www.thalesgroup.com

EGF

Eclipse Generation Factories

Introduction to EGF

Benoît Langlois / Thales Global Services

THALES

Agenda

Introduction

EGF Architecture

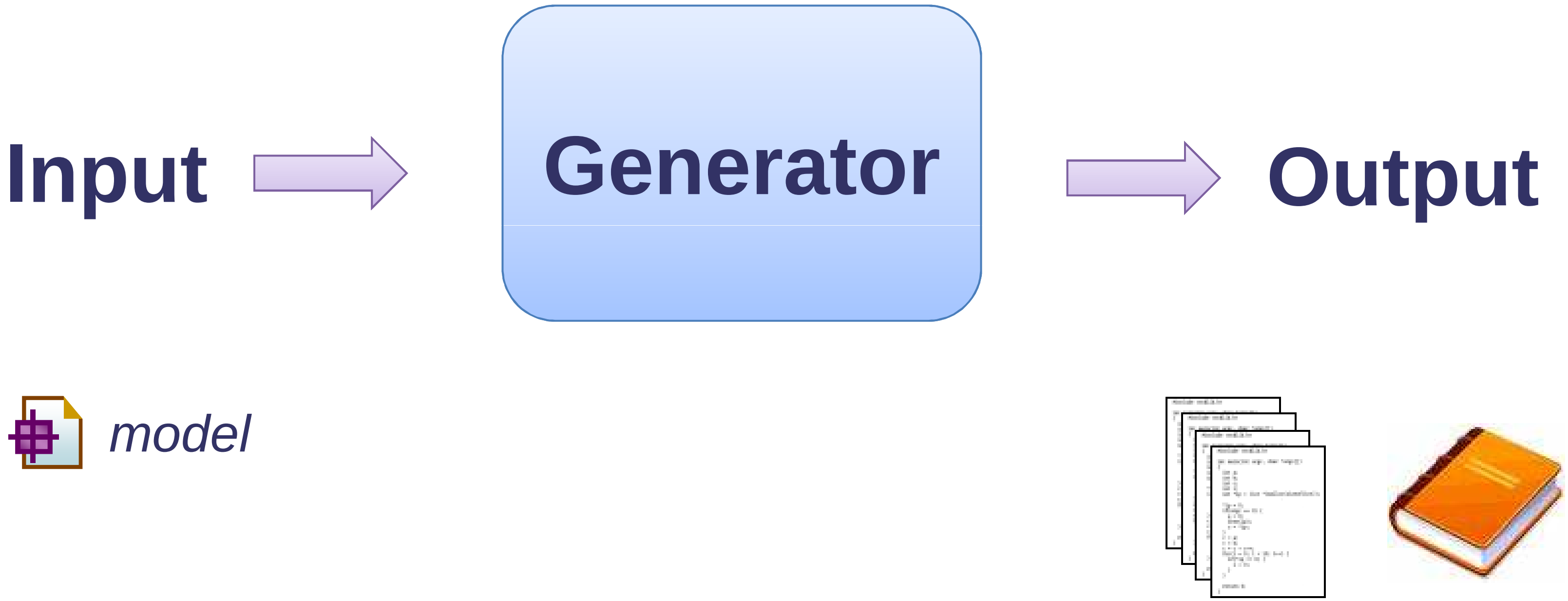
Concepts & Practice

EGF Portfolios

Agenda

- Introduction
- EGF Architecture
- Concepts & Practice
- EGF Portfolios

Generation, Basically…



Input

Generator

Output

model

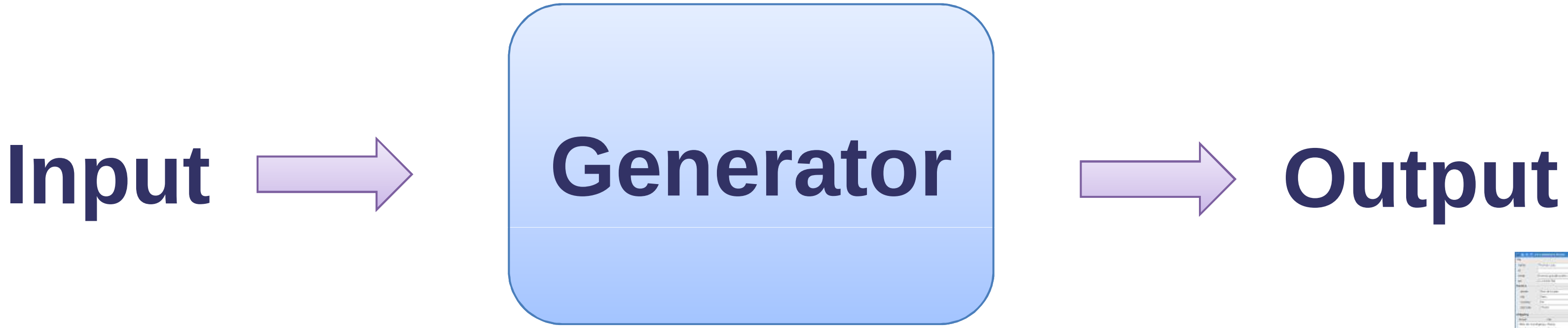
But...

Integration of heterogeneous kinds of know-how

Different types of input

Orchestration

Different types of output



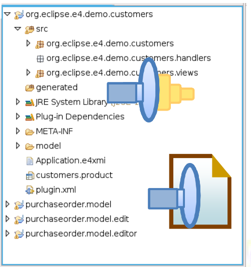
- *model*
- *File*
- *Plug-in*
- …

Different languages & Tools

Java, Ant, Ruby, Acceleo, Jython

Framework

Variability



```
▽ org.eclipse.e4.demo.customers
  ▽ src
    ▷ org.eclipse.e4.demo.customers
      org.eclipse.e4.demo.customers.handlers
    ▷ org.eclipse.e4.demo.customers.views
    generated
  ▷ JRE System Library
  ▷ Plug-in Dependencies
  ▷ META-INF
  ▷ model
    Application.e4xmi
    customers.product
    plugin.xml
▷ purchaseorder.model
▷ purchaseorder.model.edit
▷ purchaseorder.model.editor
```

Generative Software

Generation scope?

How to develop & test?

Executability? Distribution?

Generation reusability?

Generation customization?

What target-platform?

Variability? Product lines?

Performance, scalability?

Generation orchestration?

Generation workflow?

One-click generation solution?

Generation data, which ones, where?

Best practices, guidance?

Combining [model|text|dsl]-to-[Model|text|dsl]?

Update strategy of the produced artifacts?

Multiplicity of languages and engines?

Merging Generation?

Integration of a new language?

How to deal with Generations Issues?
What are the Drivers?

Modèle presentation_epm version 1.0

THALES

Dealing with Complex Generations

EGF

Eclipse Generation Factories

A Software Factory Tool, An integrated and extensible Generation Framework

Modèle presentation_epm version 1.0

THALES

What is EGF?

- EGF (Eclipse Generation Factories) is a software factory tool with the purpose to generate software artifacts, such as code or application
- It is an Eclipse open source component project in incubation under the EMFT project
- Objectives:
  - Definition and execution of software factories
  - Orchestration of software generation activities
  - Promotion of software factory portfolios
  - Extensibility of the EGF framework in order to support new generation formalisms and functions

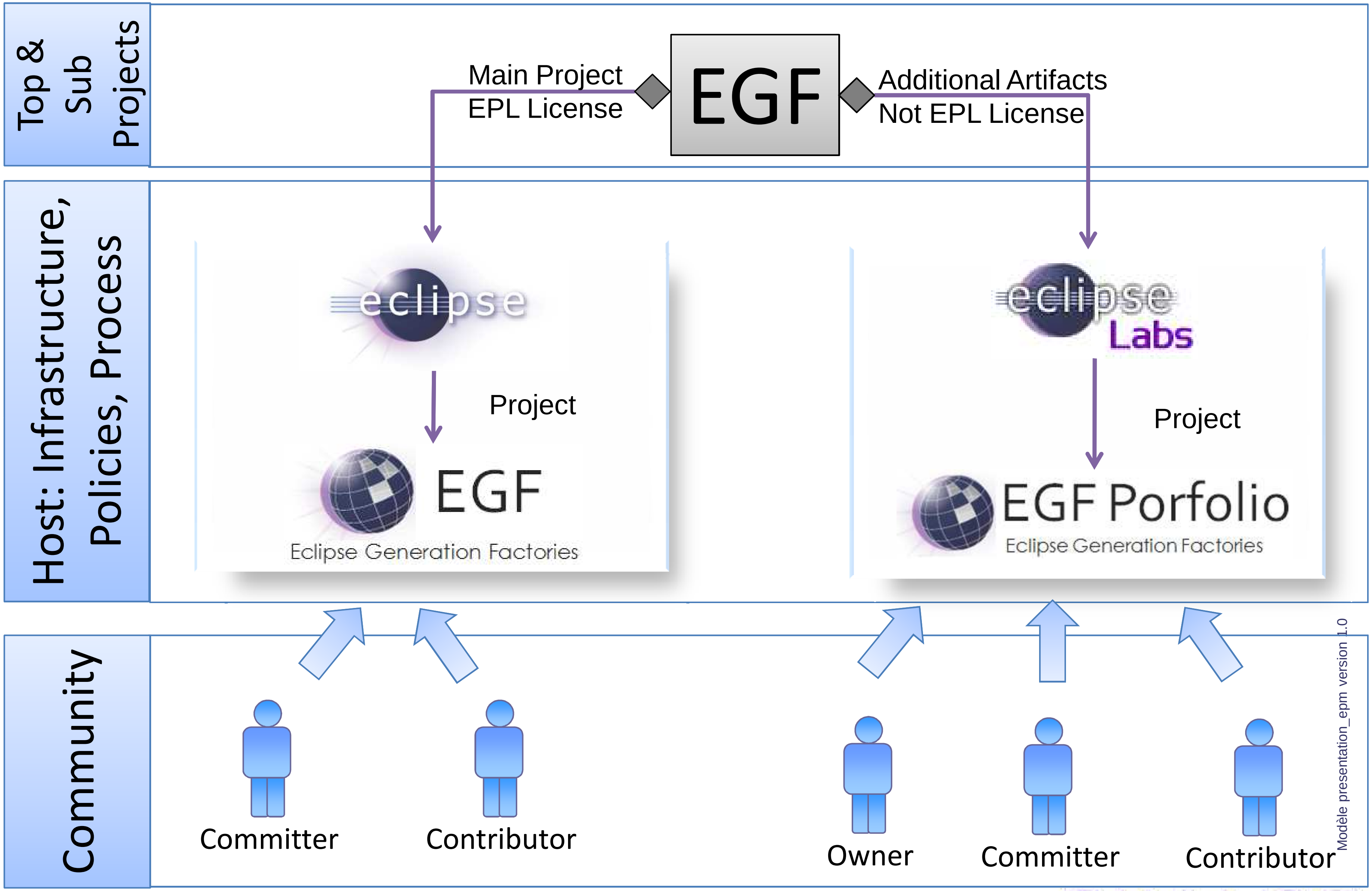
General info about EGF

EGF
Eclipse Generation Factories

Project page: http://www.eclipse.org/egf
Wiki: http://wiki.eclipse.org/EGF

Download: by update site
http://wiki.eclipse.org/EGF_Installation

Project Organization



THALES

Agenda

- Introduction
- EGF Architecture
- Concepts & Practice
- EGF Portfolios

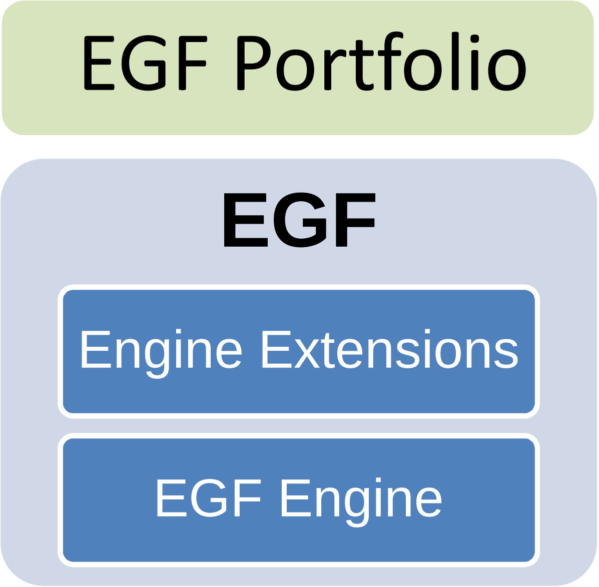
Modèle presentation_epm version 1.0

THALES

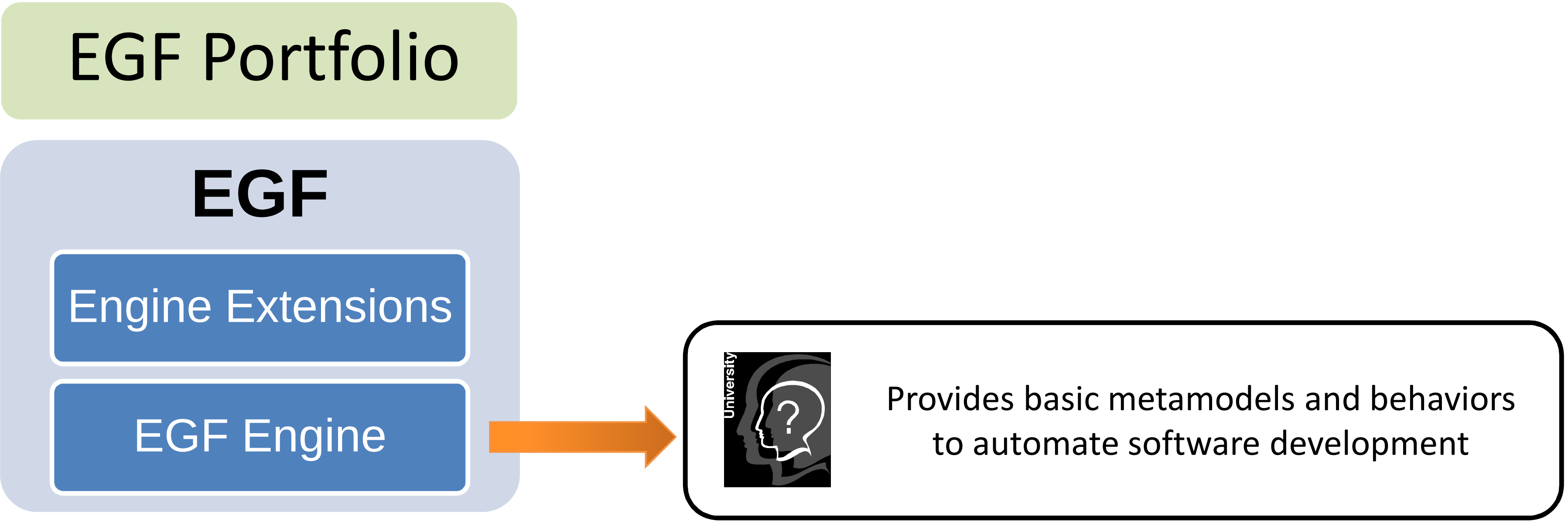
Agenda

- Introduction
- EGF Architecture
  - Architecture
  - Some issues addressed by EGF
- Concepts & Practice
- EGF Portfolios

EGF Architecture



EGF Architecture



Provides basic metamodels and behaviors to automate software development

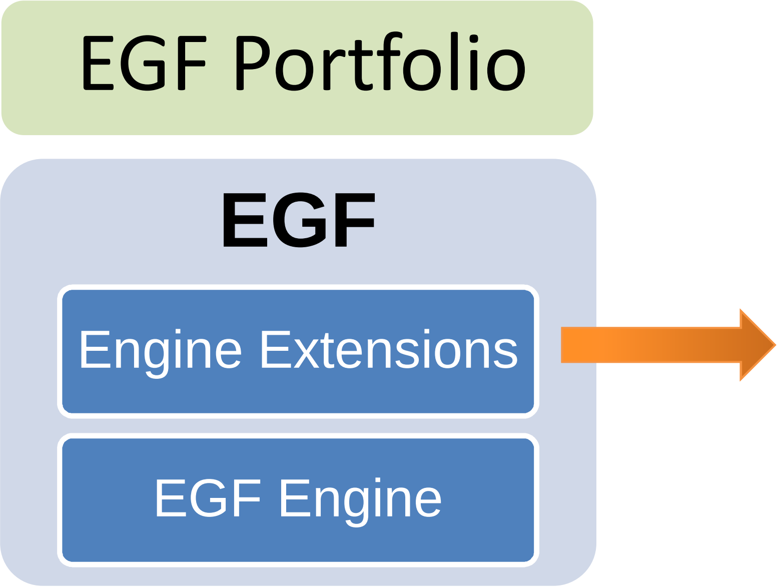
EGF Metamodel

Basic behaviors, dynamic execution

Factory component, task

Pattern

EGF Architecture

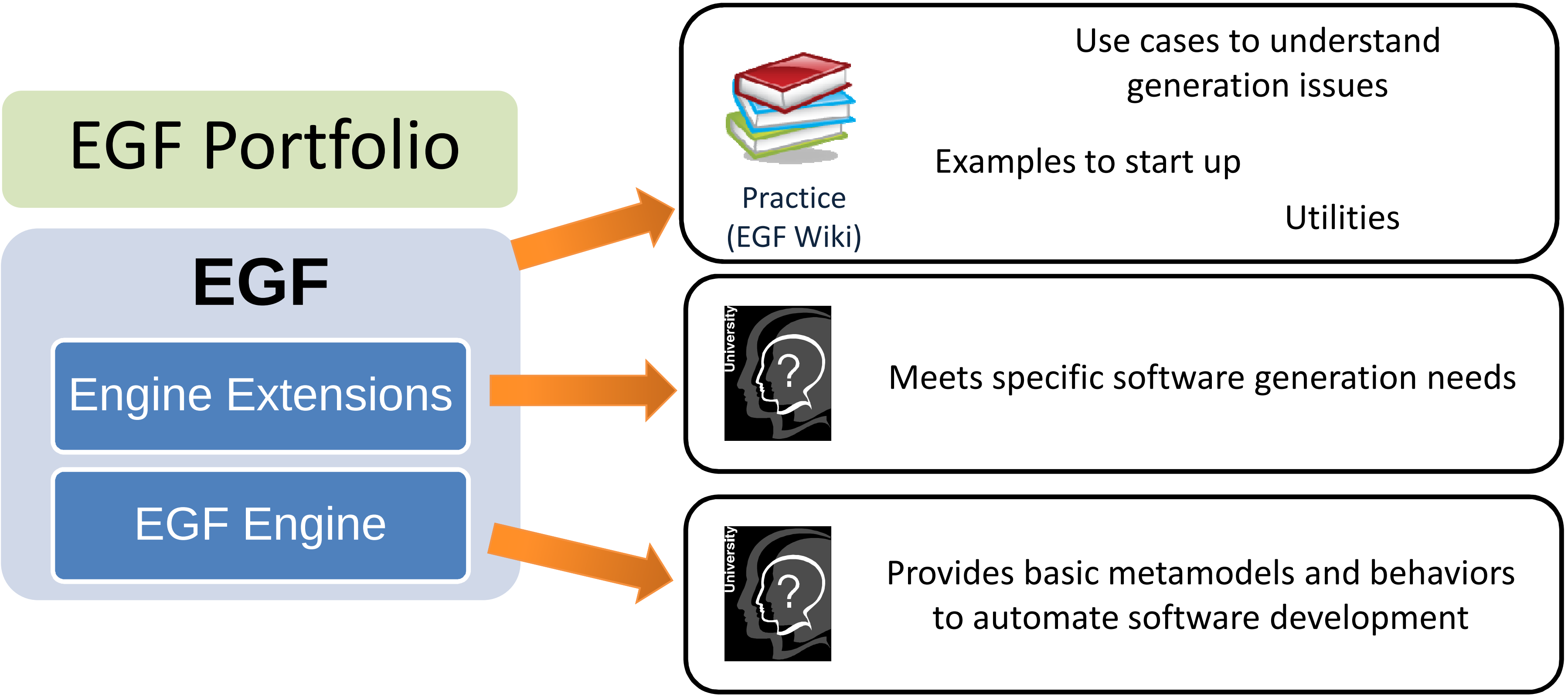


Meets specific software generation needs

Language & tools interoperability

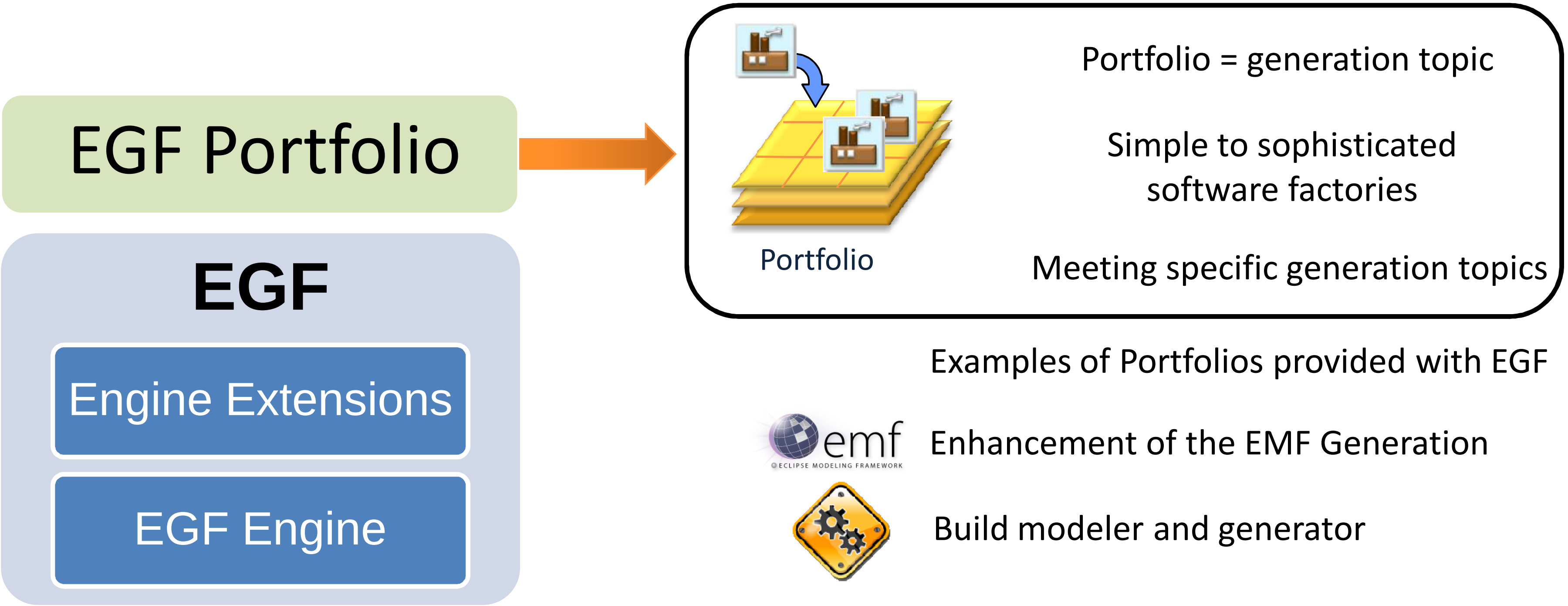
New generation formalisms

EGF Architecture



EGF Portfolio
EGF
Engine Extensions
EGF Engine
Practice
(EGF Wiki)
Use cases to understand generation issues
Examples to start up
Utilities
University
Meets specific software generation needs
University
Provides basic metamodels and behaviors to automate software development

EGF Architecture



Portfolio = generation topic

Simple to sophisticated software factories

Meeting specific generation topics

Examples of Portfolios provided with EGF

- Enhancement of the EMF Generation
- Build modeler and generator

Agenda

Introduction

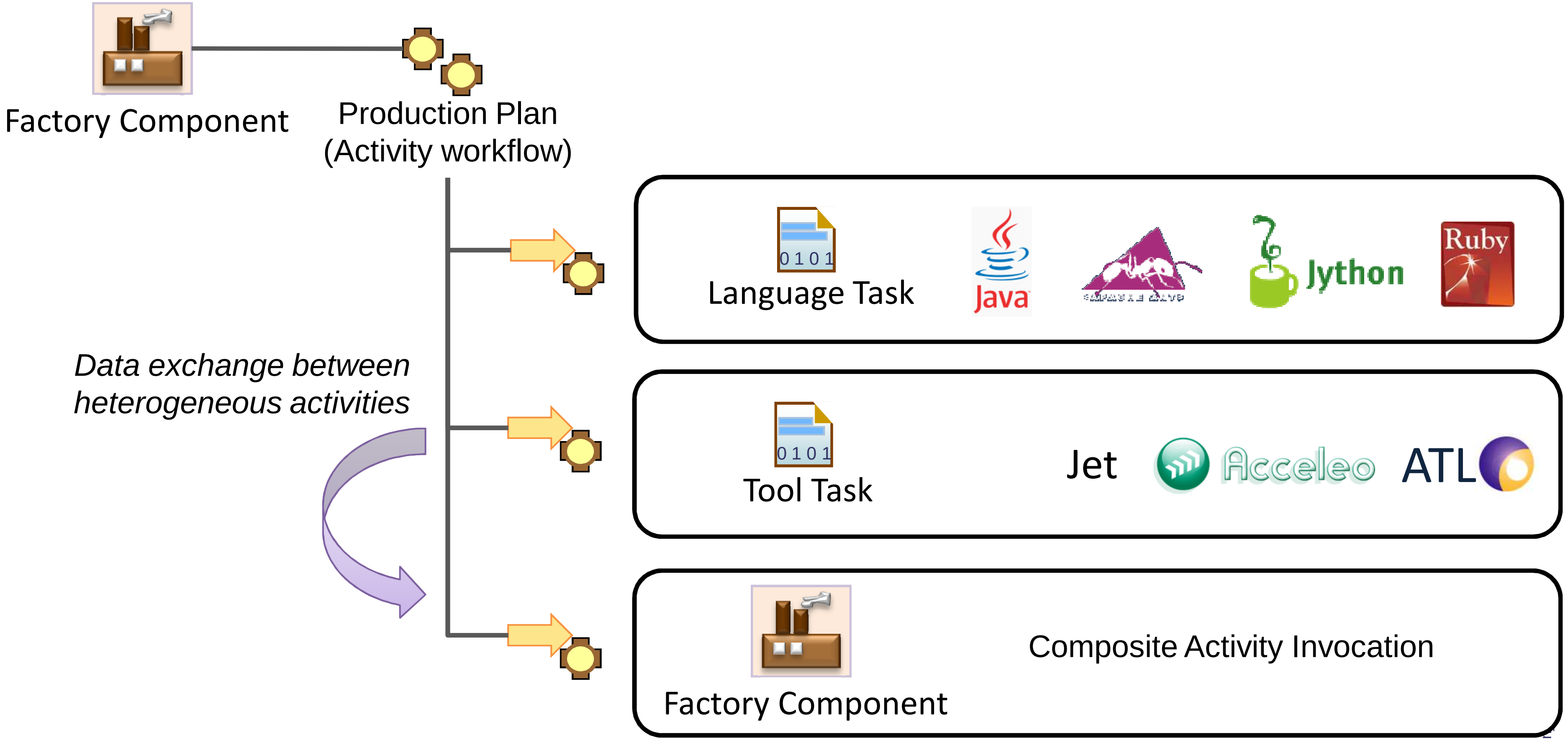
EGF Architecture

- Architecture
- Some issues addressed by EGF

Concepts & Practice

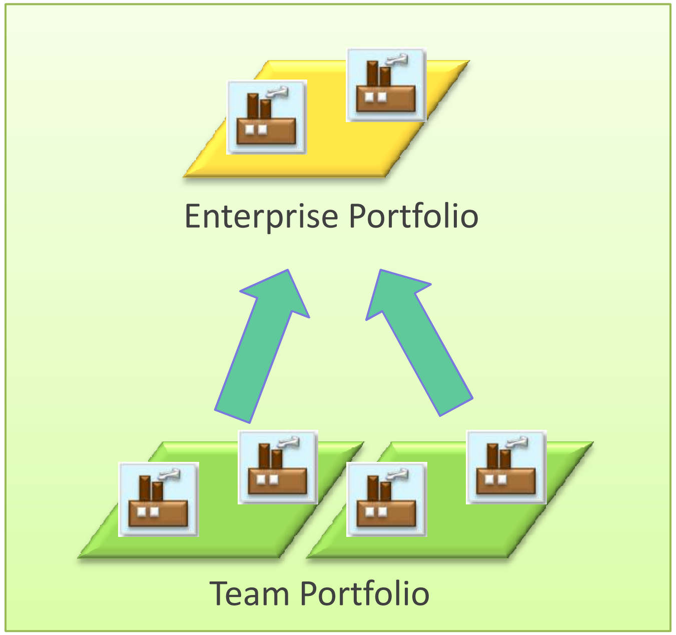
EGF Portfolios

Generation Composition, Language and Tool Interoperability



Activity Workflow with Java and Ruby: http://vimeo.com/15705526

Code Generation Families



Several levels of Customization

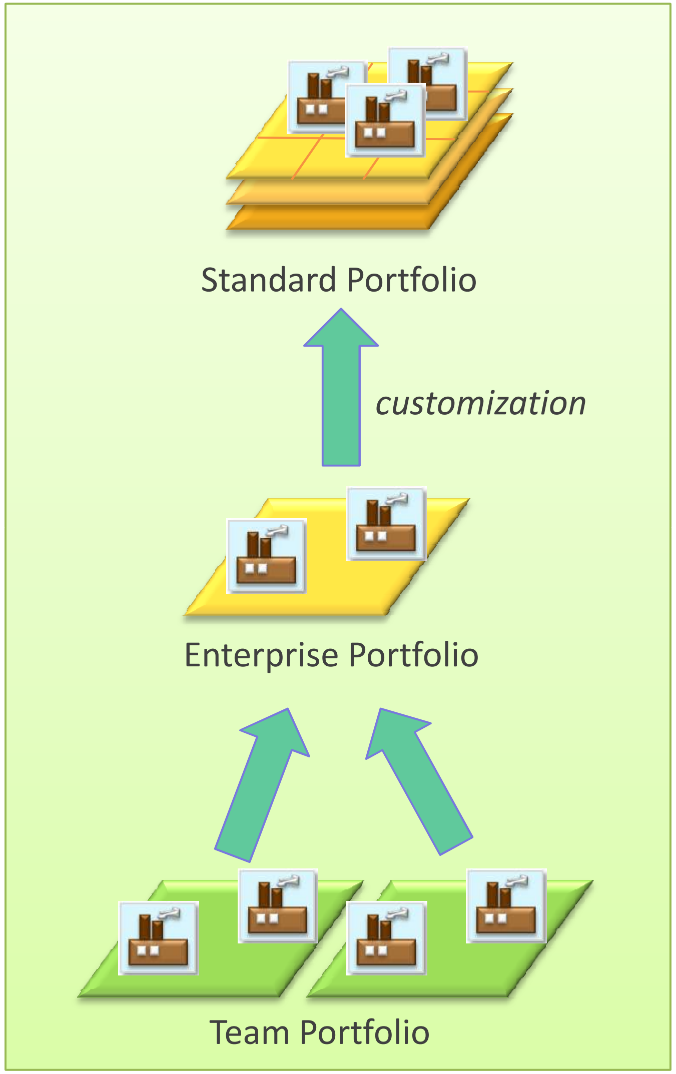
Example

Code/textual generation for my organization

extends

Specific generation for my project

Code Generation Families



Several levels of Customization

Example

EMF Generation

↑ *extends*

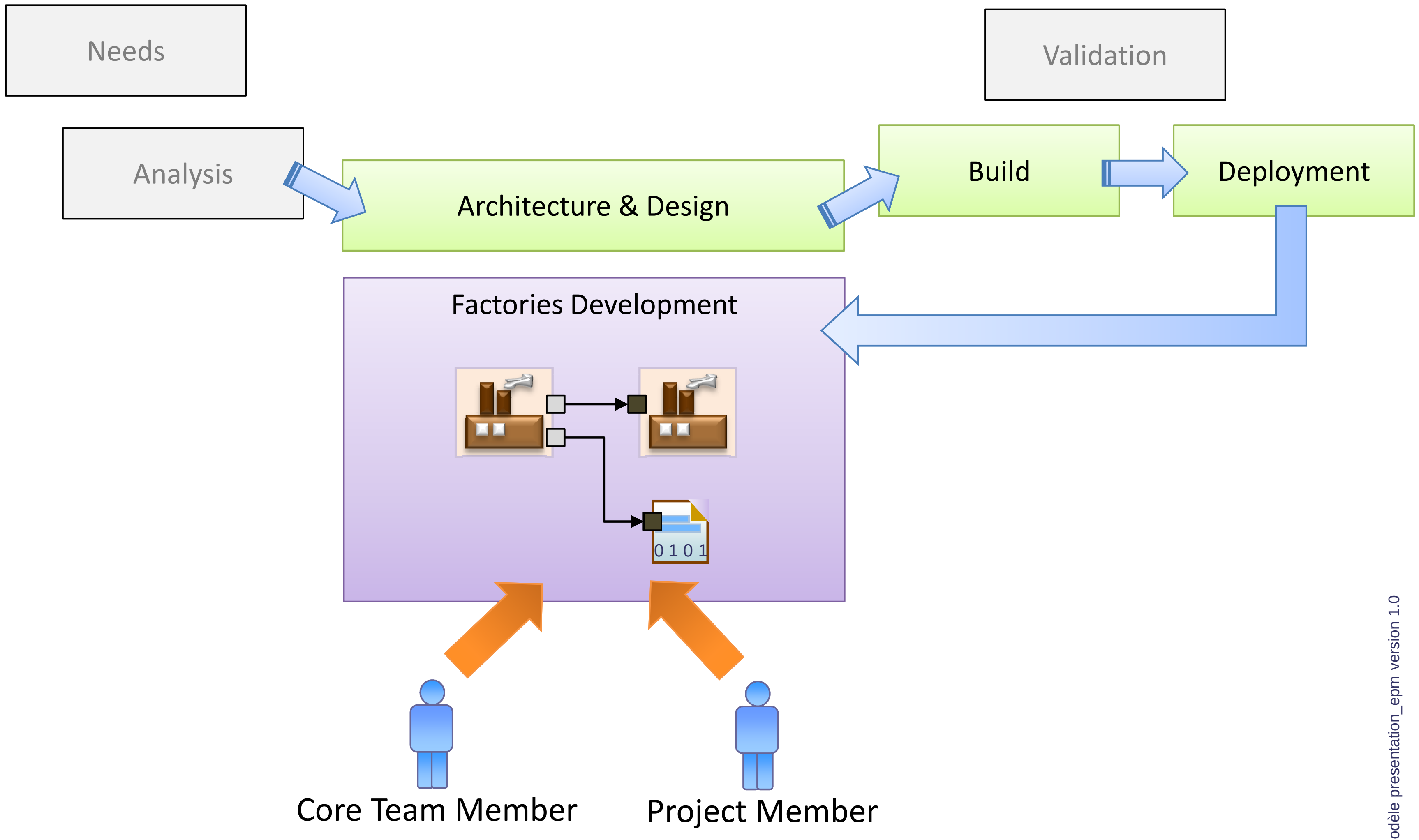
EMF Generation
for my organization

↑ *extends*

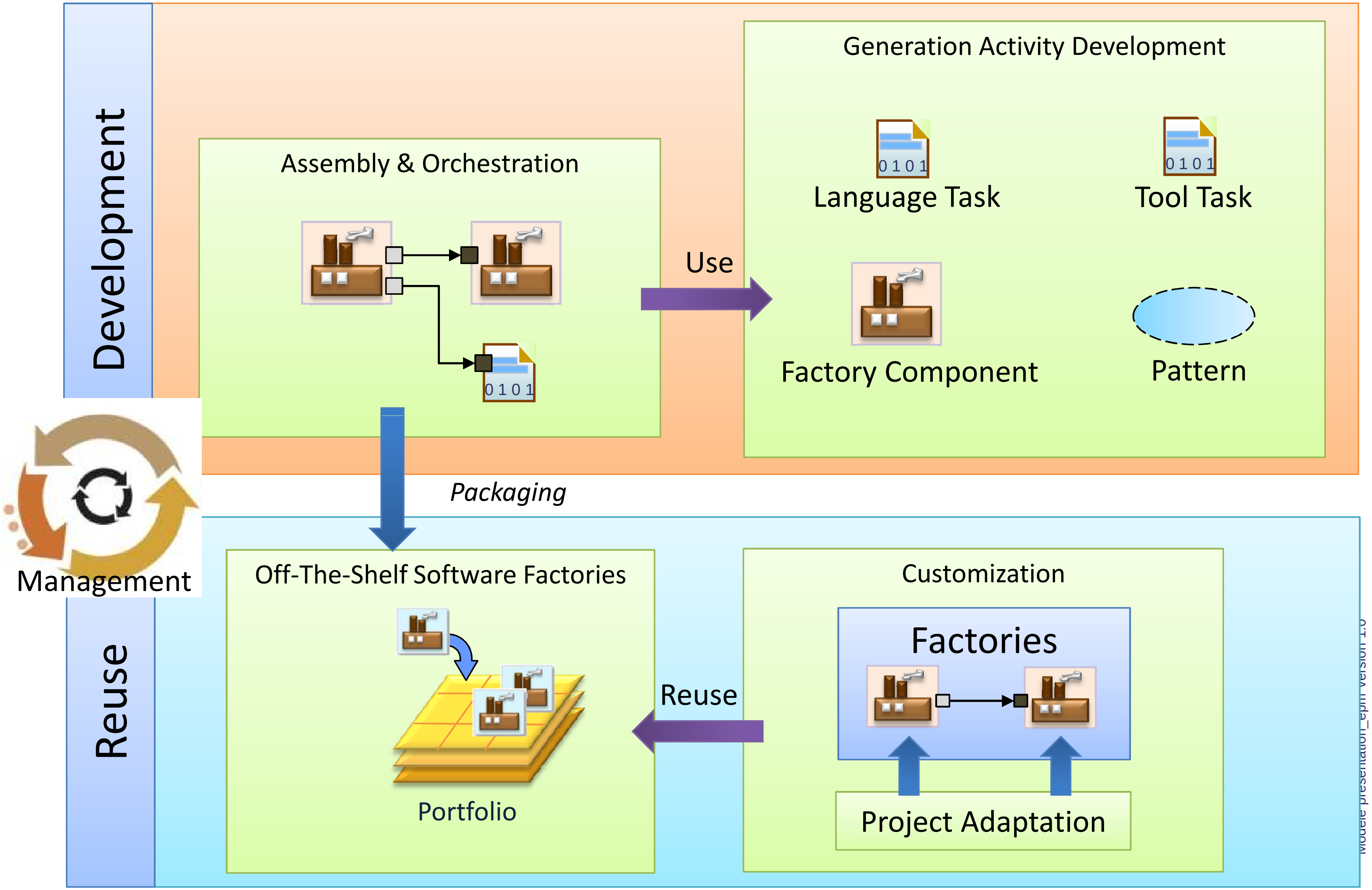
EMF Generation
for my application

THALES

Generation Development in Multi-User Mode

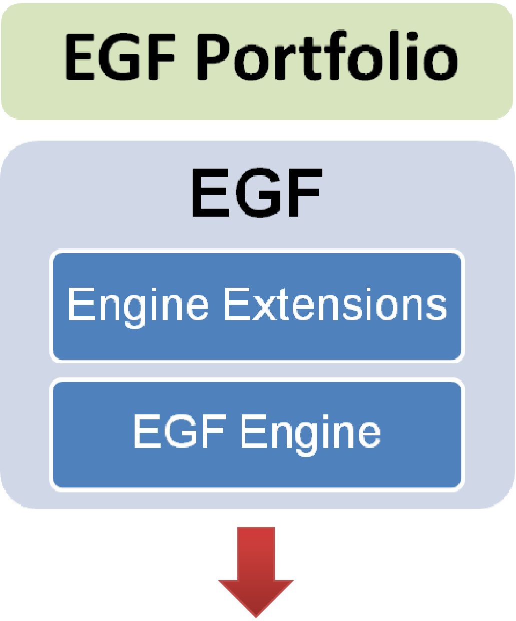


Software Factory Engineering



Software Factory Engineering

General Architecture of EGF



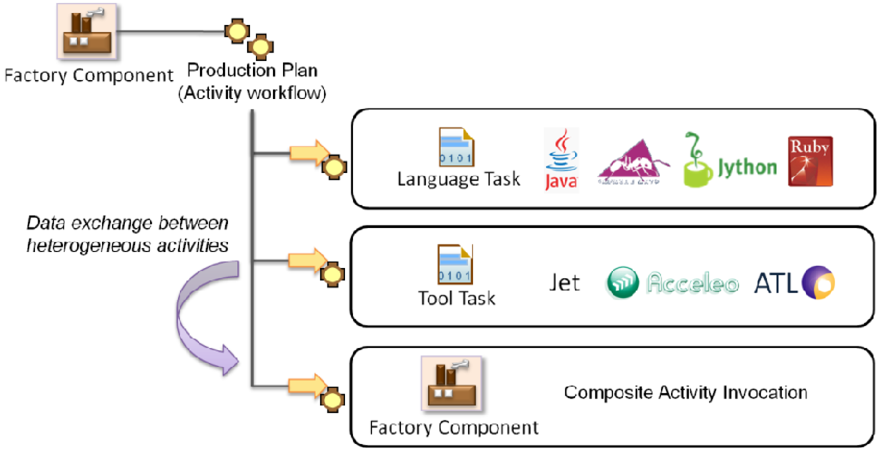
EGF Portfolio

EGF

Engine Extensions

EGF Engine

Example of EGF Factory



Factory Component

Production Plan
(Activity workflow)

Data exchange between heterogeneous activities

Language Task

Java

Jython

Ruby

Tool Task

Jet

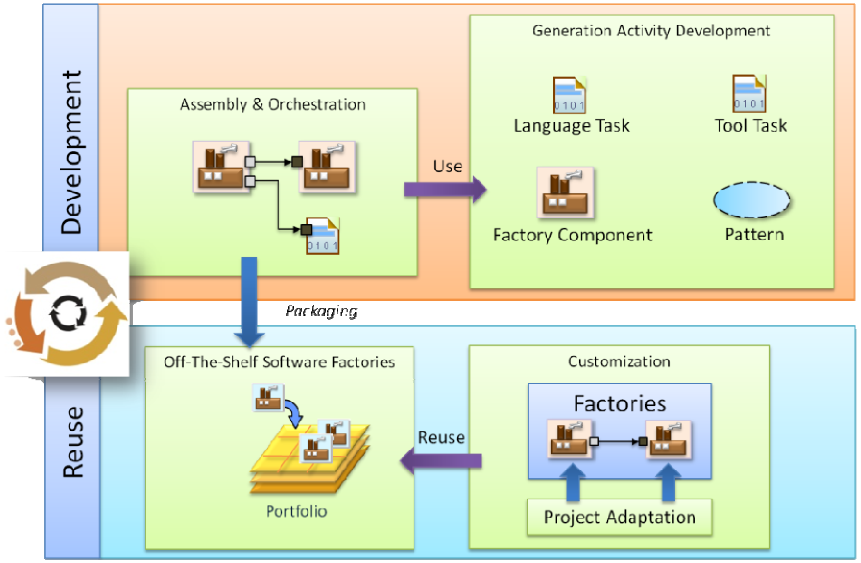
Acceleo

ATL

Factory Component

Composite Activity Invocation

Development & Reuse Process with EGF



Development

Assembly & Orchestration

Use

Generation Activity Development

Language Task

Tool Task

Factory Component

Pattern

Packaging

Reuse

Off-The-Shelf Software Factories

Portfolio

Reuse

Customization

Factories

Project Adaptation

THALES

Agenda

- Introduction
- EGF Architecture
- Concepts & Practice
- EGF Portfolios

Modèle presentation_epm version 1.0

Installation of EGF

Installation by update site

- Eclipse Update site
- Update site from Amalgam
  - [Eclipse] Help / Install Modeling Components / EGF

Download EGF materials

- Download EGF update site, dropins, examples
- Location: http://wiki.eclipse.org/EGF_Installation

Examples and Portfolio

Installation of the Examples

- Install the examples File/New/Example…/EGF
- A plug-in example contains a set of generation use cases on a specific topic

Presentation of the EGF Portfolios

- Contrarily to the examples, a portfolio is an operational solution
- http://wiki.eclipse.org/EGF/Portfolio

Agenda

Introduction

EGF Architecture

Concepts & Practice

- Generation Chain
- Activity
- Factory Component
- Task
- Pattern

EGF Portfolios

Types of Generation Units in EGF

Factory Component
Composite generation unit with an activity workflow

Task
Leaf generation unit to execute a tool (e.g., ATL, Acceleo) or code written in a language (e.g., Java, Ruby, Ant)

Portfolio
Set of factories to capitalize on a specific generation topic

Generation Chain
High generation view to organize complex generations

EGF Pattern
- Description of systematic behavior
- For definition of code generation families

Agenda

Introduction

EGF Architecture

Concepts & Practice

- Generation Chain
- Activity
- Factory Component
- Task
- Pattern

EGF Portfolios

Generation Chain - Objectives

Objectives:

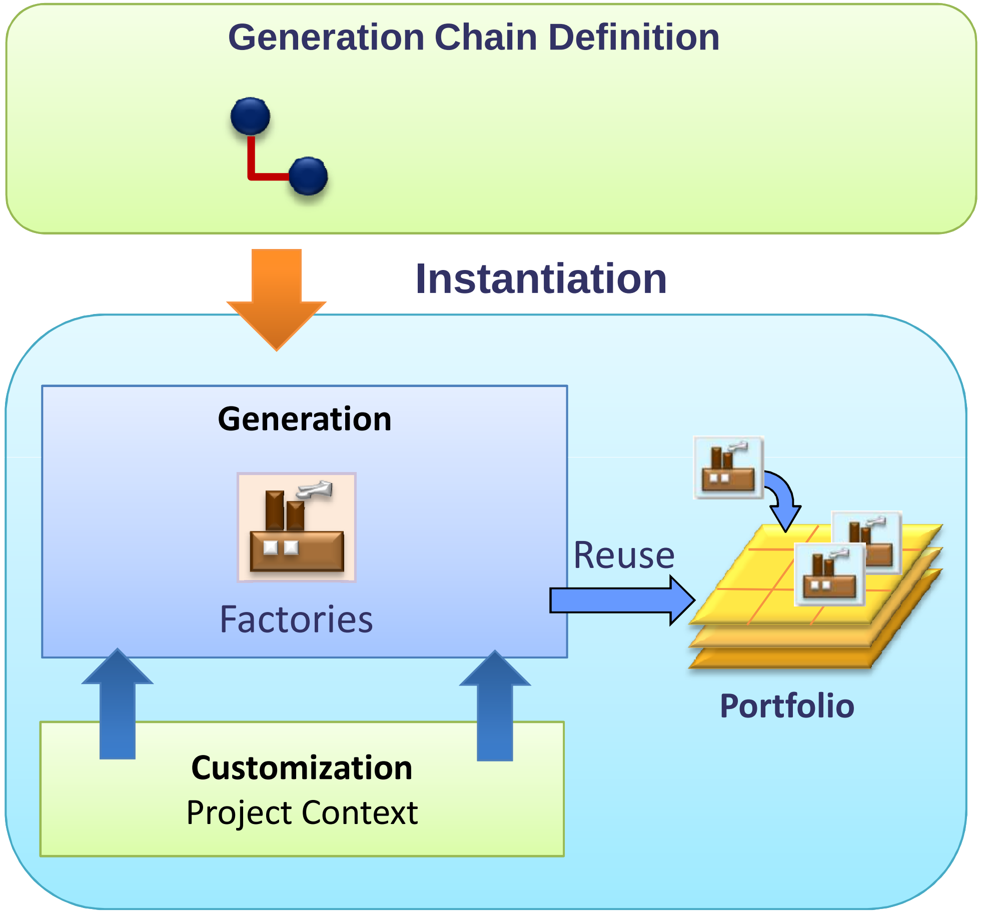
- Definition, at a high level of description, of executable generations
- Abstraction: encapsulation of the irrelevant technical details of generation. Only the features of a generation step are visible.
- Simplicity & Efficiency
  - Set generation features and next generate

Generation Chain - Principles

Technical principles:

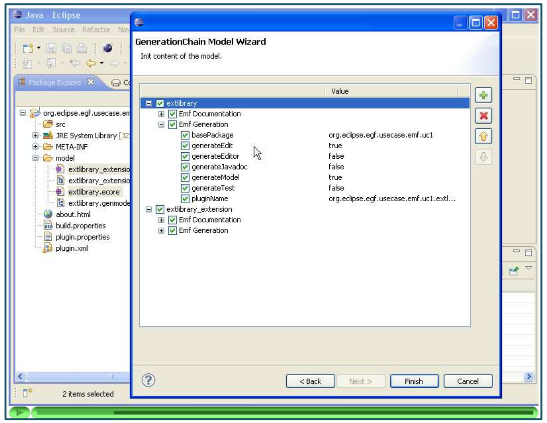
- **Storage: in a “generation chain” file**
- **Technical details:**
  - An EGF fcore file is produced from the generation chain: it contains the translation of the generation chain into factory components
  - Next, the factory components are transparently executed to produce the expected artifacts
  - It is possible to add customization later a generation with generation chains at the factory component level

Instantiation & Customization of Generation Chains



Modèle presentation_epm version 1.0

[Practice] Hands On…



Java - Eclipse

GenerationChain Model Wizard

Init content of the model.

	Value
☑ extlibrary	
☑ Emf Documentation	
☑ Emf Generation	
☑ basePackage	org.eclipse.egf.usecase.emf.uc1
☑ generateEdit	true
☑ generateEditor	false
☑ generateJavadoc	false
☑ generateModel	true
☑ generateTest	false
☑ pluginName	org.eclipse.egf.usecase.emf.uc1.extl…
☑ extlibrary_extension	
☑ Emf Documentation	
☑ Emf Generation	

2 items selected

Links:

Video: http://vimeo.com/29472598

For more explanations:

Generation Chain Tutorial

http://wiki.eclipse.org/EGF_Tutorial_and_Use_Cases

Exercice:

1. Select ecore models
2. File/New/Other…/[EGF] Generation Chain Model
3. Set the generation parameters
4. Right click on the first Generation Chain node / Run Generation Chain
5. After execution, open the fcore file in a created plug-in in order to understand how the generation is realized

Agenda

Introduction

EGF Architecture

Concepts & Practice

- Generation Chain
- Activity
- Factory Component
- Task
- Pattern

EGF Portfolios

Activity – Definition

An activity is the abstract class of EGF generation units

- Factory component and Task are activities

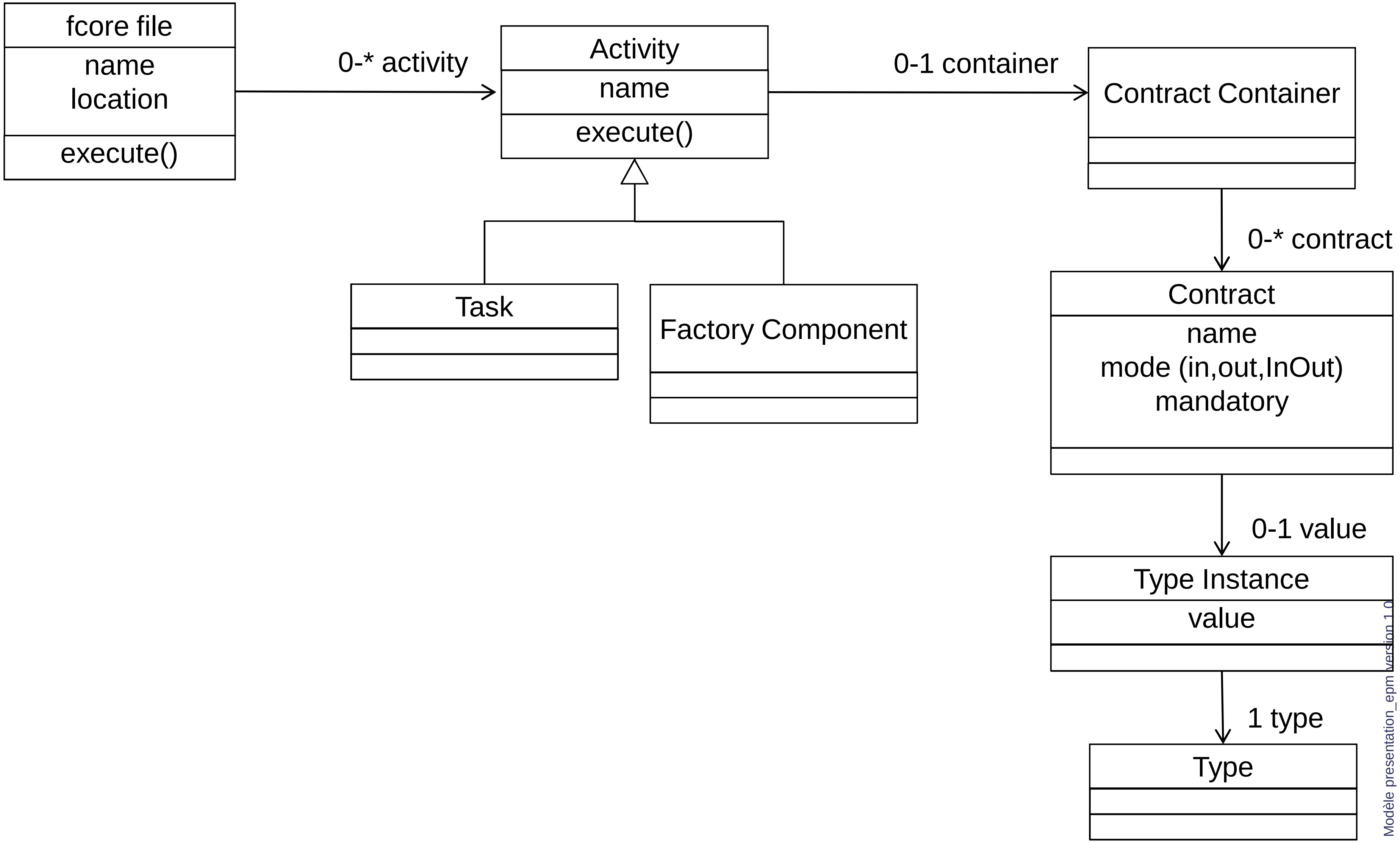
Activity storage

- Activities are stored in *fcore* files
- The same fcore file contains one or several activities

Activity properties

- Contract declaration (= parameters)

Concepts


fcore file
name
location
execute()
0-* activity
Activity
name
execute()
0-1 container
Contract Container
Task
Factory Component
0-* contract
Contract
name
mode (in,out,InOut)
mandatory
0-1 value
Type Instance
value
1 type
Type

Agenda

Introduction

EGF Architecture

Concepts & Practice

- Generation Chain
- Activity
- Factory Component
- Task
- Pattern

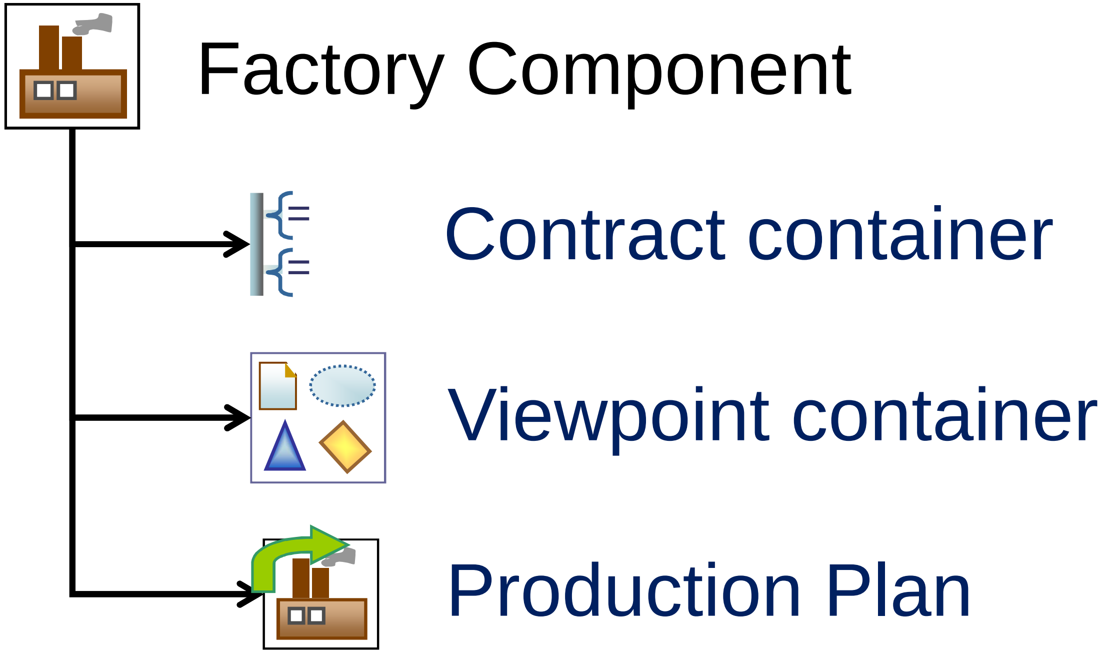
EGF Portfolios

Modèle presentation_epm version 1.0

Factory Component

- Unit of generation with a clear objective of generation
- Unit of generation with a clear contract
- Viewpoint: it contains the declaration of data used during the generation
- Production plan: it contains the orchestration of generation activities
- Factory Component Lifecycle: edition and execution

Factory Component Structure



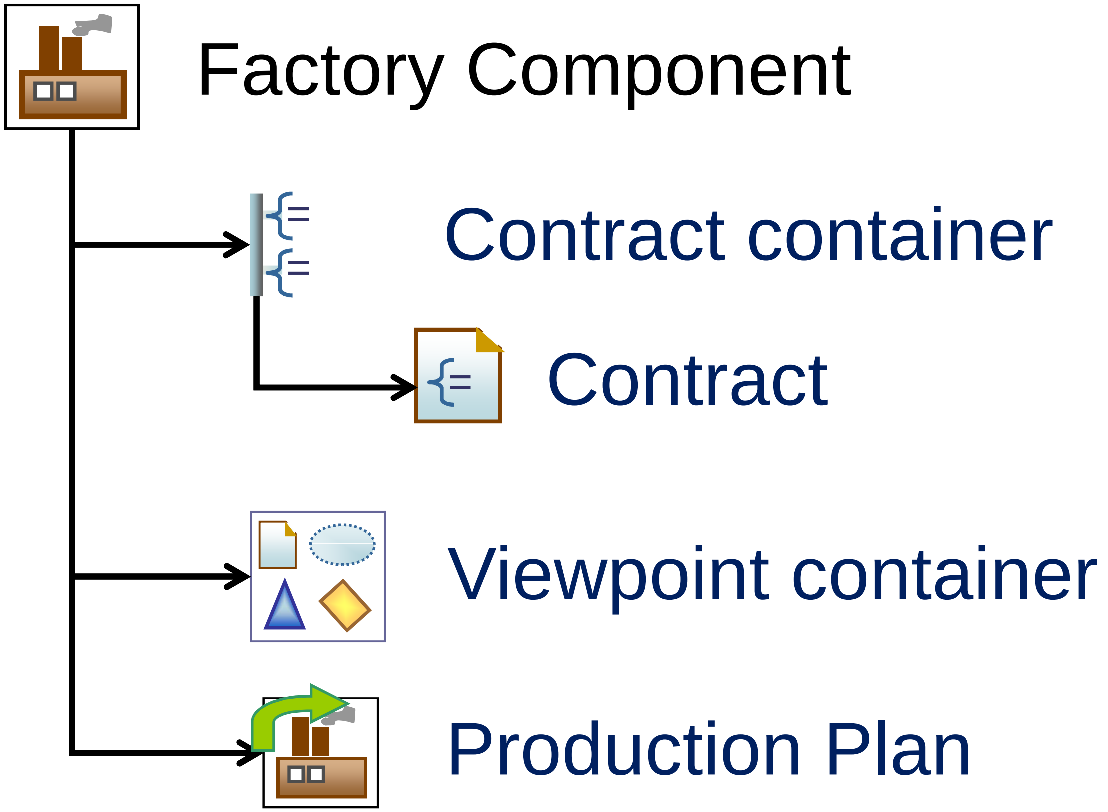
Factory Component

Contract container

Viewpoint container

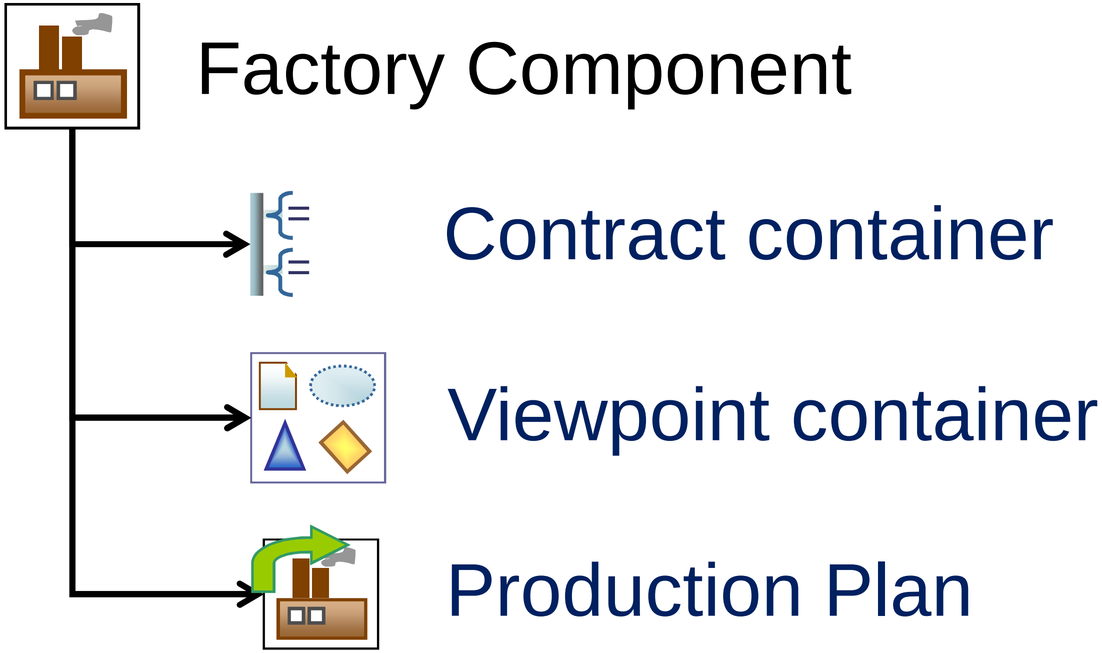
Production Plan

Factory Component Structure



- **Contract:** Factory component parameter
- A contract has a type, a passing mode (In/Out/In_Out), a default value or not, is mandatory or optional

Factory Component Structure



Factory Component

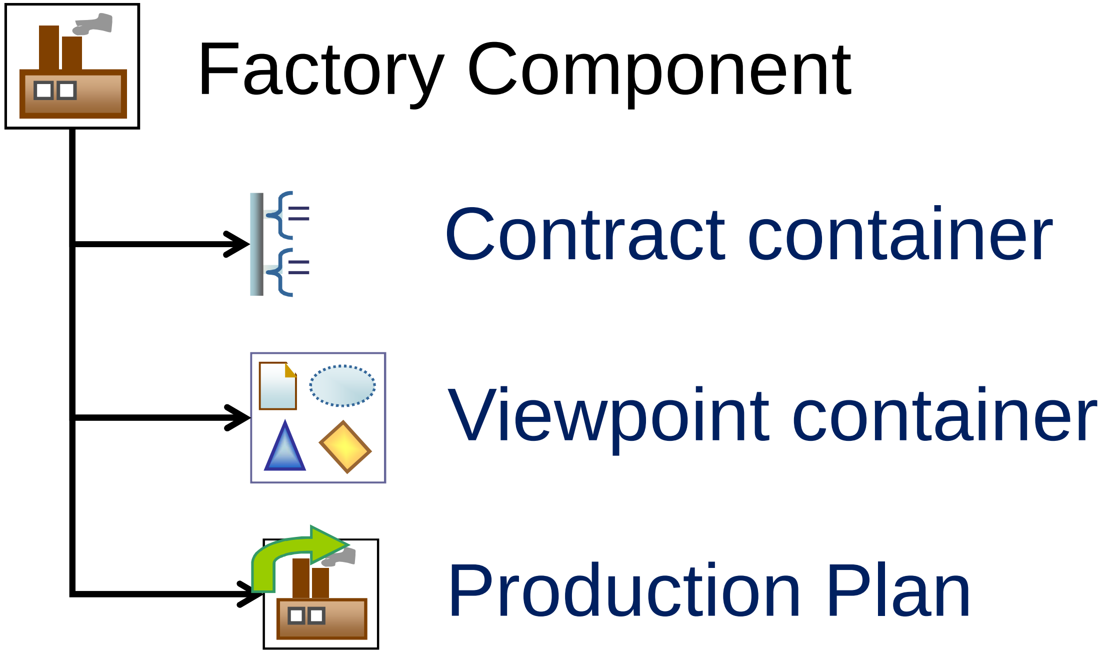
Contract container

Viewpoint container

Production Plan

- **Viewpoint:** area to declare concerns of generation data
- Examples of viewpoint:
  - Available today: domain declaration, pattern
  - Candidates: licensing, feature model

Factory Component Structure



Factory Component

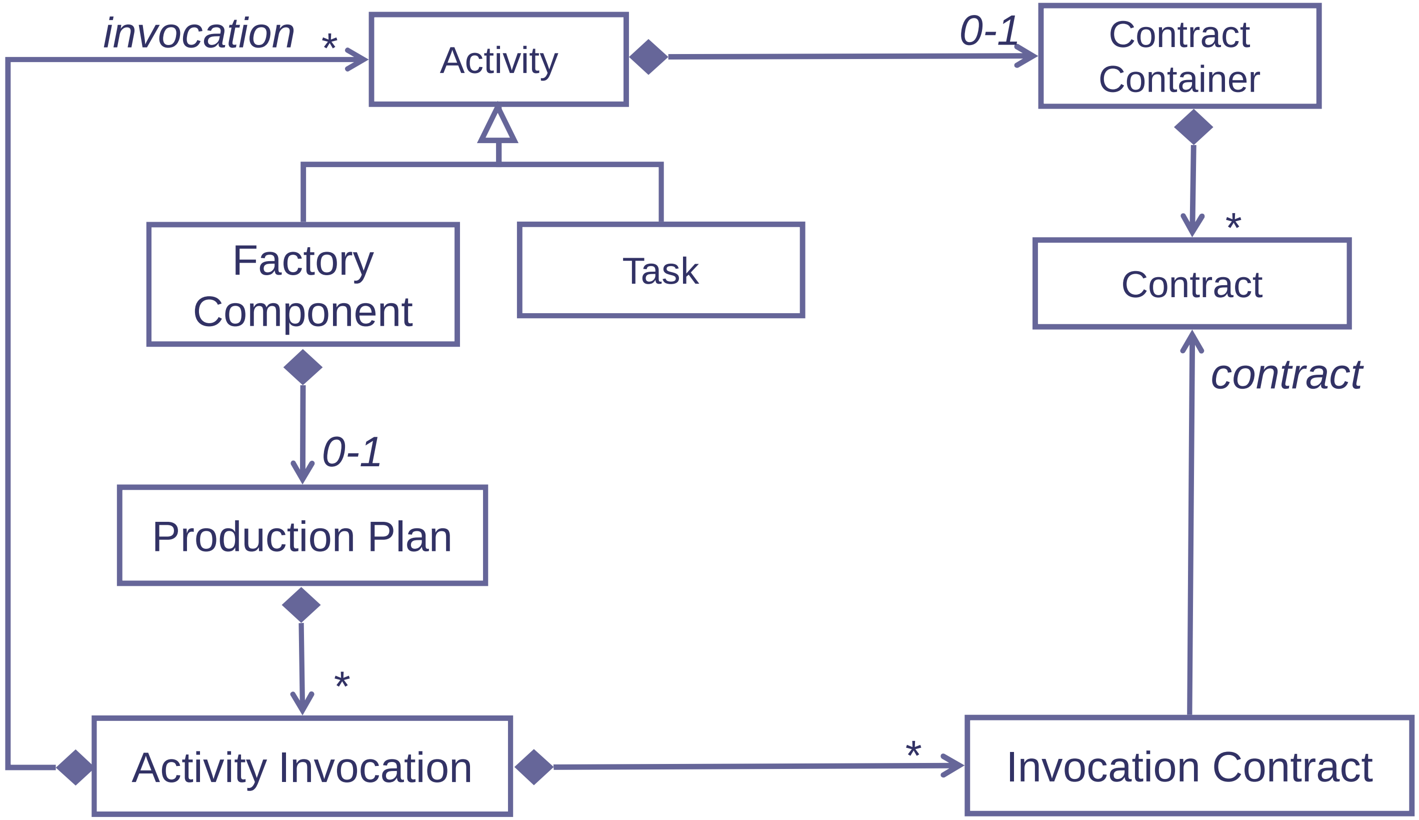
Contract container

Viewpoint container

Production Plan

- **Production Plan**: workflow to describe generation steps – Sequential today

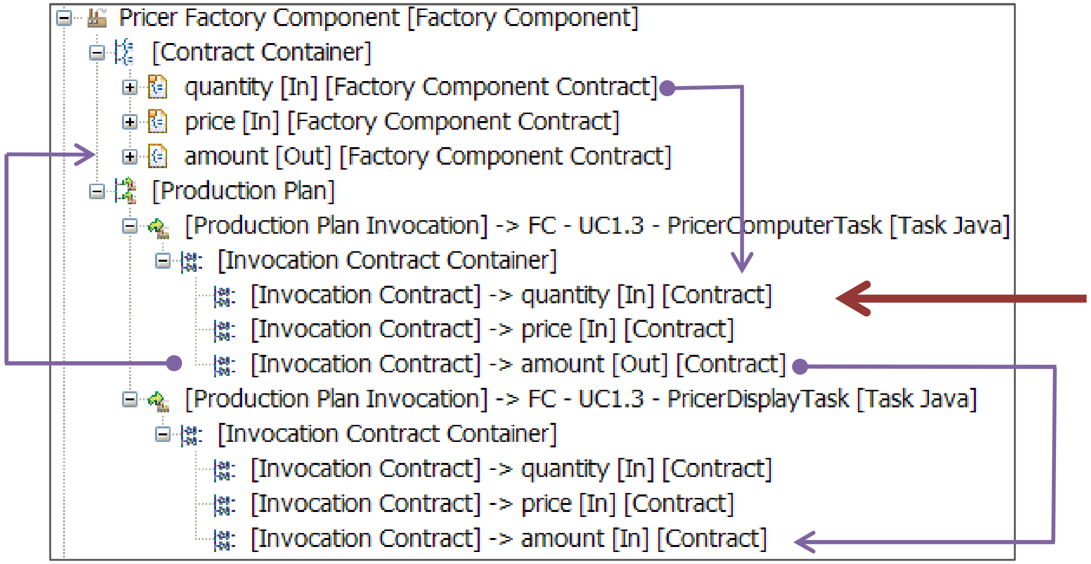
Activity and Production Plan



Modèle presentation_epm version 1.0

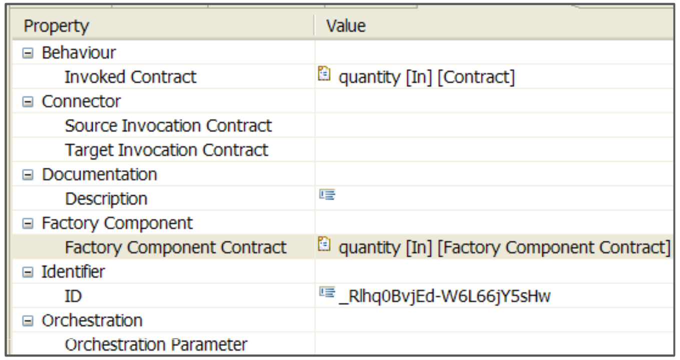
THALES

In/Output relationships – Example



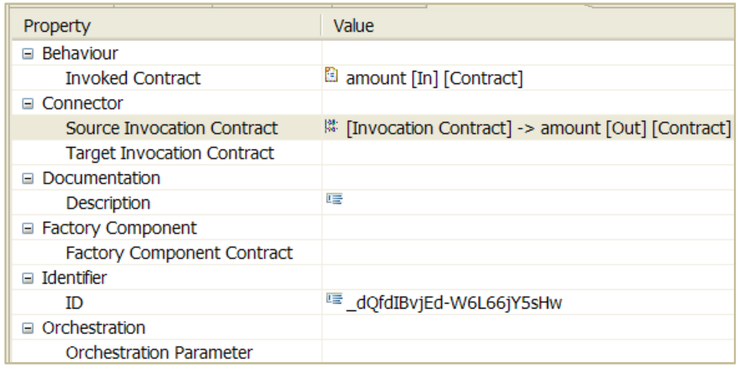
```
⊟ Pricer Factory Component [Factory Component]
   ⊟ [Contract Container]
      ⊞ quantity [In] [Factory Component Contract]
      ⊞ price [In] [Factory Component Contract]
      ⊞ amount [Out] [Factory Component Contract]
   ⊟ [Production Plan]
      ⊟ [Production Plan Invocation] -> FC - UC1.3 - PricerComputerTask [Task Java]
         ⊟ [Invocation Contract Container]
               [Invocation Contract] -> quantity [In] [Contract]
               [Invocation Contract] -> price [In] [Contract]
               [Invocation Contract] -> amount [Out] [Contract]
      ⊟ [Production Plan Invocation] -> FC - UC1.3 - PricerDisplayTask [Task Java]
         ⊟ [Invocation Contract Container]
               [Invocation Contract] -> quantity [In] [Contract]
               [Invocation Contract] -> price [In] [Contract]
               [Invocation Contract] -> amount [In] [Contract]
```

Quantity's Properties



Property	Value
⊟ Behaviour	
Invoked Contract	quantity [In] [Contract]
⊟ Connector	
Source Invocation Contract	
Target Invocation Contract	
⊟ Documentation	
Description	
⊟ Factory Component	
Factory Component Contract	quantity [In] [Factory Component Contract]
⊟ Identifier	
ID	_Rlhq0BvjEd-W6L66jY5sHw
⊟ Orchestration	
Orchestration Parameter	

Amount's Properties



Property	Value
⊟ Behaviour	
Invoked Contract	amount [In] [Contract]
⊟ Connector	
Source Invocation Contract	[Invocation Contract] -> amount [Out] [Contract]
Target Invocation Contract	
⊟ Documentation	
Description	
⊟ Factory Component	
Factory Component Contract	
⊟ Identifier	
ID	_dQfdIBvjEd-W6L66jY5sHw
⊟ Orchestration	
Orchestration Parameter	

Agenda

Introduction

EGF Architecture

Concepts & Practice

- Generation Chain
- Activity
- Factory Component
- Task
- Pattern

EGF Portfolios

Modèle presentation_epm version 1.0

THALES

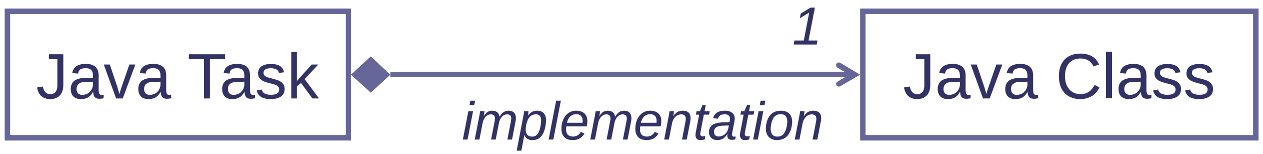
Task – Definition

A task is an atomic generation unit

- A task enables to execute code in a language
- Examples of Tasks: Java Task, Ruby Task, Ant Task

Task implementation:

- A task is associated to an implementation file
- Example: a JavaTask is implemented by a Java class (which implements ITaskProduction)

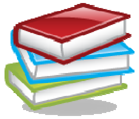


[Practice] Hands On...

Links:

[Video] Video: Activity Creation: http://vimeo.com/15639796

Examples:



[Eclipse] Help Contents! / EGF / Tutorials / Factory Component – First Steps

Exercices:

EGF Example – [Plug-in] org.eclipse.egf.usecase.fc.uc1 plugin, for definition of Factory Component & Task

Agenda

- Introduction
- EGF Architecture
- Concepts & Practice
  - Generation Chain
  - Activity
  - Factory Component
  - Task
  - Pattern
- EGF Portfolios

Modèle presentation_epm version 1.0

THALES

Pattern – Definition

Definition

- Declarative formalism to apply a behavior onto a resource

Purpose

- Dissociation of the specification (external view) from the implementation (internal view)
- The implementation conforms to a language, such as Java or Jet (for model-to-text transformation), and is executed with an engine associated to the selected language
- A set of patterns is executed by a specific activity which declares:
  - How to execute the patterns
  - The execution environment, i.e. parameters, such as the used resource (e.g., model), a reporter and post-processor when M2T, extension or redefinition of patterns to fit to a new context

Agenda

Introduction

EGF Architecture

Concepts & Practice

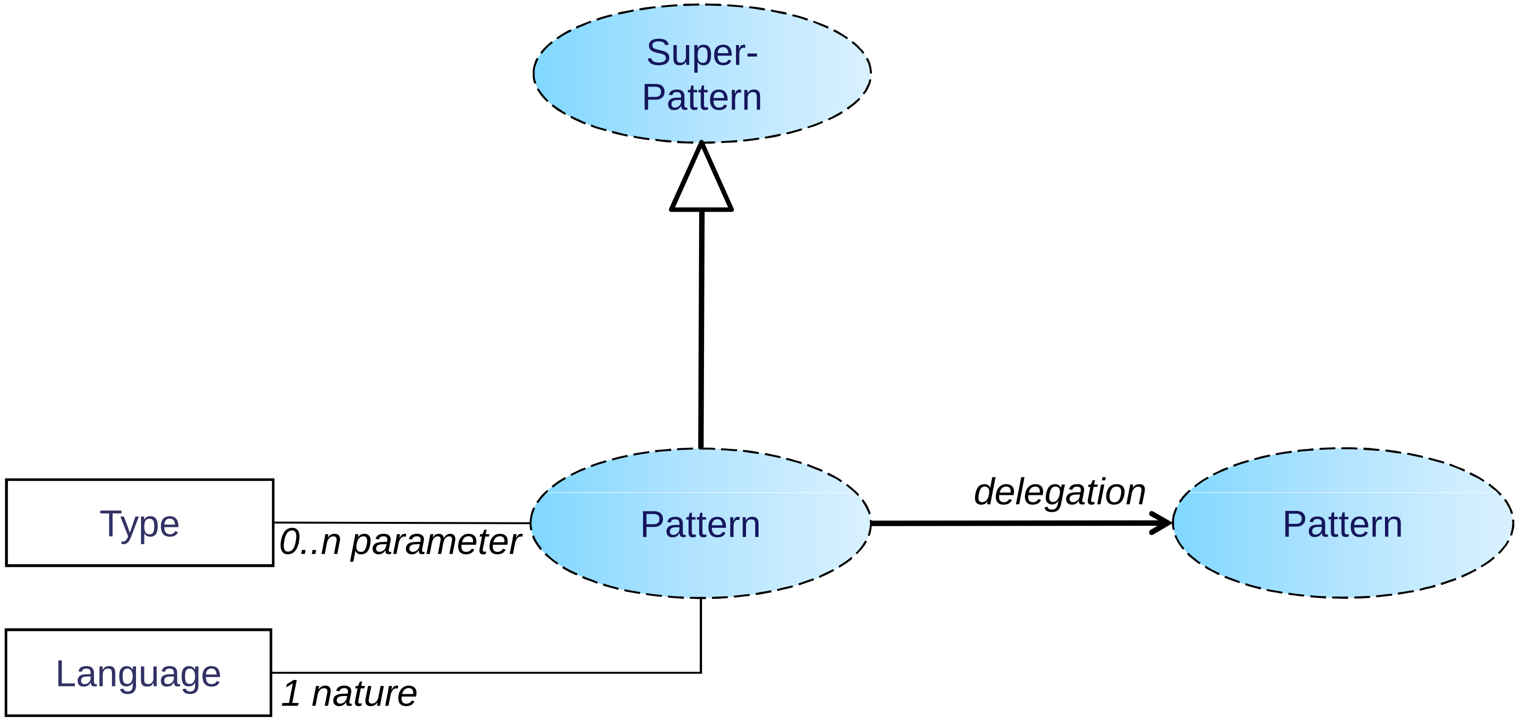
- Generation Chain
- Activity
- Factory Component
- Task
- Pattern
  - Pattern Structure

EGF Portfolios

Modèle presentation_epm version 1.0

THALES

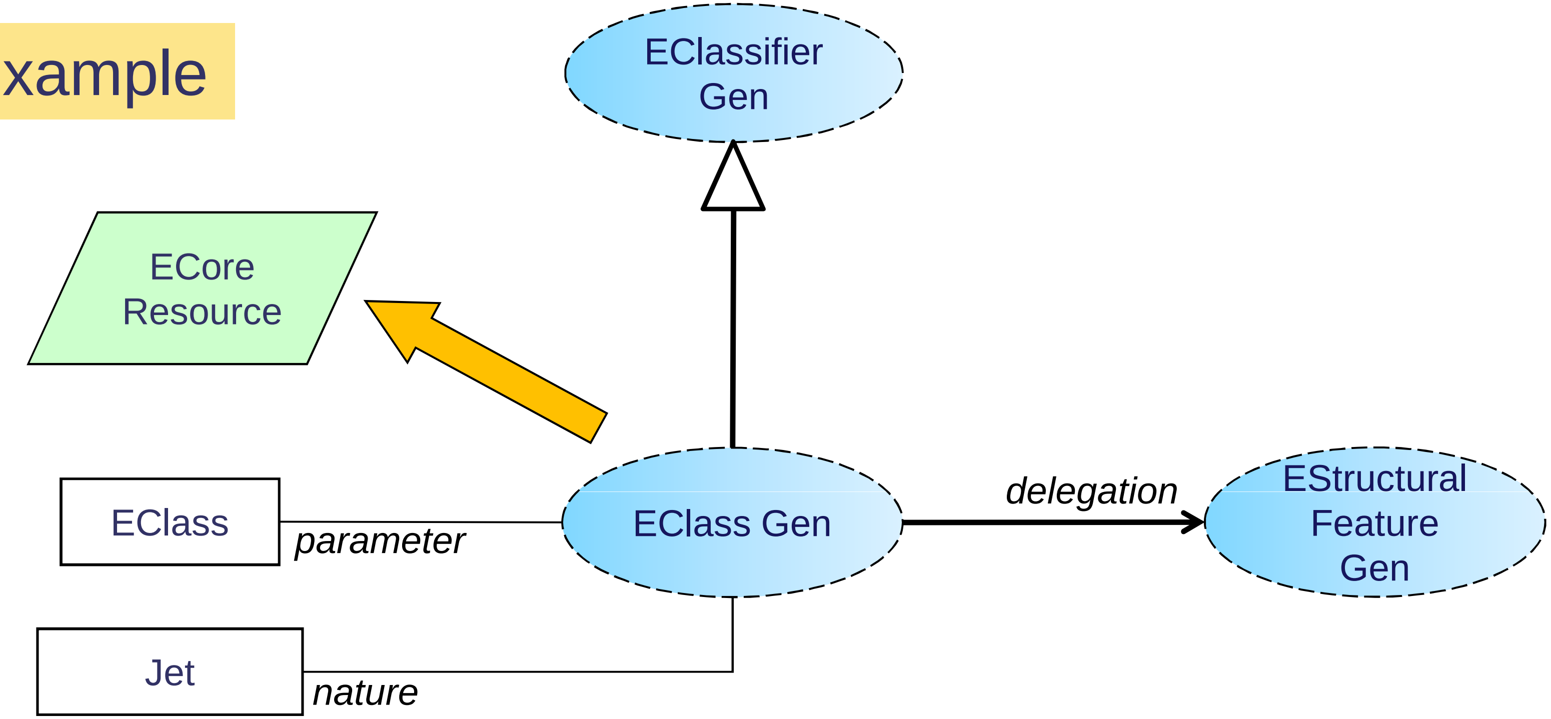
Pattern Structure – Specification Elements



- **Parameter**: Type of a query record from a query applied over a resource (e.g., a class from an ecore model, a file of a file directory)
- **Nature:** Language used for the pattern implementation (e.g., Java, Jet for model-to-text)

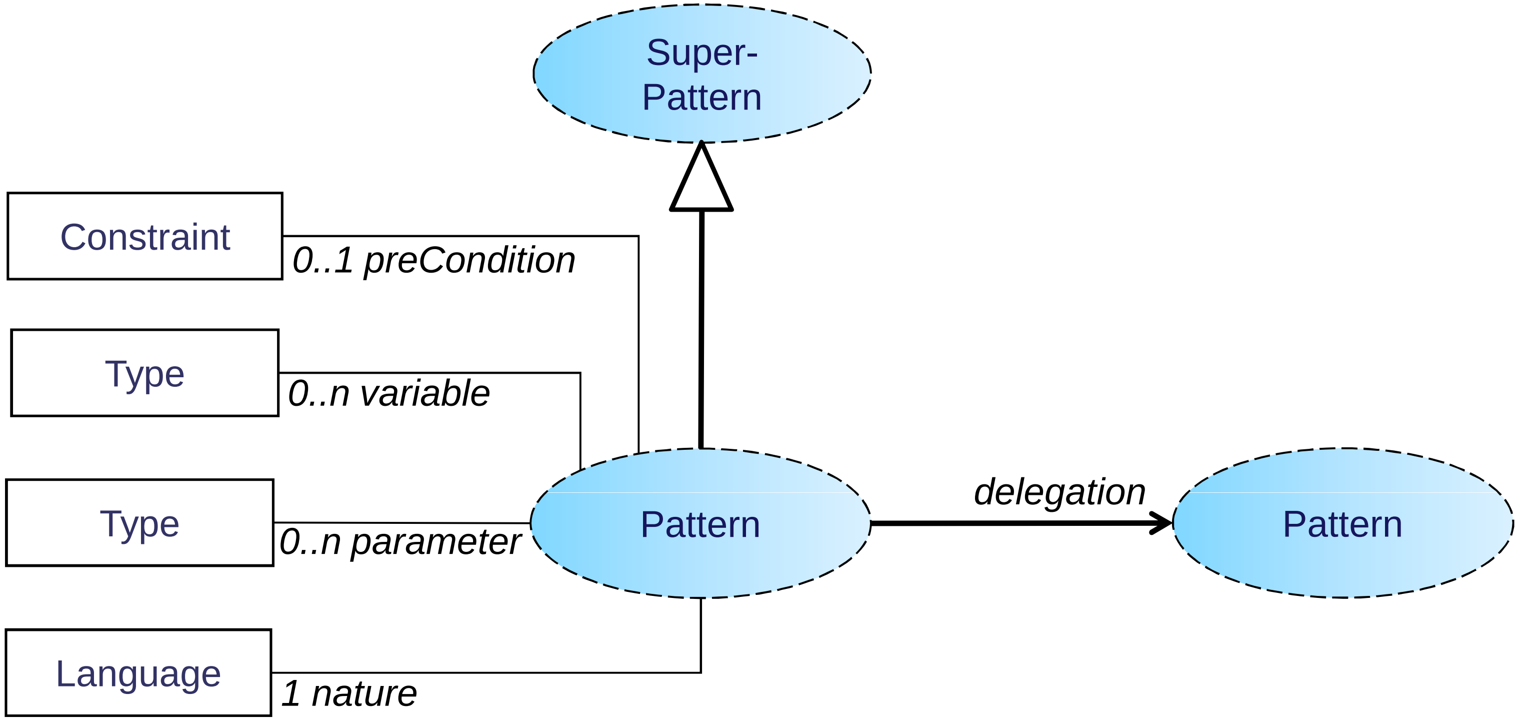
Pattern Structure – Specification Elements

Example



- The EClassGen pattern is applied onto a Ecore resource
- Objects selected on the ecore resource: EClass instances
- It specializes the EClassifierGen pattern
- It applies a model-to-text generation in Jet
- Its also applies a generation on its features by delegation to the EStructuralFeatureGen pattern

Pattern Structure – Specification Elements

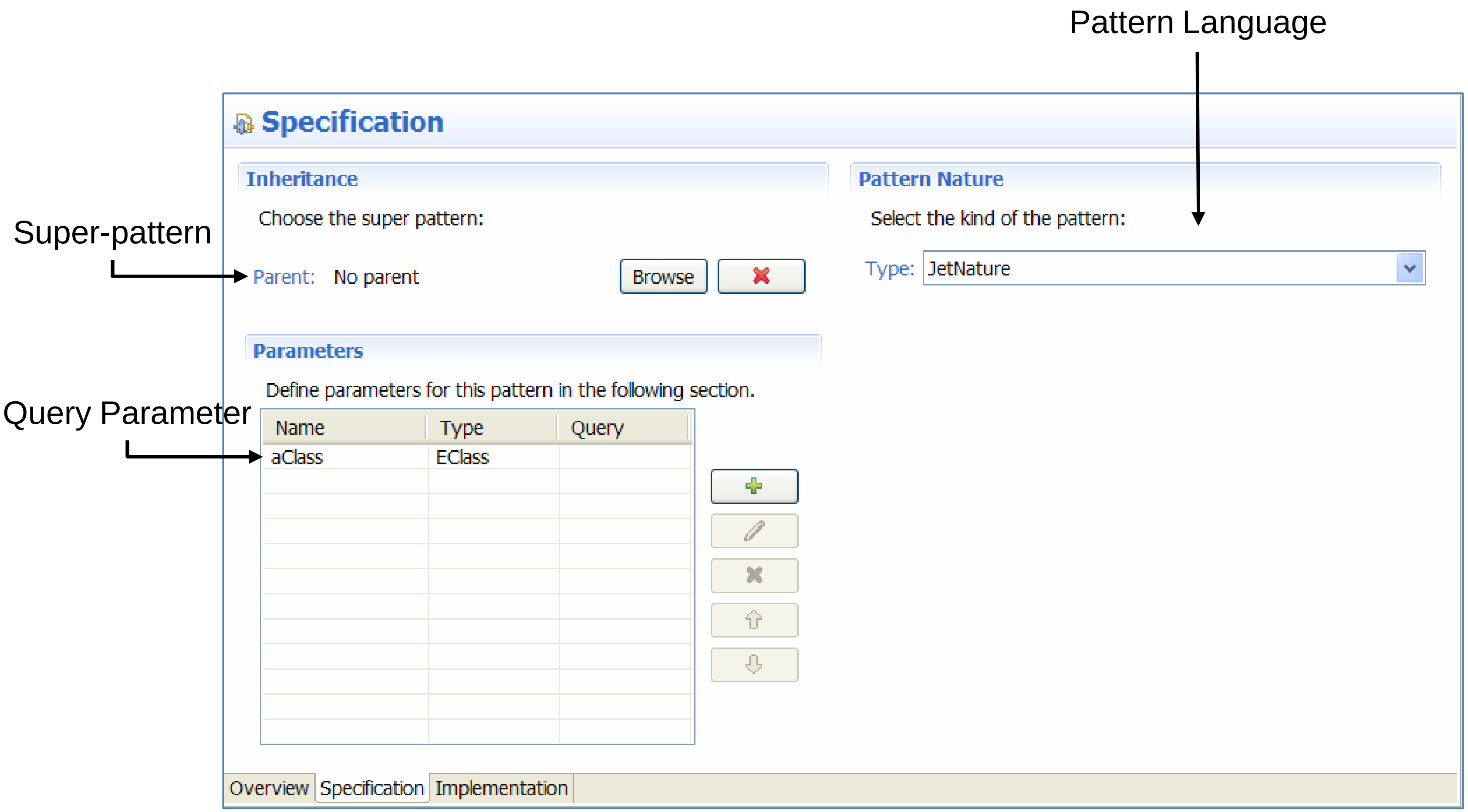


- **preCondition/Constraint:** constraint to be verified to be applied
- **variable/Type:** local variable declaration for the pattern implementation

Modèle presentation_epm version 1.0

THALES

Pattern Structure – Specification View



Specification

Inheritance

Choose the super pattern:

Parent: No parent

Pattern Nature

Select the kind of the pattern:

Type: JetNature

Parameters

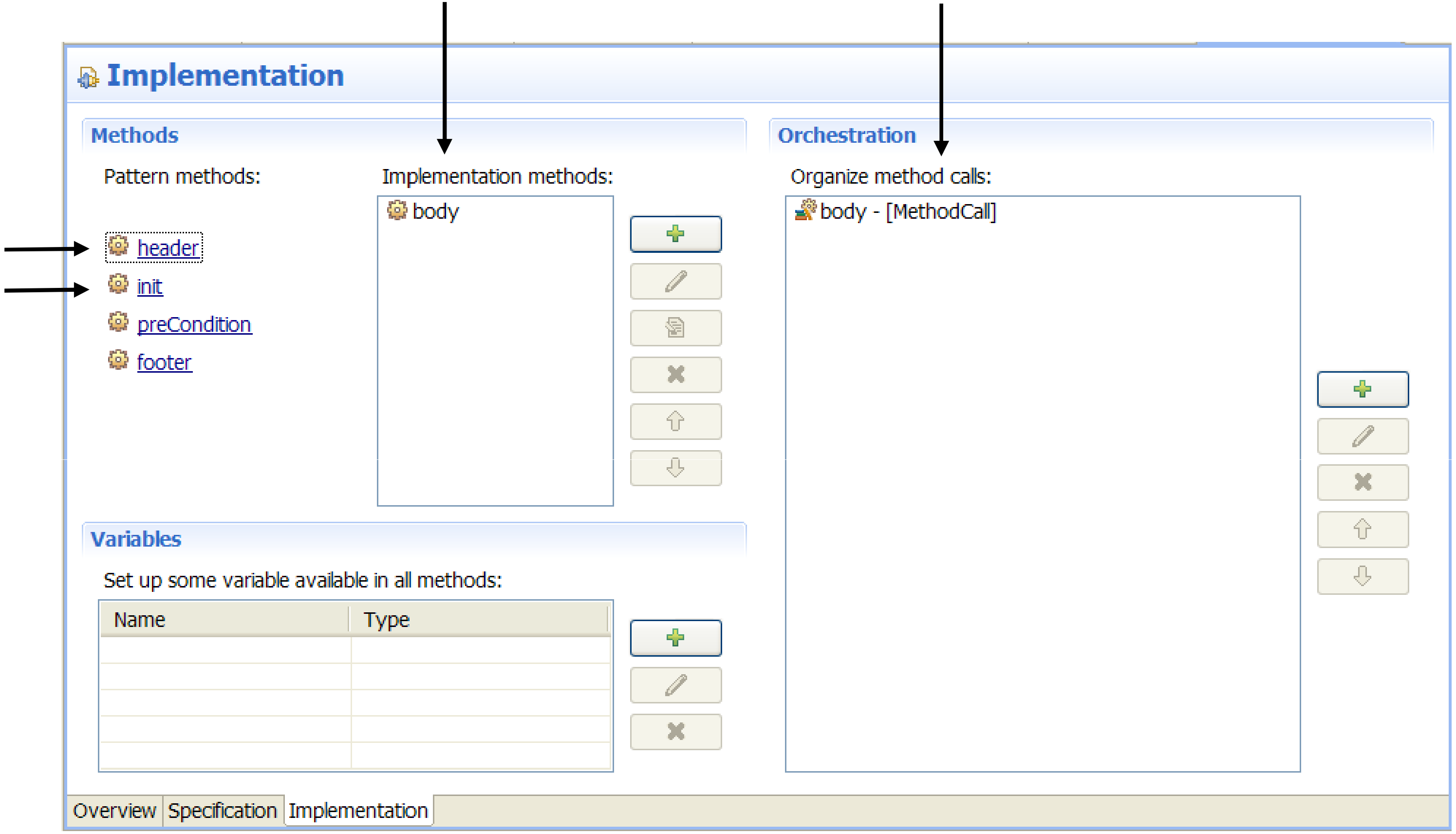
Define parameters for this pattern in the following section.

Name	Type	Query
aClass	EClass	

Pattern Structure – Implementation View

Methods which implement the pattern
They conform to the pattern language

Order to execute the methods



- **header:** typically used for the Jet header
- **init**: method for pattern initialization (e.g., variable initialization)
- A method editor allows editing pattern methods

Modèle presentation_epm version 1.0

THALES

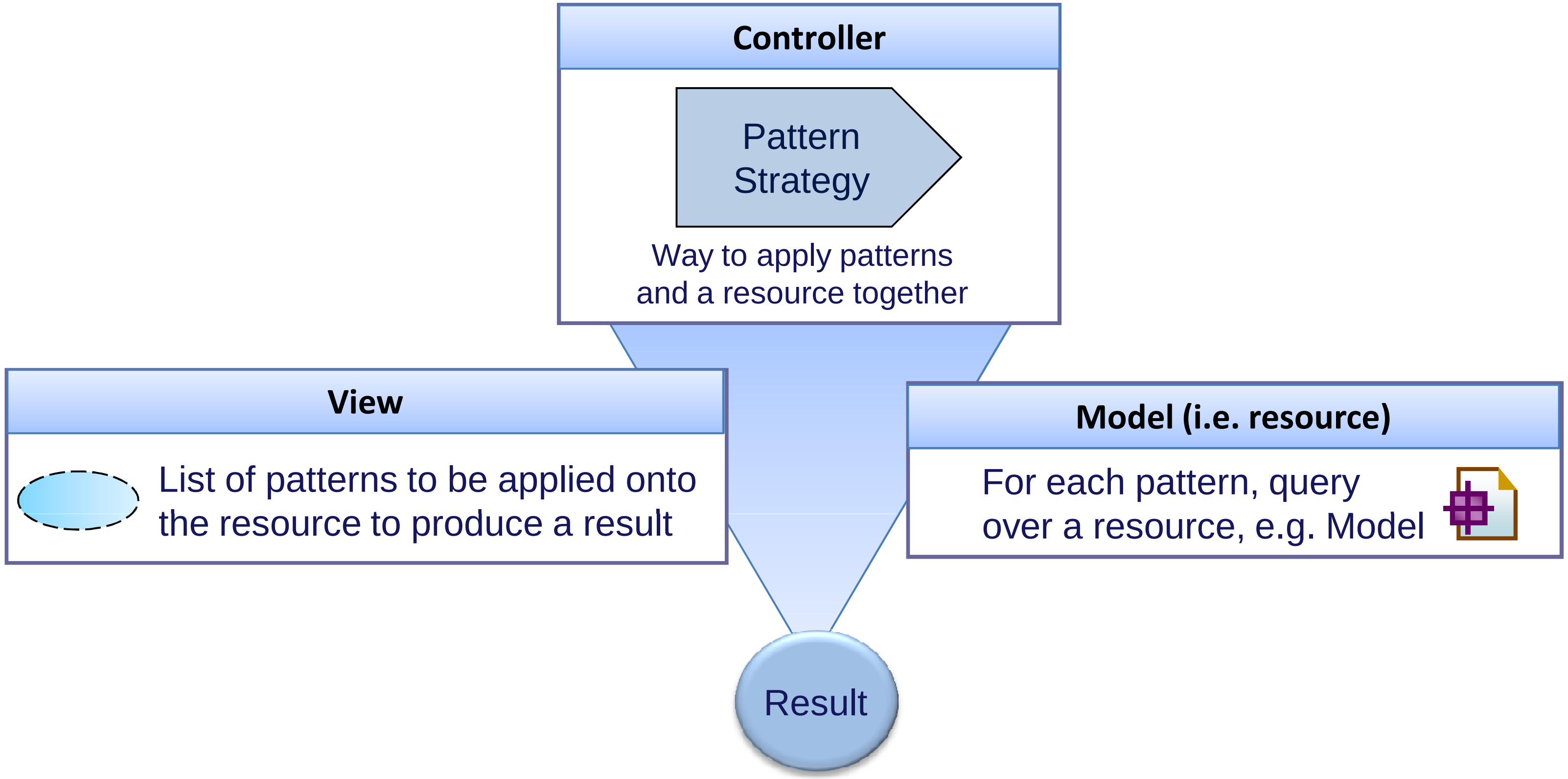
Agenda

- Introduction
- EGF Architecture
- Concepts & Practice
  - Generation Chain
  - Activity
  - Factory Component
  - Task
  - Pattern
    - Pattern Structure
    - Pattern Execution
- EGF Portfolios

Modèle presentation_epm version 1.0

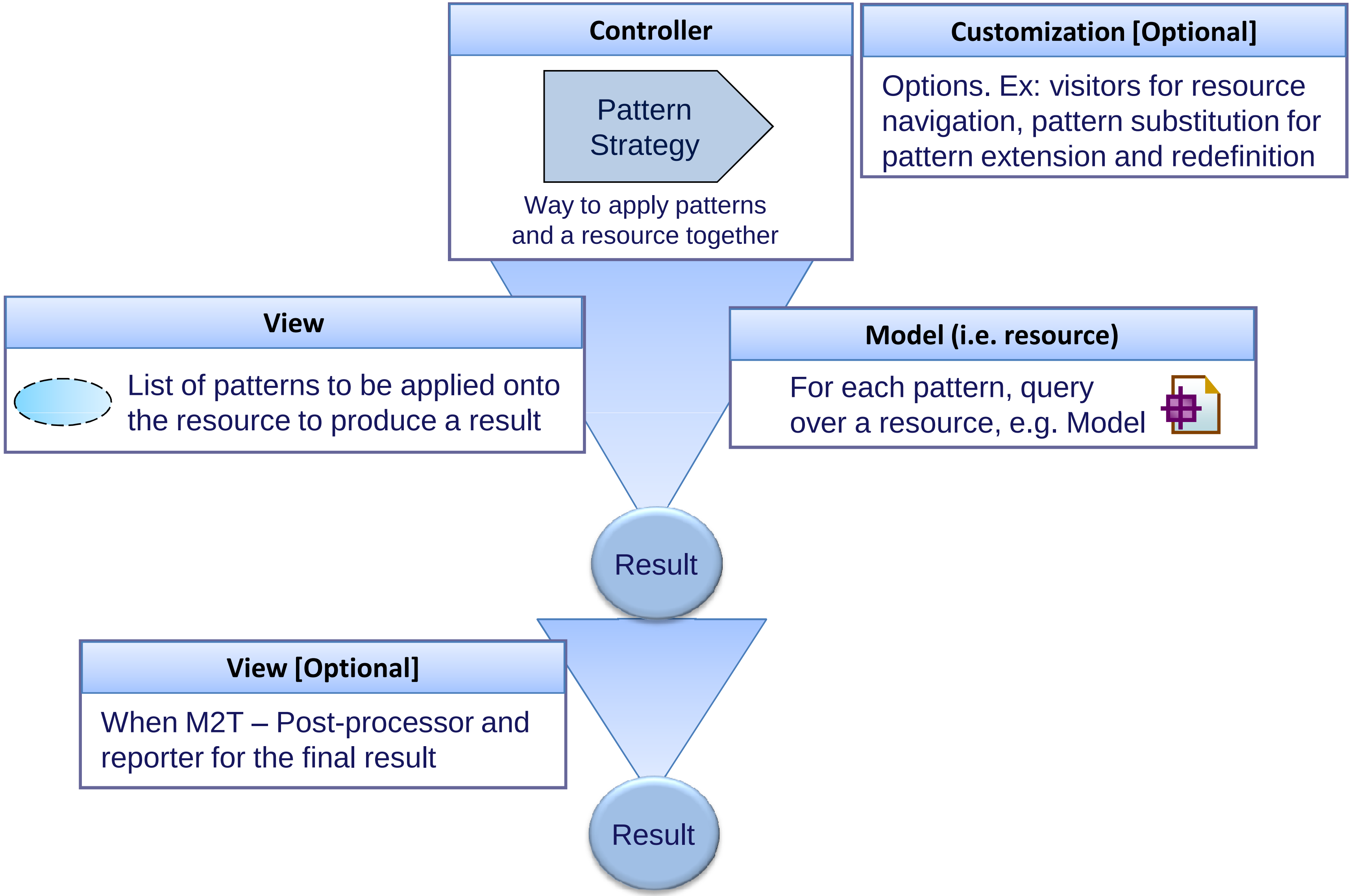
THALES

Pattern Execution – Big Picture



Controller
Pattern Strategy
Way to apply patterns and a resource together
View
List of patterns to be applied onto the resource to produce a result
Model (i.e. resource)
For each pattern, query over a resource, e.g. Model
Result

Pattern Execution – Big Picture



Controller

Pattern Strategy

Way to apply patterns and a resource together

Customization [Optional]

Options. Ex: visitors for resource navigation, pattern substitution for pattern extension and redefinition

View

List of patterns to be applied onto the resource to produce a result

Model (i.e. resource)

For each pattern, query over a resource, e.g. Model

View [Optional]

When M2T – Post-processor and reporter for the final result

Pattern Strategy

Definition: Way to apply patterns against a resource

Examples of strategies:

- **Domain-driven pattern strategy**: in-depth navigation over a resource (e.g. model), and for each resource element, applying a set of patterns
- **Pattern-driven strategy**: for each pattern, applying the pattern for each resource element

Pattern Strategy

Definition: Way to apply patterns against a resource

Strategy parameters:

- **Resource visitor**: When navigating over a resource, the visitor function specifies how to continue this navigation. Example: considering the sub-classes of the current resource instance.
- **Post-processor**: Post-processing a model-to-text transformation
- **Reporter**: Management of the ouput for a model-to-text transformation (e.g., in one or several files, file location)
- **Pattern substitution**: list of pattern substitutions for pattern redefinition or extension to customize a pattern-based transformation

Agenda

Introduction

EGF Architecture

Concepts & Practice

- Generation Chain
- Activity
- Factory Component
- Task
- Pattern
  - Pattern Structure
  - Pattern Execution
  - Pattern Relationships

EGF Portfolios

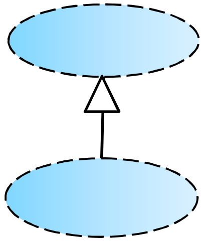
Pattern Relationships – Introduction

Patterns can be related together (e.g., pattern inheritance, pattern call)

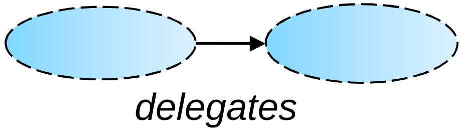
The next slides present the different kinds of pattern relationships

Pattern Relationships

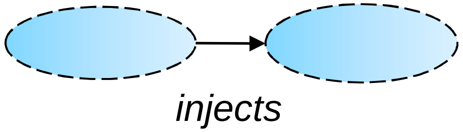
Pattern Inheritance



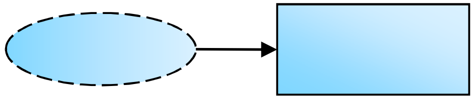
Pattern Delegation



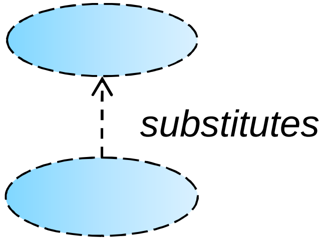
Pattern Injection



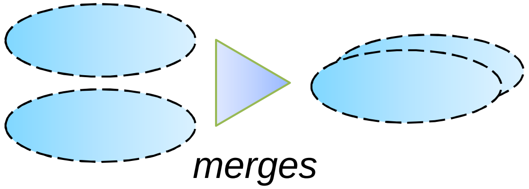
Pattern Callback



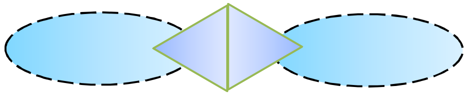
Pattern Substitution



Pattern Merge

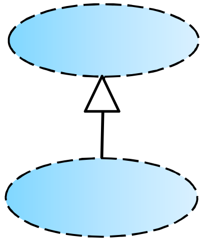


Pattern Comparison



Pattern Inheritance

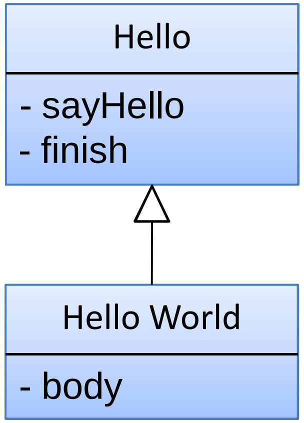
Pattern inheritance



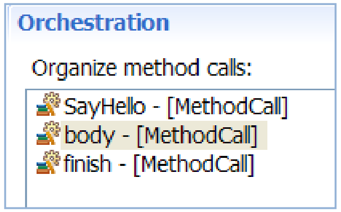
Case 1. Reuse of super-pattern methods

Same mechanism than Class inheritance
Selection of methods from the super-pattern hierarchy

Example



Orchestration of HelloWorld



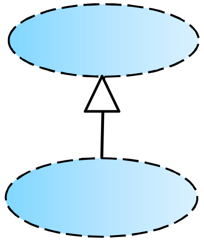
Orchestration

Organize method calls:

SayHello - [MethodCall]
body - [MethodCall]
finish - [MethodCall]

Pattern Inheritance

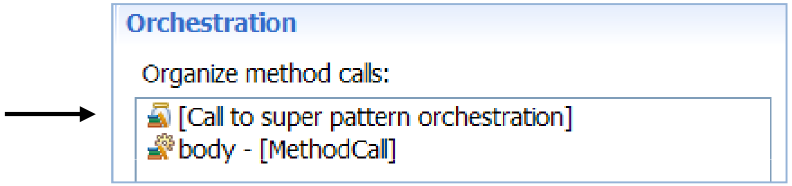
Pattern inheritance



Case 2. Reuse of super-pattern orchestration

Reuse of method and orchestration defined in the super-pattern
This abstracts the super-pattern orchestration
This avoids rewriting pattern orchestration
Just adding the methods of the current pattern

Example



Orchestration

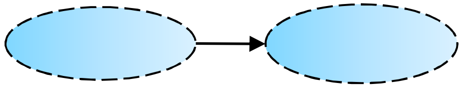
Organize method calls:

[Call to super pattern orchestration]
body - [MethodCall]

Modèle presentation_epm version 1.0

Pattern Delegation

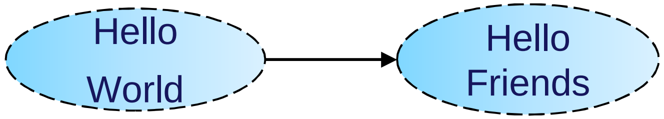
Pattern delegation

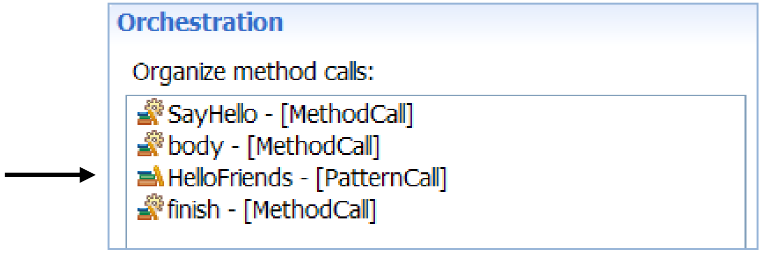


Case. For Problem decomposition & Reuse of pattern

- The same pattern is reused in different pattern contexts
- The orchestration of the called pattern is applied
- The Pattern caller provides parameter values to the called pattern
- The parameter values are statically declared at the pattern definition

Example 1



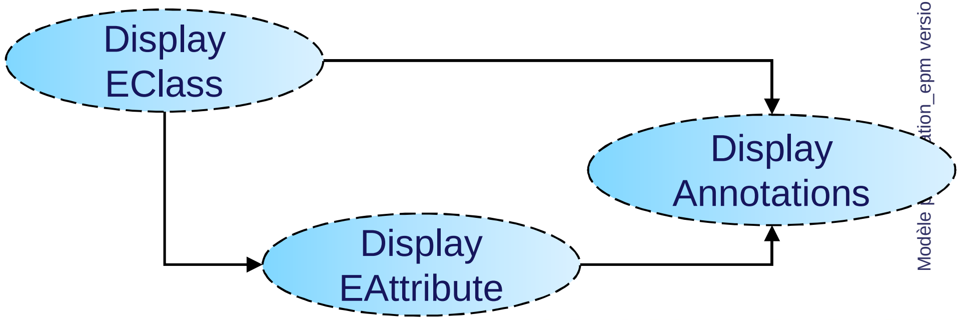


Orchestration

Organize method calls:

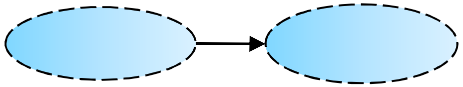
SayHello - [MethodCall]
body - [MethodCall]
HelloFriends - [PatternCall]
finish - [MethodCall]

Example 2



Multilingual Call

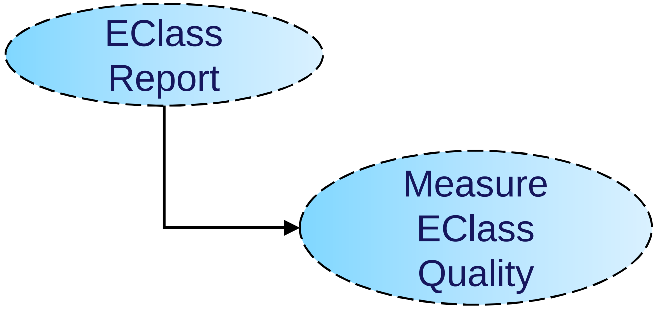
Pattern delegation



Case. Pattern delegation when implementation languages are different

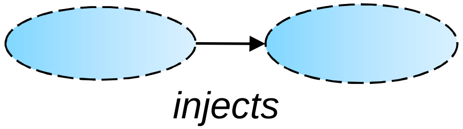
This corresponds to a Pattern Delegation where Pattern natures are different. For instance, a Pattern with a Jet nature calls a Pattern with a Java nature in order to differently process the same resource.
It is impossible to have different natures in the same Pattern inheritance hierarchy.

Example



Pattern Injection

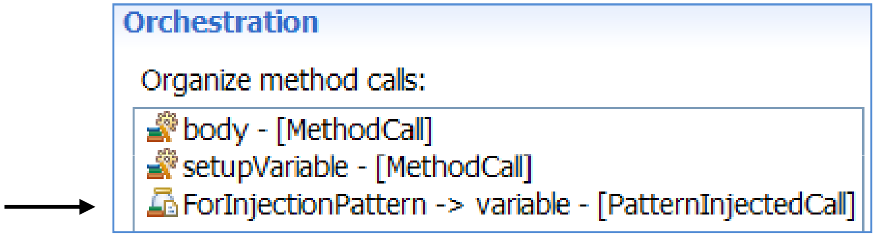
Pattern injection



Case. Reuse of pattern with a dynamic resolution of the injected context

- A Pattern injection corresponds to a Pattern Delegation, but
- The parameter values are dynamically set at pattern execution

Example



Orchestration

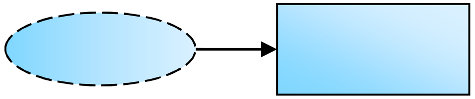
Organize method calls:

body - [MethodCall]
setupVariable - [MethodCall]
ForInjectionPattern -> variable - [PatternInjectedCall]

In this example, the “setupVariable” method sets the injection context

Pattern Callback

Pattern Callback

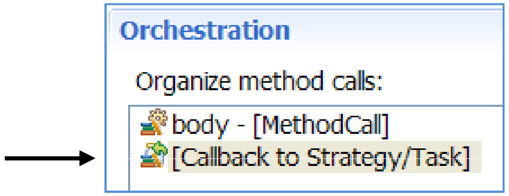


Case 1. Applying a Java call

The callback indicates where the callback on a Java Class is applied

Example

Pattern orchestration

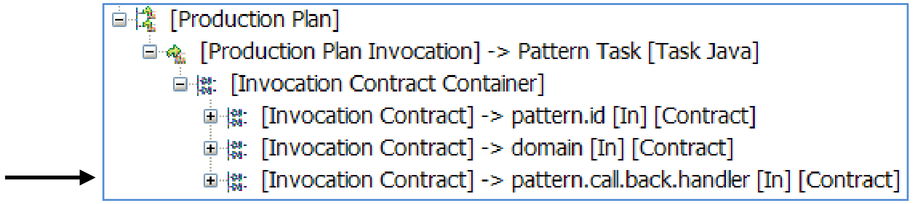


Orchestration

Organize method calls:

- body - [MethodCall]
- [Callback to Strategy/Task]

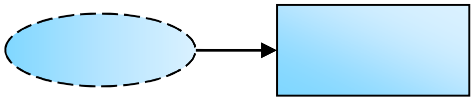
Specification of the Java Class in the production plan



```
[Production Plan]
  [Production Plan Invocation] -> Pattern Task [Task Java]
    [Invocation Contract Container]
      [Invocation Contract] -> pattern.id [In] [Contract]
      [Invocation Contract] -> domain [In] [Contract]
      [Invocation Contract] -> pattern.call.back.handler [In] [Contract]
```

Modèle presentation_epm version 1.0

Pattern Callback



Case 2. Combination with the Pattern Strategy

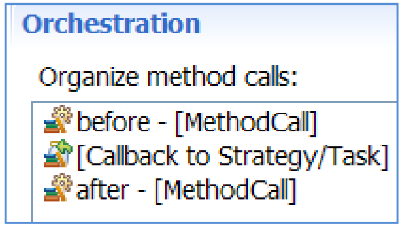
A strategy determines how to apply patterns and how to navigate over a resource. In an orchestration, a callback is the moment before and after a cycle of pattern application, and allows to discriminate the methods to apply before and after it.

Example

Scenario:
The following generation result can be realized with a callback.
- The model-driven strategy navigates over the model
- There is a pattern for each kind of model element with the following pattern orchestration
A generation action is realized before (open) and after (close) the callback.

```
<EPackage name=“P”>
   <EClass name=“C1”>
      <EAttribute = “A1”>
               …
      </EAttribute = “A1”>
   </EClass name=“C1”>
</EPackage name=“P”>
```

Generation result



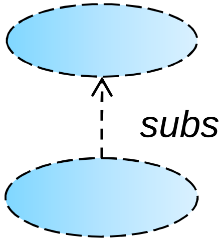
Orchestration

Organize method calls:

- before - [MethodCall]
- [Callback to Strategy/Task]
- after - [MethodCall]

Pattern Substitution

Pattern substitution



Case. Customization of a pattern-based generation

- A substitution replaces a pattern by a list of patterns
- This list can be empty (for annihilating a pattern), another pattern, or a list of other patterns (for replacing one pattern by several)
- This mechanism enables to adapt a generation to a specific context
- It is used for definition of families of code generation with patterns

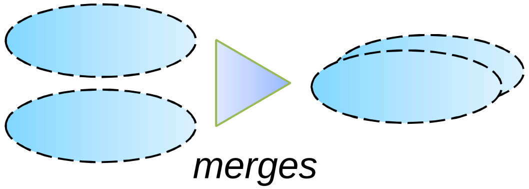
For deeper understanding

[Tutorial]

http://wiki.eclipse.org/EGF_Tutorial_and_Use_Cases#EGF_Patterns

Pattern Merge

Pattern Merge



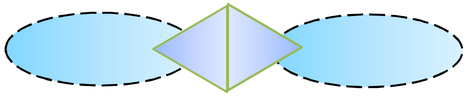
Case. Combination of pattern lists

- Two patterns lists are merged into one list
- Examples: for customization, merging a local substitution with a pattern list in parameter of factory component

For deeper understanding

EGF Example – [Plug-in] org.eclipse.egf.usecase.emf.uc3

Pattern Comparison

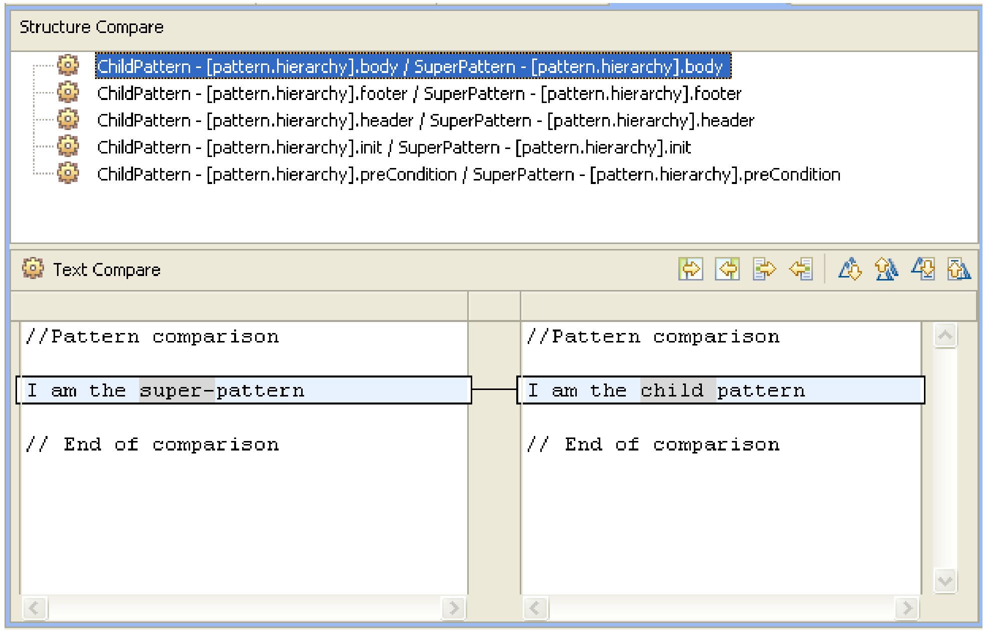


Case. Used during pattern edition – Face pattern evolution when pattern-based generation scales up

- Comparison of patterns in a hierarchy or of cousin patterns

Example

Scenario:
Comparison of super- and child-patterns in the same or different pattern libraries. Below, comparison of “body” methods of a ChildPattern and its SuperPattern.
Possibility of live edition when editing pattern comparison.



ChildPattern - [pattern.hierarchy].body / SuperPattern - [pattern.hierarchy].body
ChildPattern - [pattern.hierarchy].footer / SuperPattern - [pattern.hierarchy].footer
ChildPattern - [pattern.hierarchy].header / SuperPattern - [pattern.hierarchy].header
ChildPattern - [pattern.hierarchy].init / SuperPattern - [pattern.hierarchy].init
ChildPattern - [pattern.hierarchy].preCondition / SuperPattern - [pattern.hierarchy].preCondition

```
//Pattern comparison

I am the super-pattern

// End of comparison
```

```
//Pattern comparison

I am the child pattern

// End of comparison
```

[Practice] Hands On…

Links:

[Video] Pattern Creation: http://vimeo.com/15664081

Examples:

[Eclipse] Help Contents! / EGF / Tutorials / Pattern – First Steps

Exercices:

EGF Example – [Plug-in] org.eclipse.egf.usecase.pattern.uc1 and org.eclipse.egf.usecase.pattern.uc2

Agenda

Introduction

EGF Architecture

Concepts & Practice

EGF Portfolios

- Enhancement of the EMF Generation
- Build Chain Portfolio

Modèle presentation_epm version 1.0

THALES

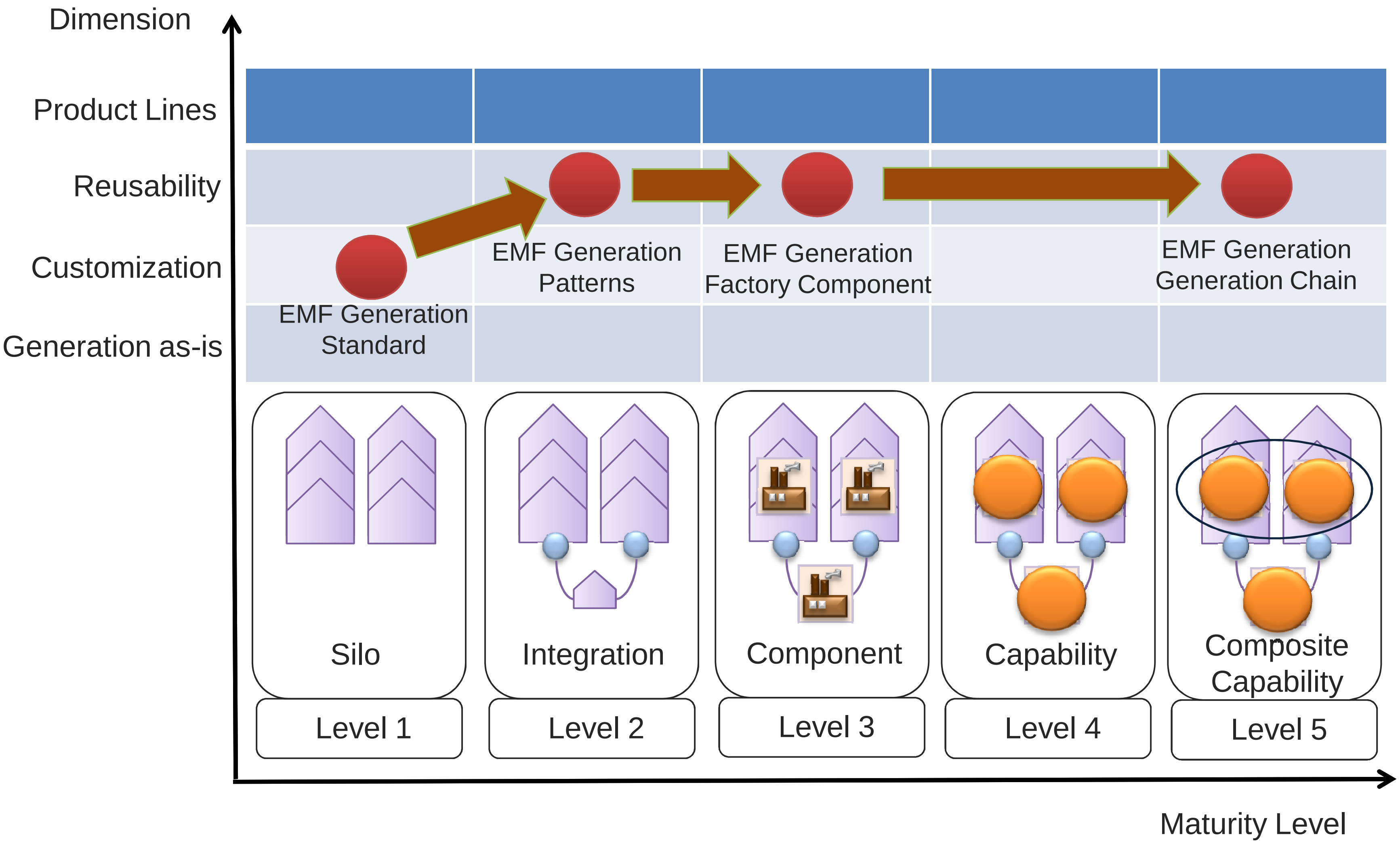
Agenda

- Introduction
- EGF Architecture
- Concepts & Practice
- EGF Portfolios
  - Enhancement of the EMF Generation
  - Build Chain Portfolio

Modèle presentation_epm version 1.0

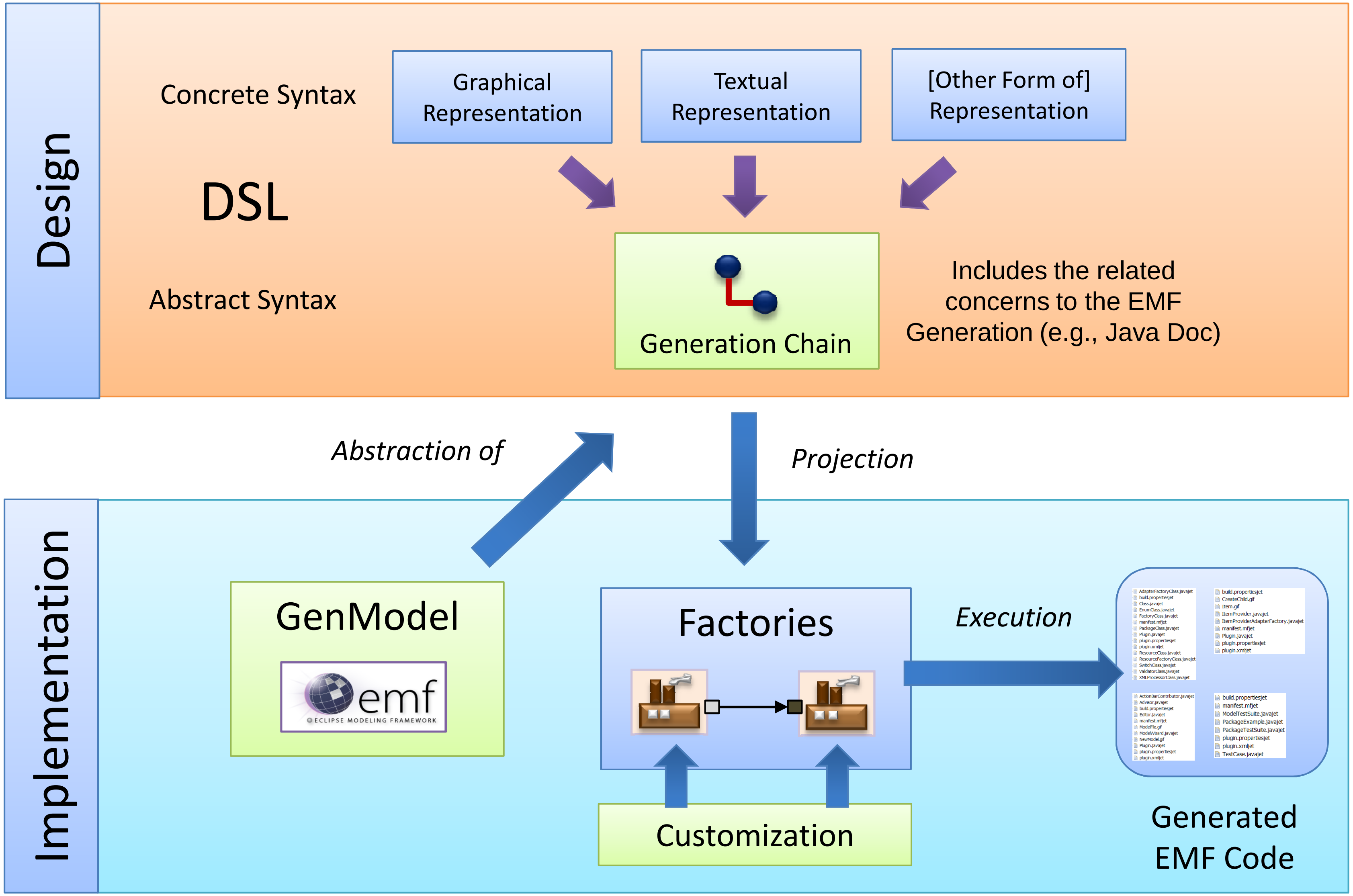
THALES

EMF Generation – Levels of Maturity



Modèle presentation_epm version 1.0

Portfolio #1 – EMF Generations Layers



Includes the related concerns to the EMF Generation (e.g., Java Doc)

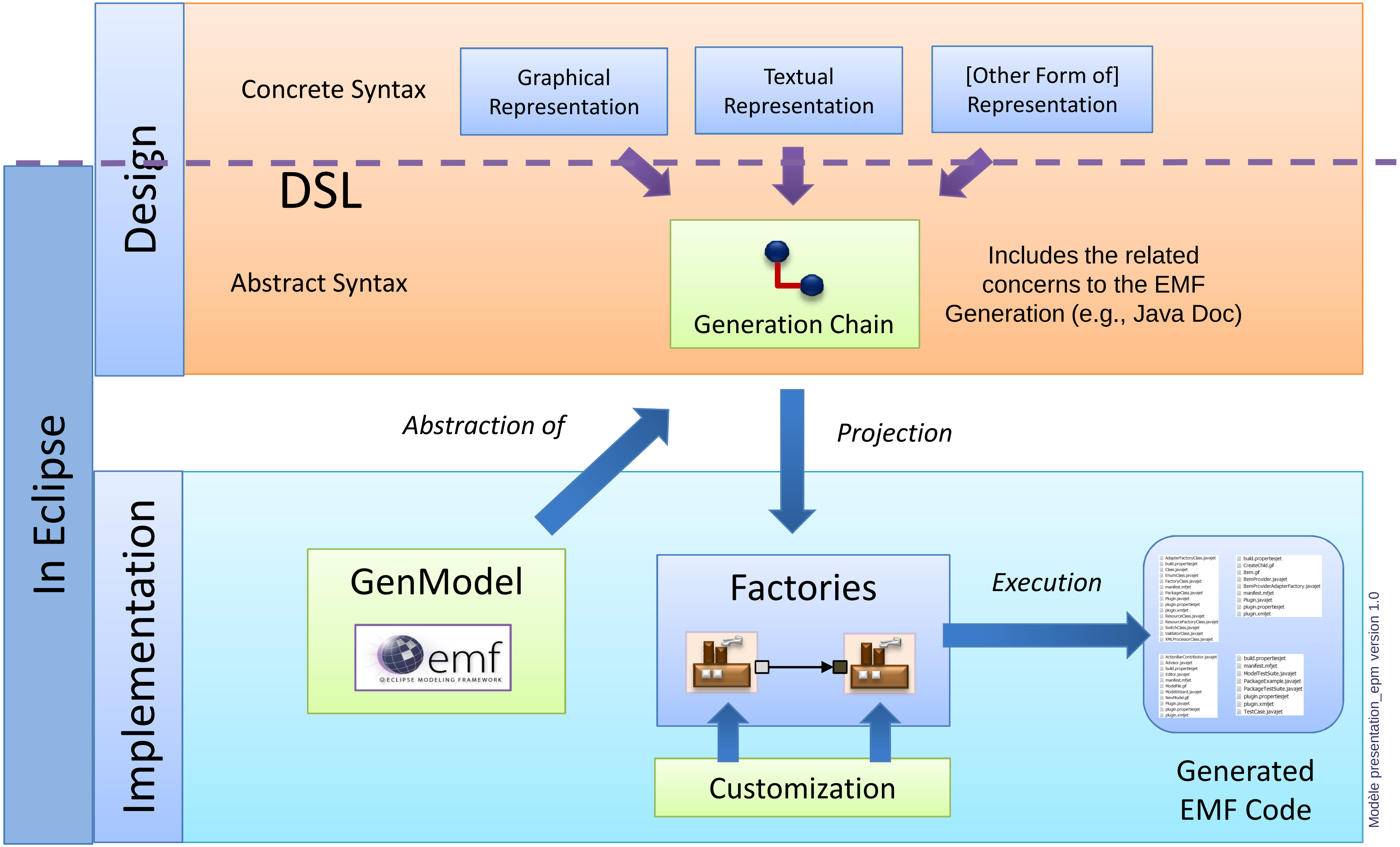
AdapterFactoryClass.javajet
build.propertiesjet
Class.javajet
EnumClass.javajet
FactoryClass.javajet
manifest.mfjet
PackageClass.javajet
Plugin.javajet
plugin.propertiesjet
plugin.xmljet
ResourceClass.javajet
ResourceFactoryClass.javajet
SwitchClass.javajet
ValidatorClass.javajet
XMLProcessorClass.javajet

build.propertiesjet
CreateChild.gif
Item.gif
ItemProvider.javajet
ItemProviderAdapterFactory.javajet
manifest.mfjet
Plugin.javajet
plugin.propertiesjet
plugin.xmljet

ActionBarContributor.javajet
Advisor.javajet
build.propertiesjet
Editor.javajet
manifest.mfjet
ModelFile.gif
ModelWizard.javajet
NewModel.gif
Plugin.javajet
plugin.propertiesjet
plugin.xmljet

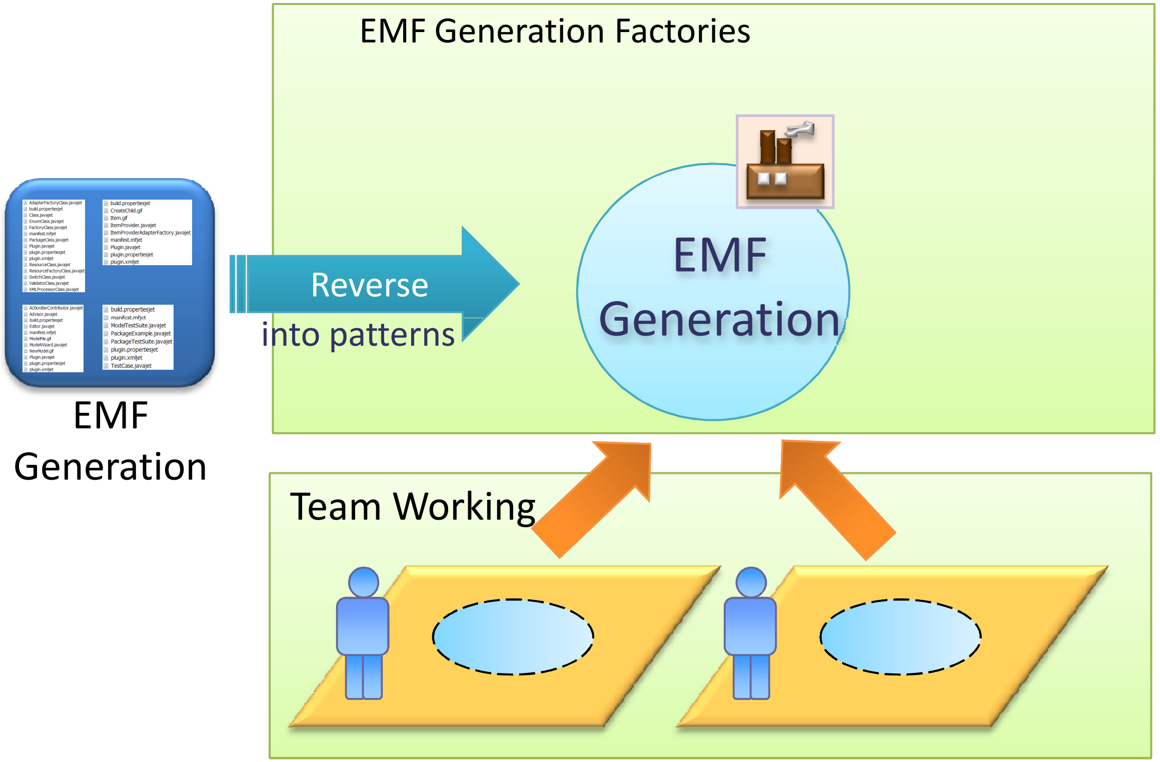
build.propertiesjet
manifest.mfjet
ModelTestSuite.javajet
PackageExample.javajet
PackageTestSuite.javajet
plugin.propertiesjet
plugin.xmljet
TestCase.javajet

Portfolio #1 – EMF Generations Layers

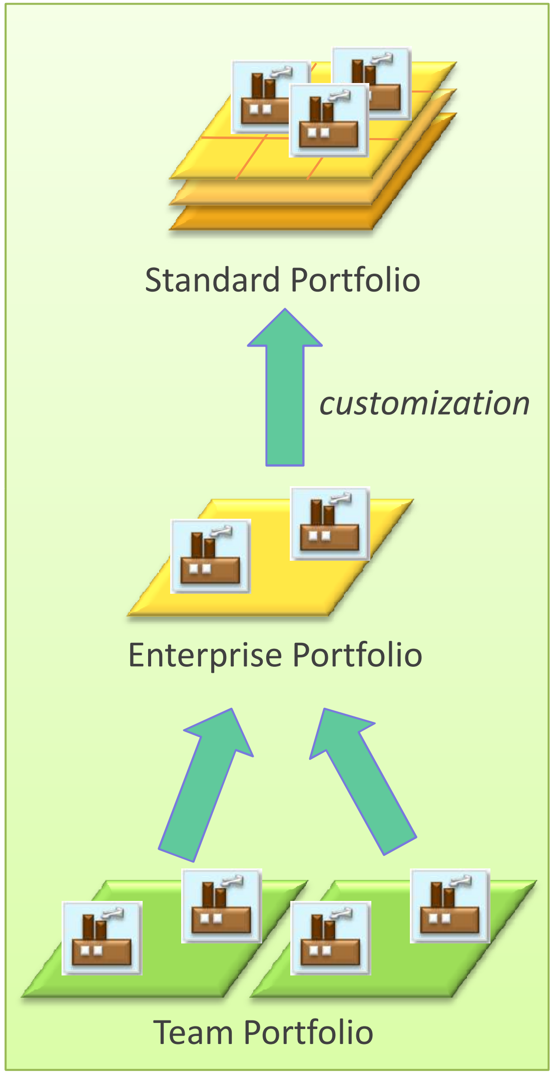


Modèle presentation_epm version 1.0

Portfolio #1 – Enhancement of EMF Generation



Several levels of Customization



[Practice] Hands On…

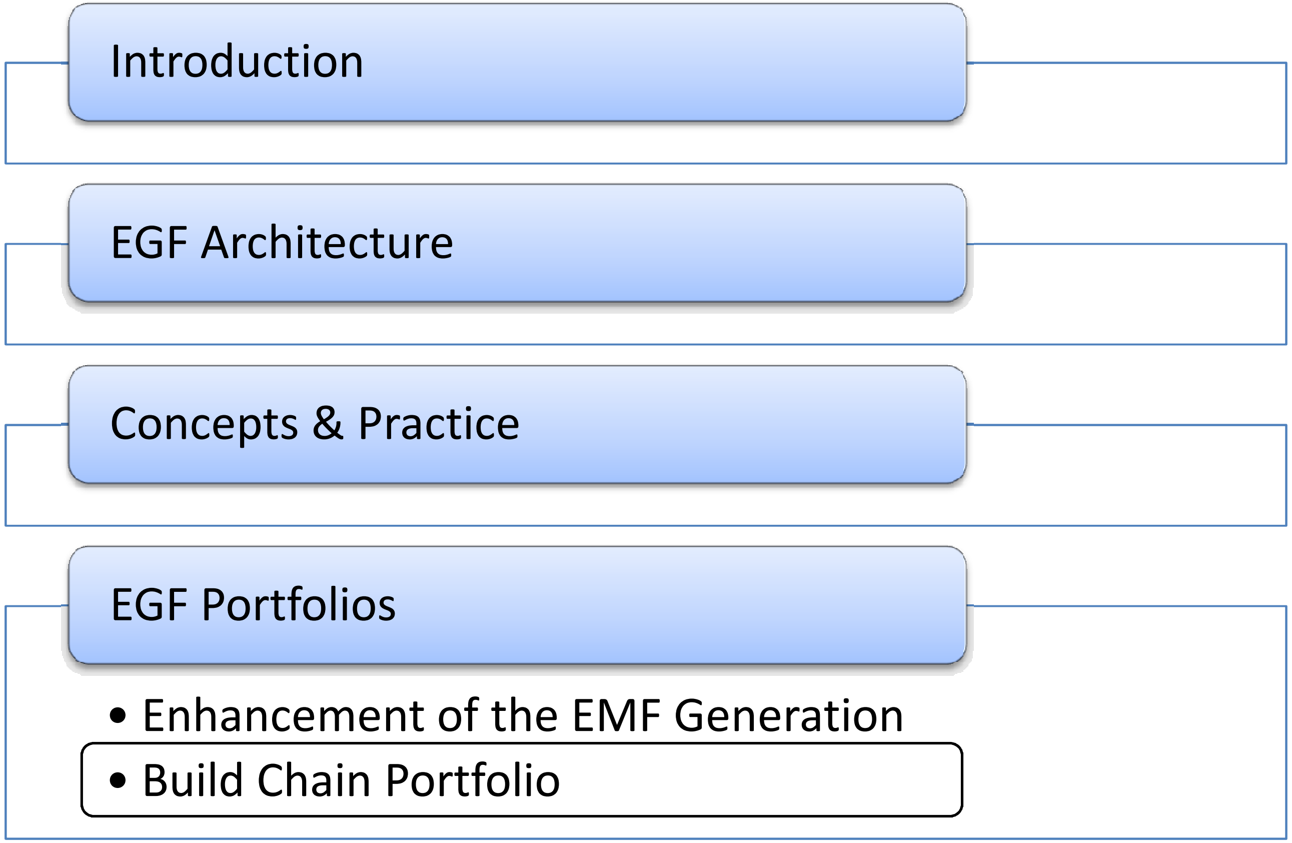
Exercices:

EGF Example – org.eclipse.egf.usecase.emf.uc1,
org.eclipse.egf.usecase.emf.uc2 and
org.eclipse.egf.usecase.emf.uc3

Download access:

http://wiki.eclipse.org/EGF_Tutorial_and_Use_Cases#Enhanced_EMF_Generation

Agenda



Introduction

EGF Architecture

Concepts & Practice

EGF Portfolios

- Enhancement of the EMF Generation
- Build Chain Portfolio

Modèle presentation_epm version 1.0

THALES

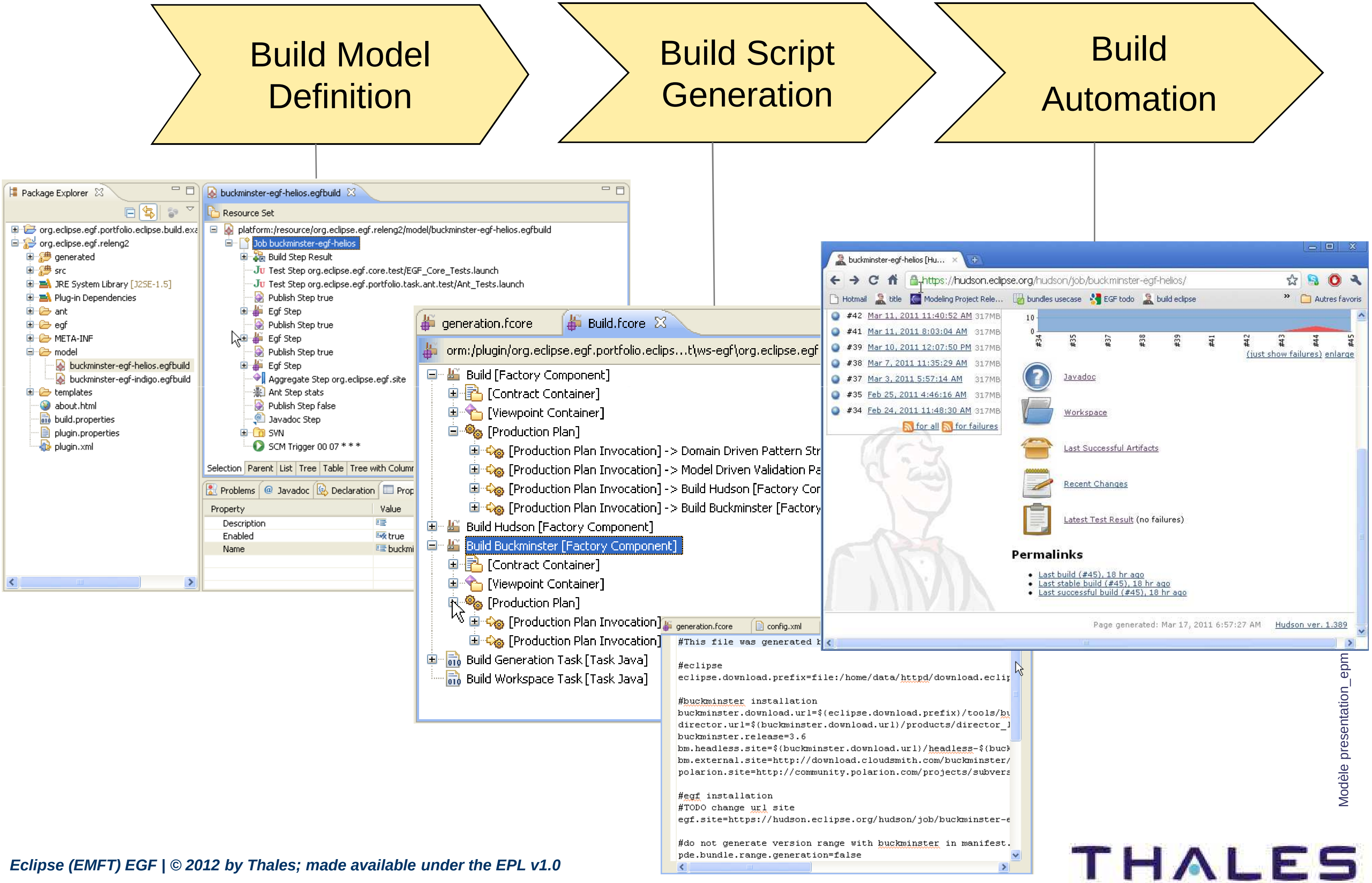
Portfolio #2 – Build Chain Editor & Generator

Objective of the Build Portfolio provided by EGF:

- **Facilitating the definition of build chain:**

  1. A **build editor** describes a build chain
  2. A **generator** targets a build platform, here **Hudson / Jenkins and Buckminster**
  3. Use of the build chain

Portfolio #2 – Build Chain Editor & Generator



Build Model Definition

Build Script Generation

Build Automation

```
org.eclipse.egf.portfolio.eclipse.build.exa
org.eclipse.egf.releng2
  generated
  src
  JRE System Library [J2SE-1.5]
  Plug-in Dependencies
  ant
  egf
  META-INF
  model
    buckminster-egf-helios.egfbuild
    buckminster-egf-indigo.egfbuild
  templates
  about.html
  build.properties
  plugin.properties
  plugin.xml
```

buckminster-egf-helios.egfbuild

```
platform:/resource/org.eclipse.egf.releng2/model/buckminster-egf-helios.egfbuild
  Job buckminster-egf-helios
    Build Step Result
    Test Step org.eclipse.egf.core.test/EGF_Core_Tests.launch
    Test Step org.eclipse.egf.portfolio.task.ant.test/Ant_Tests.launch
    Publish Step true
    Egf Step
    Publish Step true
    Egf Step
    Publish Step true
    Egf Step
    Aggregate Step org.eclipse.egf.site
    Ant Step stats
    Publish Step false
    Javadoc Step
    SVN
    SCM Trigger 00 07 * * *
```

Property	Value
Description	
Enabled	true
Name	buckmi

generation.fcore | Build.fcore

```
orm:/plugin/org.eclipse.egf.portfolio.eclips...t\ws-egf\org.eclipse.egf
  Build [Factory Component]
    [Contract Container]
    [Viewpoint Container]
    [Production Plan]
      [Production Plan Invocation] -> Domain Driven Pattern Str
      [Production Plan Invocation] -> Model Driven Validation Pa
      [Production Plan Invocation] -> Build Hudson [Factory Cor
      [Production Plan Invocation] -> Build Buckminster [Factory
  Build Hudson [Factory Component]
  Build Buckminster [Factory Component]
    [Contract Container]
    [Viewpoint Container]
    [Production Plan]
      [Production Plan Invocation]
      [Production Plan Invocation]
  Build Generation Task [Task Java]
  Build Workspace Task [Task Java]
```

generation.fcore | config.xml

```
#This file was generated k

#eclipse
eclipse.download.prefix=file:/home/data/httpd/download.eclip

#buckminster installation
buckminster.download.url=${eclipse.download.prefix}/tools/bu
director.url=${buckminster.download.url}/products/director_I
buckminster.release=3.6
bm.headless.site=${buckminster.download.url}/headless-${buck
bm.external.site=http://download.cloudsmith.com/buckminster/
polarion.site=http://community.polarion.com/projects/subvers

#egf installation
#TODO change url site
egf.site=https://hudson.eclipse.org/hudson/job/buckminster-e

#do not generate version range with buckminster in manifest.
pde.bundle.range.generation=false
```

buckminster-egf-helios [Hu...

https://hudson.eclipse.org/hudson/job/buckminster-egf-helios/

#42	Mar 11, 2011 11:40:52 AM	317MB
#41	Mar 11, 2011 8:03:04 AM	317MB
#39	Mar 10, 2011 12:07:50 PM	317MB
#38	Mar 7, 2011 11:35:29 AM	317MB
#37	Mar 3, 2011 5:57:14 AM	317MB
#35	Feb 25, 2011 4:46:16 AM	317MB
#34	Feb 24, 2011 11:48:30 AM	317MB

for all for failures

Javadoc

Workspace

Last Successful Artifacts

Recent Changes

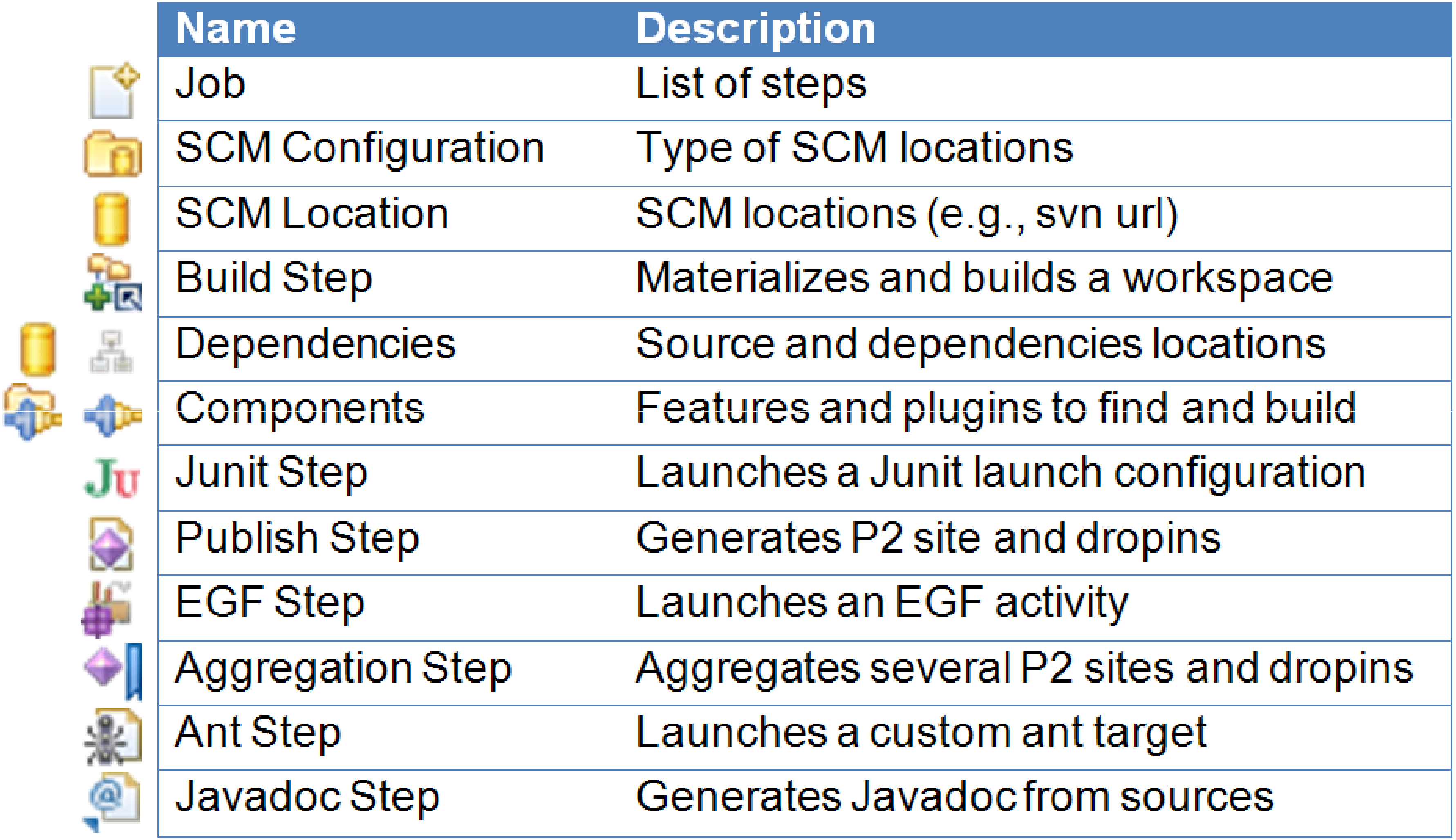
Latest Test Result (no failures)

Permalinks

- Last build (#45), 18 hr ago
- Last stable build (#45), 18 hr ago
- Last successful build (#45), 18 hr ago

Page generated: Mar 17, 2011 6:57:27 AM Hudson ver. 1.389

Portfolio #2 – Build Chain Editor & Generator – Nomenclature



Name	Description
Job	List of steps
SCM Configuration	Type of SCM locations
SCM Location	SCM locations (e.g., svn url)
Build Step	Materializes and builds a workspace
Dependencies	Source and dependencies locations
Components	Features and plugins to find and build
Junit Step	Launches a Junit launch configuration
Publish Step	Generates P2 site and dropins
EGF Step	Launches an EGF activity
Aggregation Step	Aggregates several P2 sites and dropins
Ant Step	Launches a custom ant target
Javadoc Step	Generates Javadoc from sources

[Practice] Hands On…

Links:

[Video] Build Chain Creation: http://vimeo.com/22033124

Examples:

[Eclipse] http://wiki.eclipse.org/EGF_Build_Portfolio

Exercices:

EGF Example – [Plug-in]
org.eclipse.egf.portfolio.eclipse.build.examples

General info about EGF

EGF
Eclipse Generation Factories

Project page: http://www.eclipse.org/egf
Wiki: http://wiki.eclipse.org/EGF
Blog: http://blanglois.blogspot.com/
Twitter: @LangloisBenoit