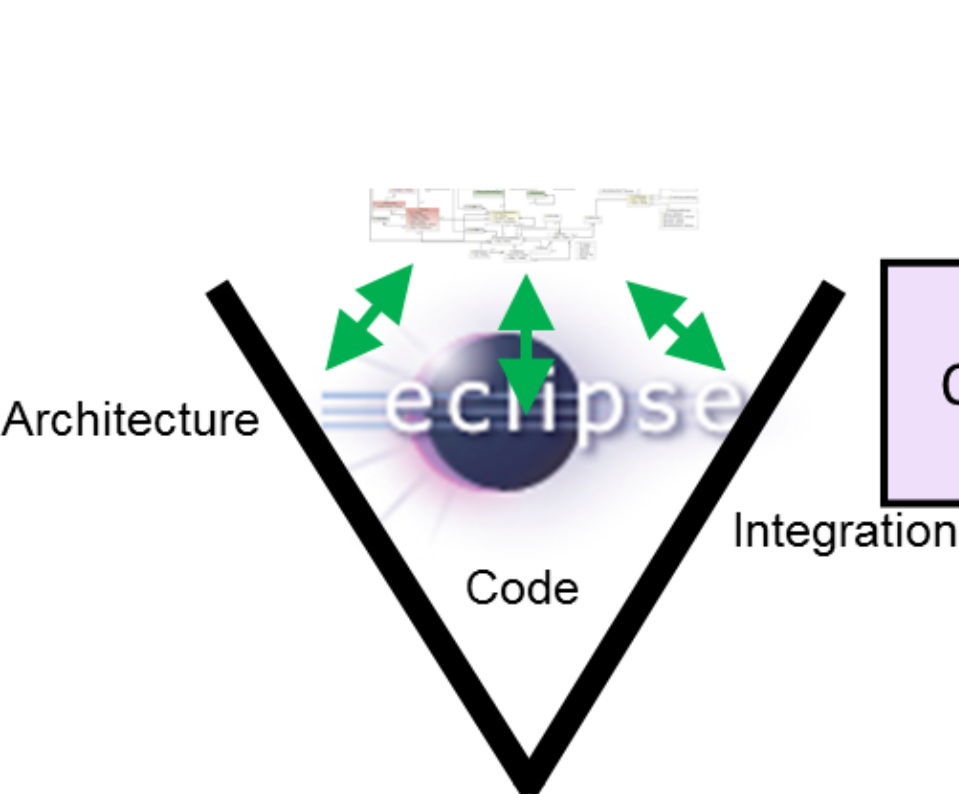
# Development With Eclipse?



- ▶ Currently Eclipse does not support qualification
- ▶ There is a road towards tool qualification for Eclipse, see [http://wiki.eclipse.org/Auto IWG WP5](http://wiki.eclipse.org/Auto_IWG_WP5)
- ▶ DO-330 is a safety standard for tools

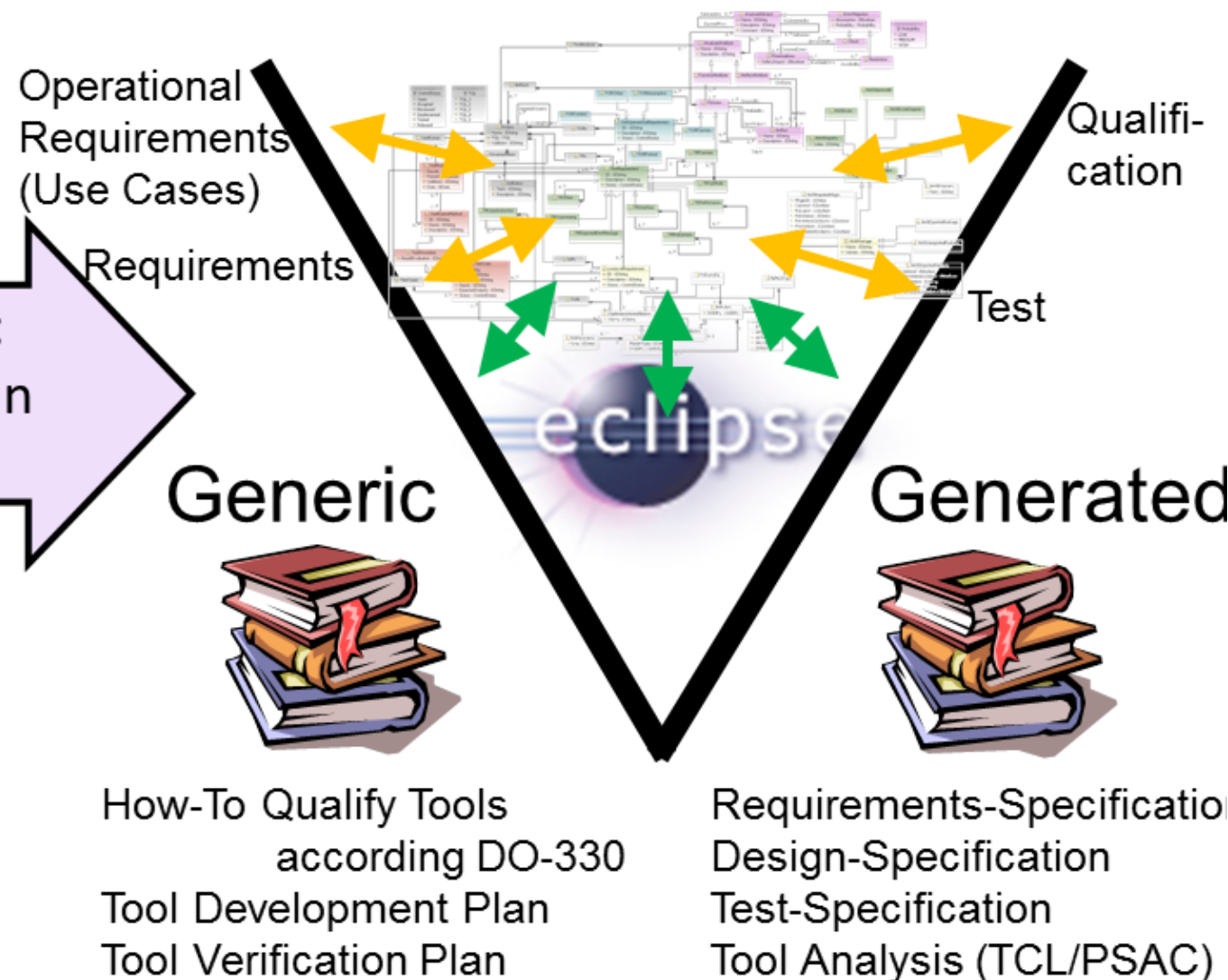
Model-based  
Tool  
Qualification

## Current Metamodel

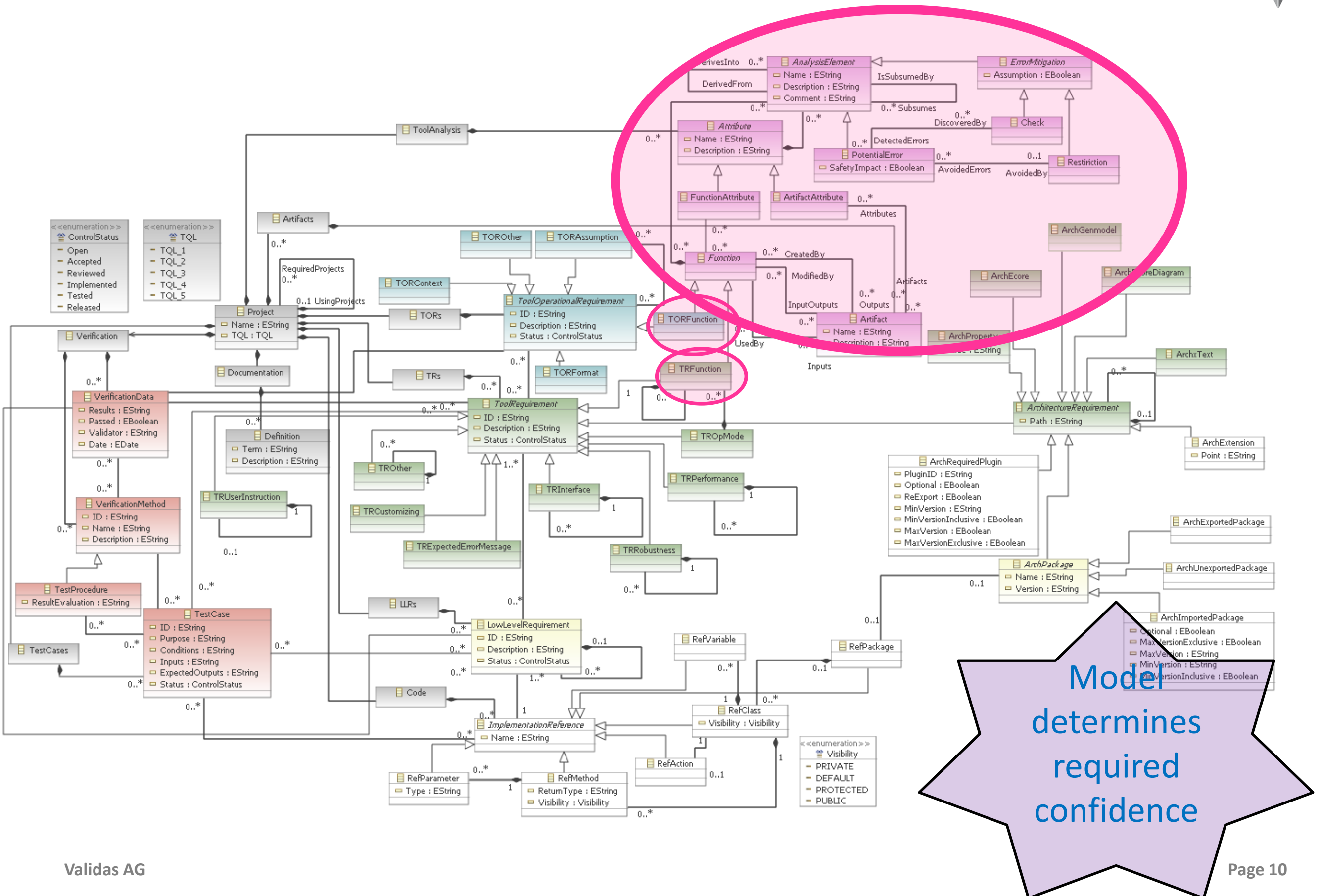


Eclipse Project:  
Qualifiable Plugin  
Projects (QPP)

## New Extended Metamodel



# Planning: Analysis Model for PSAC



# Current Eclipse Metadata



Eclipse is  
(partially)  
model-based  
development

## Overview

### General Information

This section describes general information about this plug-in.

ID:

Version:

Name:

Provider:

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.

[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)

Add...

Remove

Properties...

Add Action

Add Class

Add Method

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

Total: 6

Qualification model extends current Eclipse models

Overview

Dependencies

Runtime

Extensions

Extension Points

Build

MANIFEST.MF

plugin.xml

build.properties

Qualifiabe Features

Qualifcation Evidence

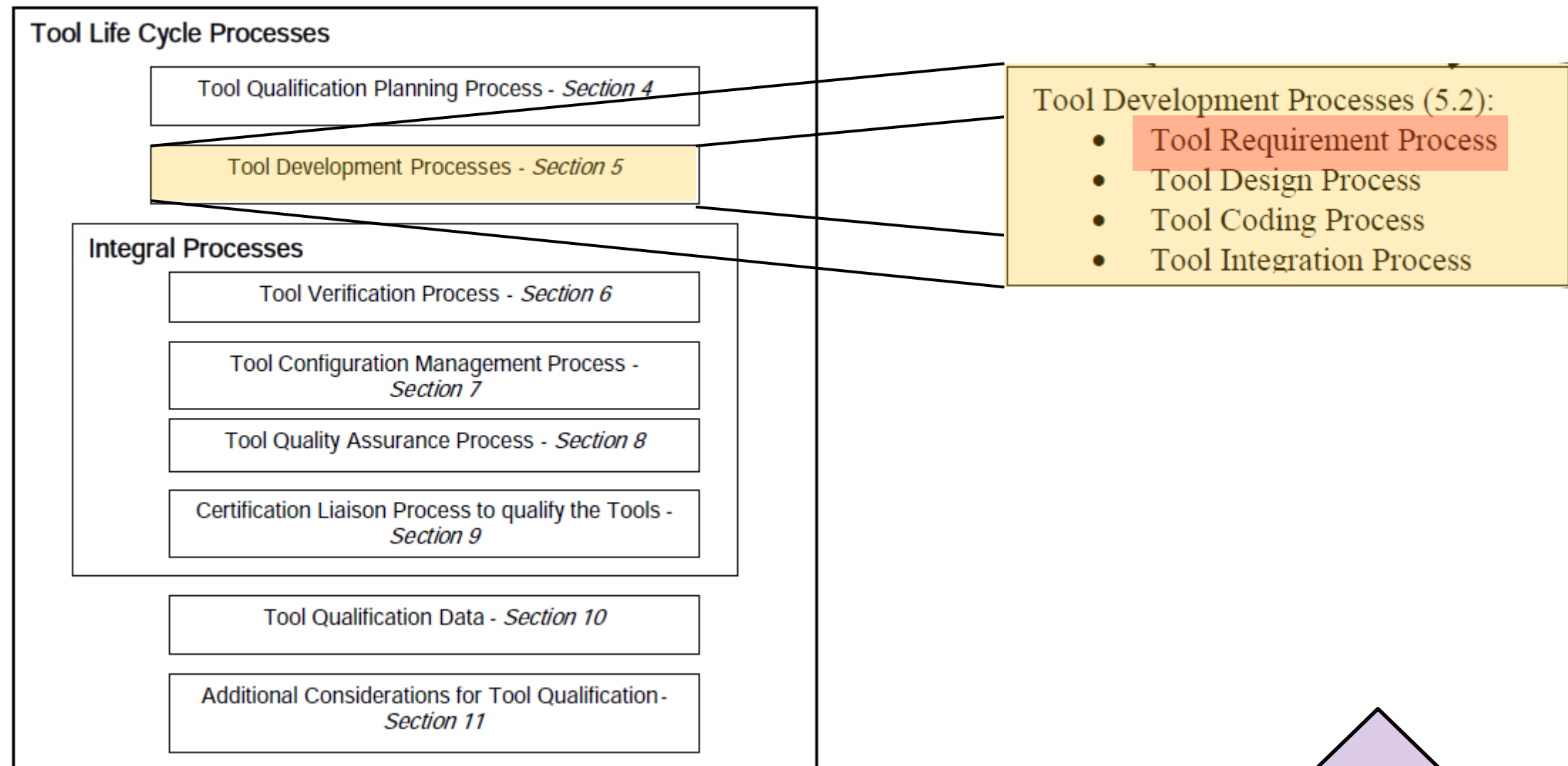
Validas AG

Page 12

# DO-330 Topics



## ► Structure of DO-330



Model  
supports  
all parts of  
DO-330



# Checklist for DO-330 compliance (refined)

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

Tracing to  
DO-330

HowToQualifyEclipseBasedTools.doc [Kompatibilitätsmodus] - Microsoft Word

Navigation

Dokument durchsuchen

- 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

## 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 <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]

- 



## Version 0.2



Tracing to  
DO-330



## 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) implementation**
  - 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

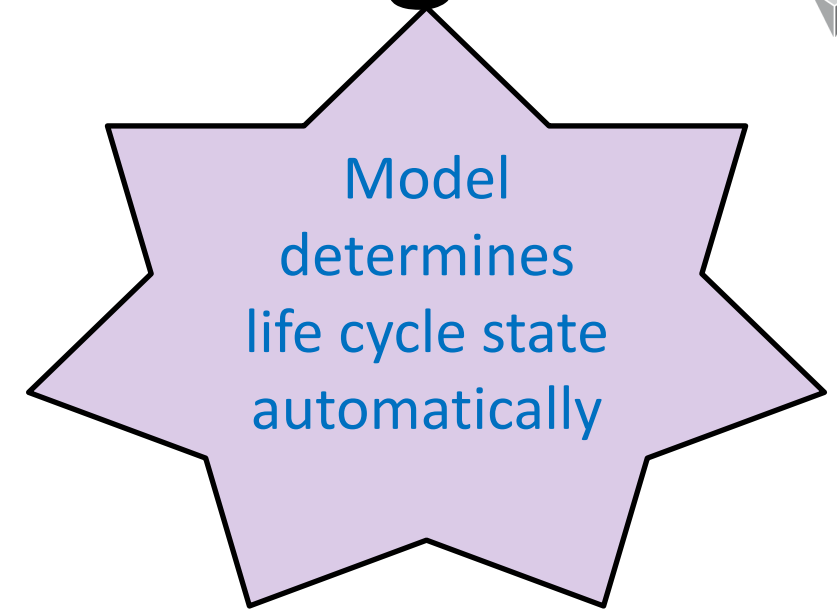
1 Document History
2 Definitions
▶ 3 Introduction
3.1 Document Extension
3.2 Multi-Function Tools
3.3 Usage of Qualified Plugins
3.4 COTS Tools
▶ 4 Standards
▶ 4.1 Project and General Information
▶ 4.2 Tool Qualification Planning
4.2.1 Overview
▶ 4.2.2 The Tool Analysis Model
4.2.3 Determination of the 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

# Tool Life Cycle for Qualifiable Plugins



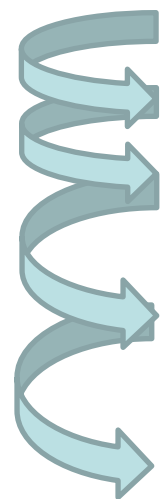
► **Combines the following processes:**

- 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* 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*.

Model  
defines tool  
life cycle  
transitions  
(formally)



# Content



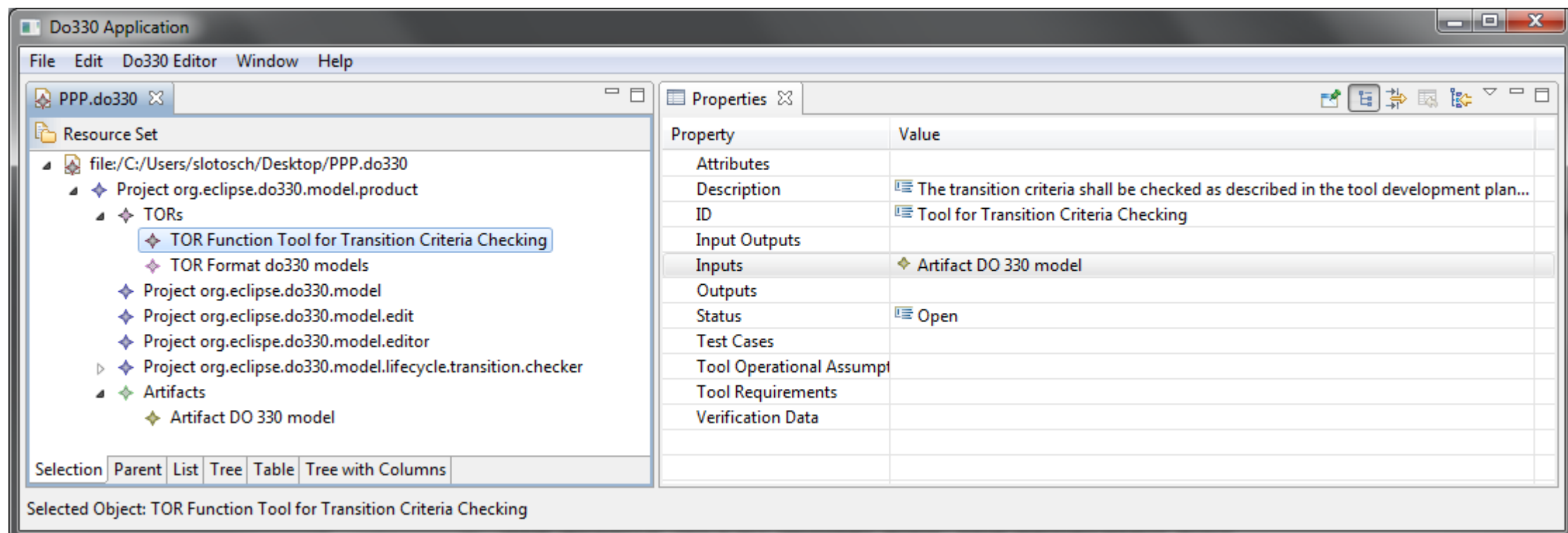
- ▶ Introduction: Validas & Eclipse Automotive IWG WP5
- ▶ The roadmap of Eclipse towards Tool Qualification
- ▶ Model-Based Tool Qualification
  - Overview
  - Model & Automatization
  - Documentation generation
- ▶ **Example: Ongoing demonstrator**
- ▶ Business model (proposal): Pay per qualification
- ▶ Summary

# Demonstrator



- ▶ **Eat your own dog food**
- ▶ **Demonstrator: DO-330 transition criteria checking**
  - Can be reused for Eclipse-Integration (QPP)
  - Can be used for tool qualification
  - Efforts monitoring
  - Access to all concept documents and models
- ▶ **Milestones defined**
- ▶ **Start model-based tool qualification**

Demonstrate  
model-based  
development /  
qualification



# Content



- ▶ Introduction: Validas & Eclipse Automotive IWG WP5
- ▶ The roadmap of Eclipse towards Tool Qualification
- ▶ Model-Based Tool Qualification
  - Overview
  - Model & Automatization
  - Documentation generation
- ▶ Example: Ongoing demonstrator
- ▶ **Business model (proposal): Pay per qualification**
- ▶ Summary

# Business Model: Pay Per Qualification



Still not clear how Eclipse qualification kits can be produced and used in an open way for user and provider

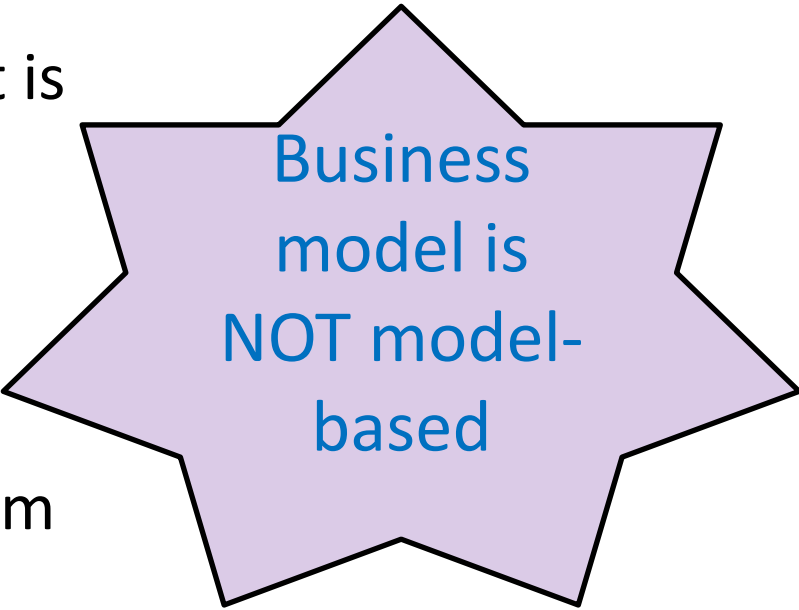
## ► Pay per Qualification:

- The owner of a qualification kit gets money when the kit is applied
  - Part 1: The analysis model, required for further classification (pay once to get it)
  - Part 2: The qualification kit (pay every time you run the kit)
- Qualification kits are available in a “qualification store”
- Application would include the application of the kit’s from required plugins (at least part 1)
- Some % from the money is collected for infrastructure topics

## ► This would make qualification kits of highly used plugins very valuable

- Companies / persons could invest in qualification kit’s like they invest into tools
- User groups can invest together into their kits

## ► This would restrict investment into qualification only for the first phases to set up the infrastructure: QPP & qualification store



Business  
model is  
NOT model-  
based

# Content



- ▶ Introduction: Validas & Eclipse Automotive IWG WP5
- ▶ The roadmap of Eclipse towards Tool Qualification
- ▶ Model-Based Tool Qualification
  - Overview
  - Model & Automatization
  - Documentation generation
- ▶ Example: Ongoing demonstrator
- ▶ Business model (proposal): Pay per qualification
- ▶ **Summary**



# Summary



- ▶ DO-330 is a safety standard for tools: Automotive, Avionics, ...
- ▶ Eclipse is (partially) model-based
- ▶ Qualification requires an extended model: Requirements , Tests,..
- ▶ Model can be used for
  - Generation of documentation
  - Checks & automatization
- ▶ Eclipse is on the road towards enabling of tool qualification
- ▶ Ongoing work



# First Tool Qualification Symposium



## ► **Presentations**

- Tool user & tool provider
- Qualification requirements & qualification kits
- Experiences from different domains & different industries
- Practical experiences & practical support

## ► **Keynote speech from F. Pothon**

- Tool Qualification Considerations and Certification  
Credits of Qualified Code Generators

## ► **Location: Munich Airport**

## ► **Registration and further information:**

- <http://toolqualification2013.eventbrite.com>

## ► **Organization: Validas AG**

- [tqs@validas.de](mailto:tqs@validas.de)
- 

# Thank You!



**VALIDAS** 

Arnulfstraße 27  
80335 München  
[www.validas.de](http://www.validas.de)  
[info@validas.de](mailto:info@validas.de)