THALES

EGF Tutorial
Generation Chain

Benoît Langlois – Thales/EPM

EGF - Thales Corporate Services - EPM

Agenda

- Principles
- Generation Customization
- Generation Chain Extensibility

THALES

Agenda

- Principles
- Generation Customization
- Generation Chain Extensibility

THALES

Rationale

- **Objective of the Generation Chain:**
  - Definition, at a high level of description, of executable generations
  - Abstraction: encapsulating the irrelevant technical details of generation
  - Simplicity & Efficiency: Reducing the number of “clicks” (i.e. the number of actions)
    - Only providing the main generation features and next generating
- **Technical principle:**
  - Generation features are captured in a “generation chain” file
  - An EGF fcore file is produced from the generation chain: it contains the translation of the generation chain into factory components
  - Next, the factory components are transparantly executed to produce the expected artifacts

THALES

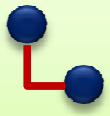
Levels of Generation

Level 1

Designer

Generation Chain

It captures the generation steps
and their features
File type: “generation chain”



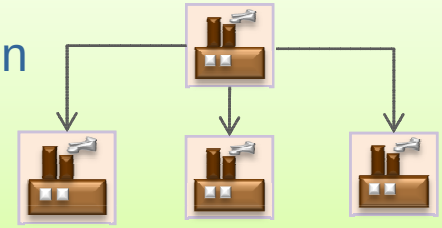
Instantiation
Generated

Level 2

Developer

Factory Components

They contain the logic of generation
File type: fcore



Instantiation
Execution of the generation

Level 3

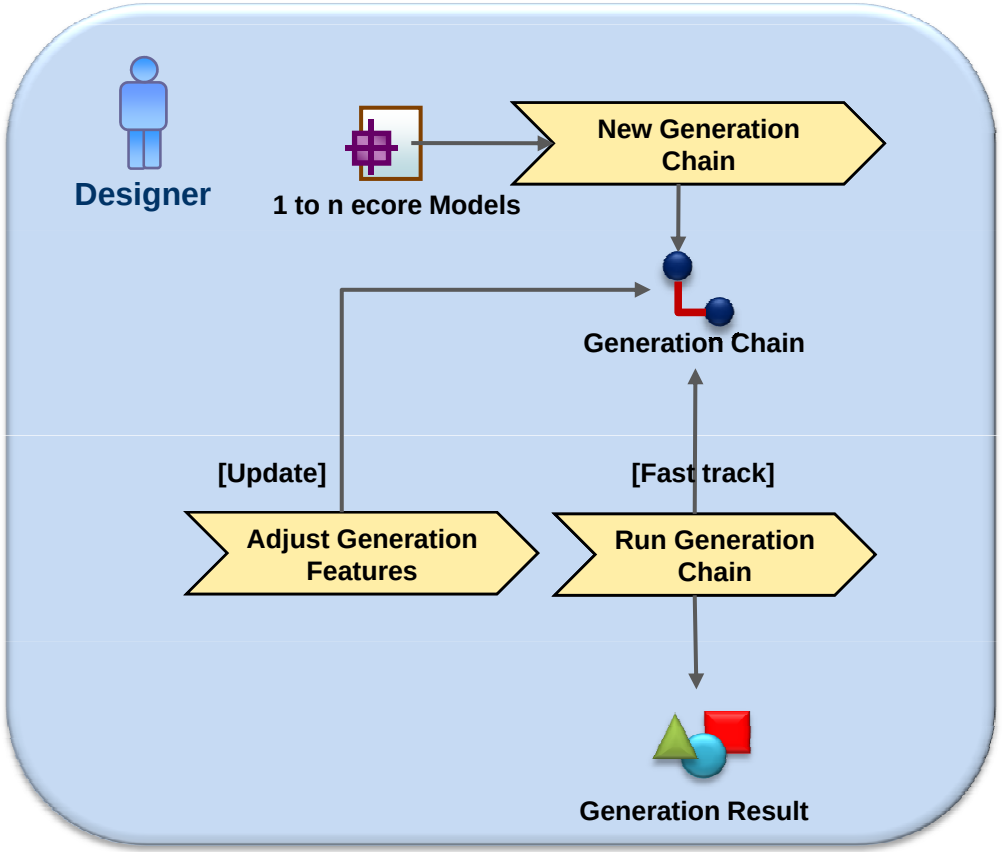
Result of the generation

The artifacts are generated

EGF: Eclipse Generation Factories – Thales Corporate Services/EPM

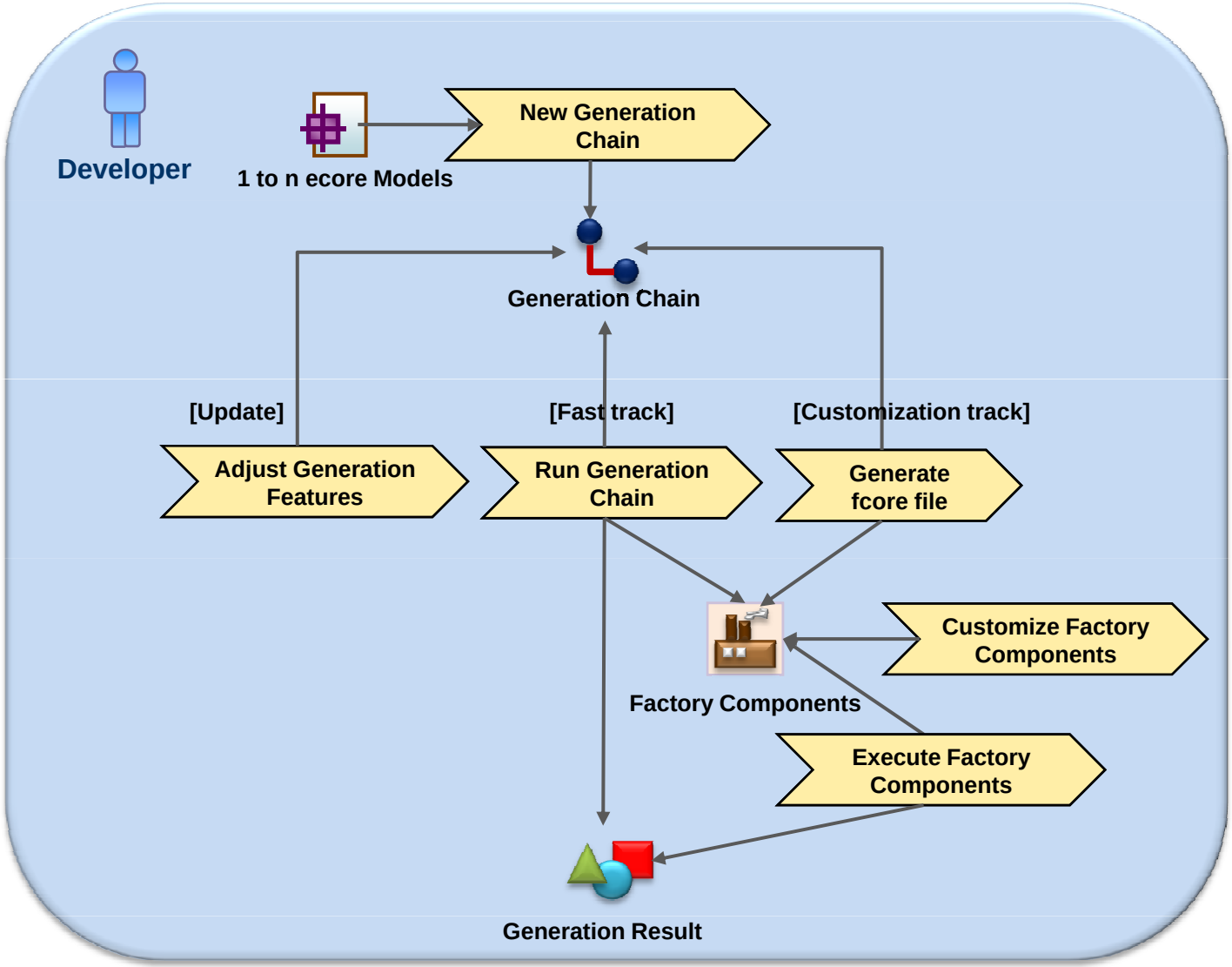
THALES

Process



THALES

Process

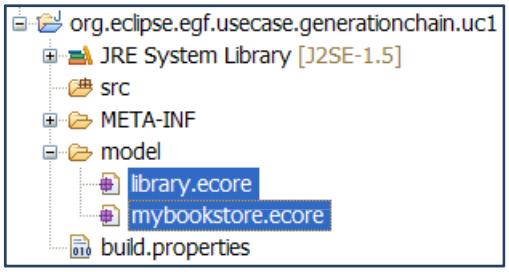


THALES

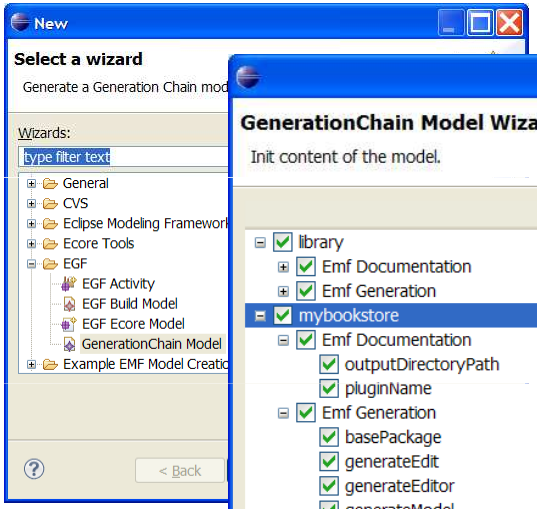
Visualization 1/3

Creation of Generation Chain

Selection of ecore models



New Generation Chain



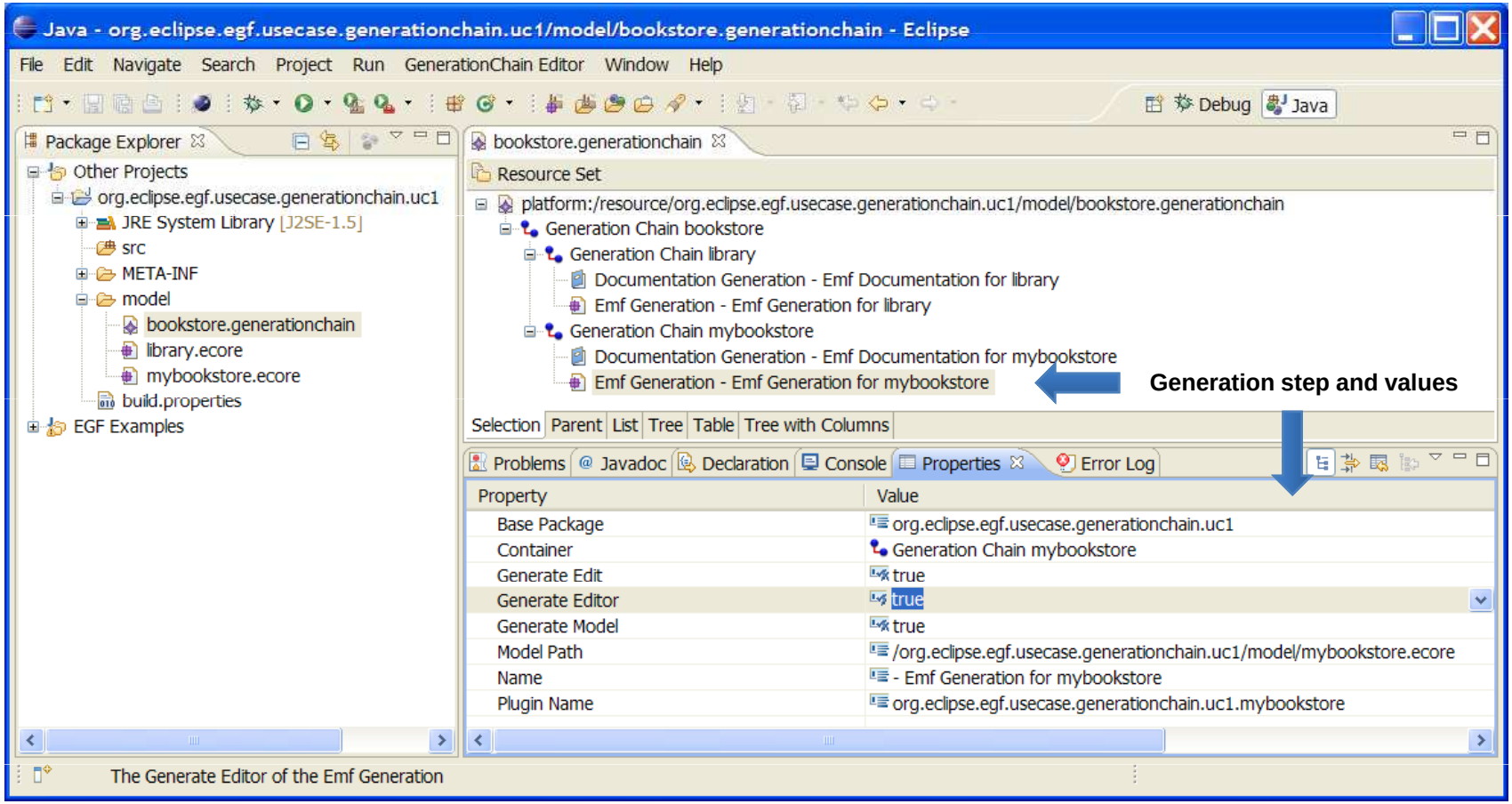
For each model, setting the generation features

THALES

Visualization 2/3

Modification of Generation Chain Features

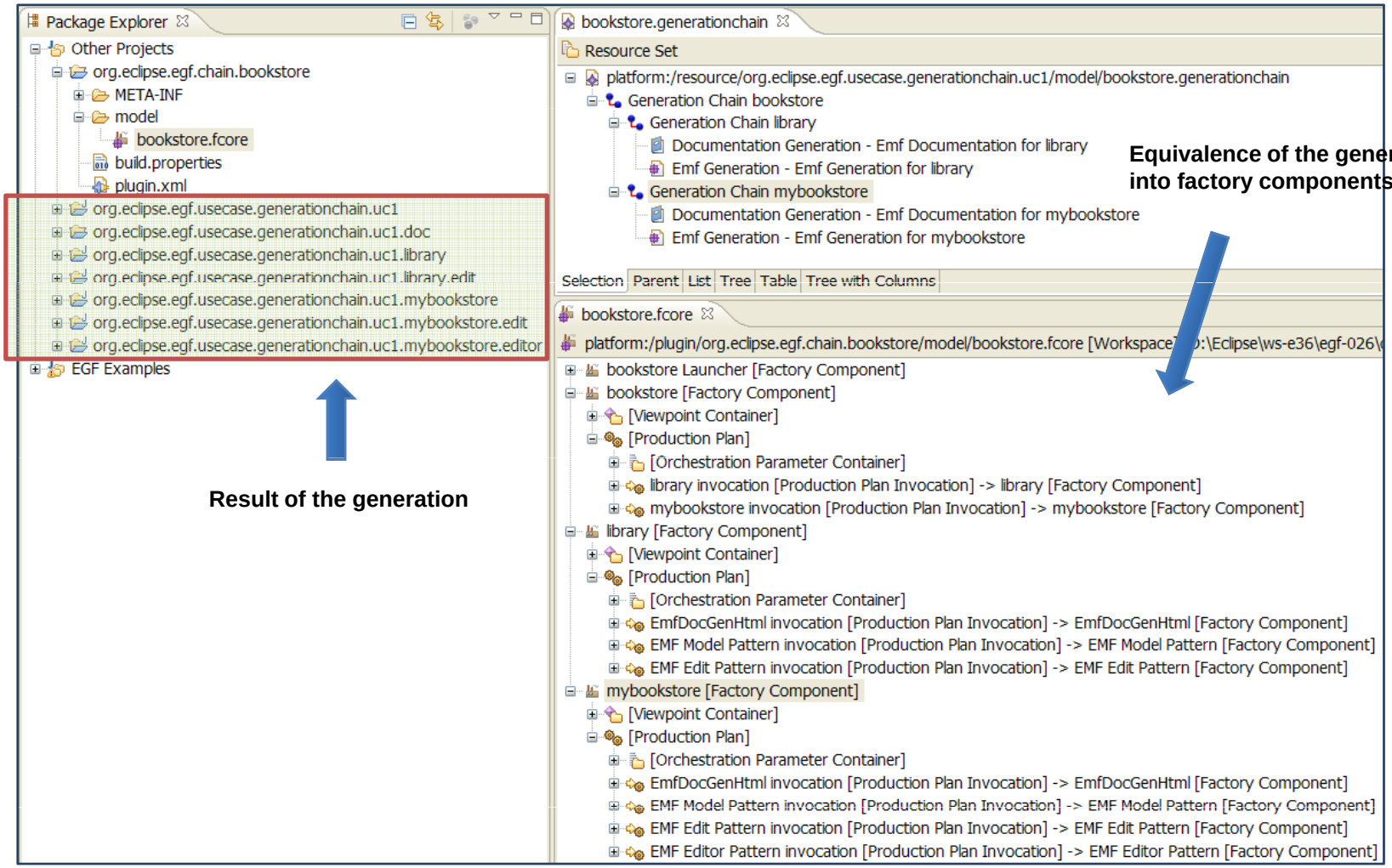
Possibility to change the generation features and add new generation steps



THALES

Visualization 3/3

Result of Generation Chain execution



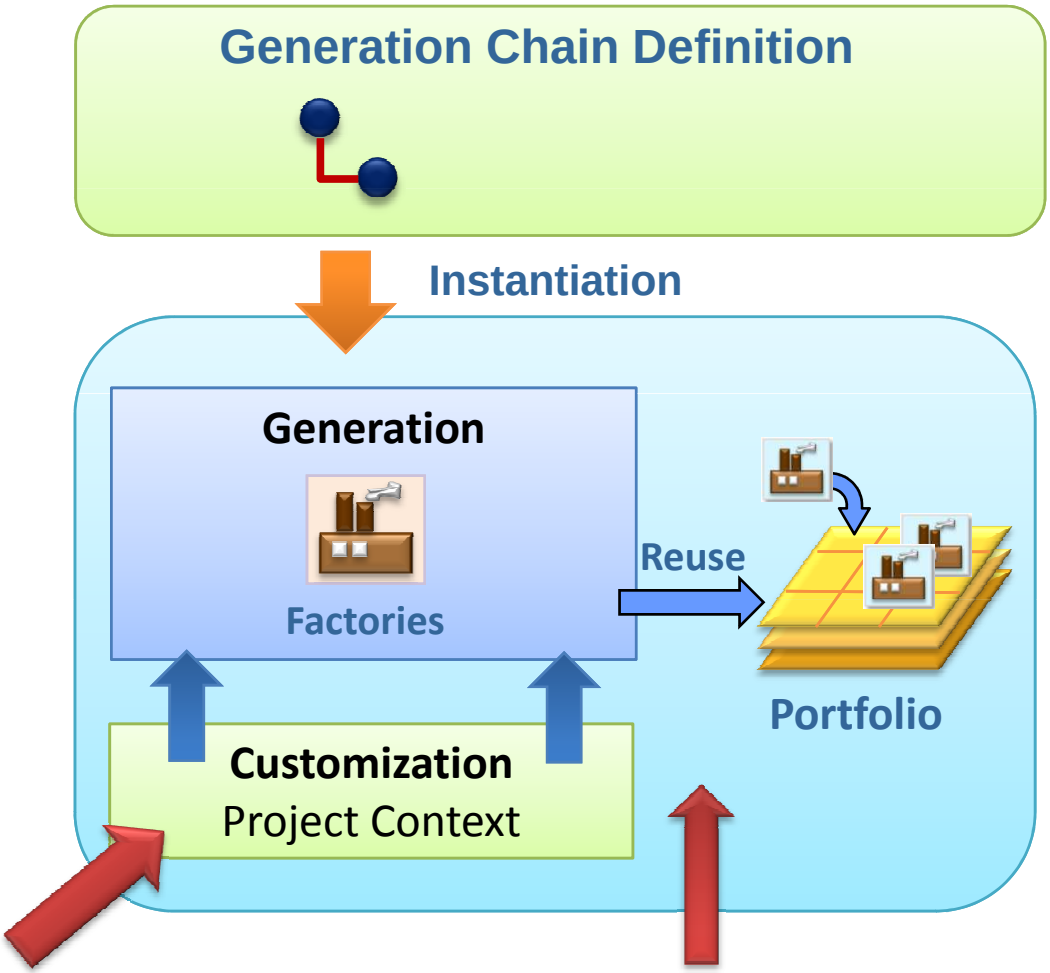
THALES

Agenda

- Principles
- Generation Customization
- Generation Chain Extensibility

THALES

Customization with Generation Chains



How to introduce customization with a generation chain-based development while a generation chain reuses a standard generation stored in a portfolio?

THALES

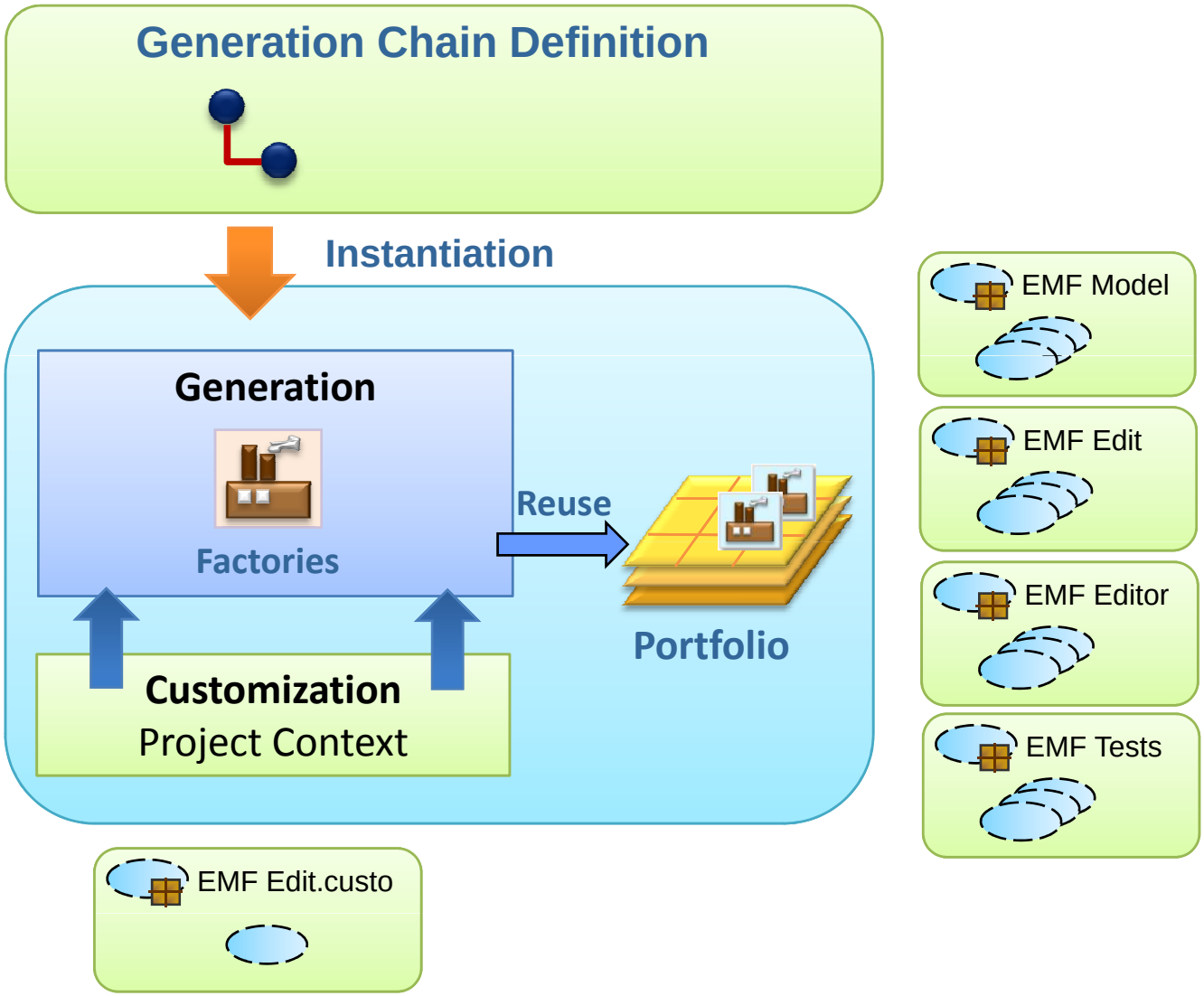
Principles of GC customization

- **Means of customization**
  - A customization is realized with patterns which specialize the standard generation, for instance model / edit / editor for the Emf generation

- **Incrementality**
  - When a feature in the generation chain is changed, the fcore is synchronized accordingly
    - Ex: when the emf model editor feature is set to “true”, the model editor generation is invoked, and in reverse is removed when this feature is set to “false”.
  - Protected elements:
    - Patterns in a Pattern Viewpoint
    - Pattern substitution in the “Orchestration Parameter Container” where the patterns for customization replace the standard patterns

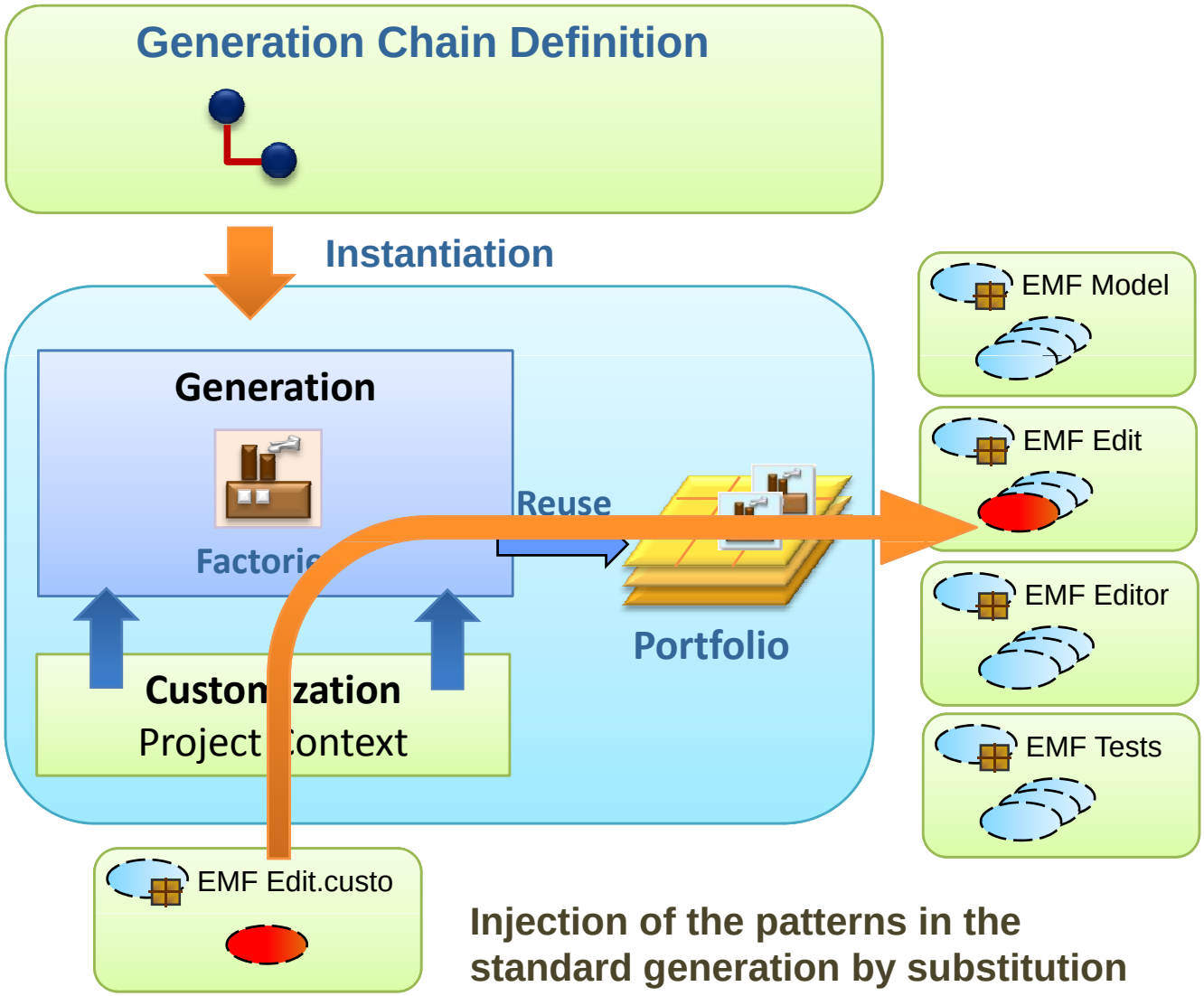
THALES

Illustration on EMF Generation



THALES

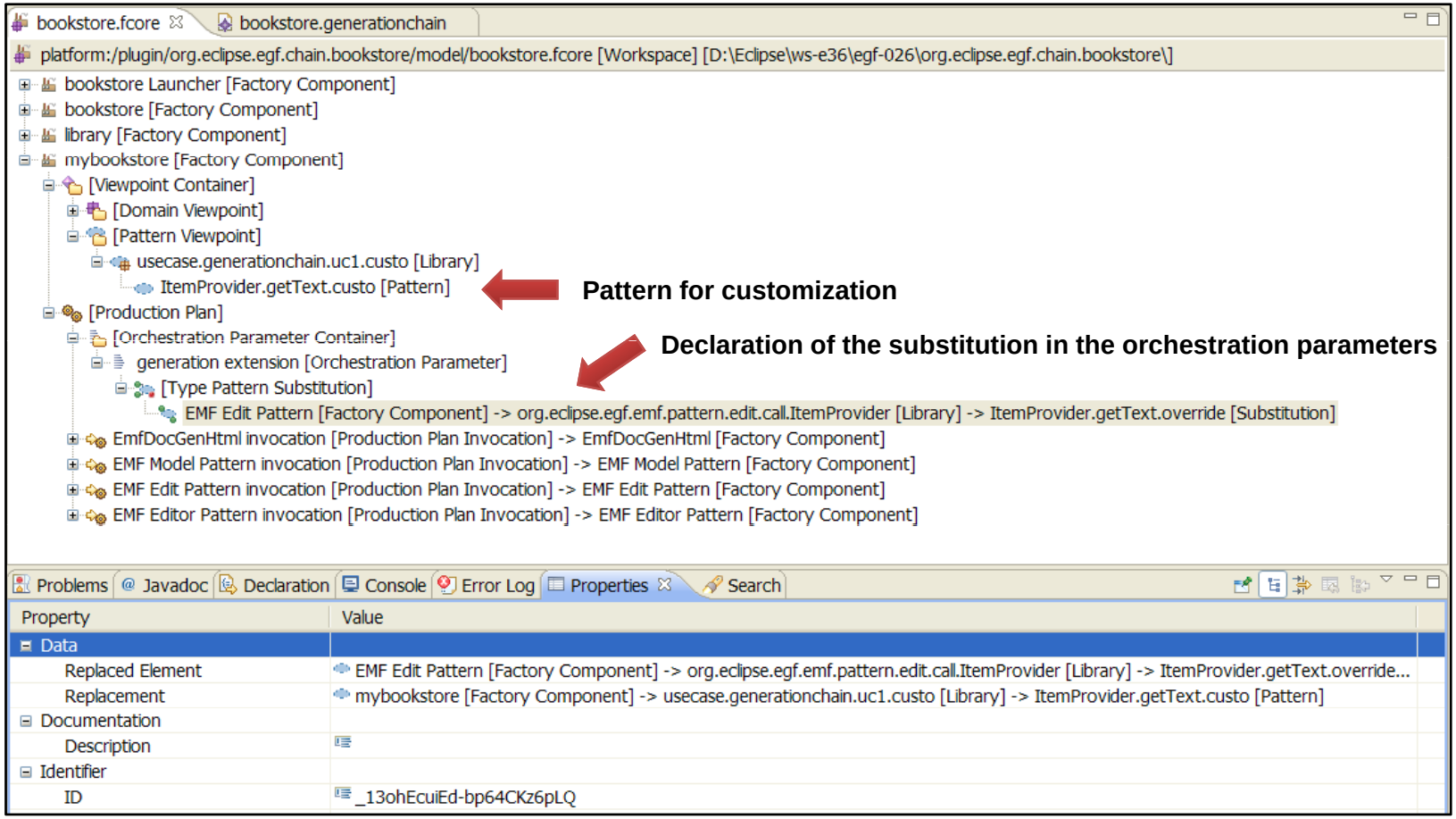
Illustration on EMF Generation



Injection of the patterns in the standard generation by substitution

THALES

Example of substitution



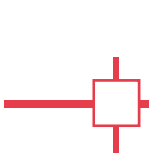
Refer to the “org.eclipse.egf.usecase.generationchain.uc1” example

THALES

Memo for a Customization with Patterns

- Open the fcore file related to the generation chain. In the Viewpoint container, create a pattern domain which will contain the pattern libraries and patterns for customization.
- Identify the standard patterns to extend. For their location, navigate for instance from the invoked factory component of the fcore file (e.g., EMF Edit) which contains the standard patterns.
- Create the pattern for customization:
  - In order to avoid rewriting everything from scratch in the new pattern, the pattern inherits from the standard pattern.
  - Add the precondition to apply the pattern; add the “imports” defined in the pattern header method; in the method for code generation (e.g., doGenerate) add the customized code.
- For pattern inheritance, in the Manifest of the plug-in which contains the fcore file, add the dependencies toward the plug-in(s) which contain(s) the standard pattern(s)

Agenda

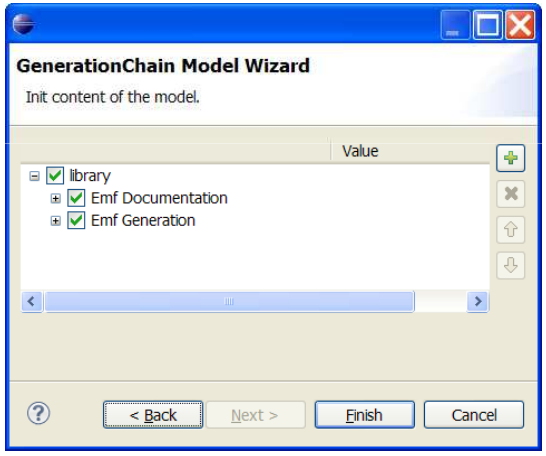


- Principles
- Generation Customization
- Generation Chain Extensibility

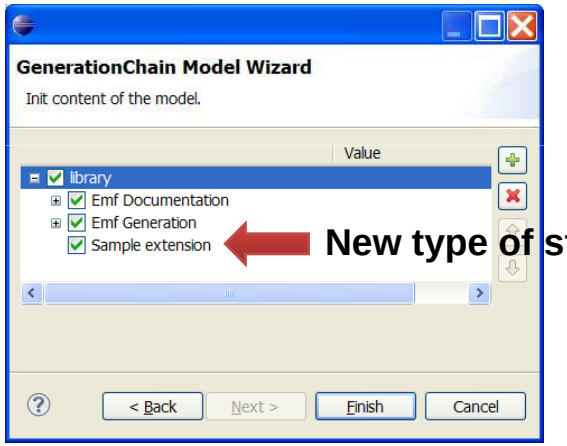
THALES

Principle of Generation Chain Extension

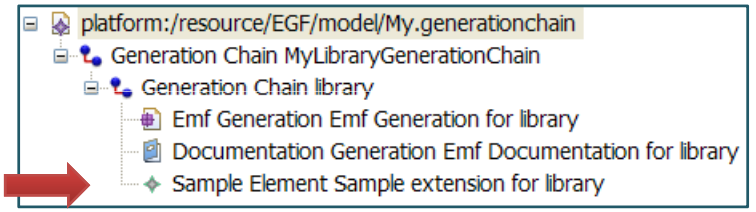
- **Objective**
  - Ability to extend generation chains with new types of generation chain step
  - Introducing new types of generations (e.g. diagram, test generation)
- **Impacts**
  - New step type in the user interface of generation chain creation
  - New step type in the generation chain model
  - Association of a generation to the new type of step



Extension of the creation UI



New type of model element



Generation Chain Extension

```
<plugin>

   <extension point="org.eclipse.emf.ecore.generated_package">
      <package
            uri="http://www.eclipse.org/egf/1.0.0/generationChainSampleExtension"
            class="org.eclipse.egf.portfolio.genchain.extension.SampleExtension.SampleExtensionPackage"
            genModel="model/sampleExtension.genmodel"/>
   </extension>

    <extension point="org.eclipse.egf.portfolio.genchain.elements">
        <helper id="sample.extension" class="org.eclipse.egf.portfolio.genchain.extension.MySampleExtension"/>
    </extension>
  <extension
        point="org.eclipse.egf.core.fcore">
     <fcore
           id="egf/sampleExtension.fcore">
     </fcore>
  </extension>

</plugin>
```

Generation Chain Extension

```
<plugin>

  <extension point="org.eclipse.emf.ecore.generated_package">
     <package
           uri="http://www.eclipse.org/egf/1.0.0/generationChainSampleExtension"
           class="org.eclipse.egf.portfolio.genchain.extension.SampleExtension.SampleExtensionPackage"
           genModel="model/sampleExtension.genmodel"/>
  </extension>

    <extension point="org.eclipse.egf.portfolio.genchain.elements">
        <helper id="sample.extension" class="org.eclipse.egf.portfolio.genchain.extension.MySampleExtension"/>
    </extension>
   <extension
         point="org.eclipse.egf.core.fcore">
      <fcore
            id="egf/sampleExtension.fcore">
      </fcore>
   </extension>

</plugin>
```

Declaration of the model extending the generation chain model

Generation Chain Extension

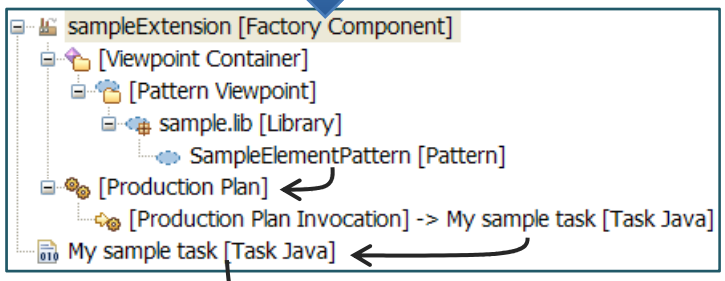
```
<plugin>

   <extension point="org.eclipse.emf.ecore.generated_package">
      <package
            uri="http://www.eclipse.org/egf/1.0.0/generationChainSampleExtension"
            class="org.eclipse.egf.portfolio.genchain.extension.SampleExtension.SampleExtensionPackage"
            genModel="model/sampleExtension.genmodel"/>
   </extension>

    <extension point="org.eclipse.egf.portfolio.genchain.elements">
        <helper id="sample.extension" class="org.eclipse.egf.portfolio.genchain.extension.MySampleExtension"/>
    </extension>
   <extension
         point="org.eclipse.egf.core.fcore">
      <fcore
            id="egf/sampleExtension.fcore">
      </fcore>
   </extension>

</plugin>
```

Factory Component used for the extension



- The "SampleElementPattern" Java pattern declares the behavior to be applied. It invokes the production plan here.

- Production Plan = Generation Behavior

Here, it just applies a simple task implemented by a Java Class.

```
public class MySampleTask implements ITaskProduction {

    public void preExecute(ITaskProductionContext productionContext, IProgressMonitor monitor) throws I

    }

    public void doExecute(ITaskProductionContext productionContext, IProgressMonitor monitor) throws In
        System.out.println("My sample task is executed.");
    }

    public void postExecute(ITaskProductionContext productionContext, IProgressMonitor monitor) throws

    }

}
```

Generation Chain Extension

```
<plugin>

   <extension point="org.eclipse.emf.ecore.generated_package">
      <package
            uri="http://www.eclipse.org/egf/1.0.0/generationChainSampleExtension"
            class="org.eclipse.egf.portfolio.genchain.extension.SampleExtension.SampleExtensionPackage"
            genModel="model/sampleExtension.genmodel"/>
   </extension>

   <extension point="org.eclipse.egf.portfolio.genchain.elements">
       <helper id="sample.extension" class="org.eclipse.egf.portfolio.genchain.extension.MySampleExtension"/>
   </extension>
  <extension
        point="org.eclipse.egf.core.fcore">
     <fcore
           id="egf/sampleExtension.fcore">
```

Java Class for extension

```
public class MySampleExtension extends ExtensionHelper {
    private static final URI PATTERN =
        URI.createPlatformPluginURI("org.eclipse.egf.portfolio.genchain.extension/egf/sampleExtension.fcore#_fMAHcKYjEd-c68Bv_MO43Q",

    @Override
    public String getLabel() {

        return "Sample extension";
    }

    @Override
    public List<Substitution> getSubstitutions() {
        EGFResourceSet set = new EGFResourceSet();
        List<Substitution> substitutions = new ArrayList<Substitution>();
        final Substitution substitution = PatternFactory.eINSTANCE.createSubstitution();
        final Pattern pattern = (Pattern) set.getEObject(PATTERN, true);
        substitution.getReplacement().add(pattern);
        substitutions.add(substitution);
        return substitutions;
    }

    @Override
    public EcoreElement createEcoreElement(Map<String, String> properties) {

        final SampleElement sampleElement = SampleExtensionFactory.eINSTANCE.createSampleElement();
        String modelPath = properties.get(MODEL_PATH);
        sampleElement.setModelPath(modelPath);
        return sampleElement;
    }
}
```

Id declaration of the pattern explained in the previous slide

Label for instance used in the creation user interface

The sequence to declare that the pattern is applied over a generation chain model when the generation chain is executed

Model action to be applied when a new step of this type is created

THALES