www.thalesgroup.com

[Kitalpha] Accuracy

A tool for validation of models

OPEN

THALES

Agenda

Introduction

Accuracy

- What is Accuracy?
- Principles
- Accuracy in practice

Example

Agenda

Introduction

Accuracy

- What is Accuracy?
- Principles
- Accuracy in practice

Example

Introduction

Context

- Today, constraints for model checking are commonly based on EMF Validation and declared in a plugin.xml file
- Managing (adding, deleting, activating and deactivating) a large set of constraints is uneasy (XML notation)
- The issue is to declare and apply model checking rules expressing in various languages (e.g., OCL, JAVA) and with a notation simpler than XML

Purpose

- Providing a framework which facilitates management and application of a set of constraints for model checking

Agenda

Introduction

Accuracy

- **What is Accuracy?**
- Principles
- Accuracy in practice

Example

What is Accuracy?

- **Accuracy is a Kitalpha component based on EMF validation which enables to declare and apply constraints on models**
- **For a designer or developer: Accuracy enables to declare high-level model constraints in order to hide implementation details**
- **For the developer of constraints: Accuracy provides a framework to describe model constraints written either in OCL or JAVA**

Example

```
CommonRootCategory=simple component
ConstraintsFolder=OwnConstraints/Constraints/

ConstraintFiles=\
constraint0001,\

#meta-information of OCL constraint
constraint0001.Invariant.hardware_component_name_size.Name=HardwareConstraintNameSize
constraint0001.Invariant.hardware_component_name_size.Message= the size of {0} name must have at least 4 characters.
constraint0001.Invariant.hardware_component_name_size.Severity=ERROR
constraint0001.Invariant.hardware_component_name_size.Categories=simple component/hardware
constraint0001.Invariant.hardware_component_name_size.Code=0001
```

Agenda

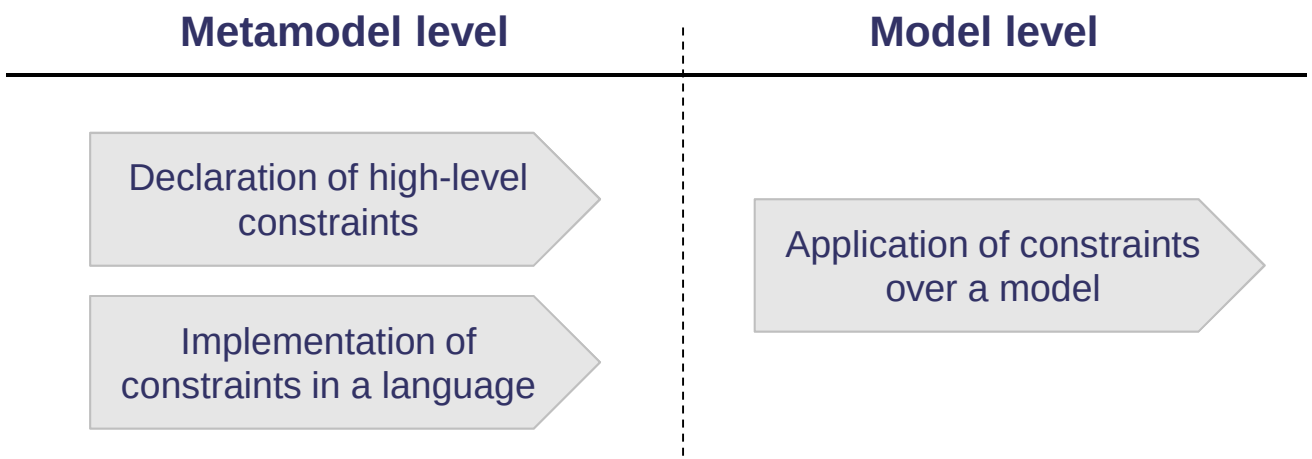
Introduction

Accuracy

- What is Accuracy?
- **Principles**
- Accuracy in practice

Example

Process

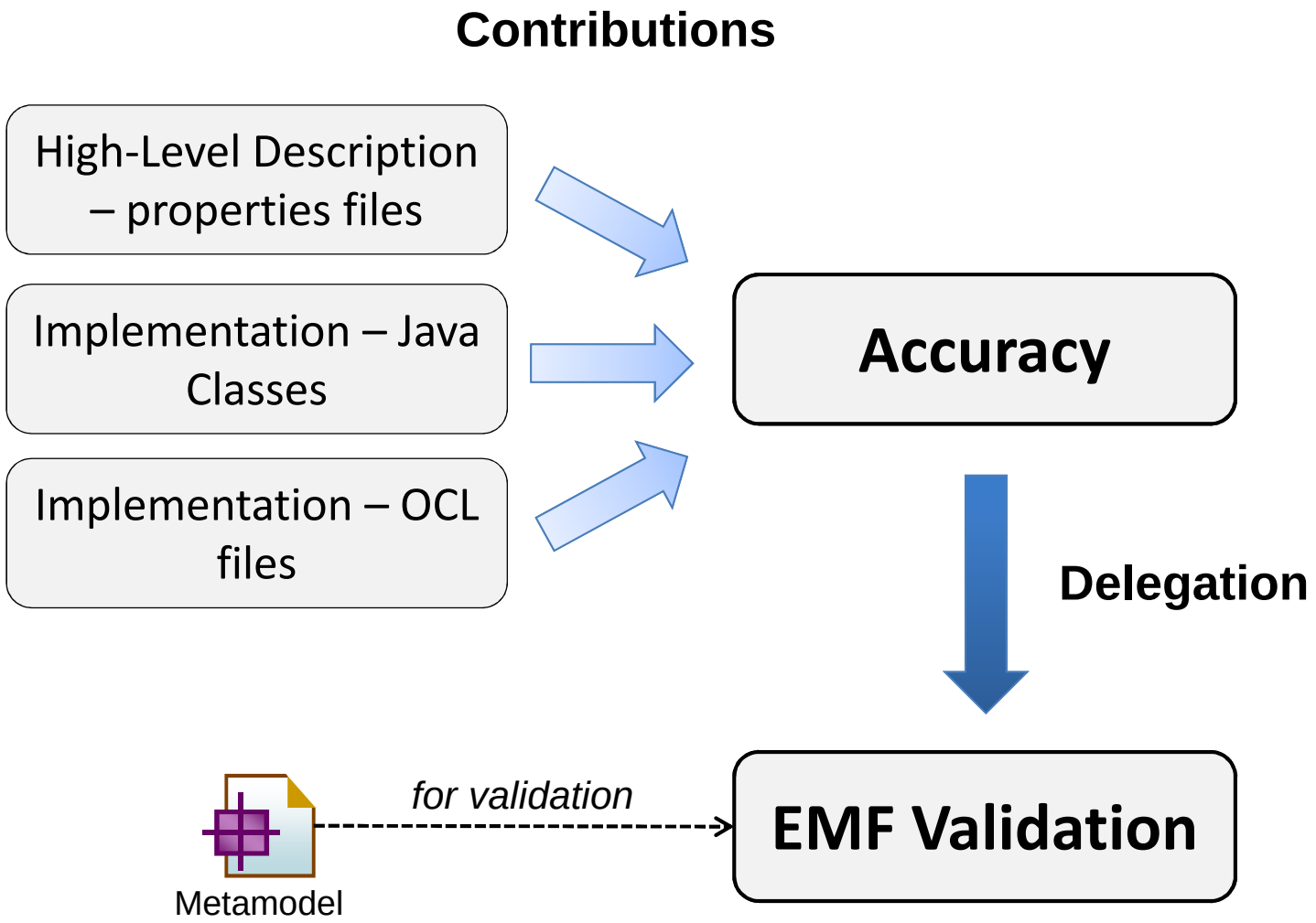


Declaration and implementation of constraints in appropriate files

Constraints are declared and written in appropriate files

- **High-level description of constraints in “Properties” files**
  - Constraint configuration (e.g., root category, mandatory rules list)
  - Meta-Information (e.g., severity, messages, categories)
- **OCL implementation in OCL files**
  - OCL expressions (e.g., invariants)
- **Java implementation in Java classes**
  - Java constraints
  - Constraints providers

Position of Accuracy and EMF Validation



Format of the Properties File

Example

```
CommonRootCategory=simple component
ConstraintsFolder=OwnConstraints/Constraints/

ConstraintFiles=\
constraint0001,\

#meta-information of OCL constraint
constraint0001.Invariant.hardware_component_name_size.Name=HardwareConstraintNameSize
constraint0001.Invariant.hardware_component_name_size.Message= the size of {0} name must have at least 4 characters.
constraint0001.Invariant.hardware_component_name_size.Severity=ERROR
constraint0001.Invariant.hardware_component_name_size.Categories=simple component/hardware
constraint0001.Invariant.hardware_component_name_size.Code=0001
```

Header

List of contraints

Header

CommonRootCategory	Required
ConstraintsFolder	Required for OCL constraints
ConstraintFiles	Required for OCL constraints
Mandatory.rules.list	Optional
Category	Optional, for filtring
Additional.rules.list	Optional, for filtring

Format of the Properties File

Example

```
CommonRootCategory=simple component
ConstraintsFolder=OwnConstraints/Constraints/

ConstraintFiles=\
constraint0001,\
```

Header

```
#meta-information of OCL constraint
constraint0001.Invariant.hardware_component_name_size.Name=HardwareConstraintNameSize
constraint0001.Invariant.hardware_component_name_size.Message= the size of {0} name must have at least 4 characters.
constraint0001.Invariant.hardware_component_name_size.Severity=ERROR
constraint0001.Invariant.hardware_component_name_size.Categories=simple component/hardware
constraint0001.Invariant.hardware_component_name_size.Code=0001
```

List of contraints

Constraint description

Name	Name of the constraint
Message	Message to display
Severity	Level of the severity
	NULL : all is OK
	INFO: informative message
	WARNING: warning message
	ERROR: error message
	CANCEL: validation was canceled
Categories	Constraint category
Code	Unique code of the constraint

Syntax for an OCL constraint

<OCL file name>.invariant.<invariant name>.<data to set> = <value>

Where Invariant name: is declared in OCL file

Syntax for a Java constraint

<Requirement ID>.invariant.<invariant ID>.<data to set> = <value>

Where Requirement ID: provided at the contribution of Java constraint

Invariant ID: provided at the contribution of Java constraint

Agenda

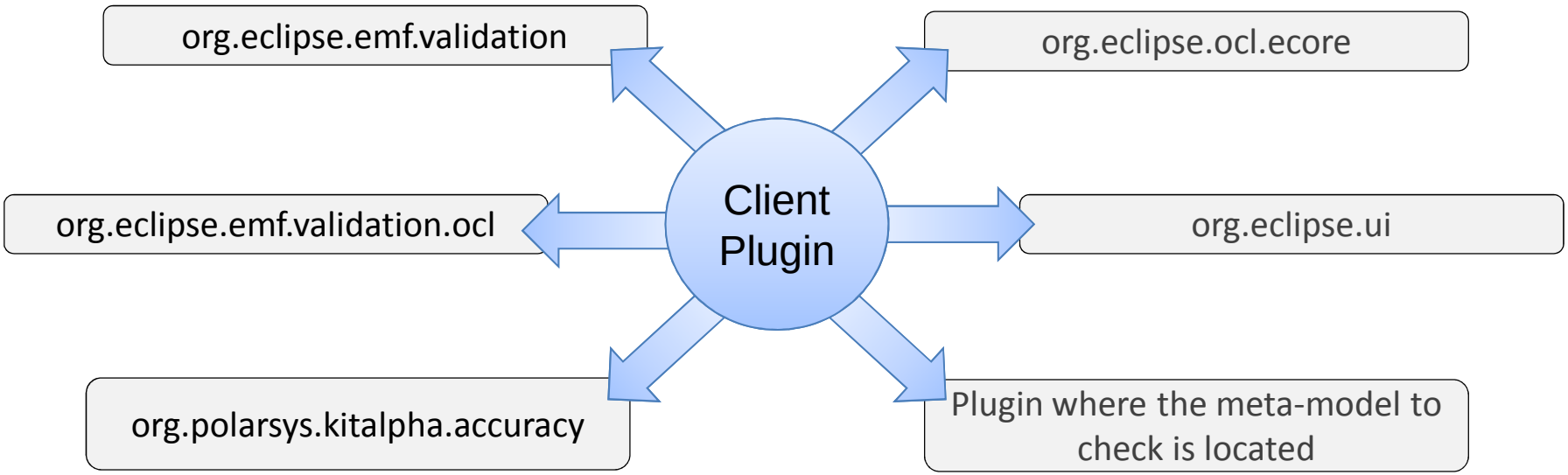
Introduction

Accuracy

- What is Accuracy?
- Principles
- **Accuracy in practice**

Example

Dependencies



Definition of extensions 1/2

org.eclipse.emf.validation.constraintBindings

- Create a new binding
- Specify a category of constraints (CommonRootCategory in properties file)
- Specify a Client context in which the constraints will be applied
  - Accuracy use com.thalesgroup.mde.validation.generic.provider.clientContext as client context

org.eclipse.emf.validation.constraintProviders

- For registering constraints provider for a specific meta-model
- The provider extends either GenericOCLConstraintProvider for OCL constraints provider or GenericJavaConstraintProvider for java constraints provider

com.thalesgroup.mde.validation.java

- Contribution for Java constraint
  - Requirement ID
  - Invariant ID
  - The Java class which implements IJavaConstraint interface

Definition of extensions 2/2

org.eclipse.ui.startup

- This extension point allows activating a plugin when the workbench starts
- It is used by Accuracy 1) to register an EValidator for a Meta-model on which constraints will be applied, and 2) to add the categories and constraints to the preferences view (Model validation node)

Best practice: use a second plugin for the Accuracy UI integration which contribute to org.eclipse.ui.startup. This keep UI and the declaration of constraints independent.

Filtering Constraints

- **Accuracy can filter (Activate/Deactivate) constraints from the preferences view, by specifying, constraints to keep activating in properties file [Sens de la phrase?].**
- **To use this feature, the properties file:**
  - Specifying the category of constraints by setting the field category
  - Specifying the additional.rules.list field which contains all the requirement IDs of the constraints to be activated
- **Then, apply the configuration**
- **All the constraints listed in the addition.rules.list and mandatory constraints fields will be activated. Others constraints will be deactivated.**

Agenda

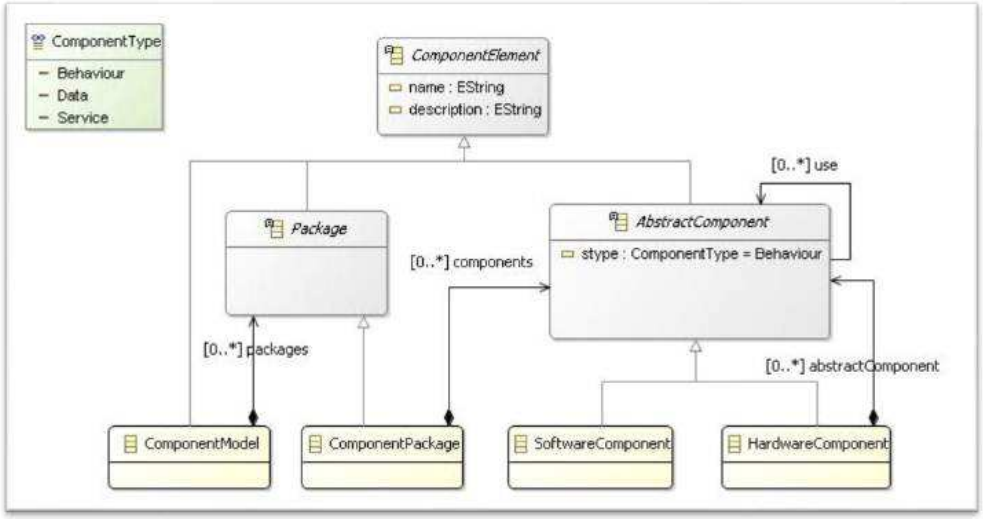
Introduction

Accuracy

- What is Accuracy?
- Principles
- Accuracy in practice

Example

Test Case



Component Sample metamodel

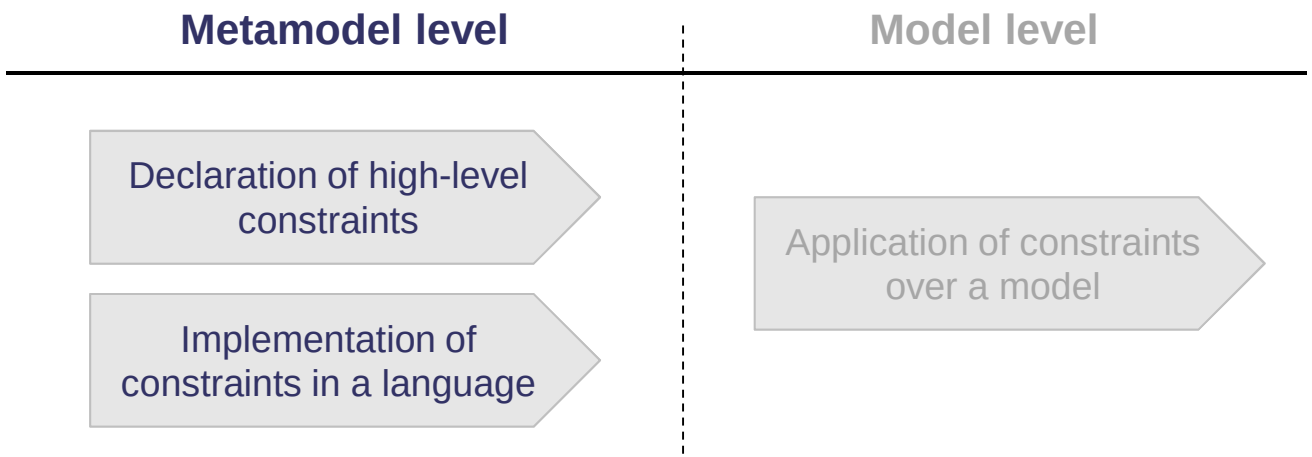
Java constraints

- The root must be instanceof ComponentModel
- Software must not have cyclic uses

OCL constraint s

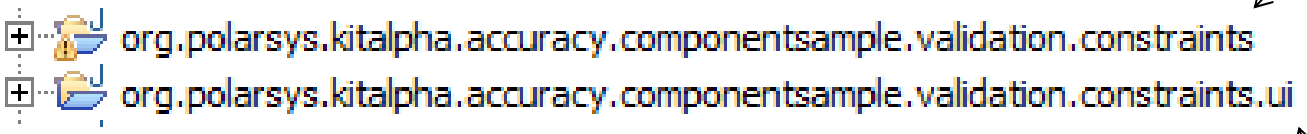
- Empty component names are not allowed
- Empty packages are not allowed

Development Phase



Plugins

Creation of two new plugins

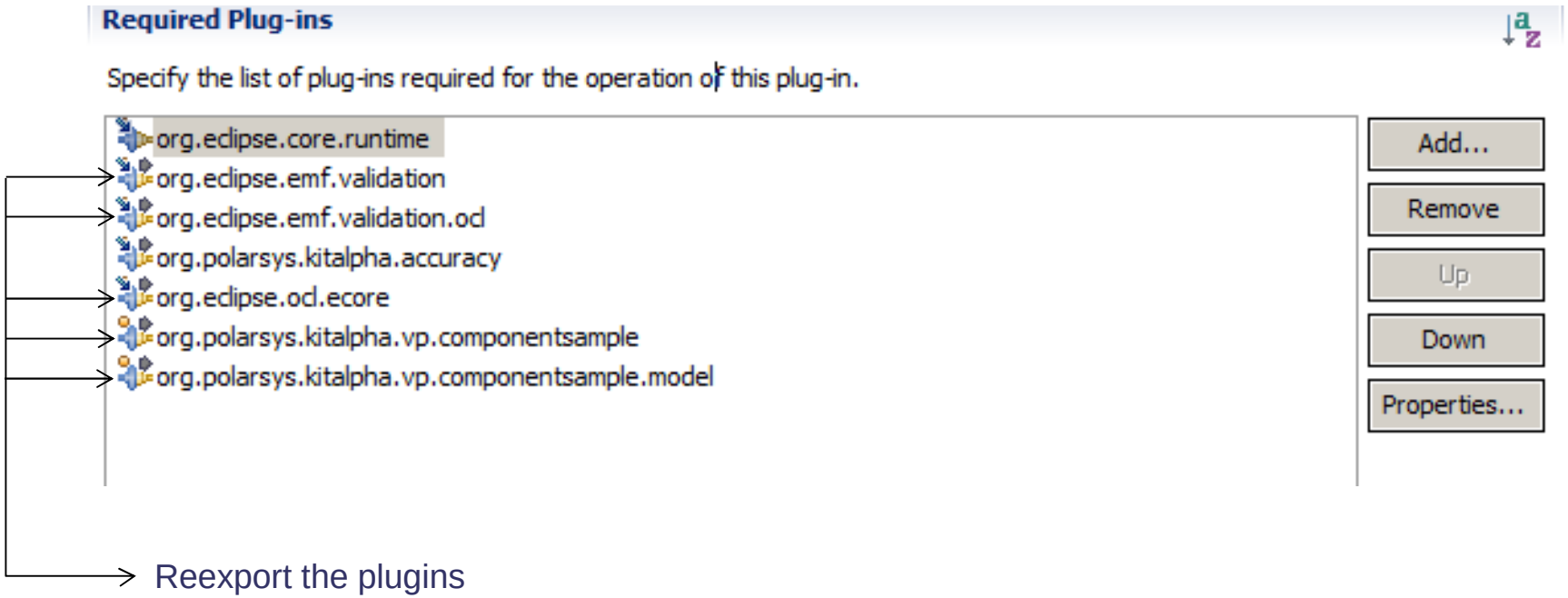


Client contributor

For register Validators and UI integration

Dependencies

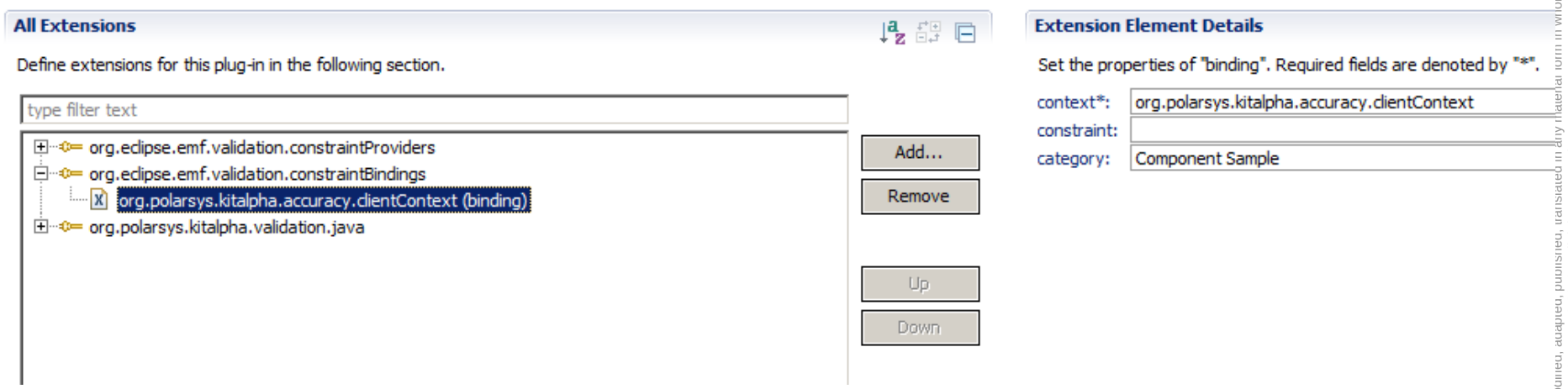
Add dependencies for contribution



EMF Validation contribution

Contribute to org.eclipse.emf.validation.constraintBindings

Add a binding to set a context, category of constraints

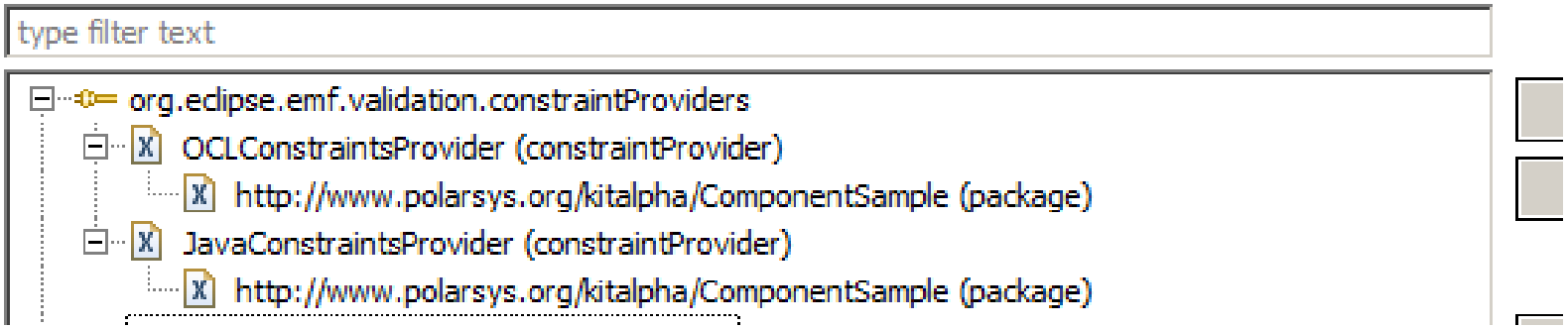


Constraints Providers – For Java – 1/4

Contribute to org.eclipse.emf.validation.constraintProviders

Java provider – How to contibute?

- Add a Java constraint provider
- Create SimpleComponentJavaConstraintProvider class that extends GenericJavaConstraintProvider
- Bind the constraint providers to meta-model by given the namespaceUri



Constraints Providers – For Java – 2/4

Java constraint provider for Simple component

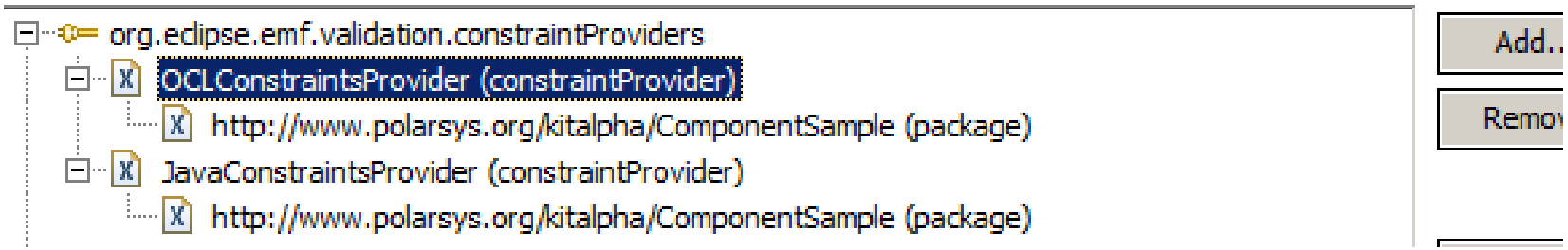
```
public class JavaConstraintsProvider extends GenericJavaConstraintProvider {

    @Override
    public ResourceBundle getConfigurationFileResourceBundle() {
        return ResourceBundle.getBundle(IConstants.CONSTRAINTS_CONFIG_FILE);
    }

    @Override
    public Bundle getContributorBundle() {
        return Activator.getDefault().getBundle();
    }

    @Override
    public URL getUrlFromPath(String filePath) {
        return Activator.getDefault().getBundle().getEntry(filePath);
    }

}
```

Constraints Providers – For OCL – 3/4

Contribute to org.eclipse.emf.validation.constraintProviders

OCL provider – How to contibute?

- Add an OCL constraint provider
- Create SimpleComponentOCLConstraintProvider class that extends GenericOCLConstraintProvider
- Bind the constraint providers to meta-model by given the namespaceUri



Constraints Providers – For OCL – 4/4

Java constraint provider for Simple component

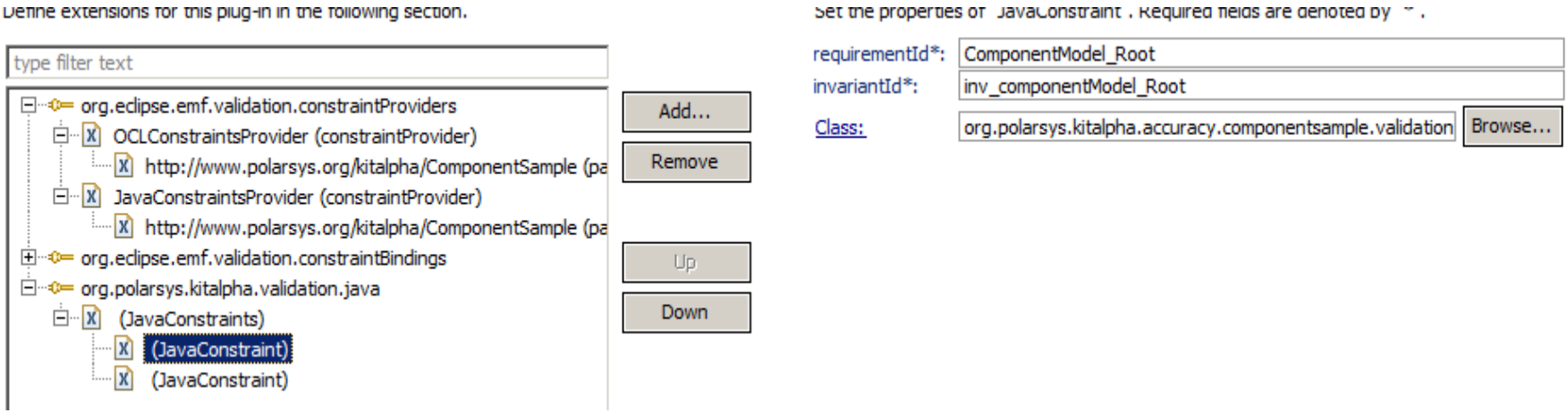
```
public class OCLConstraintsProvider extends GenericOCLConstraintProvider {

    @Override
    public ResourceBundle getConfigurationFileResourceBundle() {
        return ResourceBundle.getBundle(IConstants.CONSTRAINTS_CONFIG_FILE);
    }

    @Override
    public Bundle getContributorBundle() {
        return Activator.getDefault().getBundle();
    }

    @Override
    public URL getUrlFromPath(String filePath) {
        return Activator.getDefault().getBundle().getEntry(filePath);
    }

}
```

Java Contribution

Contribute to org.polarsys.kitalpha.validation.java

Java constraint – How to contribute?

- Set requirement ID of Java constraint
- Set invariant ID of constraint
- Create SoftwareNameStartWithS class that extends IJavaConstraint



OCL Contribution

Contribute to org.polarsys.kitalpha.validation.java

OCL constraint – How to contribute?

- Create a new folder to contain OCL constraints
- Create a new OCL file
- Write the OCL constraint in this file (code snippet below)

```
package ComponentSample

context ComponentElement

inv component_null_empty_name: not(name = null)

endpackage
```

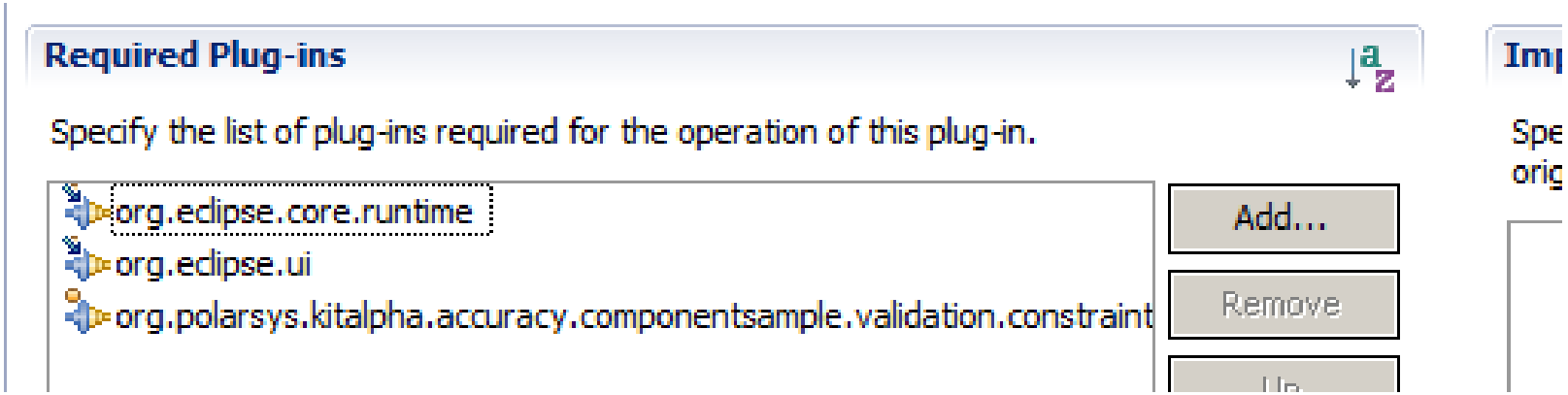
Description of High-level Model Constraints

Contribute to Properties file

```
CommonRootCategory=Component Sample/allconstraints

ConstraintsFolder=componentSampleConstraints/constraints/

ConstraintFiles=emptyPackage,\
emptyNames

mandatory.rules.list=\
emptyNames,\
ComponentModel_Root



emptyPackage.Invariant.component_package_not_empty.Name=ComponentPackageNotEmpty
emptyPackage.Invariant.component_package_not_empty.Message= The Component Package ({0}) is empty
emptyPackage.Invariant.component_package_not_empty.Severity=WARNING
emptyPackage.Invariant.component_package_not_empty.Categories=Component Sample/allconstraints/Component
Packages
emptyPackage.Invariant.component_package_not_empty.Code=0001

...
```

UI Contribution

Add dependencies



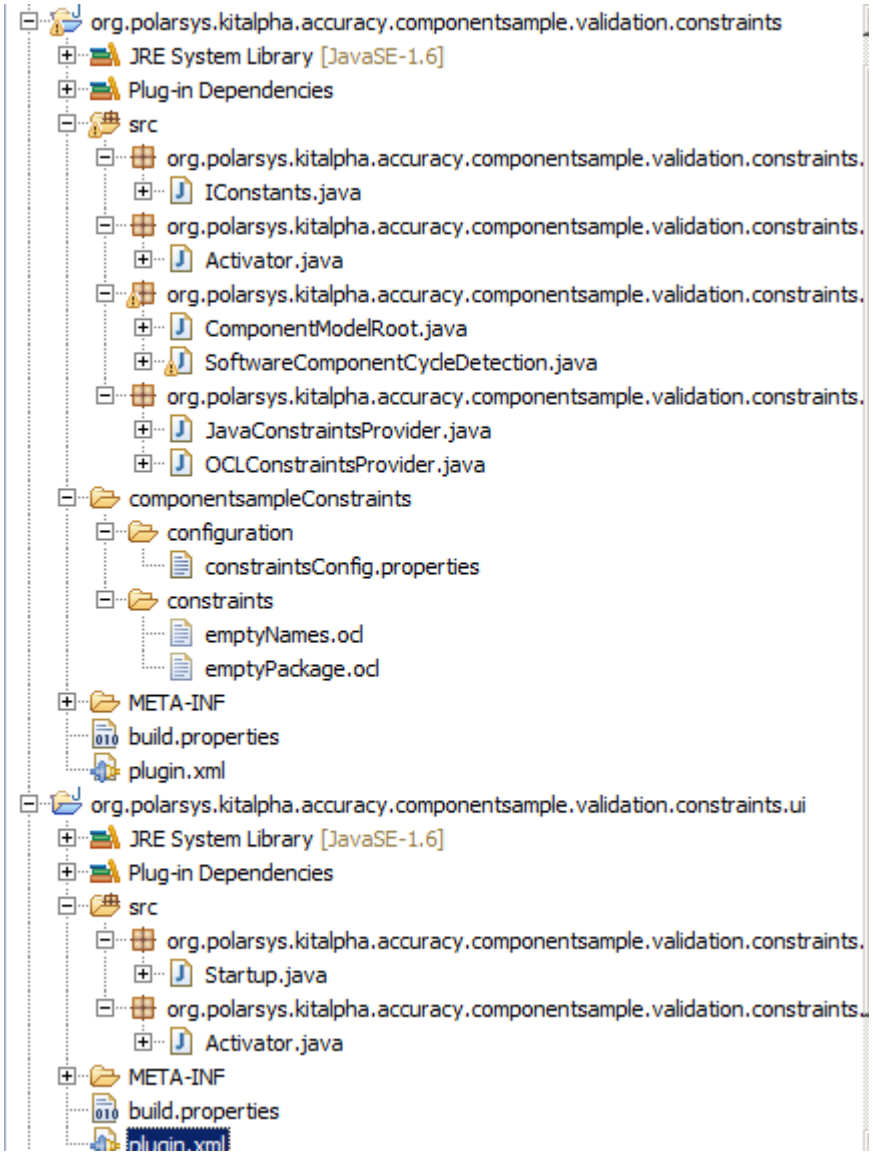
Contribution to Startup

Startup – How to contribute?

- Contribute to org.eclipse.ui.startup
- Create Startup class that implements IStartup

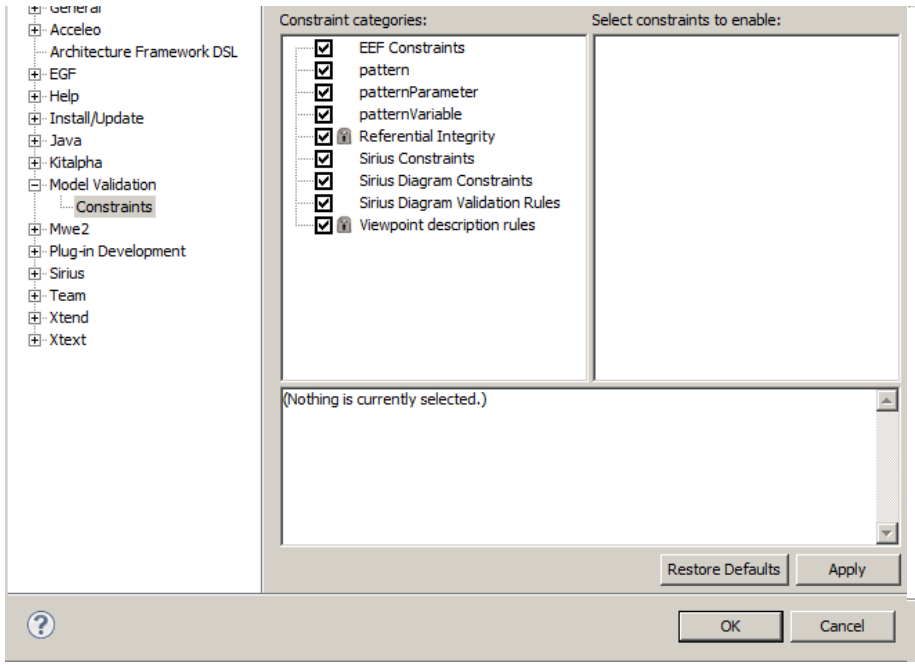
```
public class Startup implements IStartup {

    @Override
    public void earlyStartup() {

        //force the registration of the metamodel
        ComponentSamplePackage.eINSTANCE.eClass();

        //Register all packages to the same validation adapter
        EValidatorAdapter validationAdaptor = new EValidatorAdapter();

        EValidator.Registry.INSTANCE.put(ComponentSamplePackage.eINSTANCE, validationAdaptor);
        GenericConstraintProviderService.getInstance().registerProvider(new OCLConstraintsProvider());
        GenericConstraintProviderService.getInstance().registerProvider(new JavaConstraintsProvider());
    }

}
```

Package and Code Hierarchy



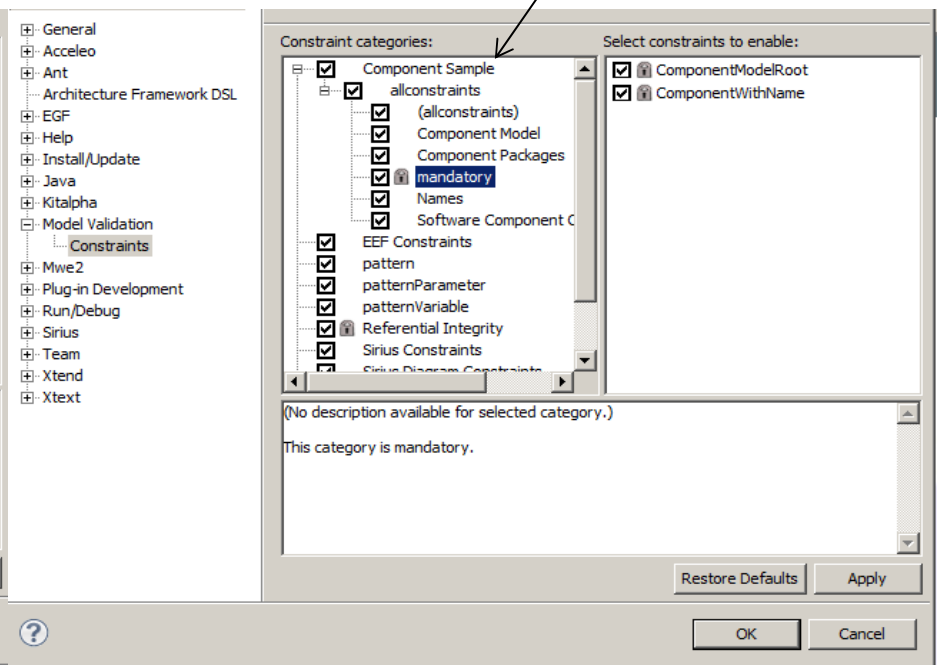
Preferences View

Before contribution

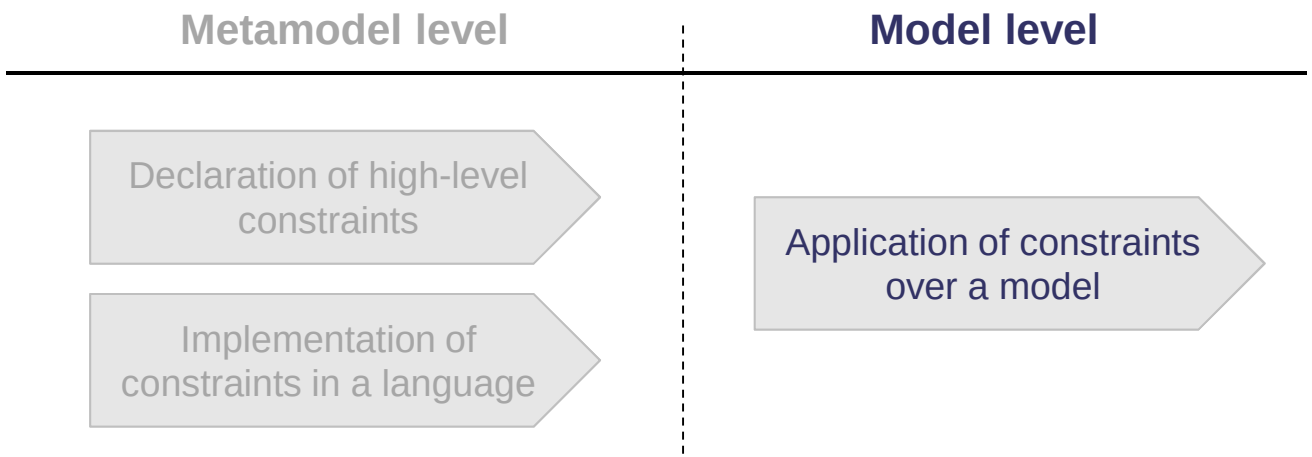


After contribution

New Contribution



Use Phase



Validation Result

Input instance

Validation Result