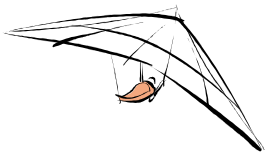
www.thalesgroup.com



EGF Start Up

Eclipse Generation Factories

Benoît Langlois / Thales Global Services

THALES

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

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

Generative Software

Generation scope?

How to develop & test?

Generation reusability?

Executability? Distribution?

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?

Issue	Dealing with generation complexity
Proposed solution	An integrated and extensible software factory framework
Purpose	Software Industrialization

What is EGF?

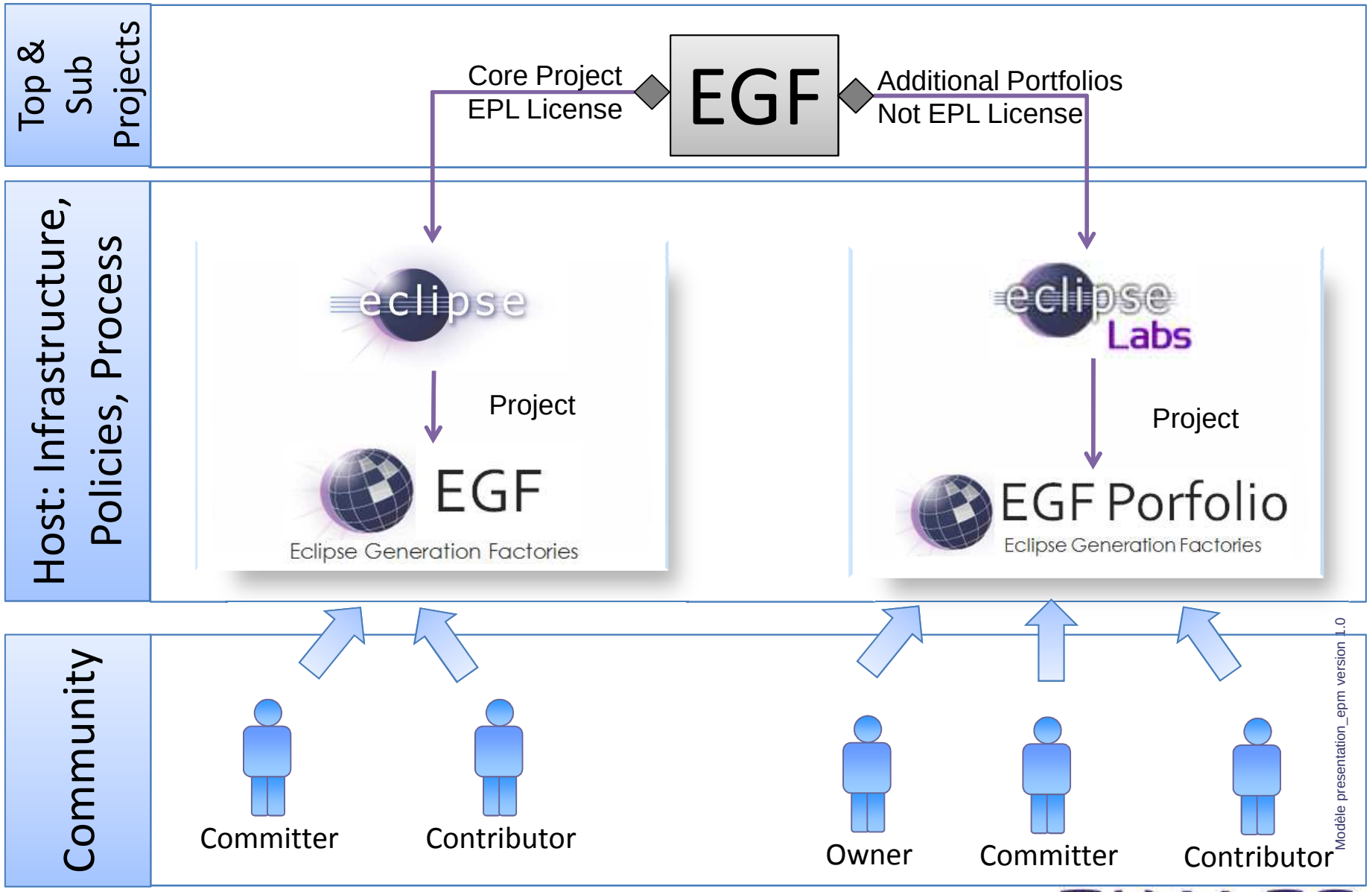
- **EGF (Eclipse Generation Factories) is an Eclipse open source component project in incubation under the EMFT project**
- **Purpose: providing a model-based generation framework**
- **Objectives:**
  - Supporting complex, large-scale and customizable generations
  - Promoting generation portfolios in order to capitalize on generation solutions
  - Providing an extensible generation structure

General info about EGF

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

Download: by update site from Indigo
or by update site from Amalgam

Project Organization

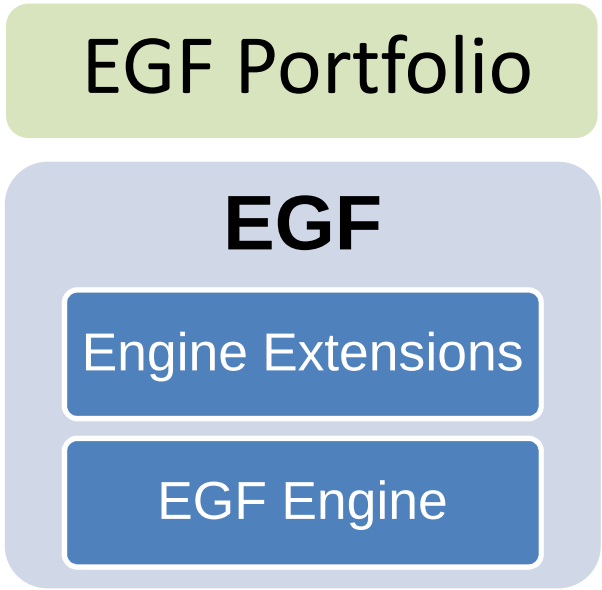


Modèle presentation_epm version 1.0

THALES

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

EGF Architecture



EGF Architecture

EGF Portfolio

EGF

Engine Extensions

EGF Engine

Provides basic metamodels and behaviors to automate software production

EGF Metamodel

Basic behaviors, dynamic execution

Factory component, task

Pattern

Modèle presentation_epm version 1.0

THALES

EGF Architecture

EGF Portfolio

EGF

Engine Extensions

EGF Engine

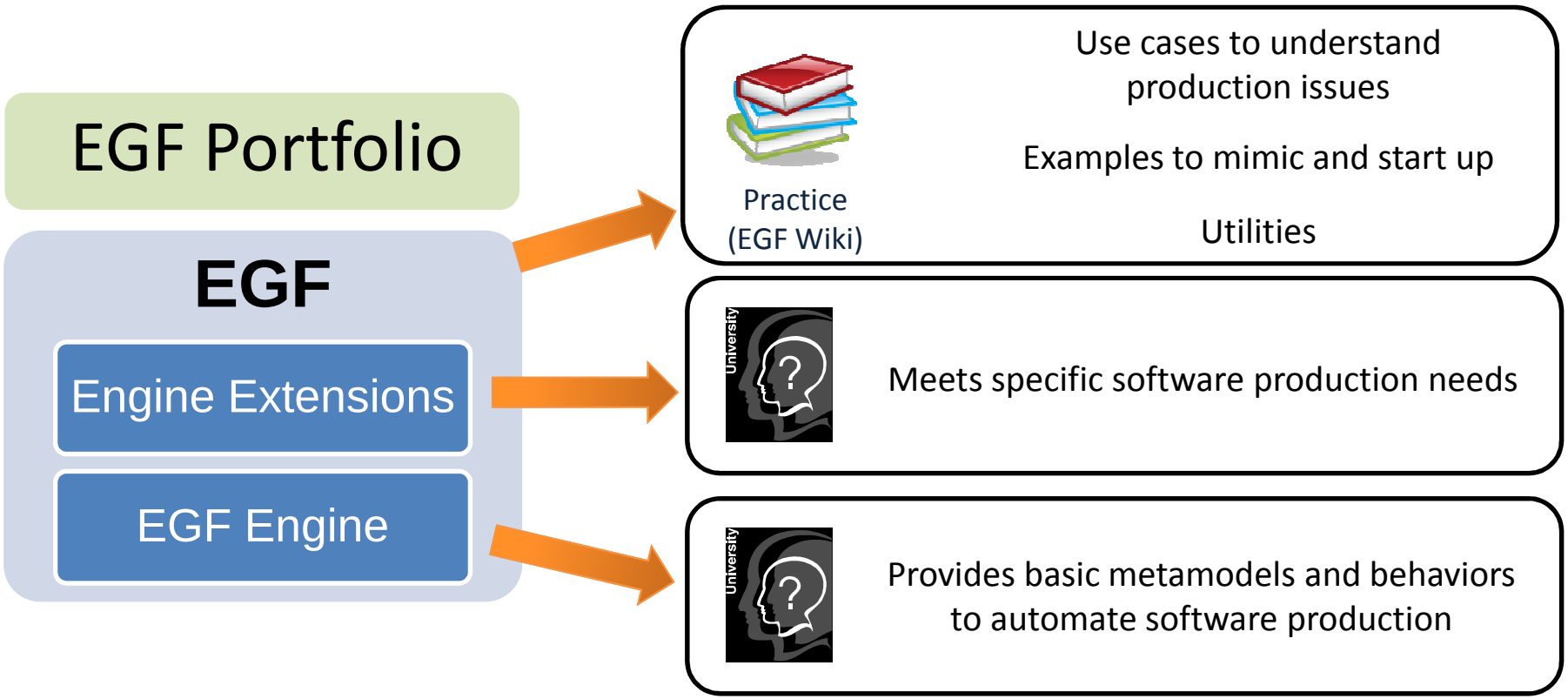
Meets specific software production needs

Language & tools interoperability

New types of generation formalisms

THALES

EGF Architecture



Modèle presentation_epm version 1.0

THALES

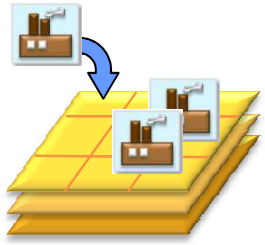
EGF Architecture

EGF Portfolio

EGF

Engine Extensions

EGF Engine



Portfolio

Portfolio = generation topic

Simple to off-the-shelf software factories

Meeting specific generation topics

Examples of Portfolios provided with EGF

Enhancement of the EMF Generation

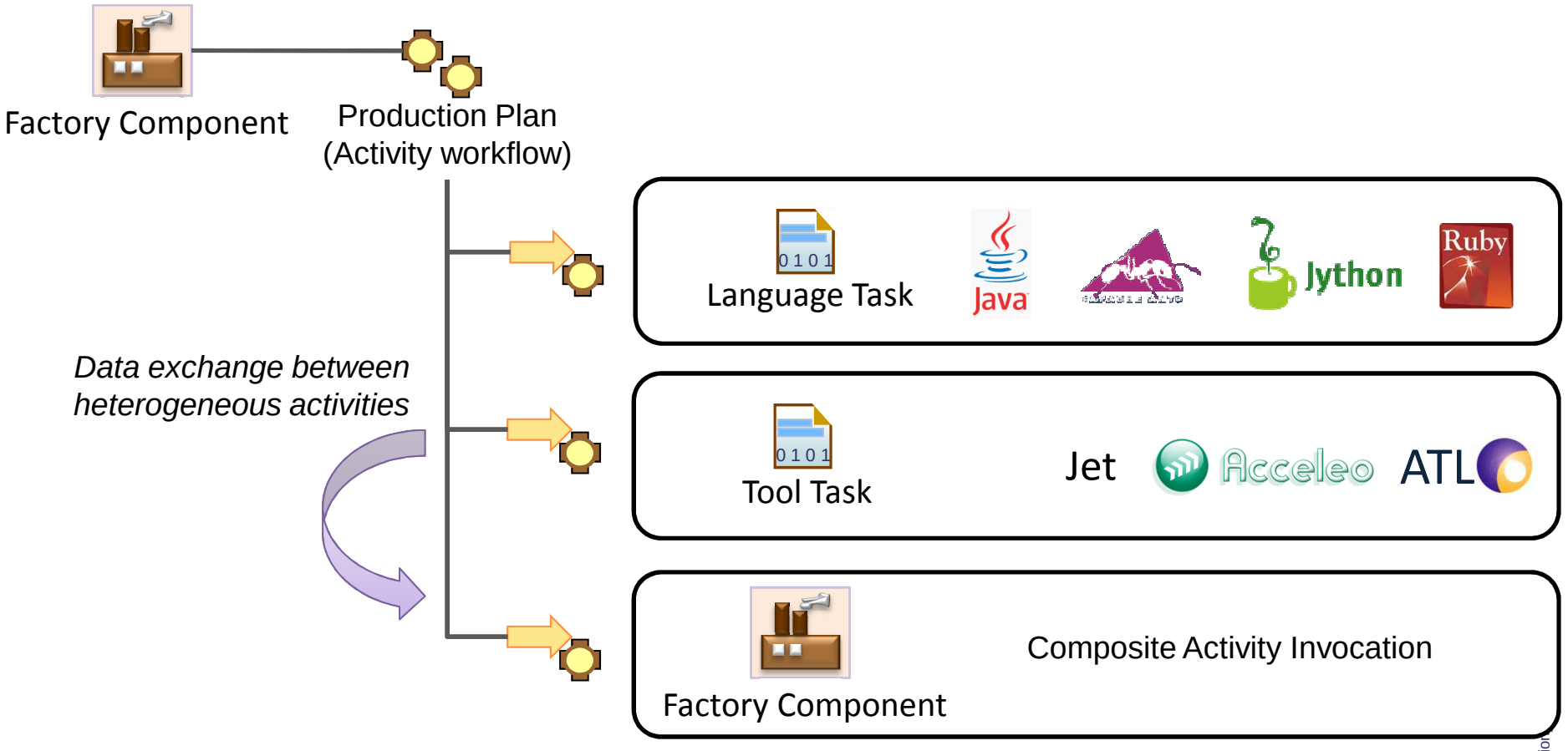
Build modeler and generator

Modèle presentation_epm version 1.0

THALES

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

Generation Composition, Language and Tool Interoperability

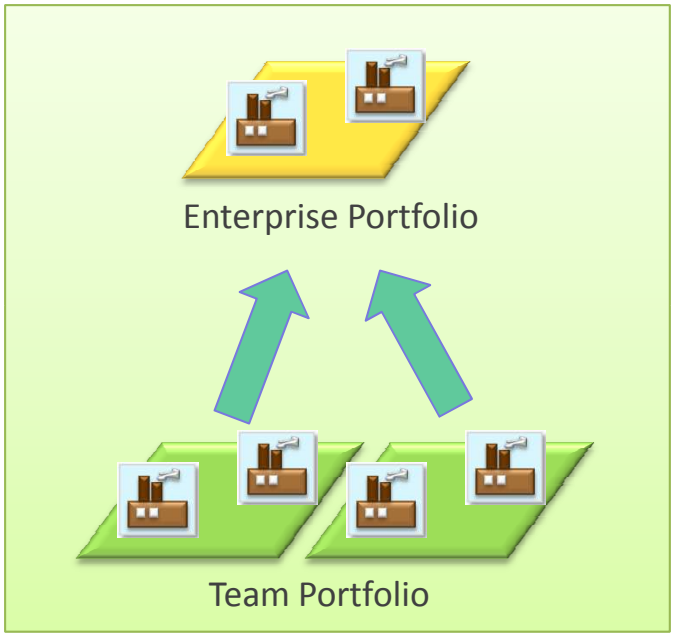


Activity Workflow with Java and Ruby: http://vimeo.com/15705526

Modèle presentation

THALES

Code Generation Families



Several levels of Customization

Example

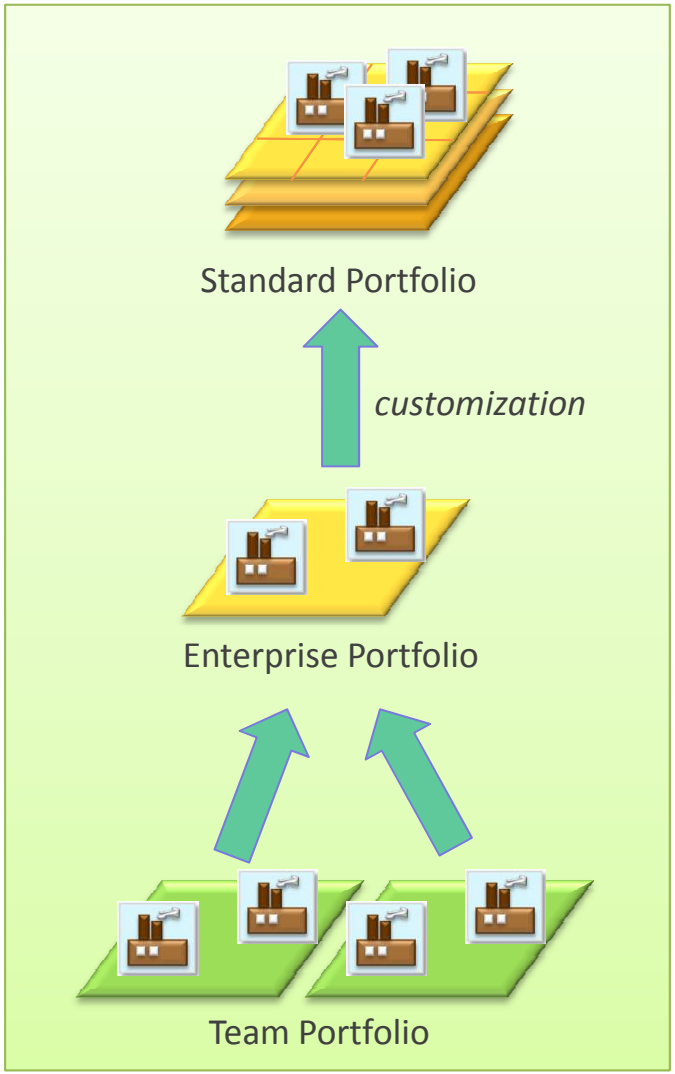
Code/textual generation
for my organization

extends

Specific generation
for my project

THALES

Code Generation Families



Example

EMF Generation

extends

EMF Generation
for my organization

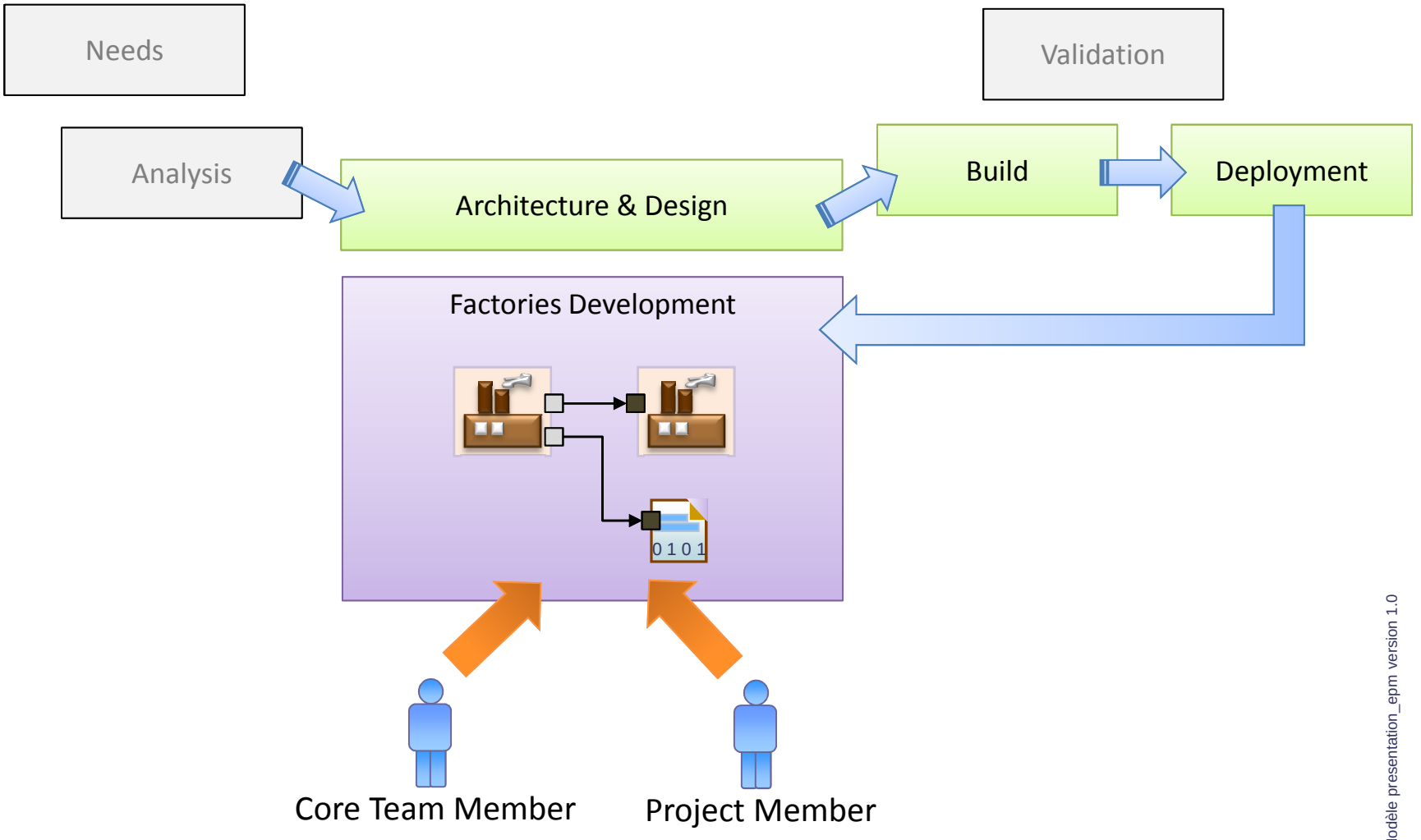
extends

EMF Generation
for my application

Several levels of Customization

THALES

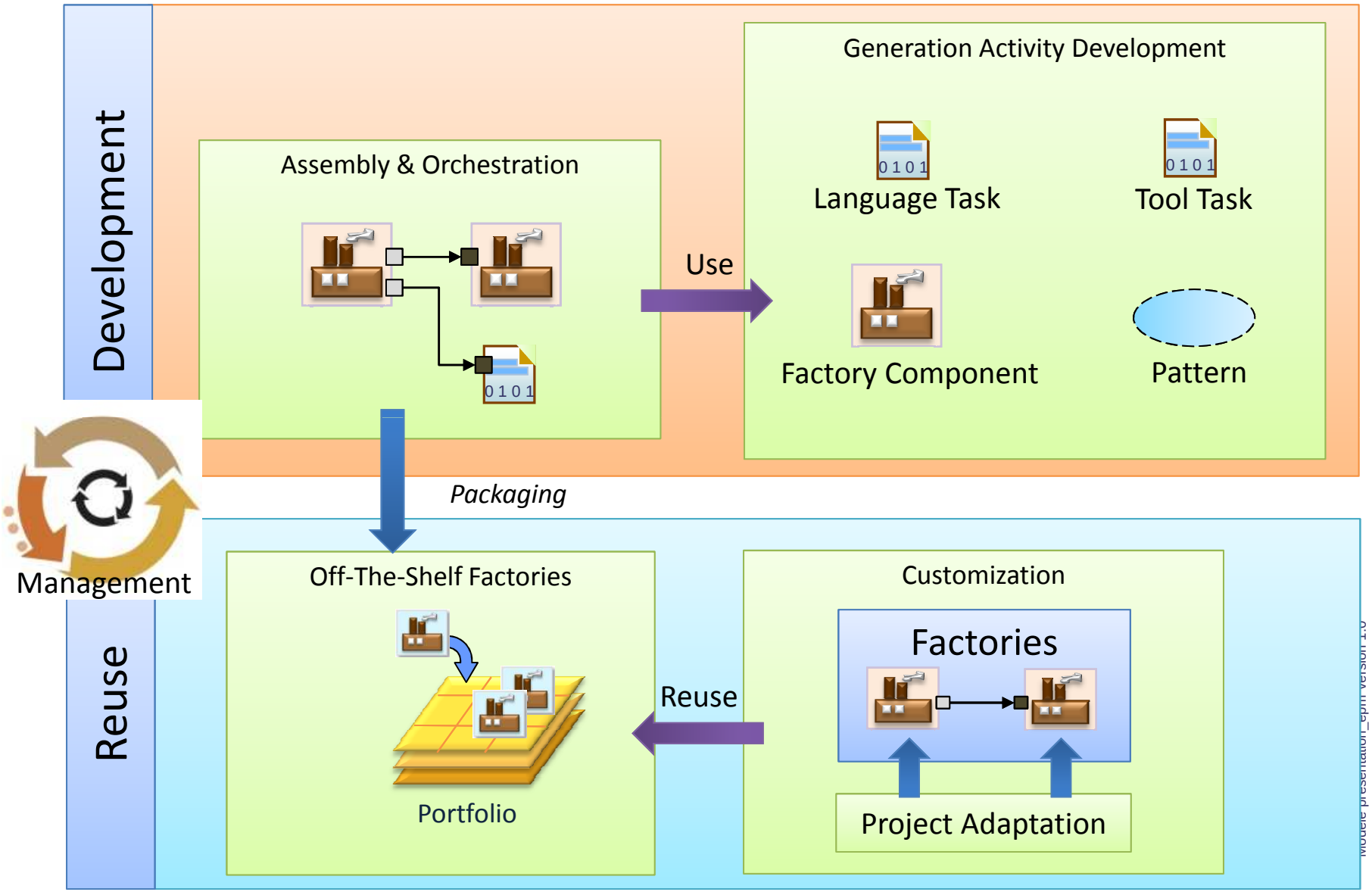
Generation Development in Multi-User Mode



Modèle presentation_epm version 1.0

THALES

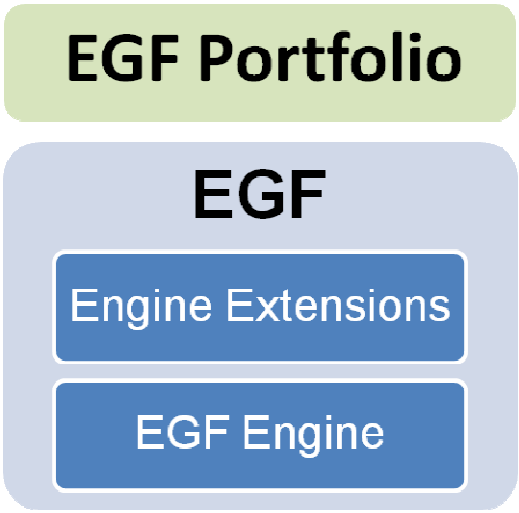
Software Factory Engineering



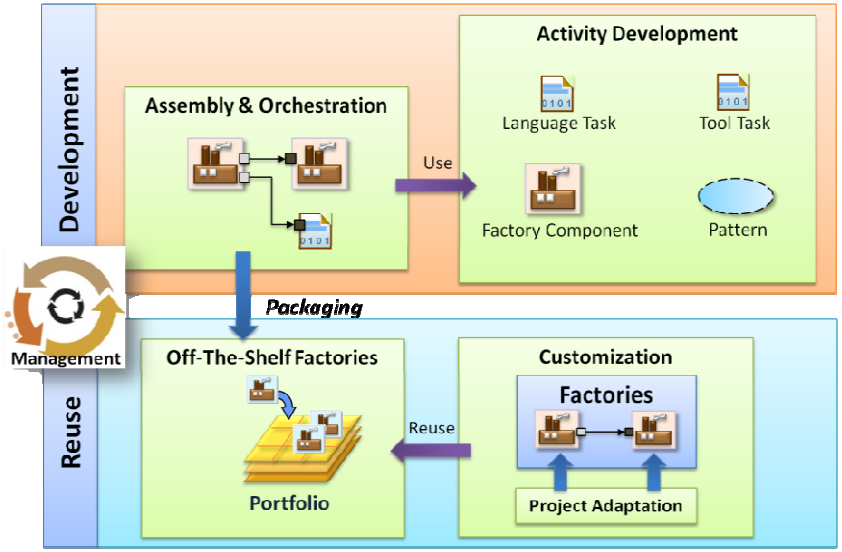
THALES

Software Factory Engineering

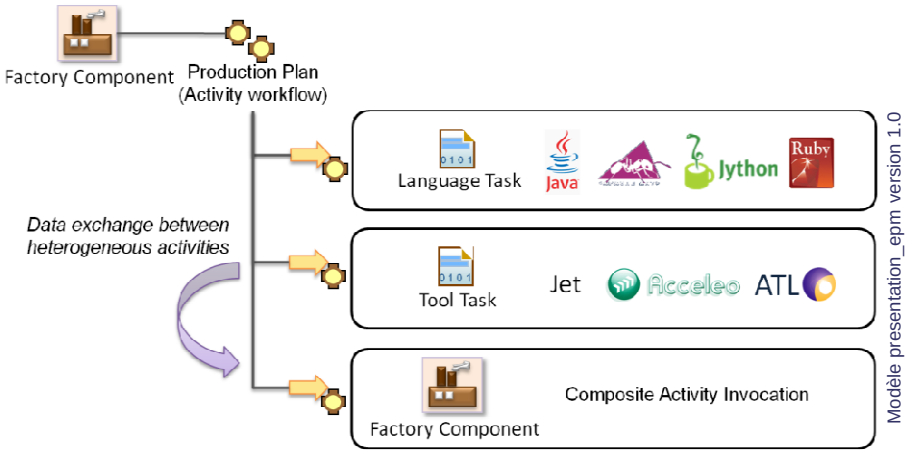
General Architecture of EGF



Development & Reuse Process with EGF



Example of EGF Factory



THALES

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

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

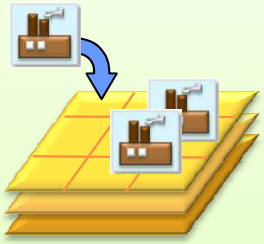
Presentation of the EGF Portfolios

- http://wiki.eclipse.org/EGF/Portfolio

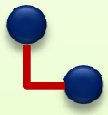
- Introduction
- EGF Architecture
- ***Concepts & Practice***
  - Generation Chain
  - Activity
  - Factory Component
  - Task
  - Pattern
- EGF Portfolios

Types of Generation Units in EGF

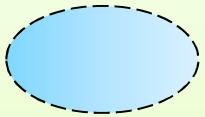
Factory Component
Composite generation unit with an activity workflow
Task
Leaf generation unit written in a language (e.g., Java, Ruby)



Portfolio
Capitalization 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

Modèle presentation_epr

THALES

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

Generation Chain - Objectives

Objectives:

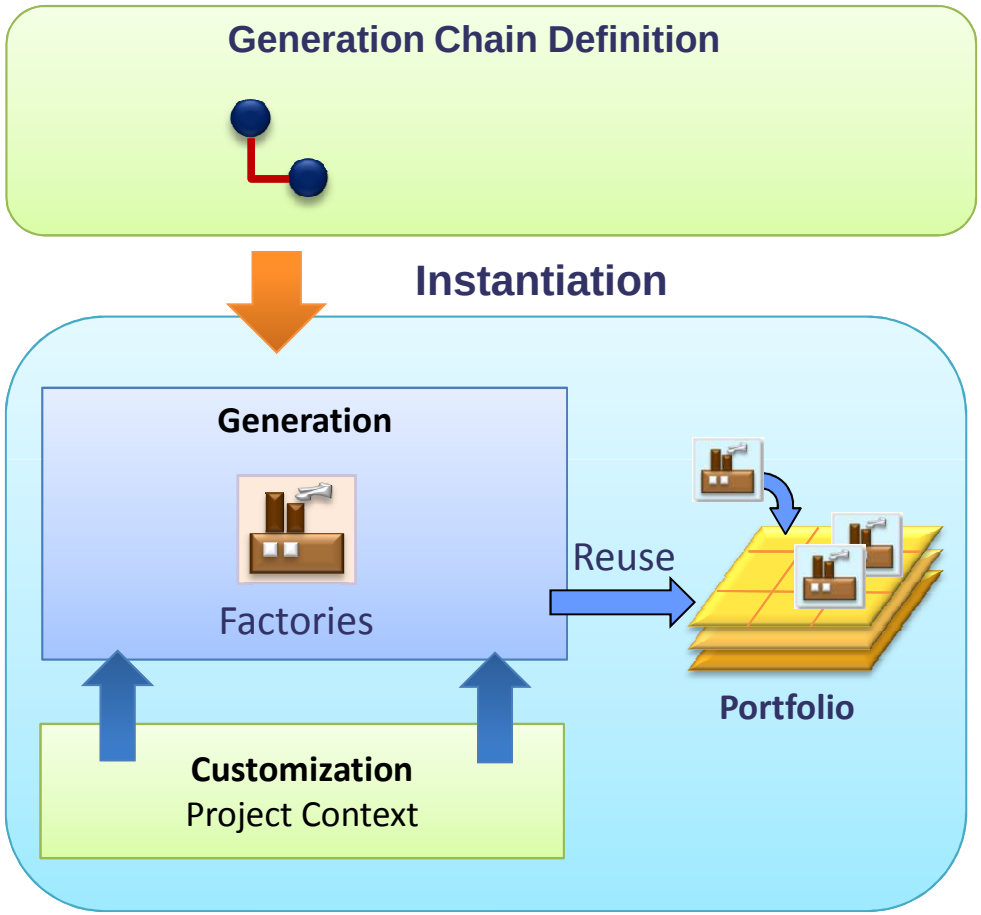
- **Definition, at a high level of description, of executable generations**
- **Abstraction: encapsulating the irrelevant technical details of generation**
- **Simplicity & Efficiency: Reducing the number of “clicks” (i.e. the number of actions)**
  - Only providing the main generation features and next generating

Generation Chain - Principles

Technical principles:

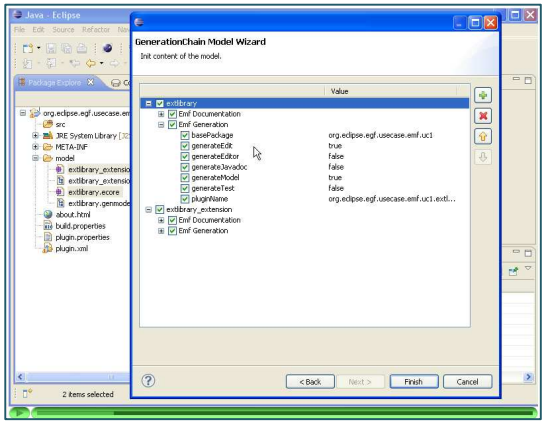
- Generation features are captured in a “generation chain” file
- An EGF fcore file is produced from the generation chain: it contains the translation of the generation chain into factory components
- Next, the factory components are transparently executed to produce the expected artifacts
- It is possible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

Instantiation & Customization of Generation Chains



THALES

[Practice] Hands On…



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

THALES

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

Activity – Definition

An activity is the abstract class of executable EGF generation units

- Factory component and Task are activities

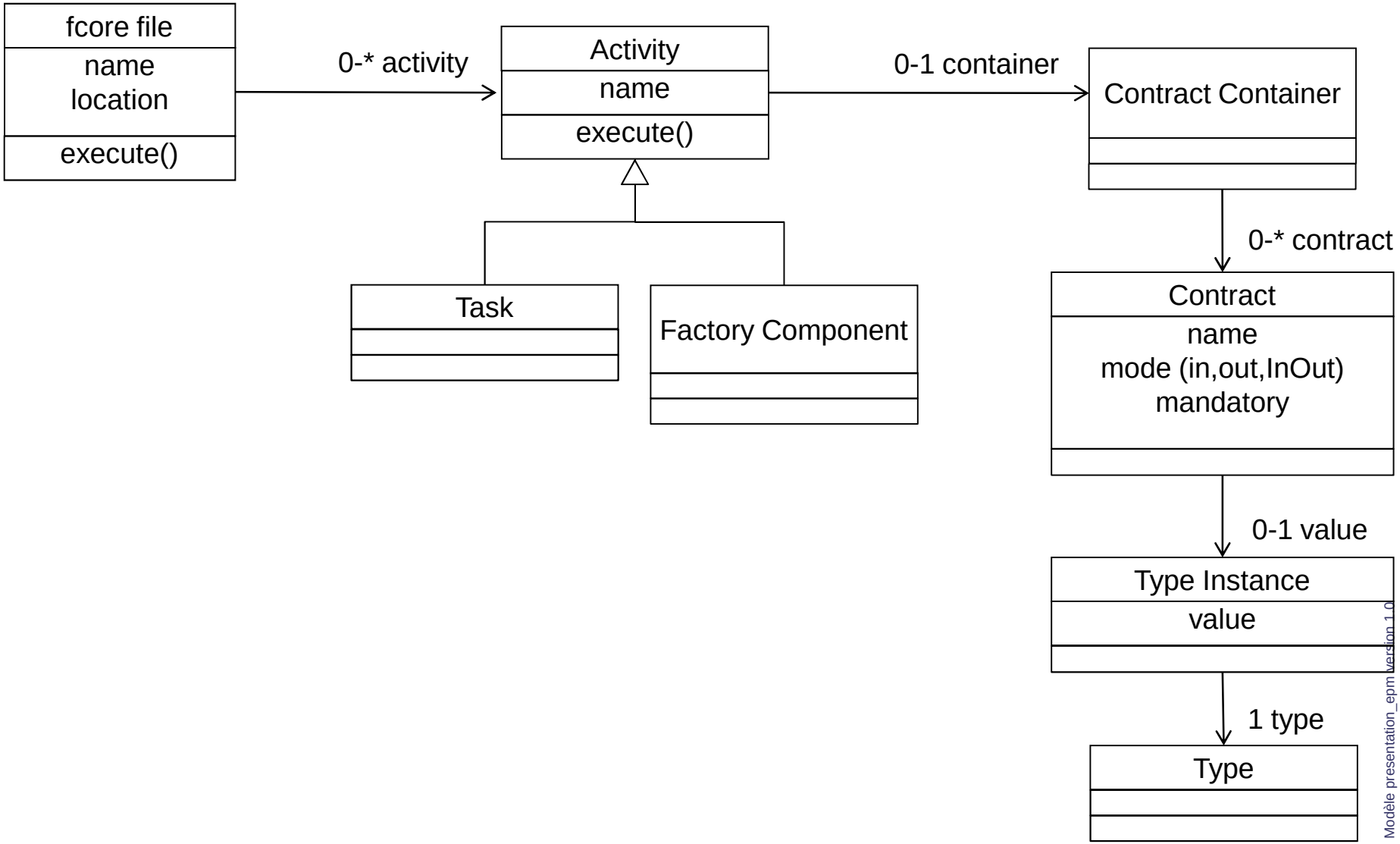
Activity storage

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

Activity properties

- Contract declaration
- Ability to be invoked and to execute a generation action

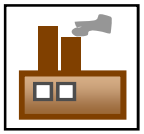
Concepts



THALES

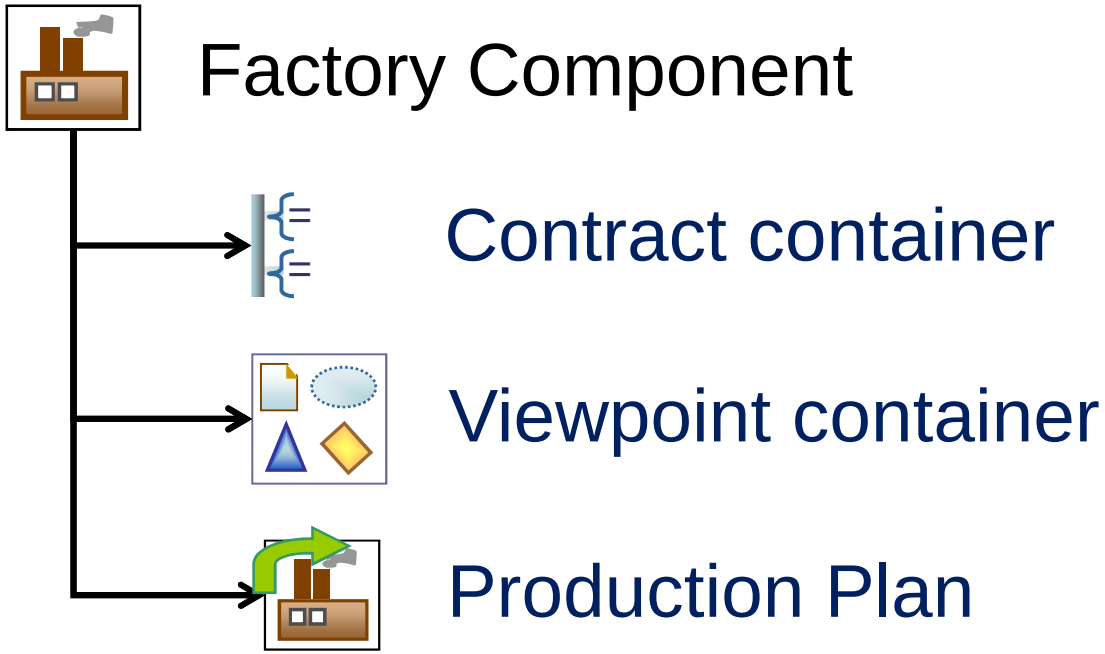
- Introduction
- EGF Architecture
- ***Concepts & Practice***
  - Generation Chain
  - Activity
  -  **Factory Component**
  - Task
  - Pattern
- EGF Portfolios

Factory Component

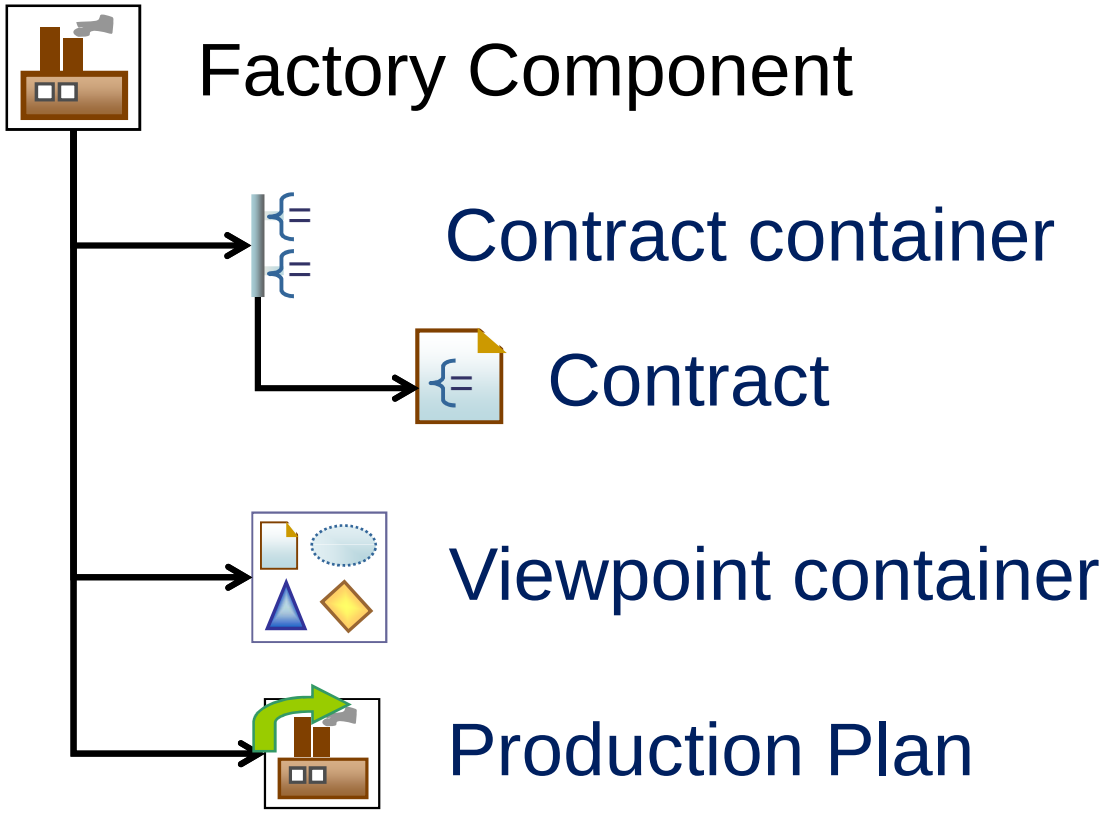


- **Unit of generation with a clear objective of generation**
- **Unit of generation with a clear contract**
- **Assembly of factory components**
  - Delegation to other activities
  - Creation of heterogeneous and complex generations
- **Generation workflow located in a production plan**
- **Explicit declaration of generation data organised by viewpoints**
- **A Factory Component can be edited and executed in the same Eclipse session**

Factory Component Structure

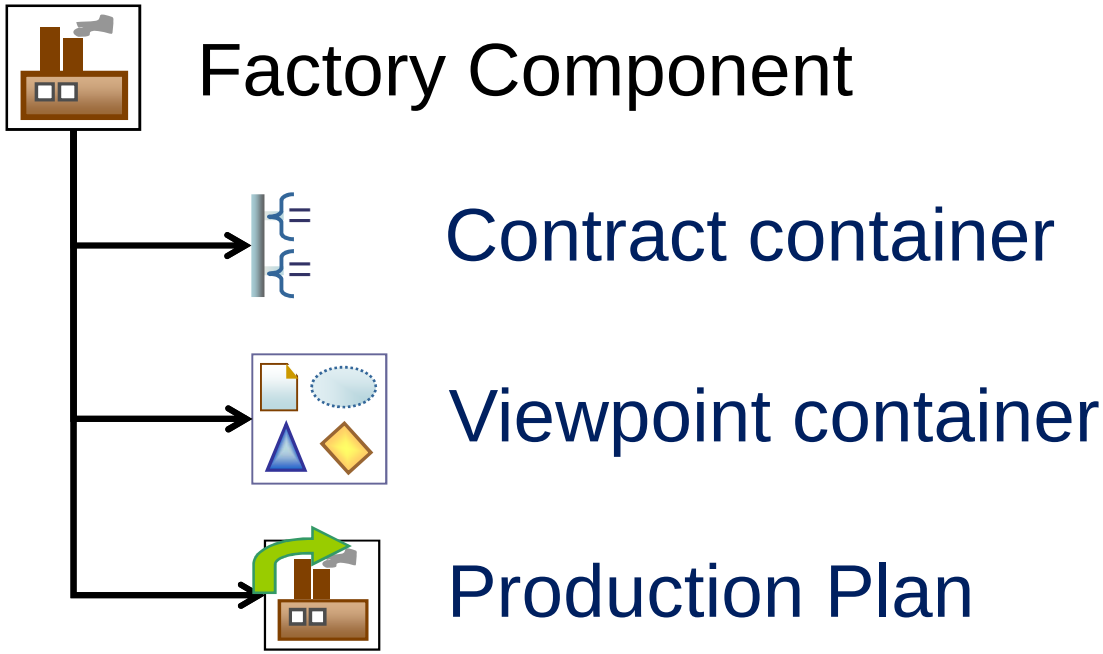


Factory Component Structure



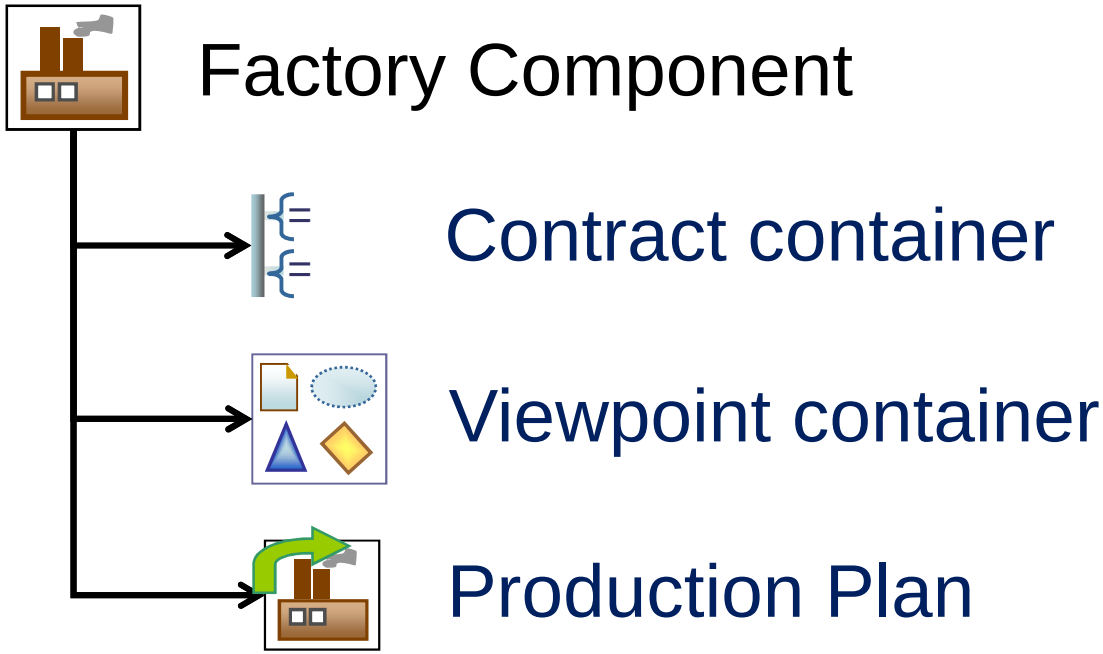
- **Contract:** Factory component parameter
- A contract has a type, a passing mode (In/Out/In_Out), a default value or not, is mandatory or optional

Factory Component Structure



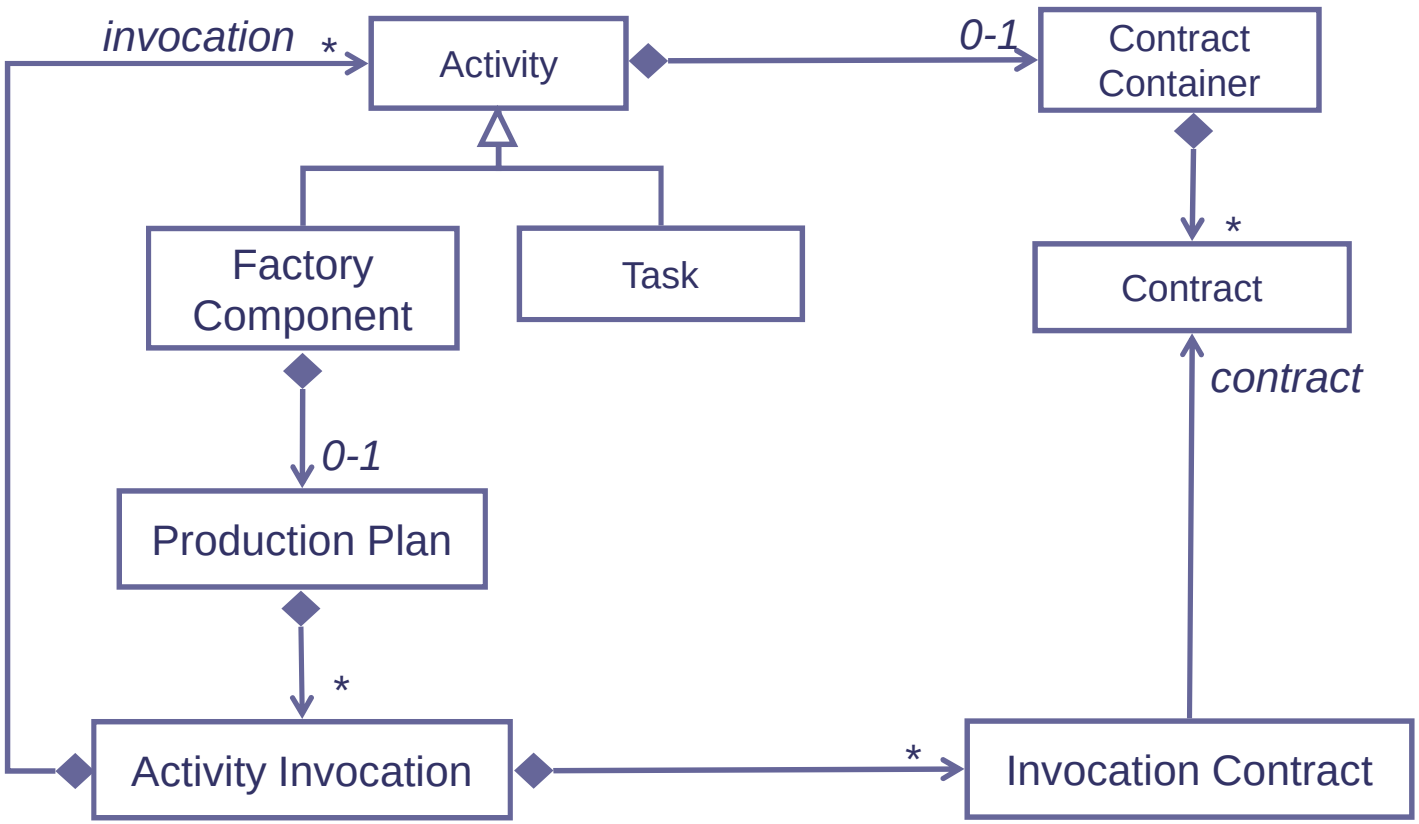
- **Viewpoint:** area to declare concerns of generation data
- Examples of viewpoint:
  - Available today: domain declaration, pattern
  - Candidates: licensing, feature model

Factory Component Structure



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

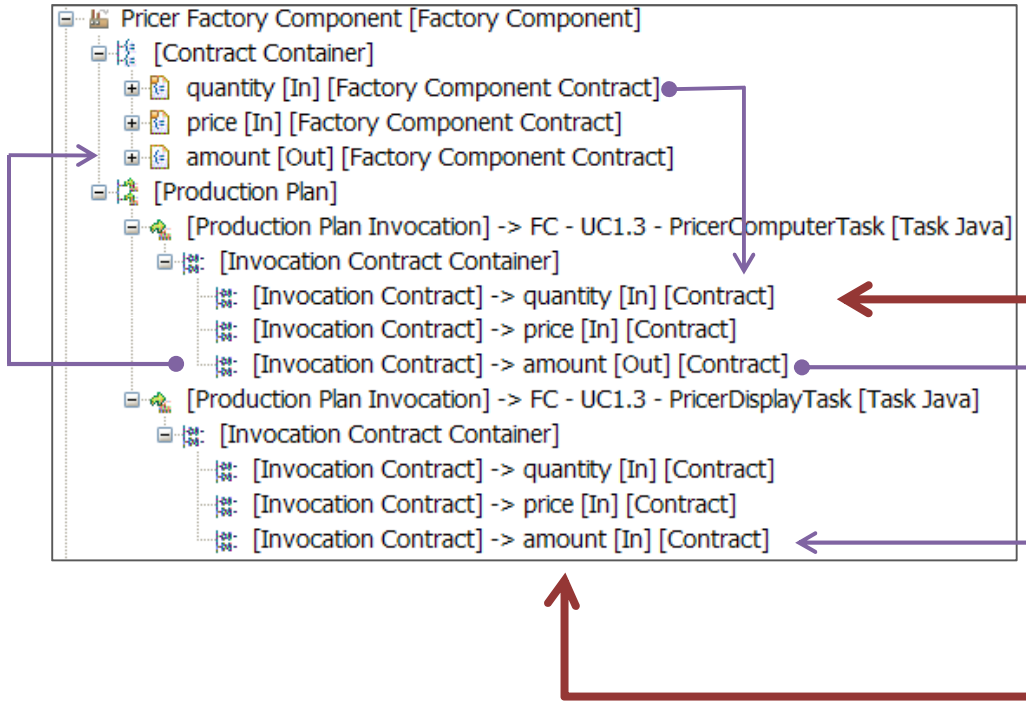
Activity and Production Plan



Modèle presentation_epm version 1.0

THALES

In/Output relationships – Example



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	

THALES

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

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:

- An implementation is associated to a task
- 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

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

THALES

Pattern – Definition

Definition

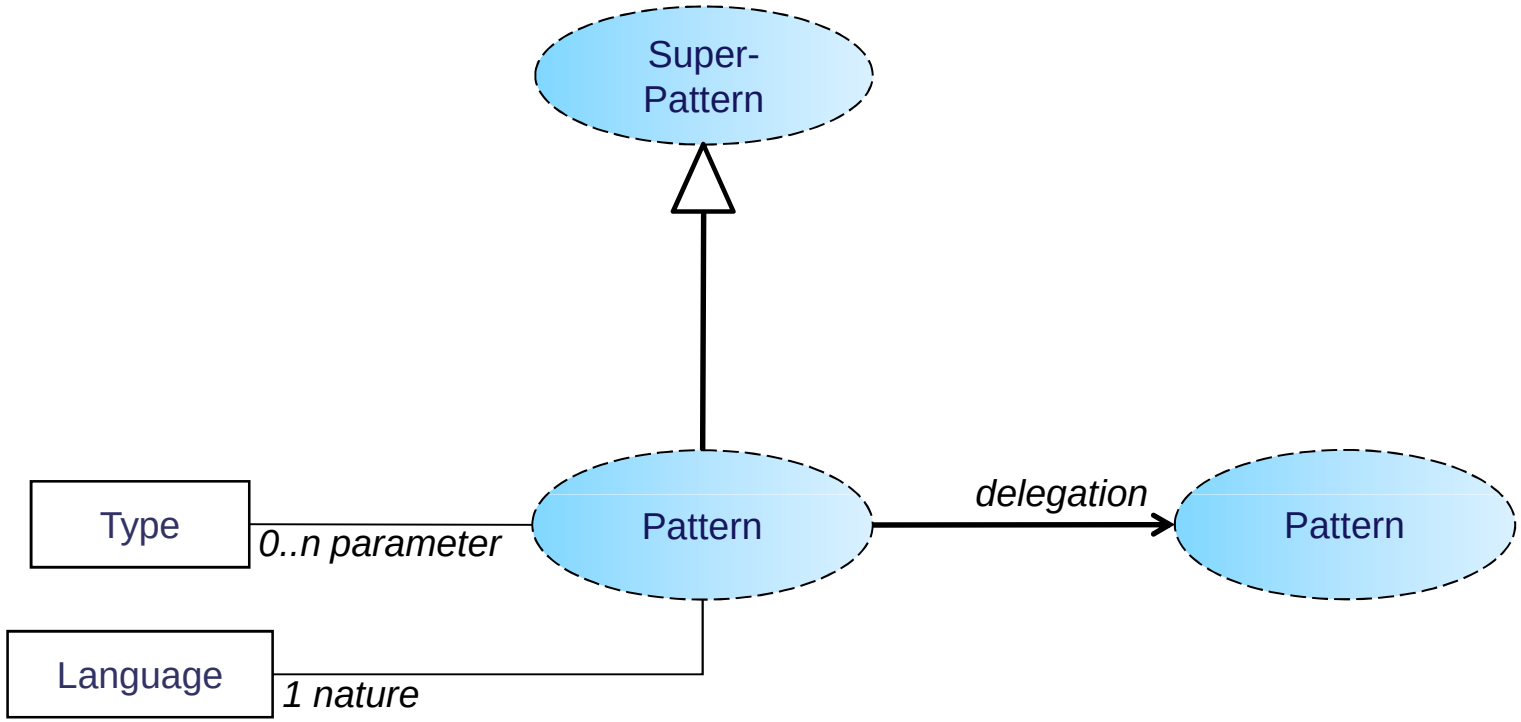
- A pattern is a solution to a recurrent generation problem

Purpose

- Applying a systematic behavior onto a resource
- Clearly dissociating the specification (external view) from the implementation (internal view) of the behavior
- Reusing and customizing a pattern in different contexts
- Supporting multilingual patterns in order to apply the best programming language to a situation, and then supporting multi-paradigm (M2T, M2M, T2M, T2T)
  - Java and Jet are the two first languages supported today

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

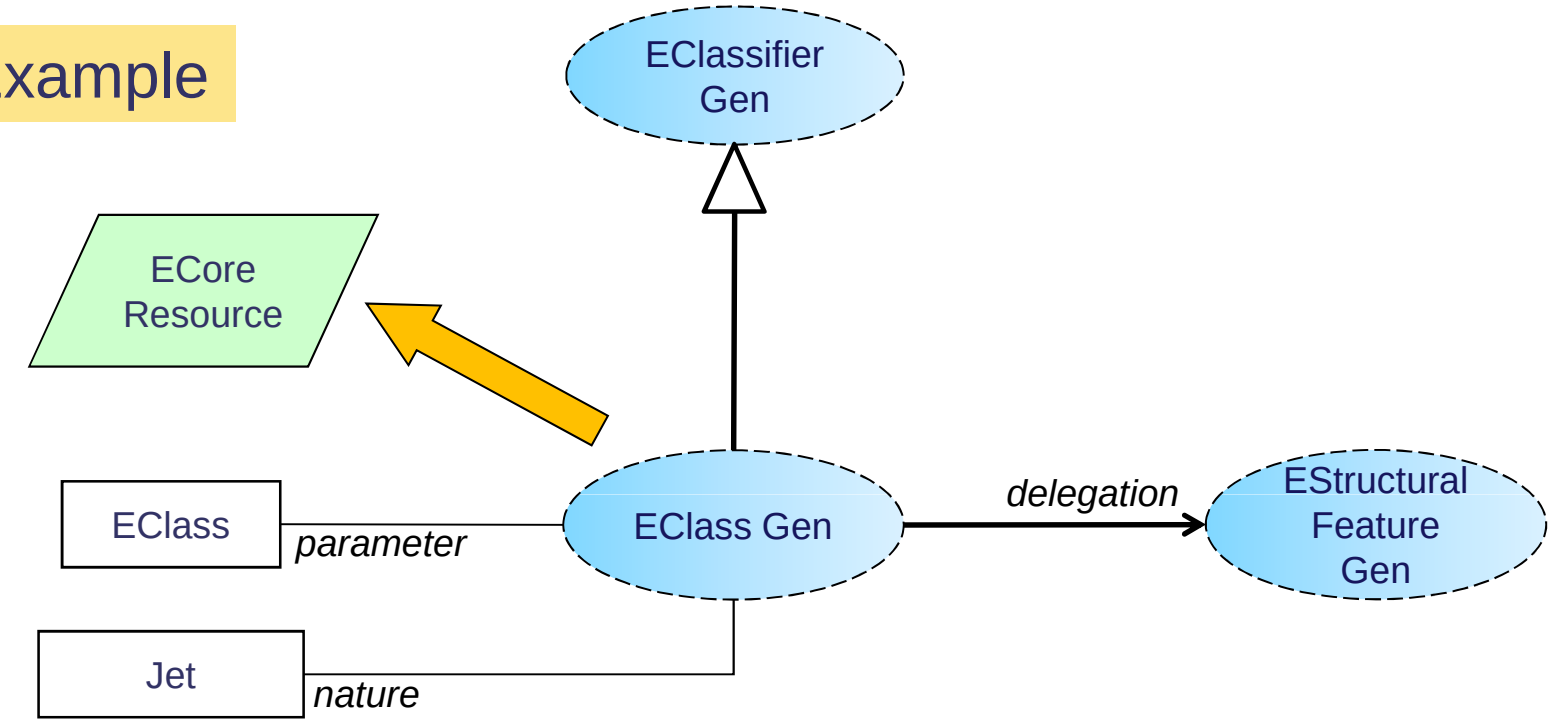
Pattern Structure – Specification Elements



- **Parameter**: Object 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)

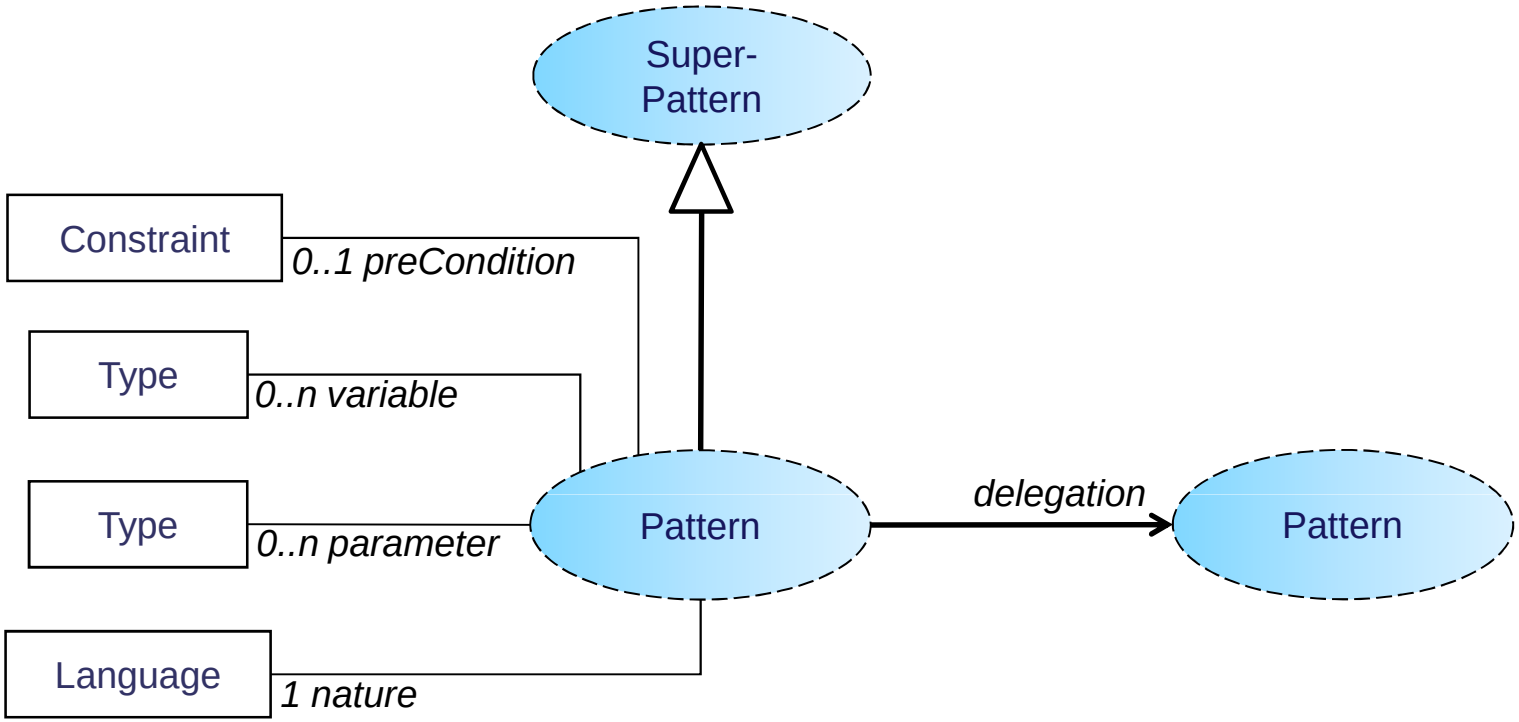
Pattern Structure – Specification Elements

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

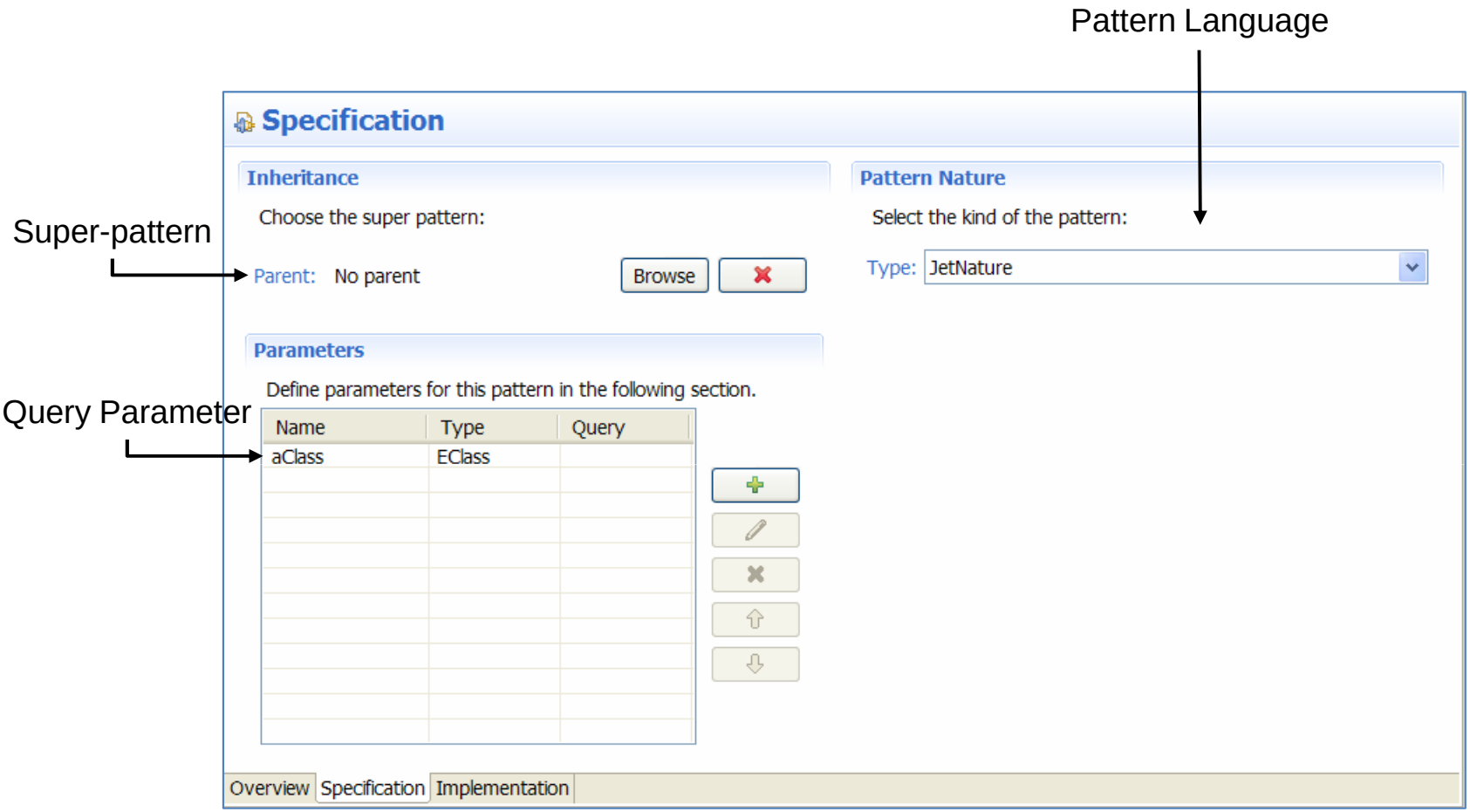
Pattern Structure – Specification Elements



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

THALES

Pattern Structure – Specification View

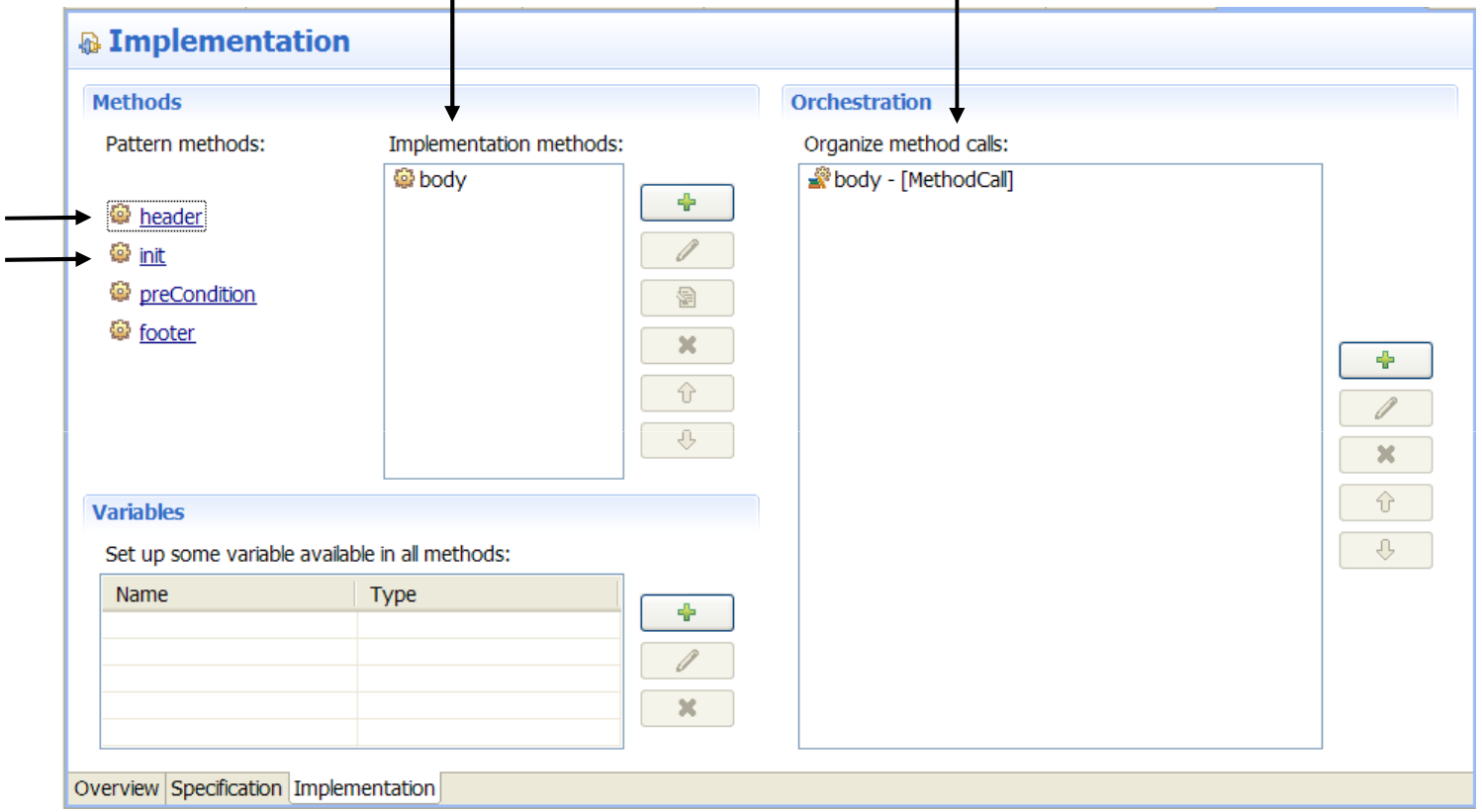


THALES

Pattern Structure – Implementation View

Methods which implement the pattern
They conform to the pattern language

Order to execute the methods



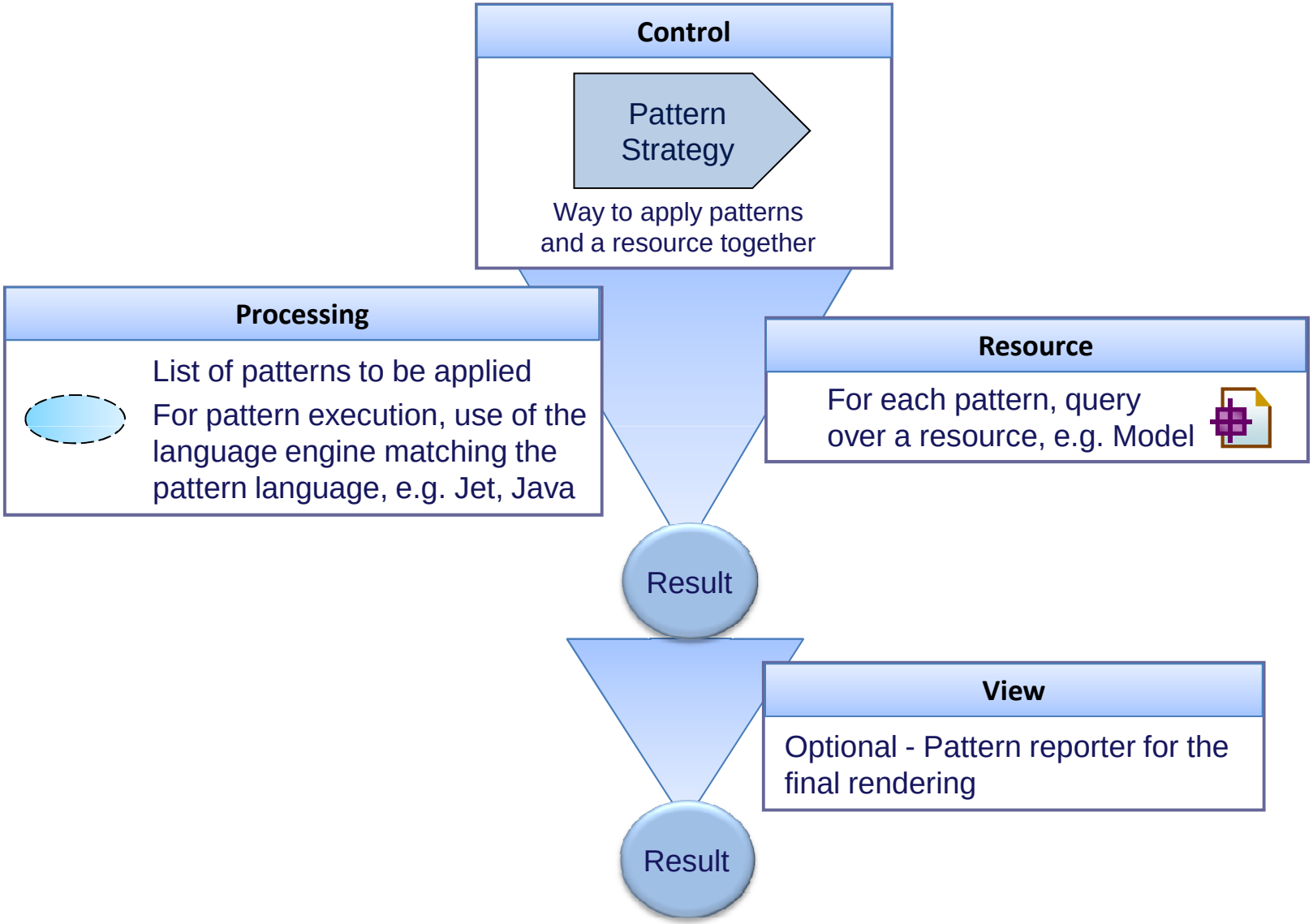
- **header:** typically used for the Jet header
- **init**: method for pattern initialization (e.g., variable initialization)
- A method editor allows editing pattern methods

THALES

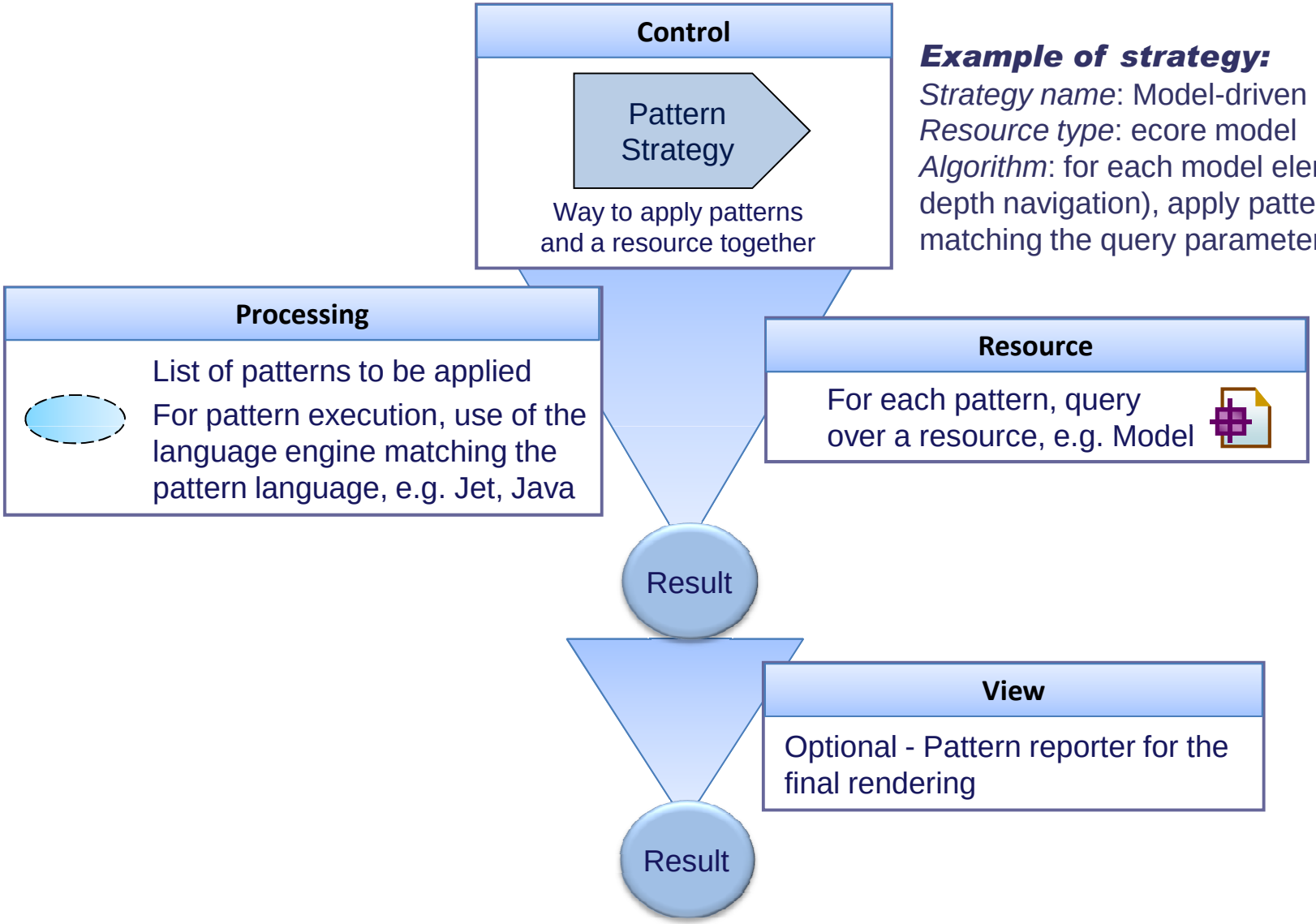
- Introduction
- EGF Architecture
- ***Concepts & Practice***
  - Generation Chain
  - Activity
  - Factory Component
  - Task
  - **Pattern**
    - Pattern Structure
    - **Pattern Execution**
- EGF Portfolios

THALES

Pattern Execution – Big Picture



Pattern Execution – Big Picture



Example of strategy:

Strategy name: Model-driven Strategy
Resource type: ecore model
Algorithm: for each model element (in-depth navigation), apply patterns matching the query parameters

Pattern Strategy

Definition: Way to apply patterns against a resource

Examples of strategies:

- **Model-driven pattern strategy**: in-depth navigation over a model, and for each model element, applying a set of patterns
- **Pattern-driven strategy**: for each pattern, applying the pattern for each model element
- **[Data type]-driven strategy**: generalization of the approach; instead of model, it could be any type of resource (e.g., file directory)

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.

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

THALES

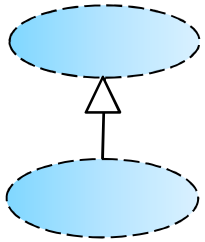
Pattern Relationships – Introduction

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

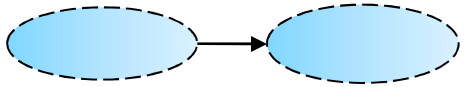
The next slides present the different kinds of pattern relationships

Pattern Relationships

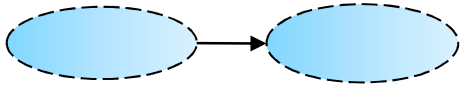
Pattern Inheritance



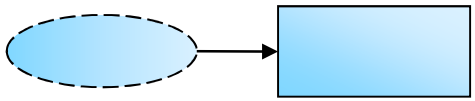
Pattern Delegation



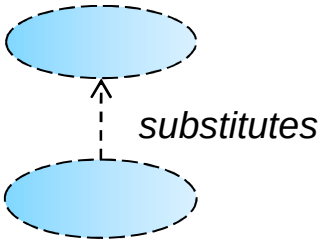
Pattern Injection



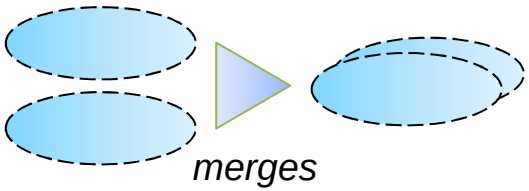
Pattern Callback



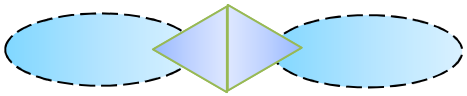
Pattern Substitution



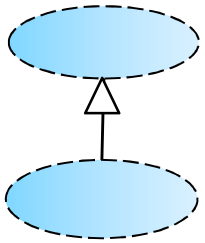
Pattern Merge



Pattern Comparison



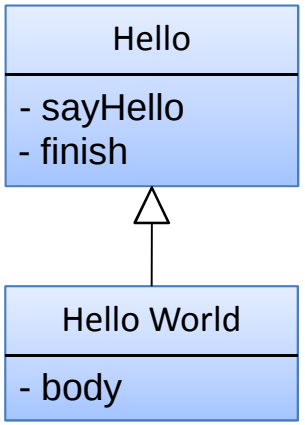
Pattern inheritance



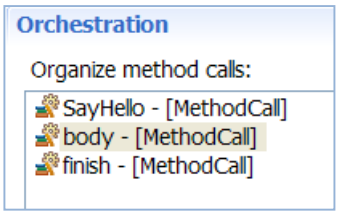
Case 1. Reuse of super-pattern methods

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

Example



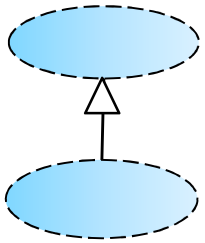
Orchestration of HelloWorld



THALES

Pattern Inheritance

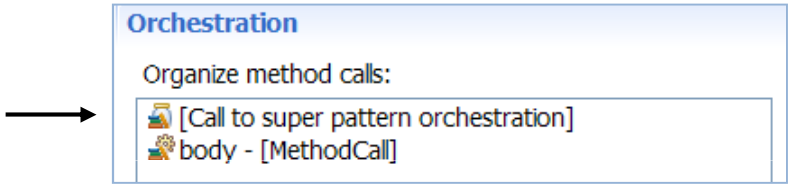
Pattern inheritance



Case 2. Reuse of super-pattern orchestration

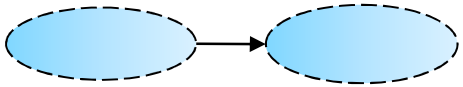
Reuse of method and orchestration defined in the super-pattern
This abstracts the super-pattern orchestration
This avoids rewriting pattern orchestration
Just adding the methods of the current pattern

Example



Pattern Delegation

Pattern delegation

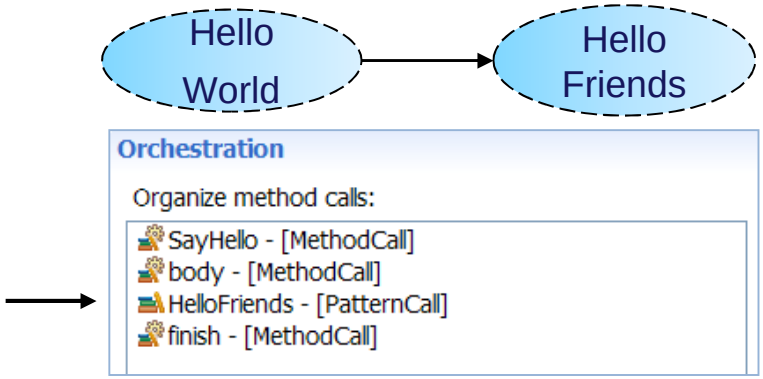


delegates

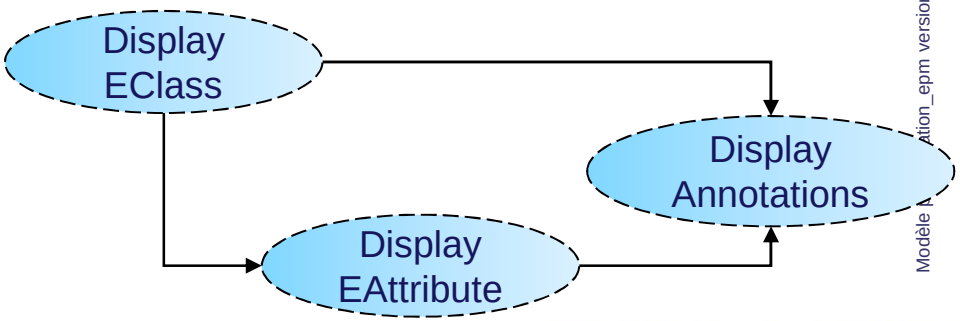
Case. For Problem decomposition & Reuse of pattern

- 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

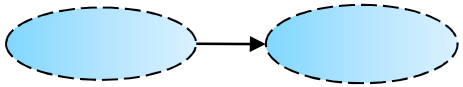


Example 2



THALES

Pattern delegation

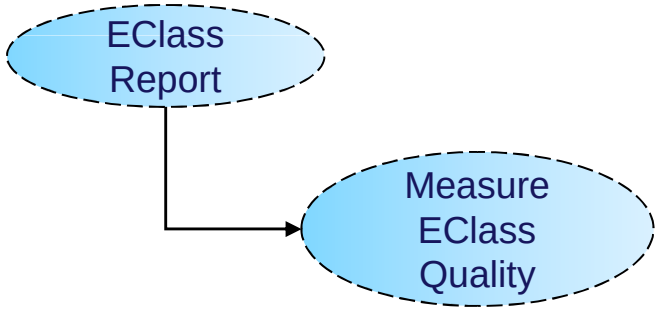


delegates

Case. Pattern delegation when implementation languages are different

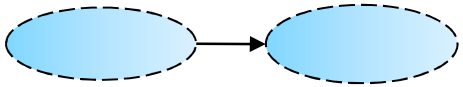
This corresponds to a Pattern Delegation where Pattern natures are different. For instance, a Pattern with a Jet nature calls a Pattern with a Java nature in order to differently process the same resource.
It is impossible to have different natures in the same Pattern inheritance hierarchy.

Example



Pattern Injection

Pattern injection

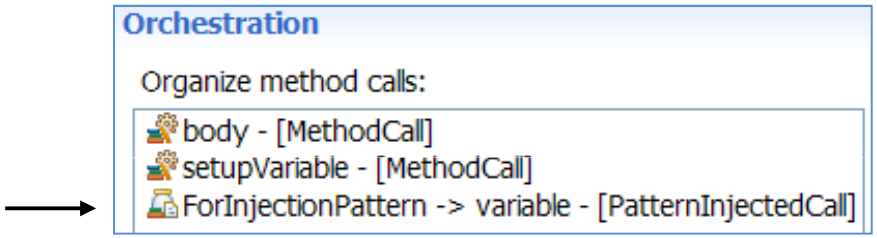


injects

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

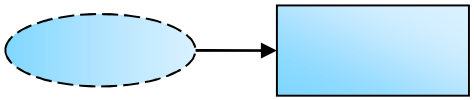
- A Pattern injection corresponds to a Pattern Delegation, but
- The parameter values are dynamically set at pattern execution

Example



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

Pattern Callback

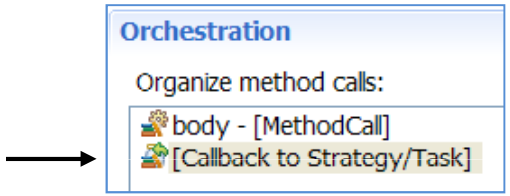


Case 1. Applying a Java call

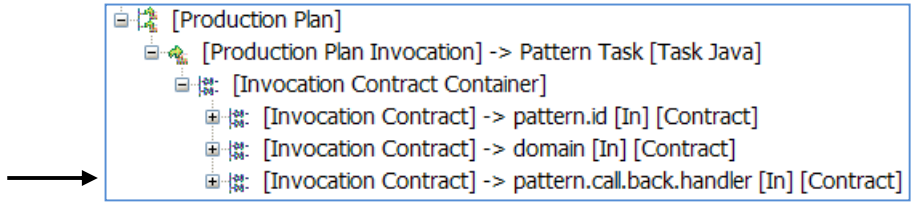
The callback indicates where the callback on a Java Class is applied

Example

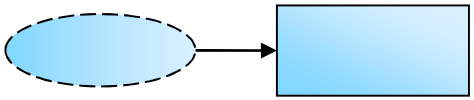
Pattern orchestration



Specification of the Java Class in the production plan



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.

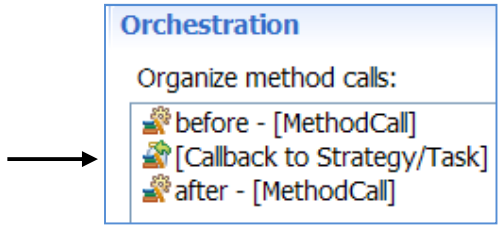
Example

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">
```

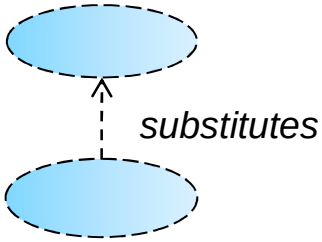
Generation result



THALES

Pattern Substitution

Pattern substitution



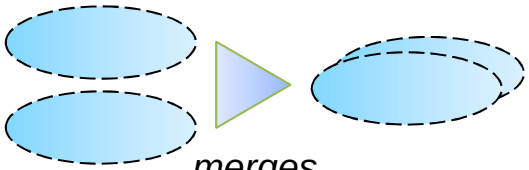
Case. Customization of a pattern-based generation

- 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



Case. Combination of pattern lists

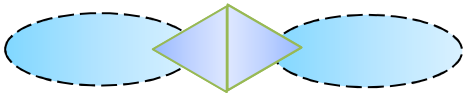
- Two patterns lists are merged into one list
- Examples: for customization, merging a local substitution with a pattern list in parameter of factory component

For deeper understanding

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

THALES

Pattern Comparison

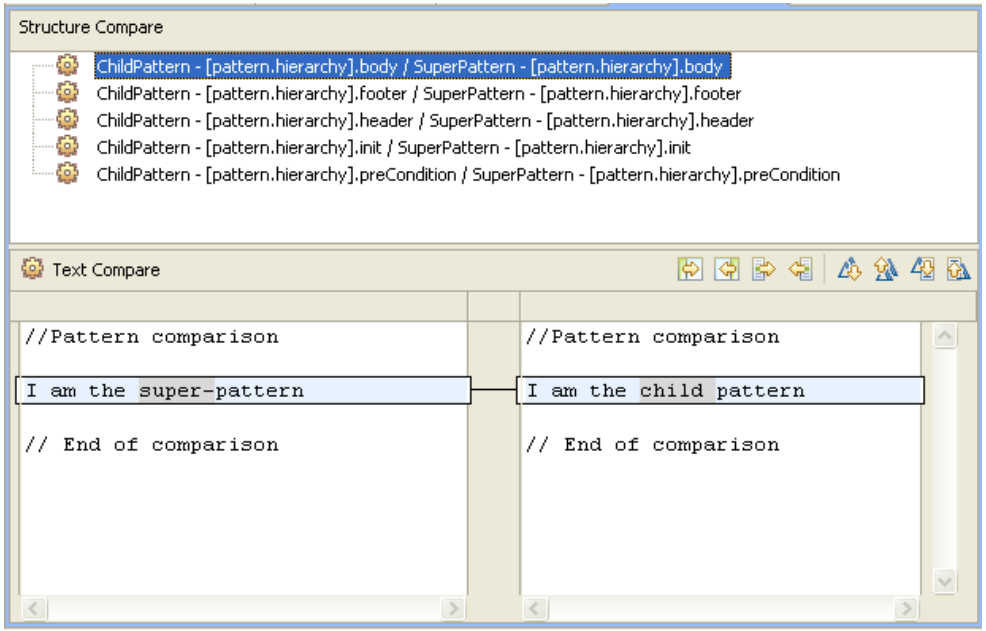


Case. Used during pattern edition – Face pattern evolution when pattern-based generation scales up

- 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.



THALES

[Practice] Hands On…

Links:

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

Examples:

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

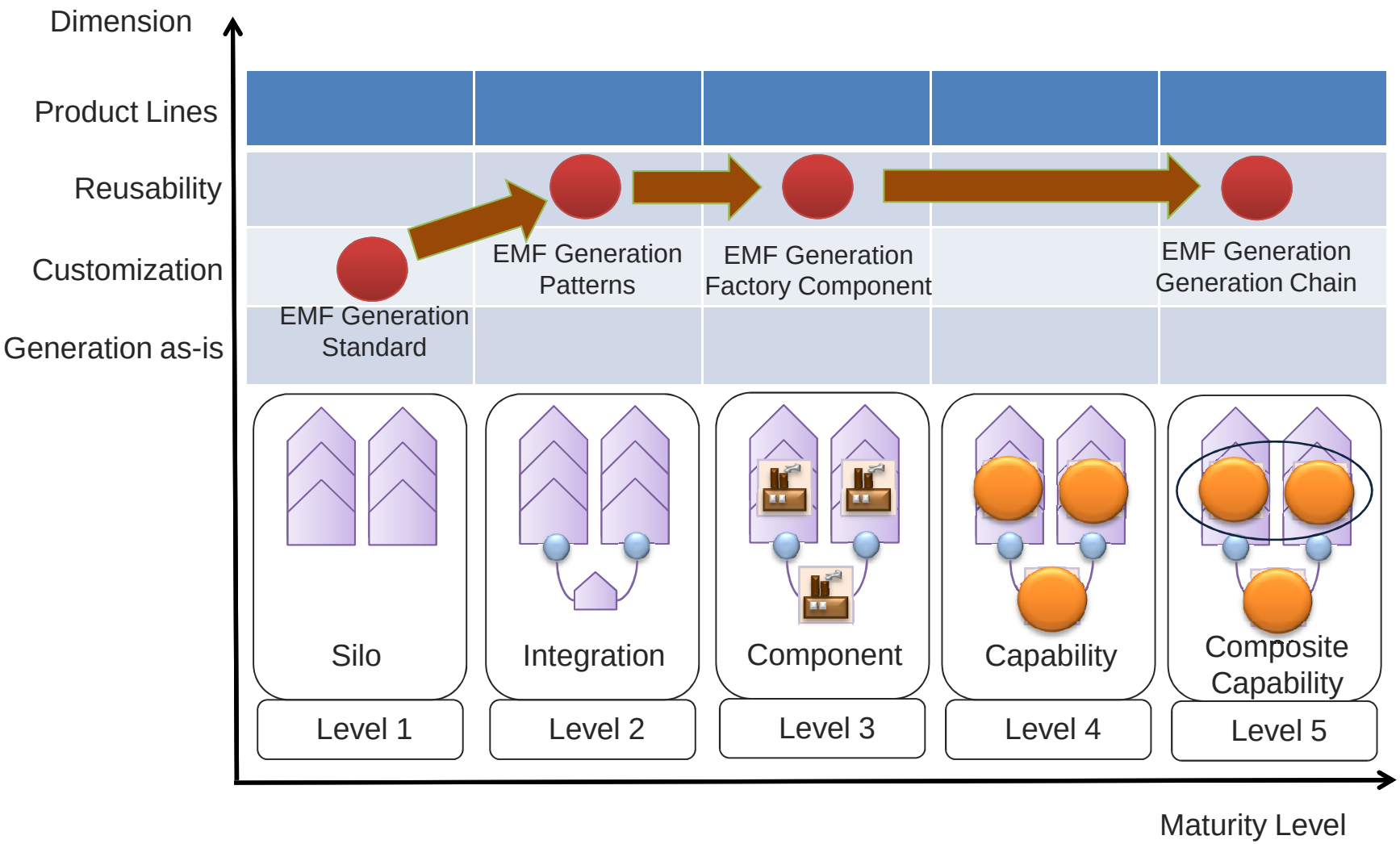
Exercices:

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

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

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

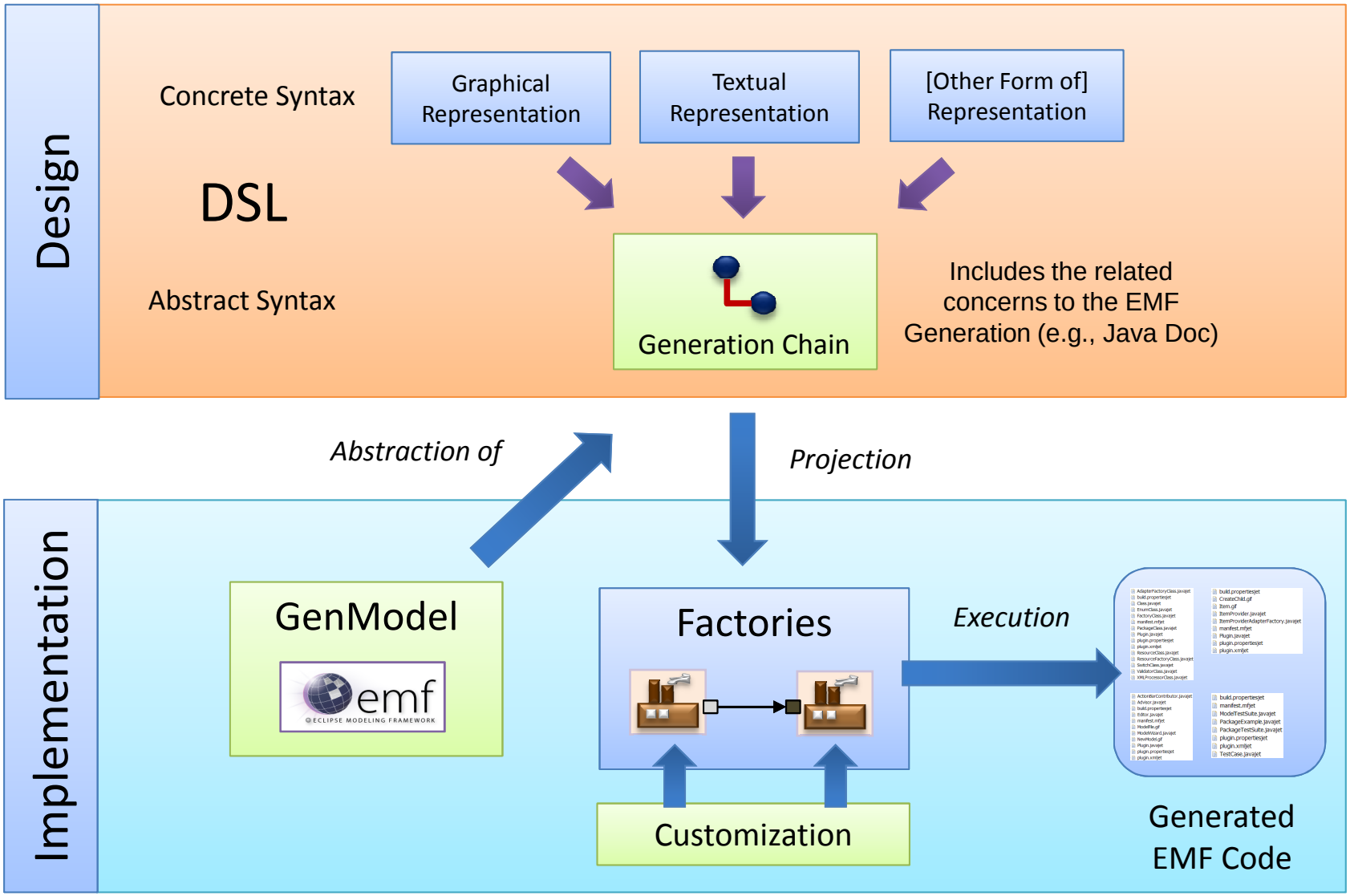
EMF Generation – Levels of Maturity



Modèle presentation_epm version 1.0

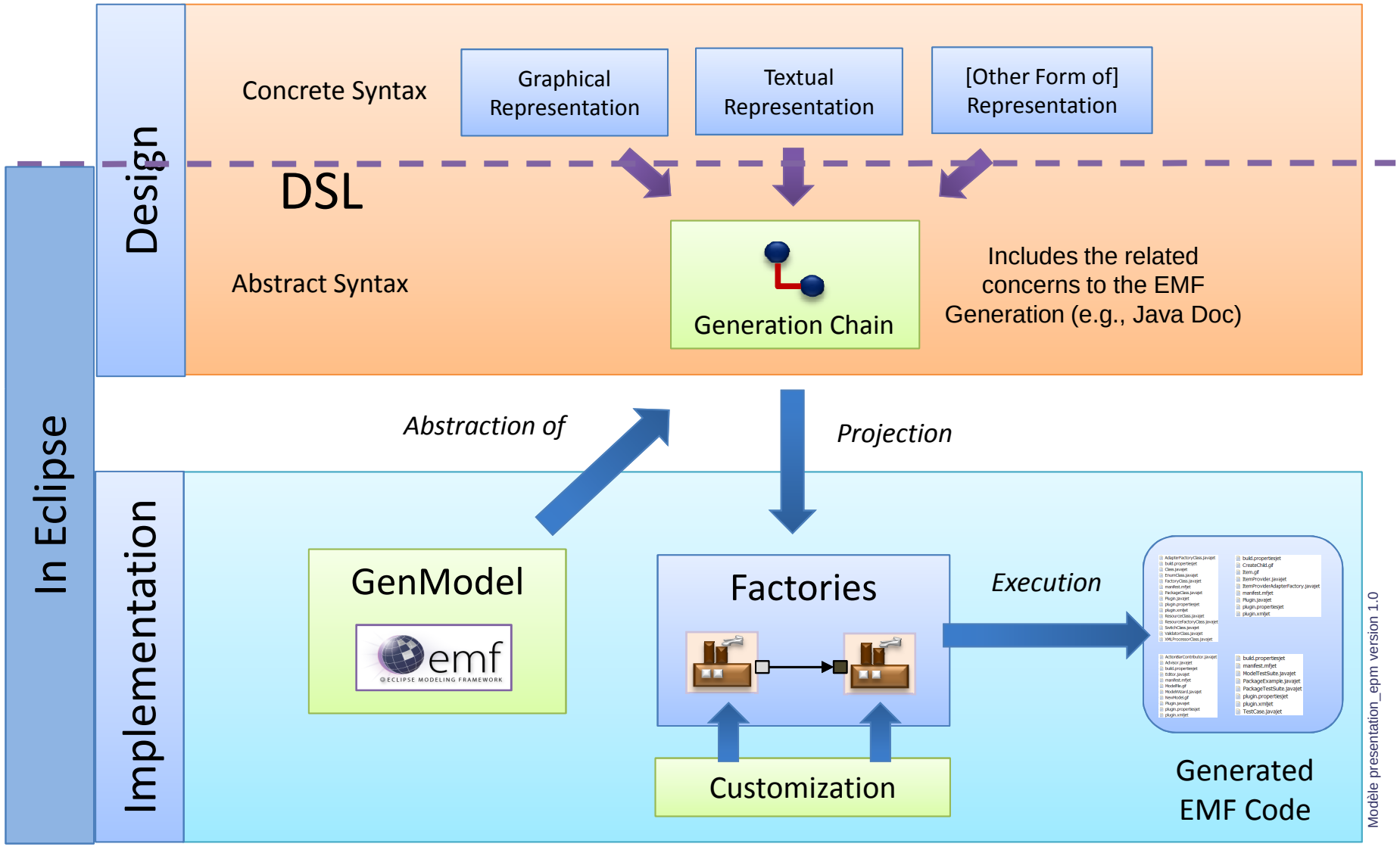
THALES

Portfolio #1 – EMF Generations Layers



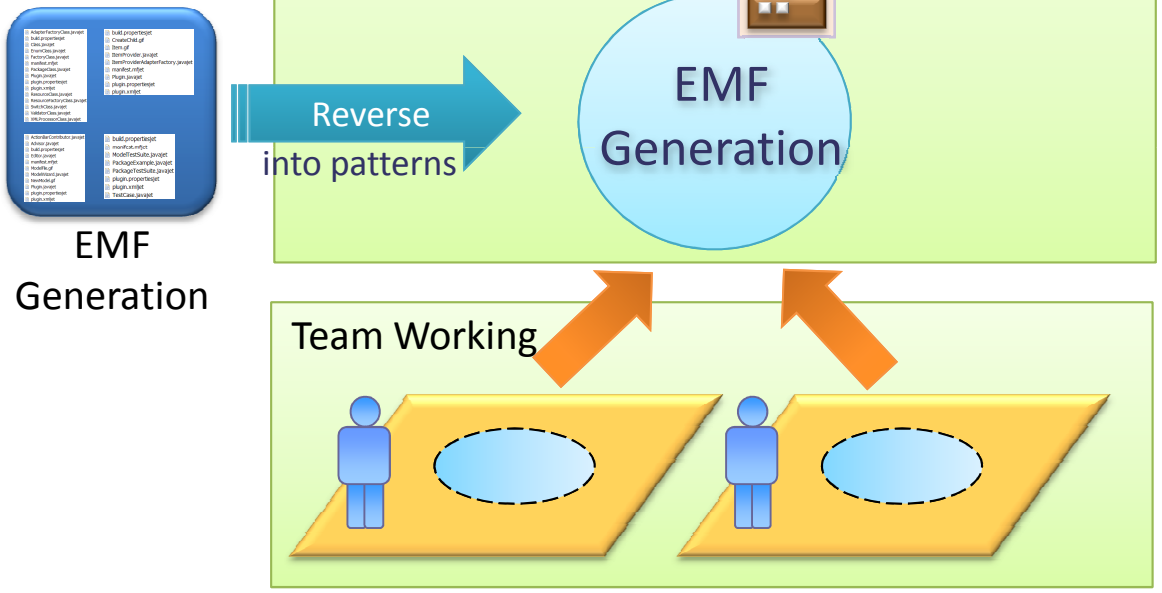
THALES

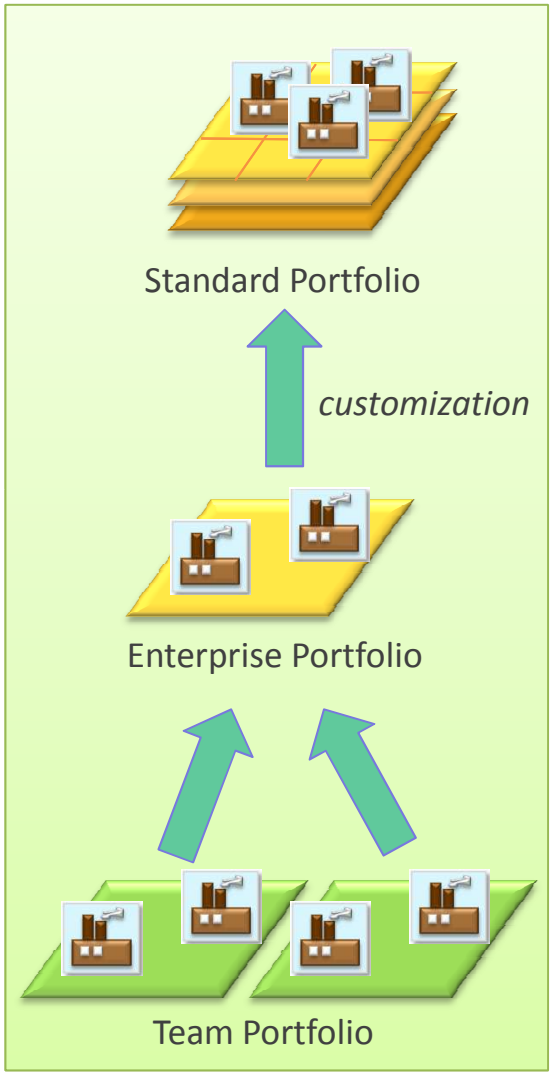
Portfolio #1 – EMF Generations Layers



THALES

Portfolio #1 – Enhancement of EMF Generation





THALES

[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

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

Portfolio #2 – Build Chain Editor & Generator

Objective of the Build Portfolio provided by EGF:

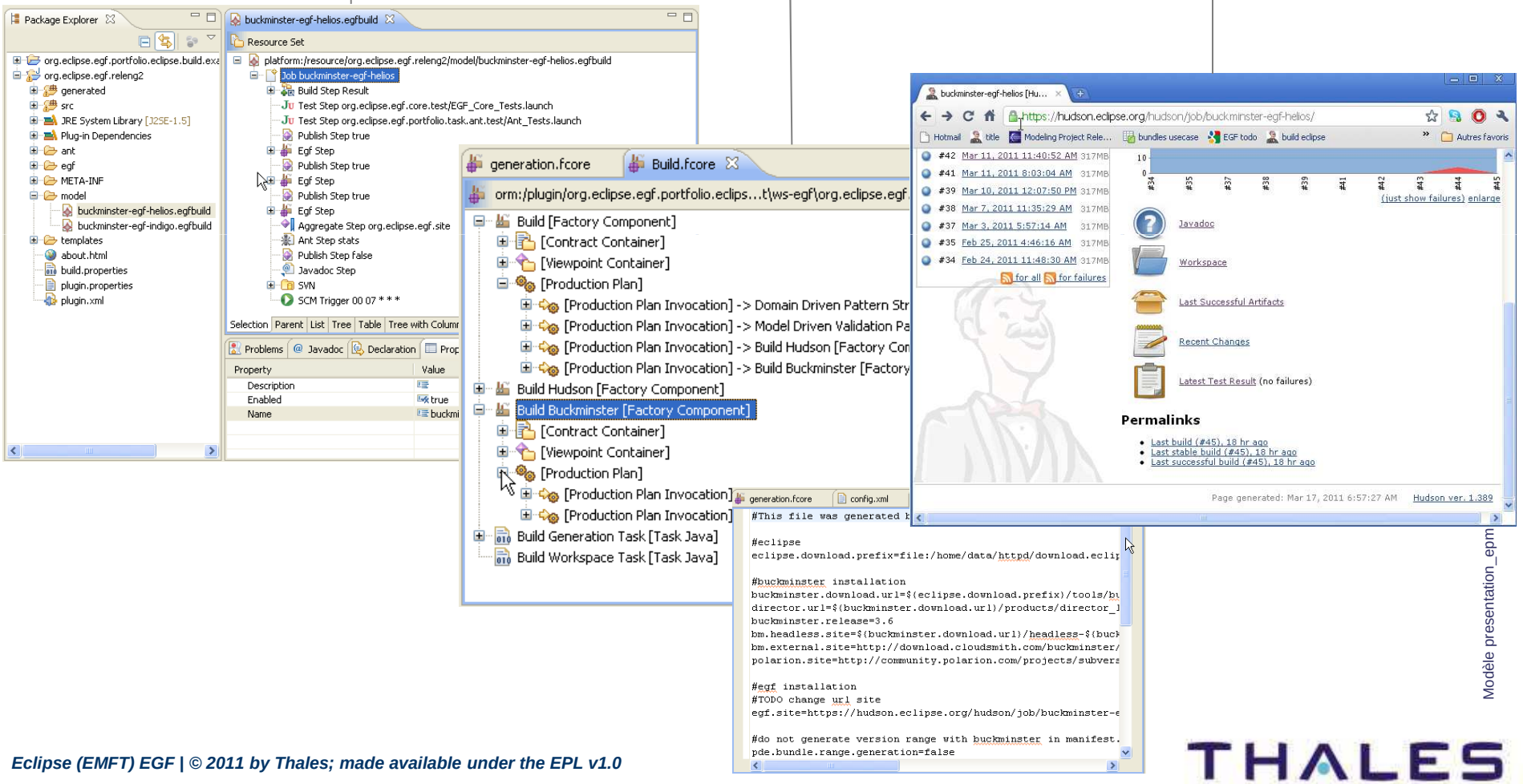
- **Facilitating the definition of build chain:**
  1. A **build editor** describes a build chain
  2. A **generator** targets a build platform, here **Hudson and Buckminster**
  3. Use of the build chain

Portfolio #2 – Build Chain Editor & Generator

Build Model Definition

Build Script Generation

Build Automation



Modèle presentation_epm

THALES

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

Modèl

THALES

[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

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