EGF
Eclipse Generation Factories

Introduction to EGF

Benoît Langlois / Thales Global Services

THALES

Introduction

EGF Architecture

Concepts & Practice
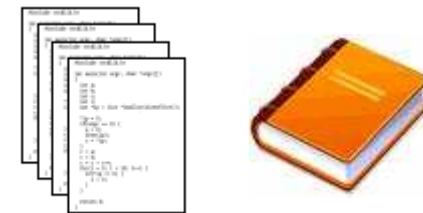
EGF Portfolios

Modèle presentation_epm version 1.0

THALES

Introduction

EGF Architecture

Concepts & Practice

EGF Portfolios

THALES

Input ⟹ **Generator** ⟹ **Output**

model

Modèle presentation_epm version 1.0

THALES

Integration of heterogeneous kinds of know-how

Different types of input

Orchestration

Different types of output
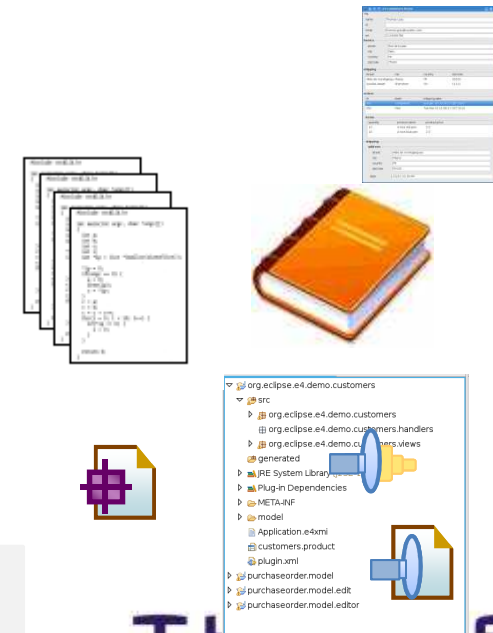
Input ➡ **Generator** ➡ **Output**

model

File

Plug-in

...

Different languages & Tools

Framework

Variability

Modèle presentation_epm version 1.0

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?

Best practices, guidance?

Generation data, which ones, where?

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

A Software Factory Tool, An integrated and extensible Generation Framework

THALES

◆ **EGF (Eclipse Generation Factories) is a software factory tool with the purpose to generate software artifacts, such as code or application**

◆ **EGF is an Eclipse Modeling component**

◆ **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
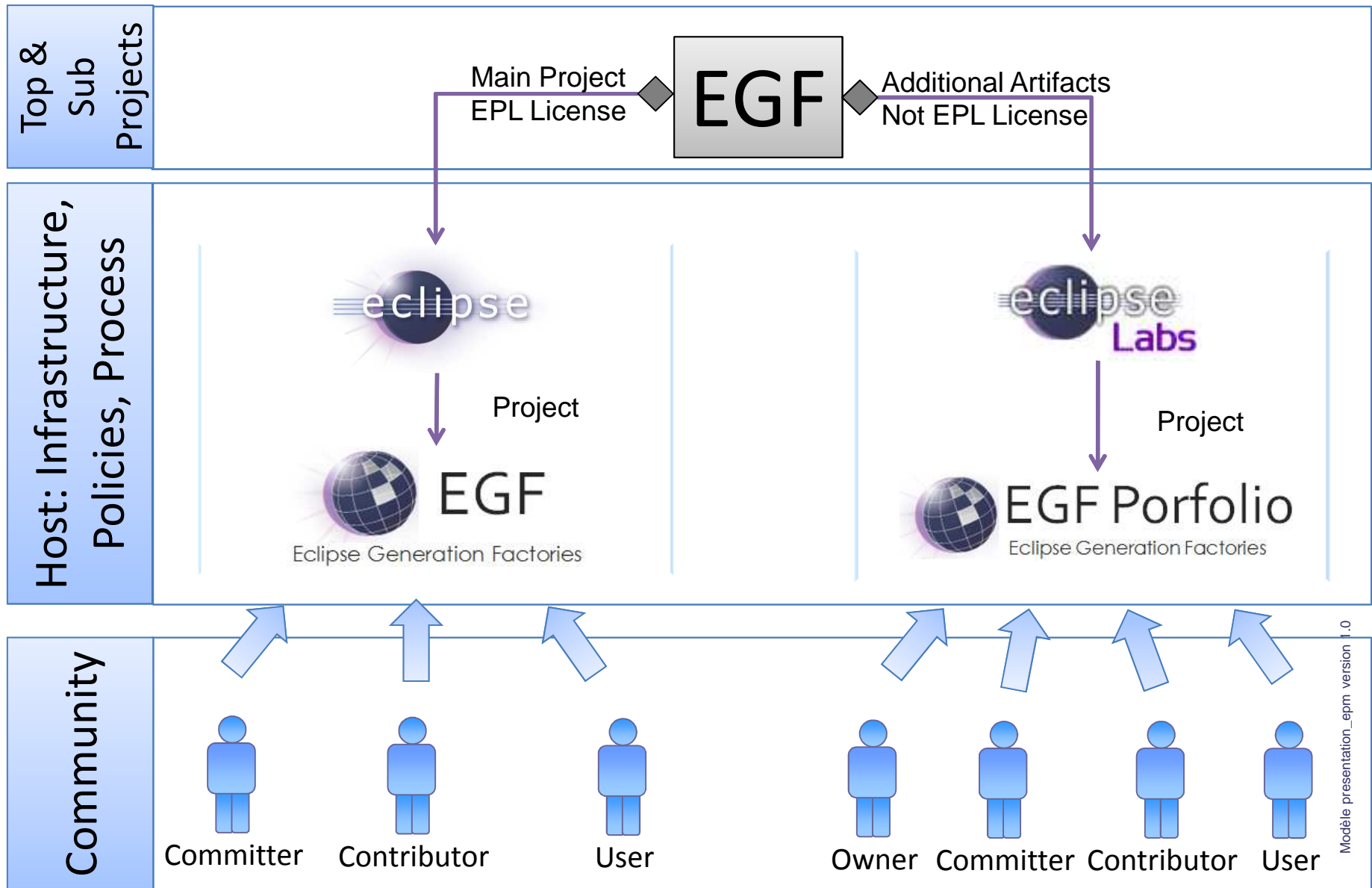
Modèle presentation_epm version 1.0

THALES

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

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

Modèle presentation_epm version 1.0

THALES

Introduction

EGF Architecture
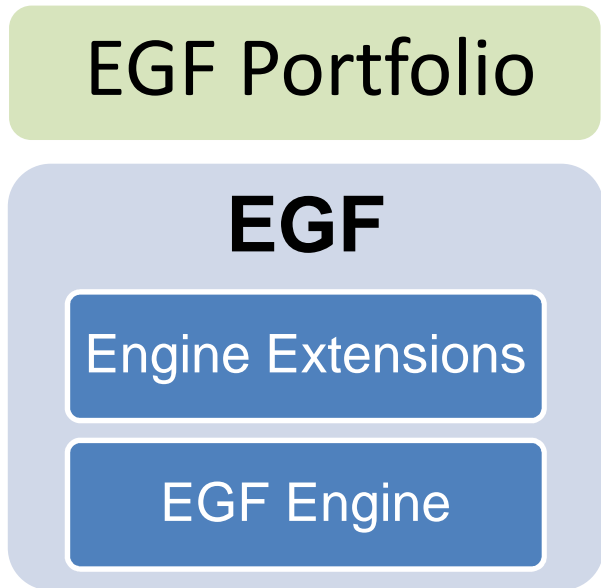
Concepts & Practice

EGF Portfolios

THALES

Introduction

EGF Architecture

- Architecture
- Some issues addressed by EGF

Concepts & Practice

EGF Portfolios

Modèle presentation_epm version 1.0

THALES

EGF Portfolio

EGF

Engine Extensions

EGF Engine

Modèle presentation_epm version 1.0

THALES

EGF Portfolio

EGF

Engine Extensions

EGF Engine ➜ **The Basics**

EGF Metamodel

Factory component, task

Basic behaviors, dynamic execution

EGF Pattern

Modèle presentation_epm version 1.0

THALES

EGF Portfolio

EGF

Engine Extensions

EGF Engine

Extensibility

New Language & tools

New generation formalisms

Modèle presentation_epm version 1.0

THALES

EGF Portfolio

EGF

Engine Extensions

EGF Engine

Practice
(EGF Wiki)

Use cases to understand generation issues

Examples to start up

Extensibility

The Basics

Modèle presentation_epm version 1.0

THALES

EGF Portfolio

EGF

Engine Extensions

EGF Engine

Portfolio

Off-the-shelf software factories

Portfolios provided with EGF

Enhanced EMF Generation

Build modeler and generator

For continuous integration
(Hudson/Jenkins, Buckminster today)
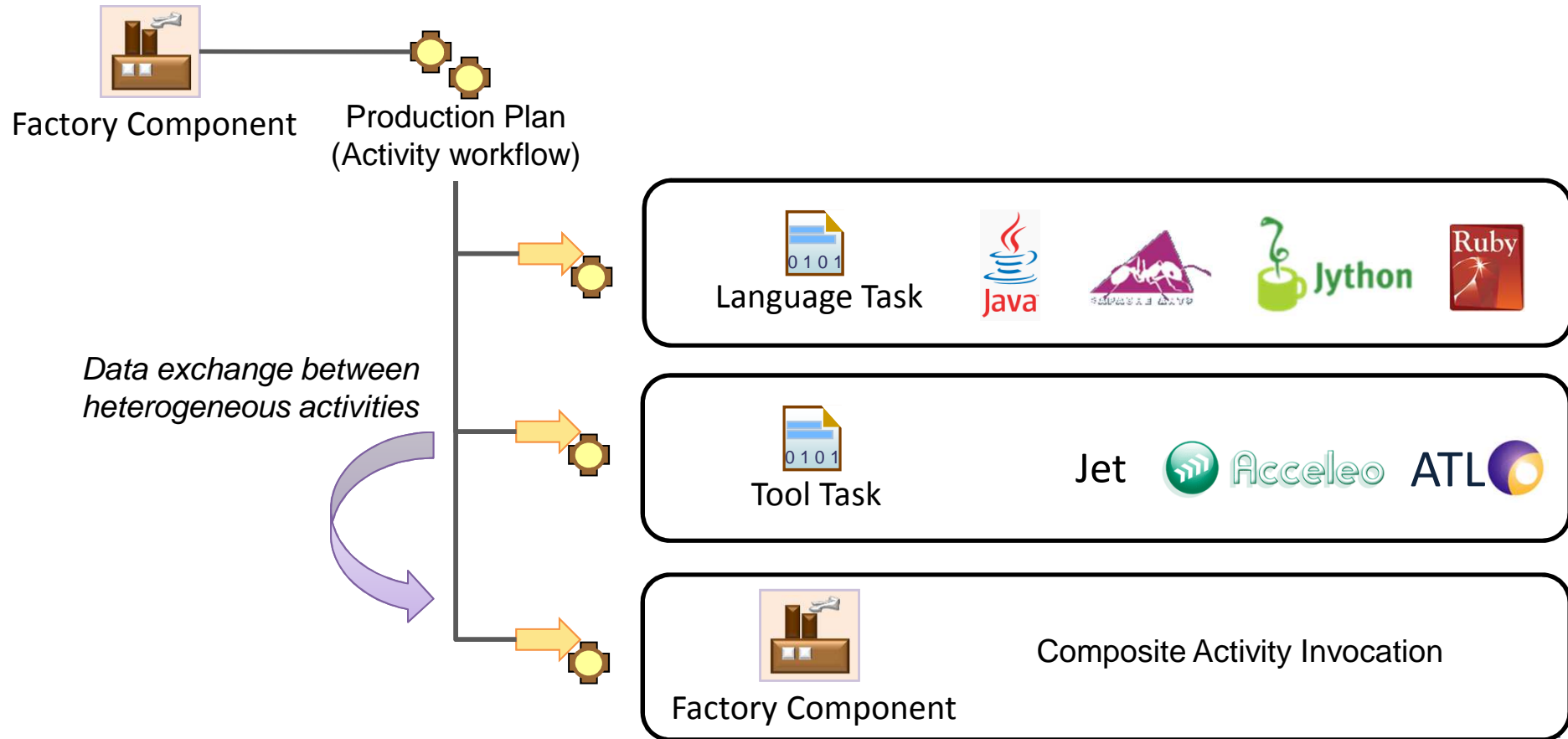
Modèle presentation_epm version 1.0

THALES

Introduction

EGF Architecture

- Architecture
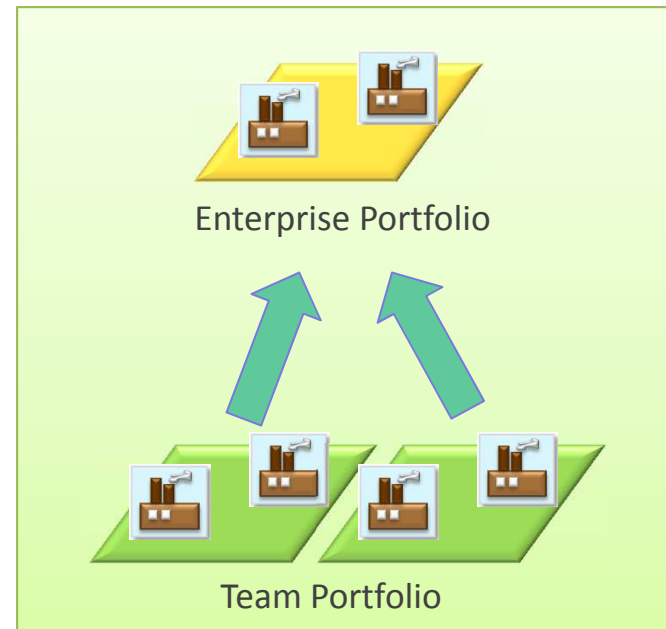- Some issues addressed by EGF

Concepts & Practice

EGF Portfolios

THALES

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

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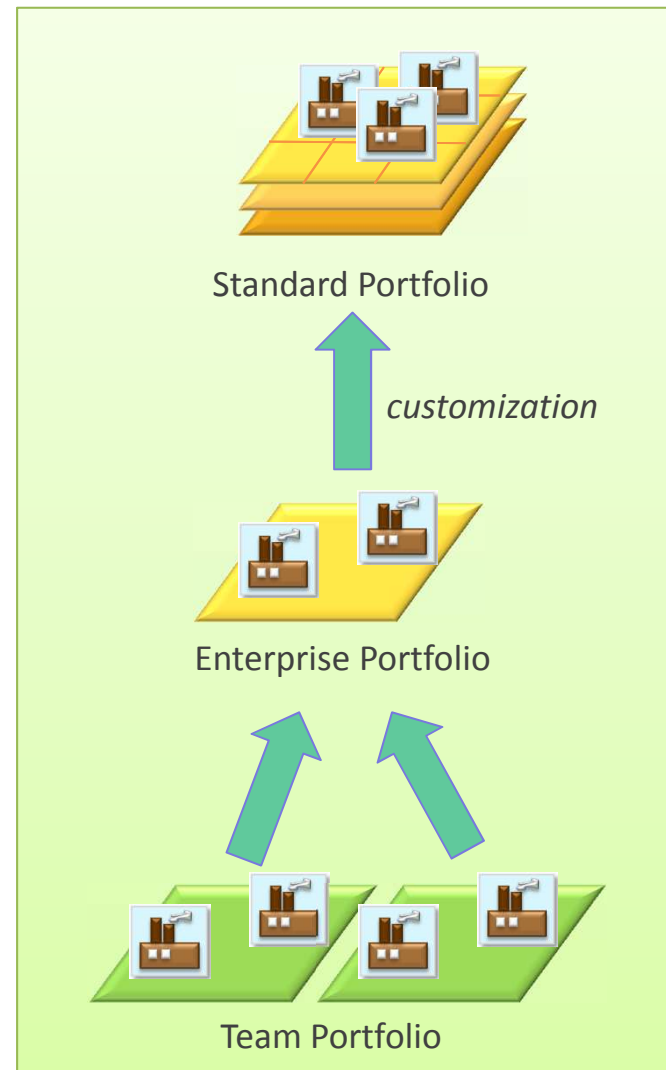
Modèle presentation

THALES

Several levels of Customization

Example

Code/textual generation
for my organization

extends

Specific generation
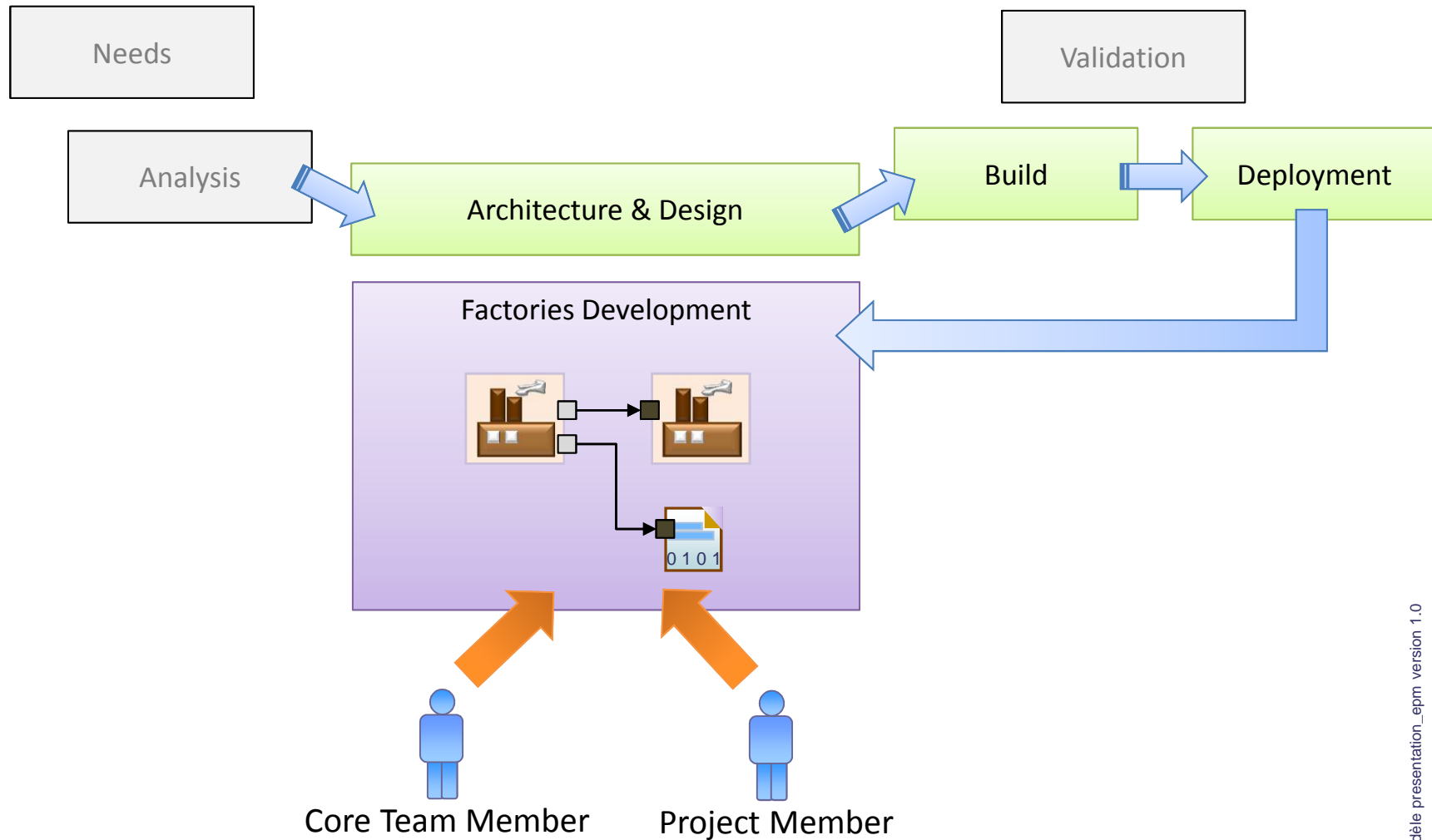for my project

THALES

Example

Standard Portfolio

customization

Enterprise Portfolio

Team Portfolio

Several levels of Customization

EMF Generation

extends

EMF Generation
for my organization

extends

EMF Generation
for my application

Modèle presentation_epm version 1.0

THALES

Needs

Validation

Analysis

Architecture & Design

Build

Deployment

Factories Development

Core Team Member

Project Member

THALES

Development

Generation Activity Development

Assembly & Orchestration

Language Task

Tool Task

Factory Component

Pattern

Use

Packaging

Reuse

Off-The-Shelf Software Factories

Customization

Factories

Portfolio

Reuse

Project Adaptation

Management

THALES

General Architecture of EGF

EGF Portfolio

EGF

Engine Extensions

EGF Engine
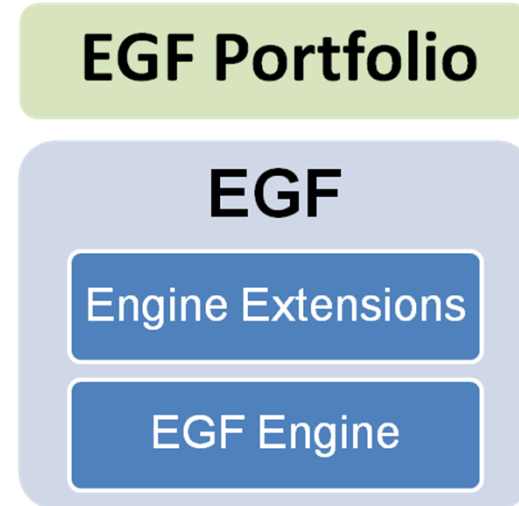
Example of EGF Factory
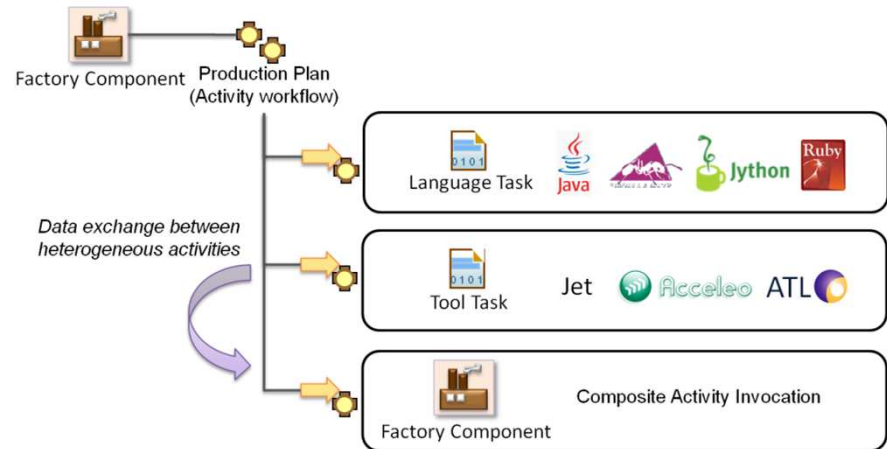
Development & Reuse Process with EGF

Modele presentation_epm version 1.0

THALES
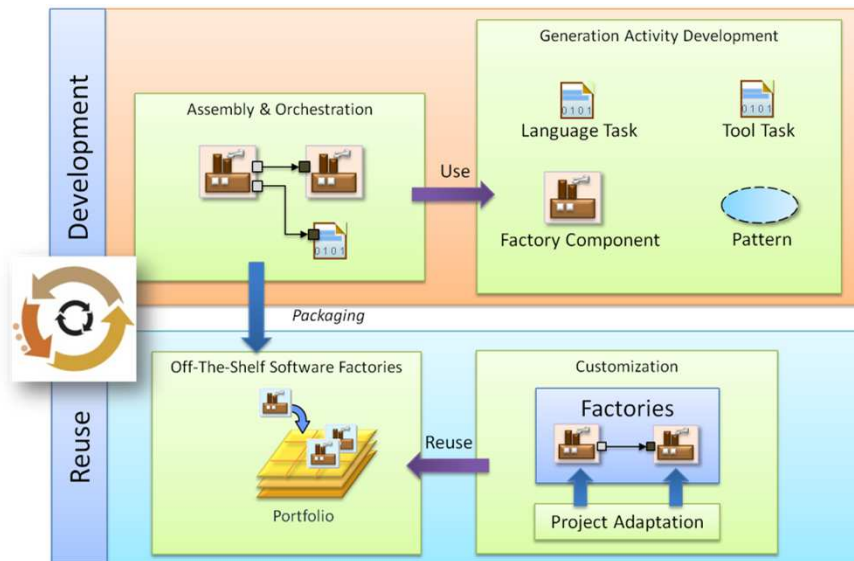
Introduction

EGF Architecture

Concepts & Practice

EGF Portfolios

THALES

Installation by update site

◆ **Eclipse Update site**

◆ **Update site from Amalgam**

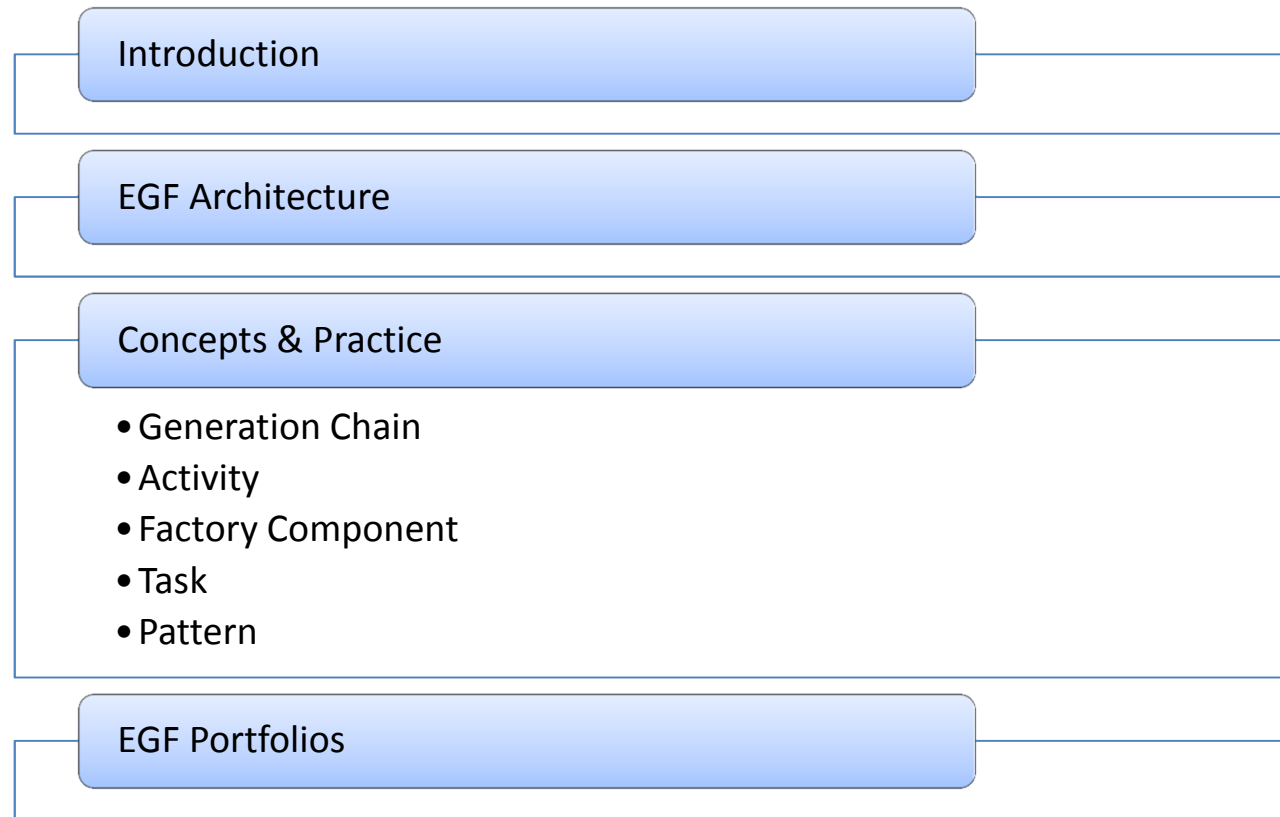   ○ [Eclipse] Help / Install Modeling Components / EGF

Download EGF materials

◆ **Download EGF update site, dropins, examples**

◆ **Location:** http://wiki.eclipse.org/EGF_Installation

Modèle presentation_epm version 1.0

THALES

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**

Modèle presentation_epm version 1.0

THALES

Introduction

EGF Architecture

Concepts & Practice

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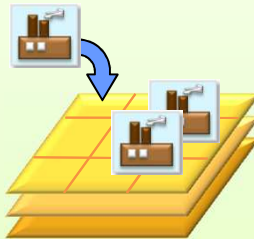- Task
- Pattern

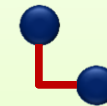EGF Portfolios

THALES

Factory Component
Composite generation unit with an activity workflow
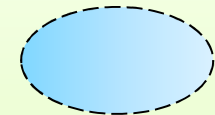**Task**
Leaf generation unit to execute a tool (e.g., ATL, Acceleo) or code written in a language (e.g., Java, Ruby, Ant)
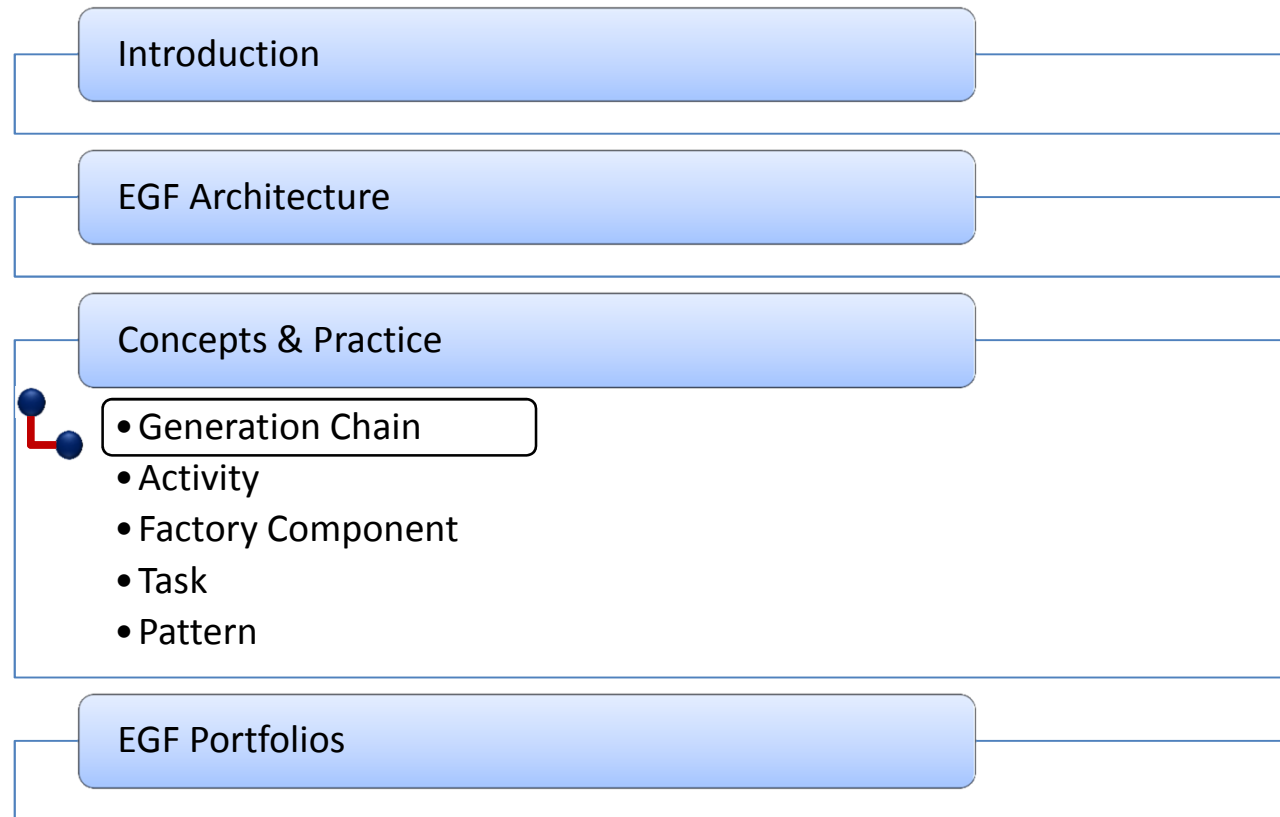
Portfolio
Set of factories to capitalize on a specific generation topic

Generation Chain
High generation view to organize complex generations

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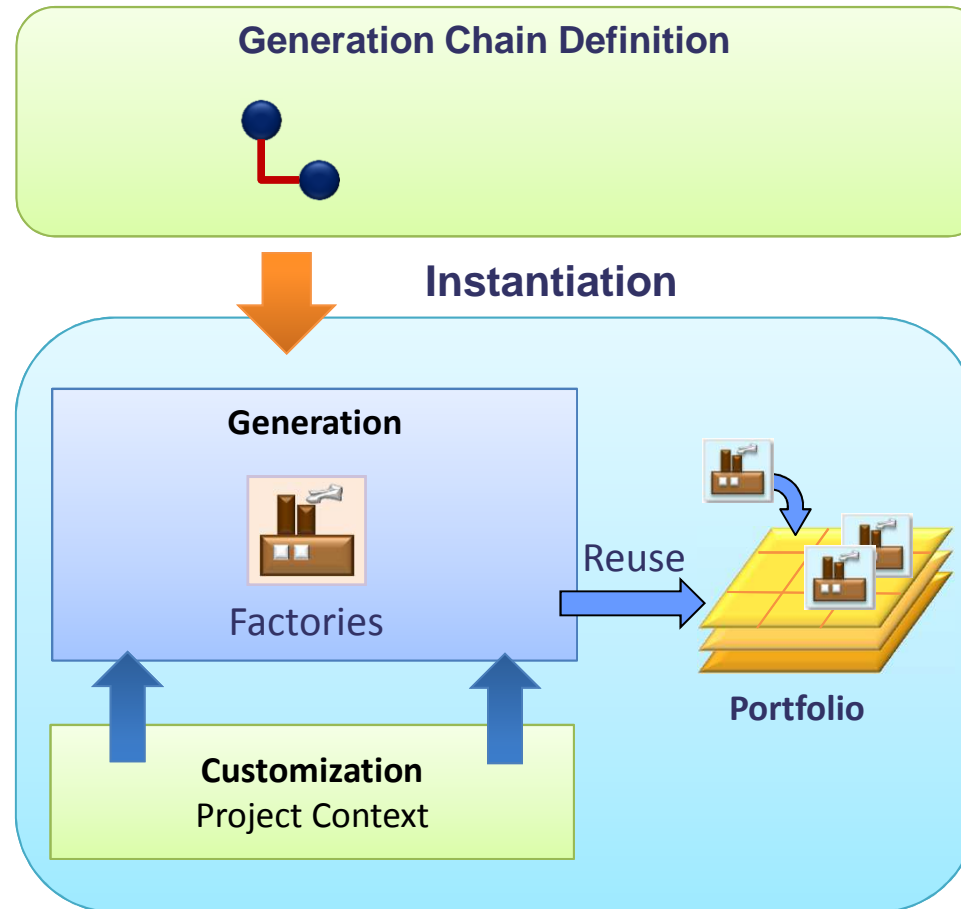- For definition of code generation families

Modèle presentation_epr

THALES

Introduction

EGF Architecture

Concepts & Practice

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

EGF Portfolios

Modèle presentation_epm version 1.0

THALES

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

THALES

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
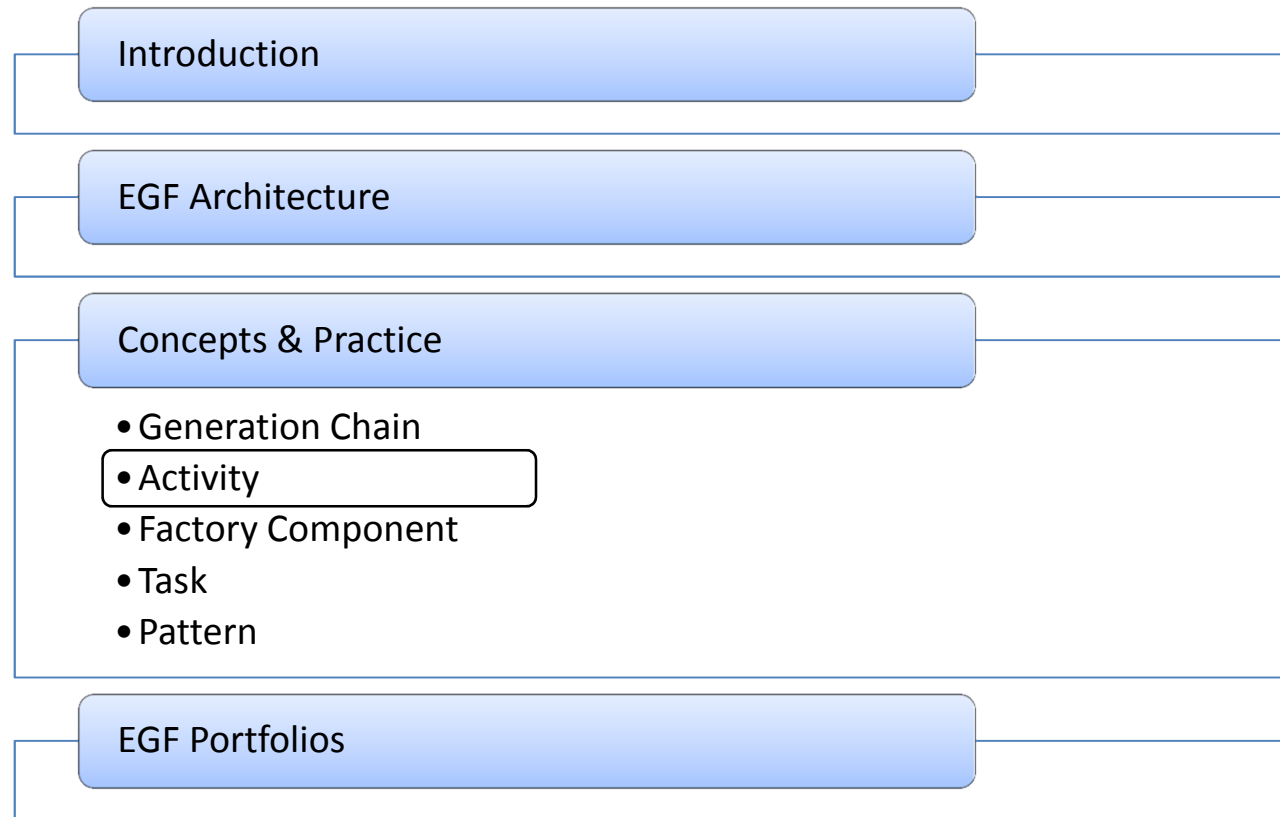
Modèle presentation_epm version 1.0

THALES

Generation Chain Definition

Instantiation

Generation

Factories

Reuse

Portfolio

Customization
Project Context

Modèle presentation_epm version 1.0

THALES

Links:

Video: http://vimeo.com/29472598

For more explanations:

Generation Chain Tutorial
http://wiki.eclipse.org/EGF_Tutorial_and_Use_Cases
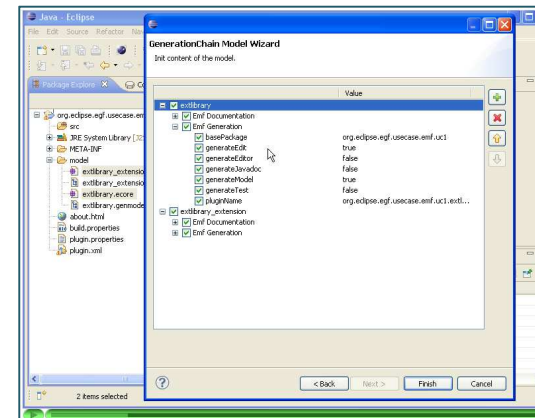
Exercice:

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

Modèle presentation_epm version 1.0

THALES

Introduction

EGF Architecture

Concepts & Practice

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

EGF Portfolios

Modèle presentation_epm version 1.0

THALES

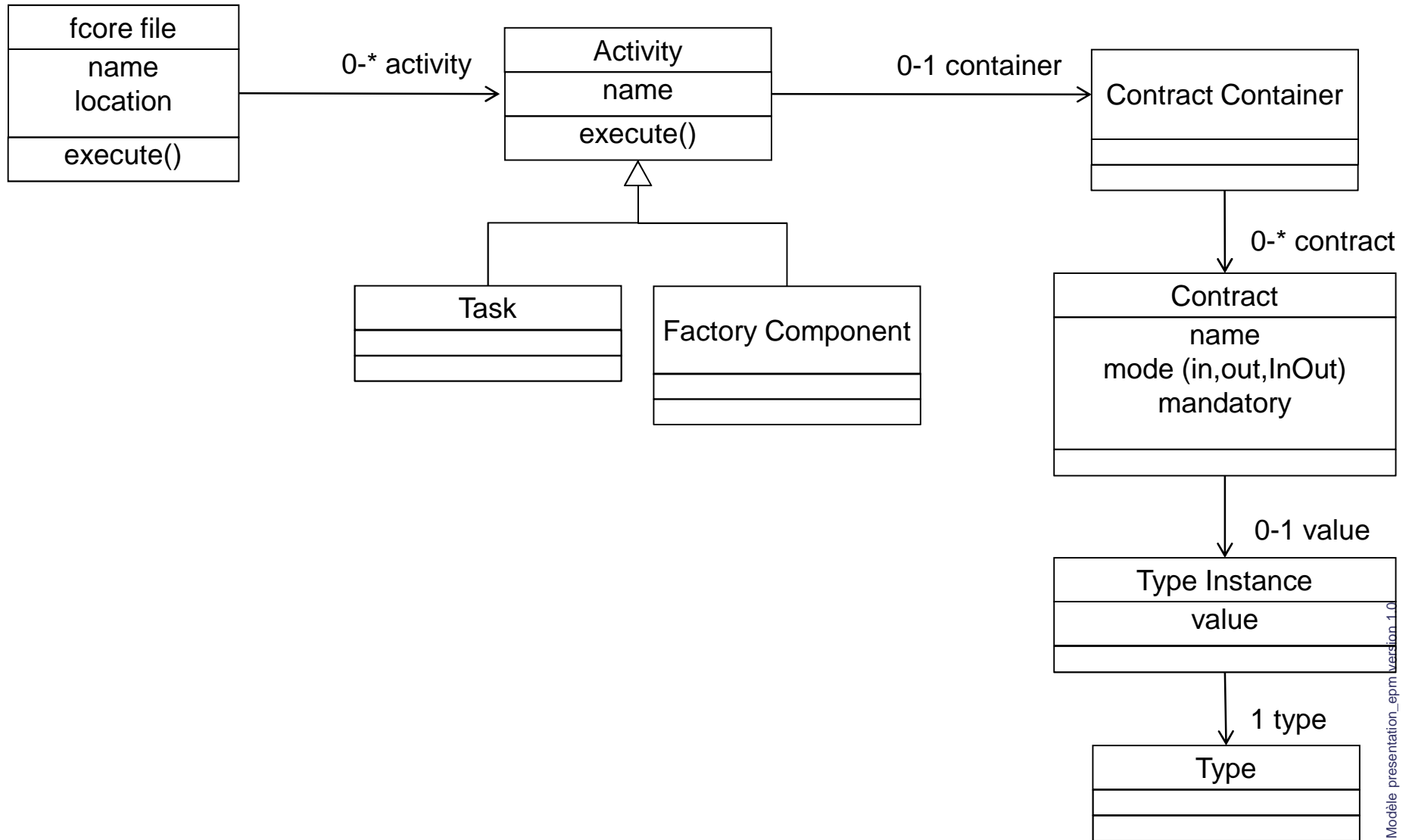**An activity is the abstract class of EGF generation units**

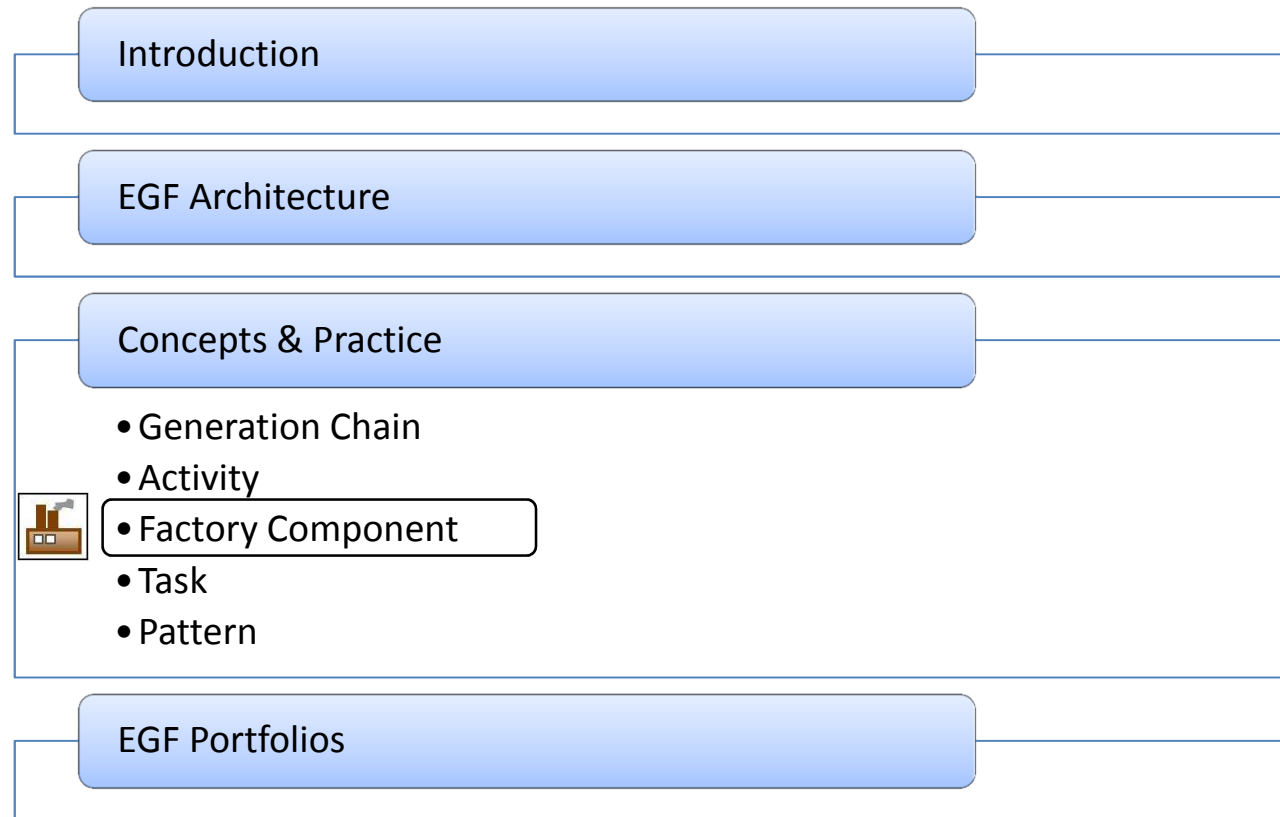◆ **Factory component and Task are activities**

Activity storage

◆ **Activities are stored in *fcore* files**

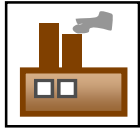◆ **The same fcore file contains one or several activities**

Activity properties
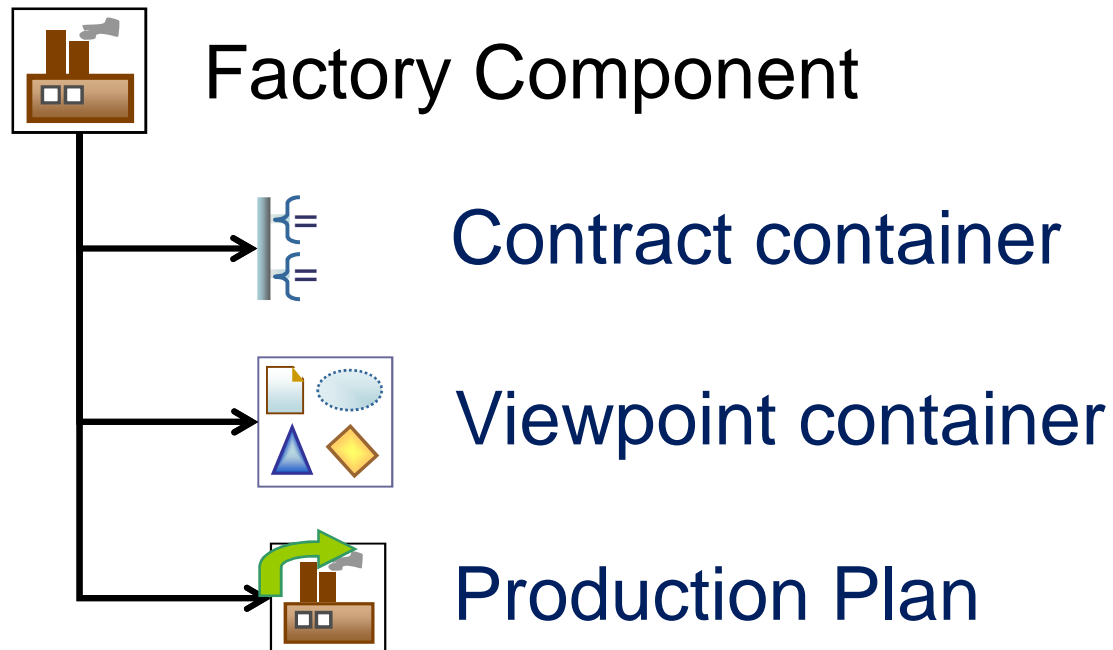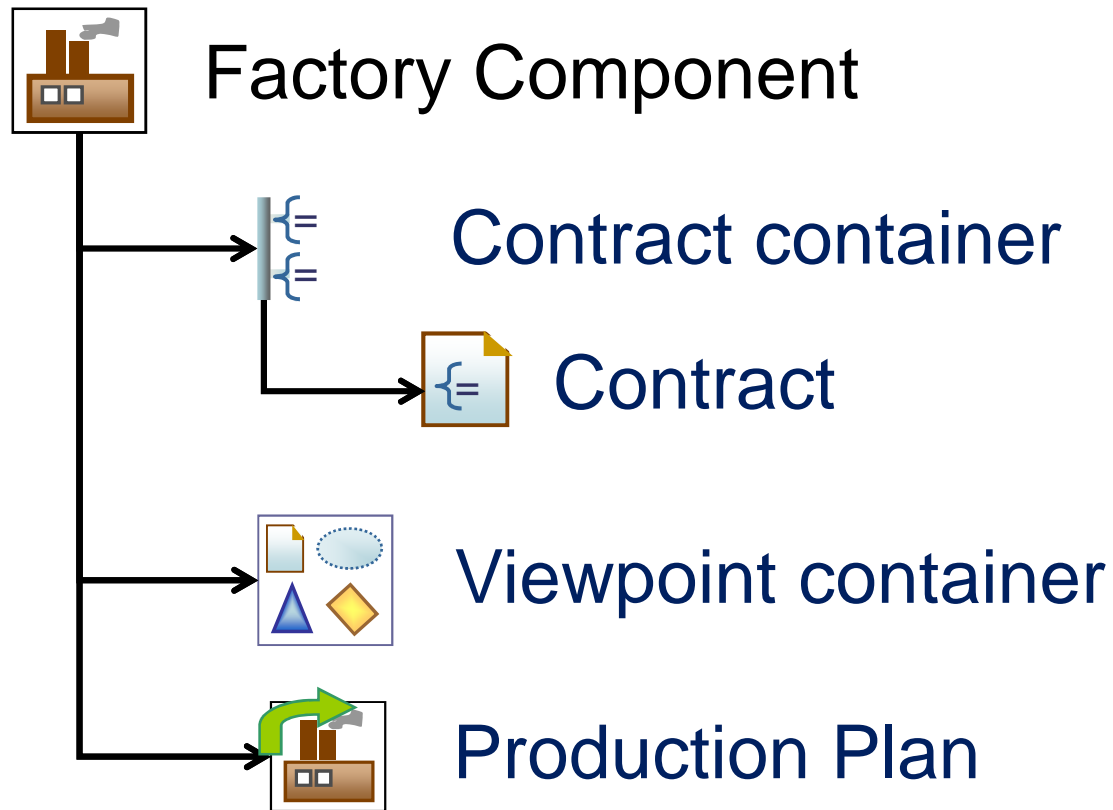
◆ **Contract declaration (= parameters)**

Modèle presentation_epm version 1.0

THALES

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

Modèle presentation_epm version 1.0

THALES

Introduction

EGF Architecture

Concepts & Practice

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

EGF Portfolios

Modèle presentation_epm version 1.0

THALES

- ◆ **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**

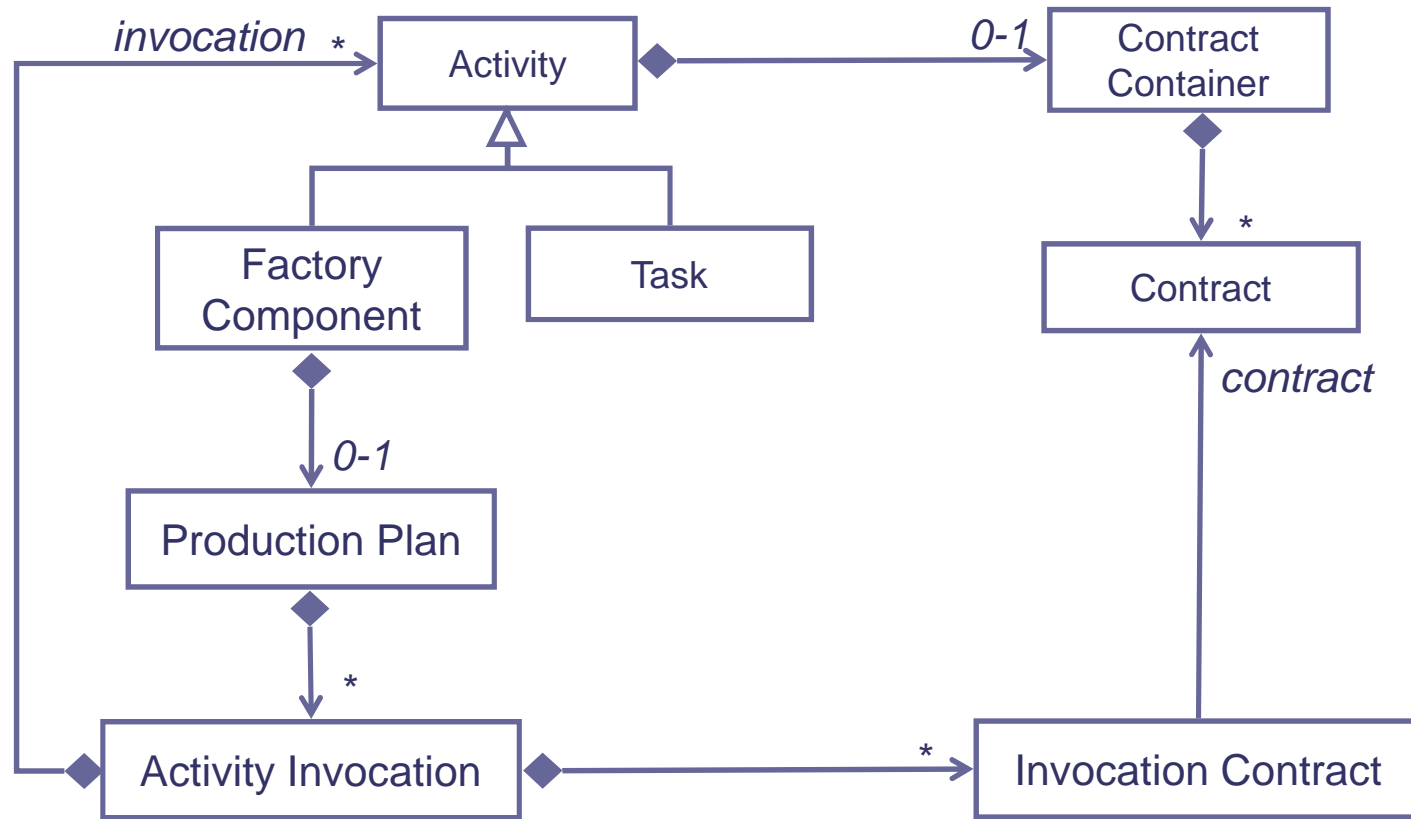- ◆ **Factory Component Lifecycle: edition and execution**

Modèle presentation_epm version 1.0

THALES

Factory Component

Contract container

Viewpoint container

Production Plan

Modèle presentation_epm version 1.0

THALES

Factory Component

Contract container

Contract

Viewpoint container

Production Plan

◆ **Contract:** Factory component parameter

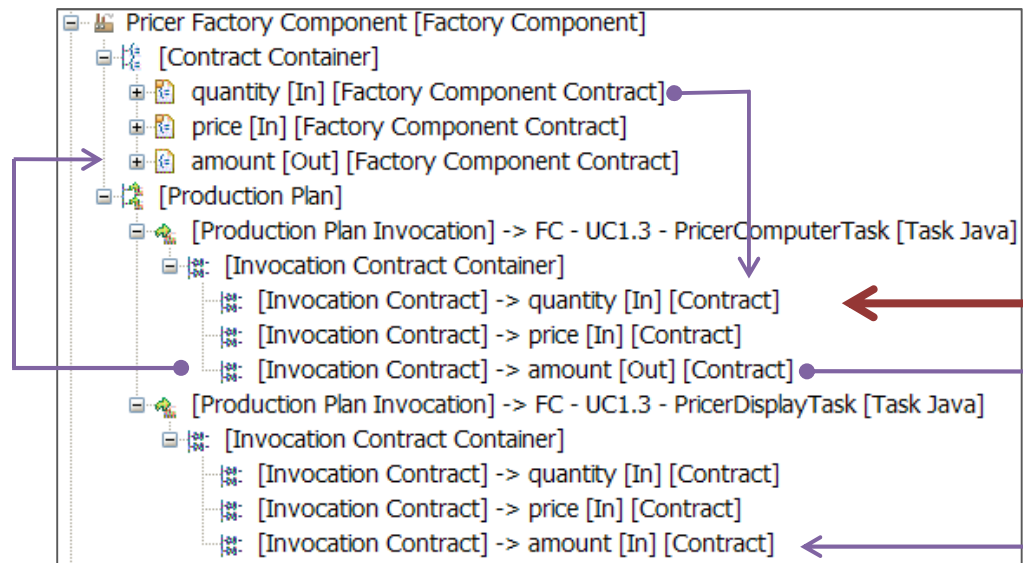◆ A contract has a type, a passing mode (In/Out/In_Out), a default value or not, is mandatory or optional

THALES

Factory Component

Contract container

Viewpoint container
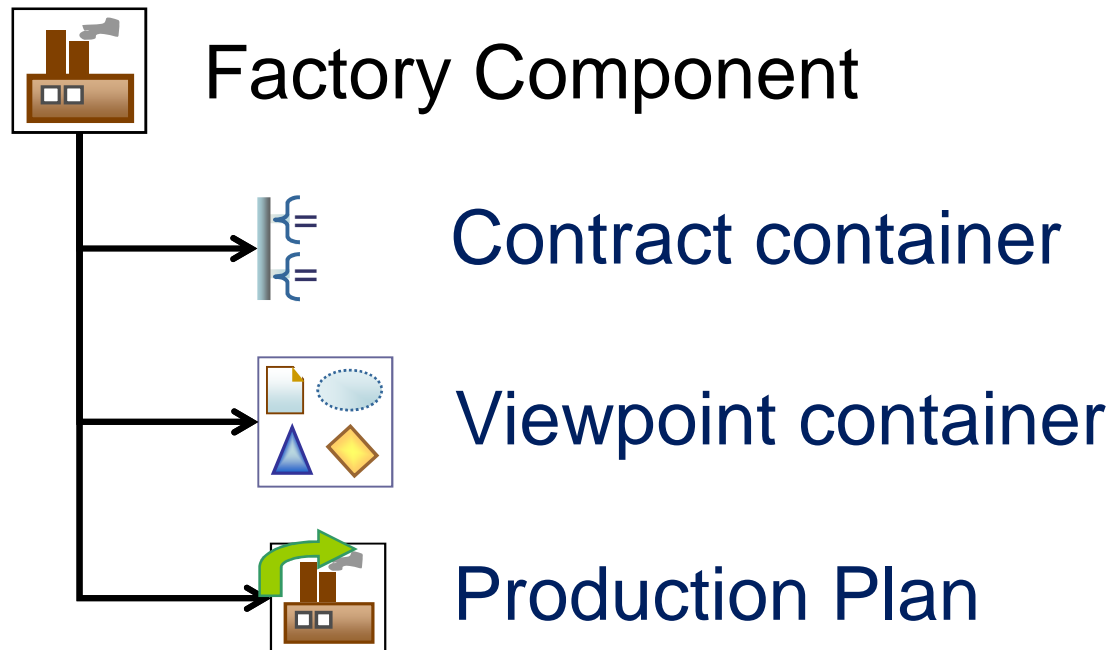
Production Plan

◆ **Viewpoint:** area to declare concerns of generation data

◆ Examples of viewpoint:

  ○ Available today: domain declaration, pattern

  ○ Candidates: licensing, feature model

Modèle presentation_epm version 1.0

THALES

Factory Component

Contract container

Viewpoint container
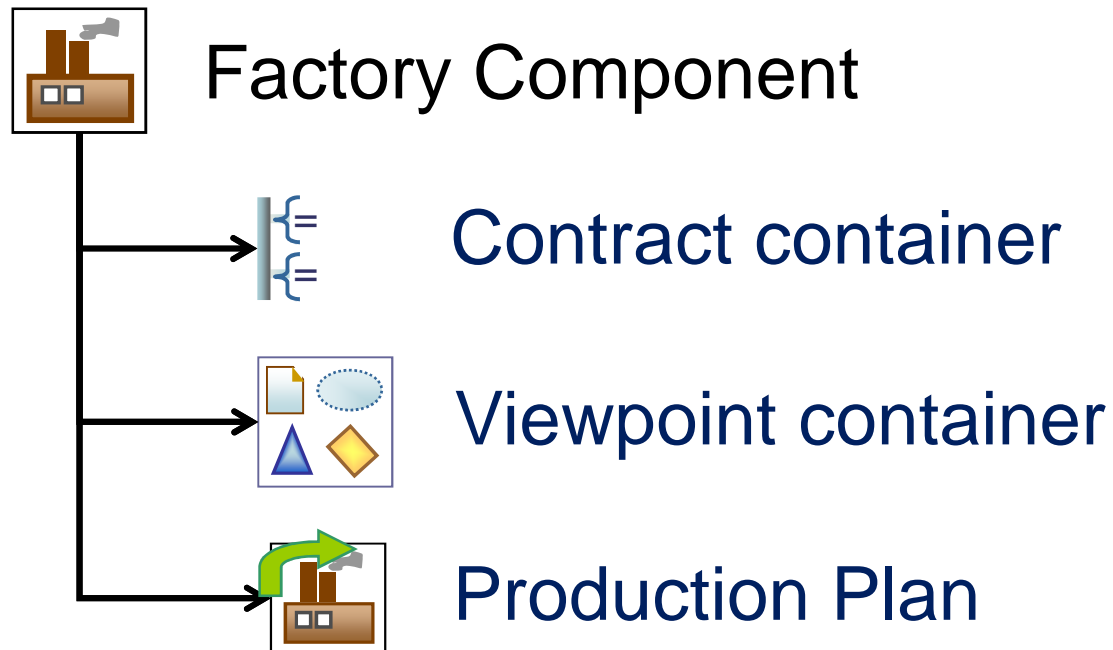
Production Plan

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

Modèle presentation_epm version 1.0

THALES

Modèle presentation_epm version 1.0

THALES

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
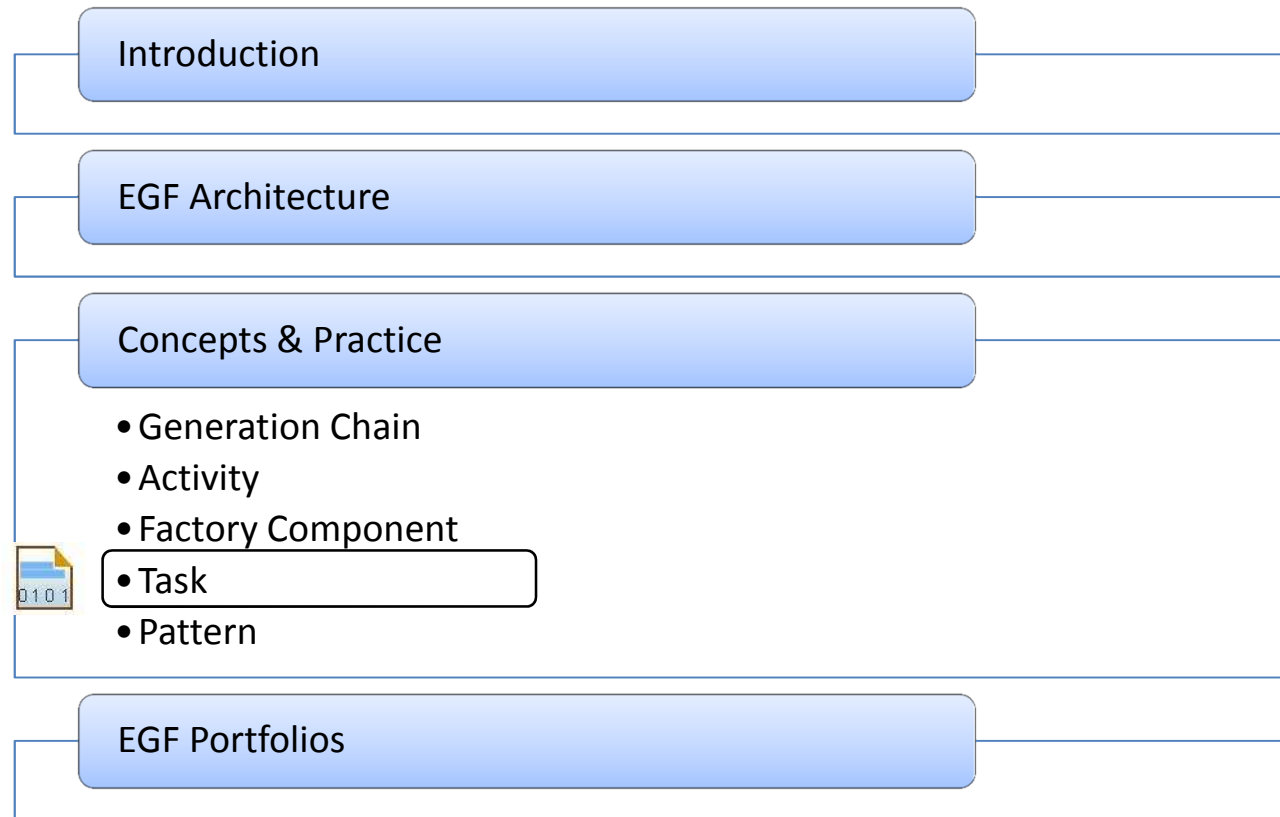
A task is an atomic generation unit

- ◆ **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**
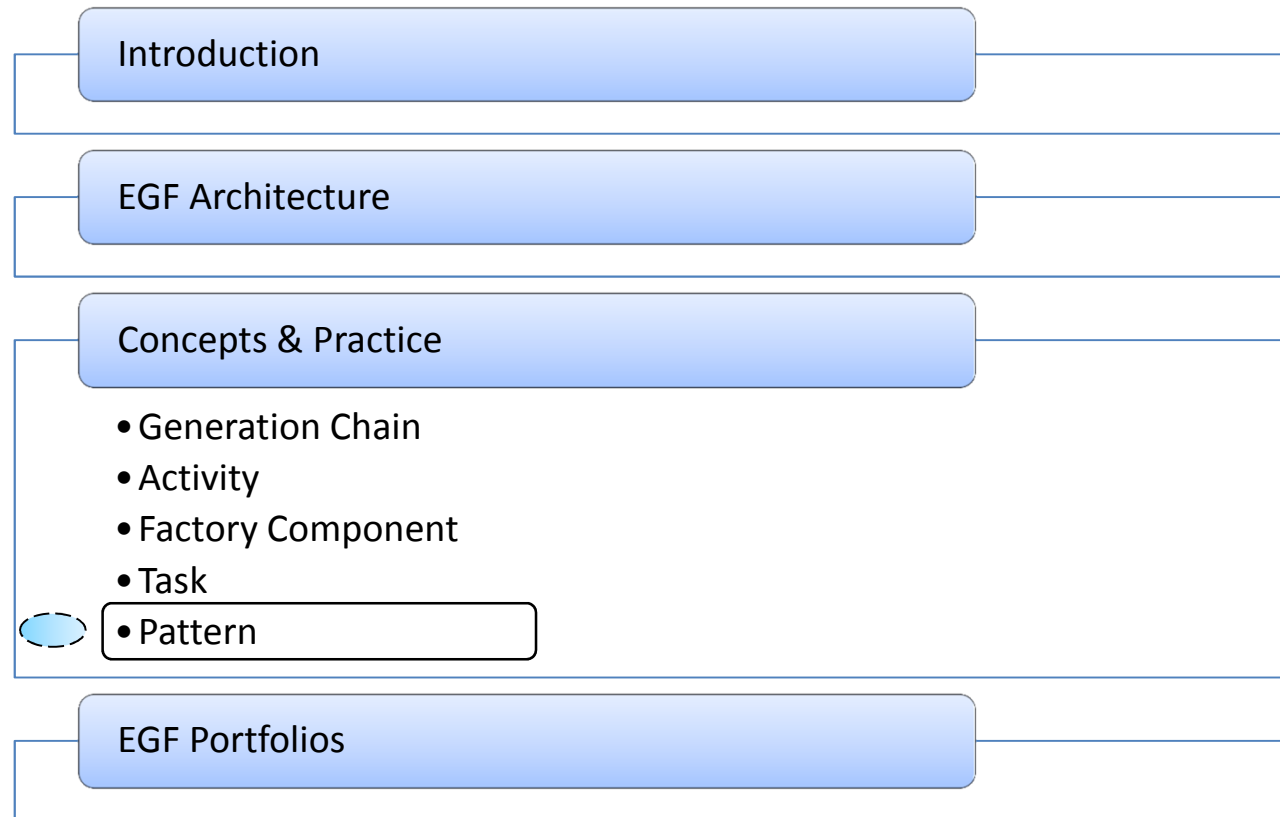- ◆ **Example: a JavaTask is implemented by a Java class (which implements ITaskProduction)**

Java Task	—— 1 ——▶	Java Class

implementation

Modèle presentation_epm version 1.0

THALES

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

Modèle presentation_epm version 1.0

THALES

Introduction

EGF Architecture

Concepts & Practice

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

EGF Portfolios

THALES

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

Modèle presentation_epm version 1.0

THALES

Introduction

EGF Architecture

Concepts & Practice

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

EGF Portfolios

THALES
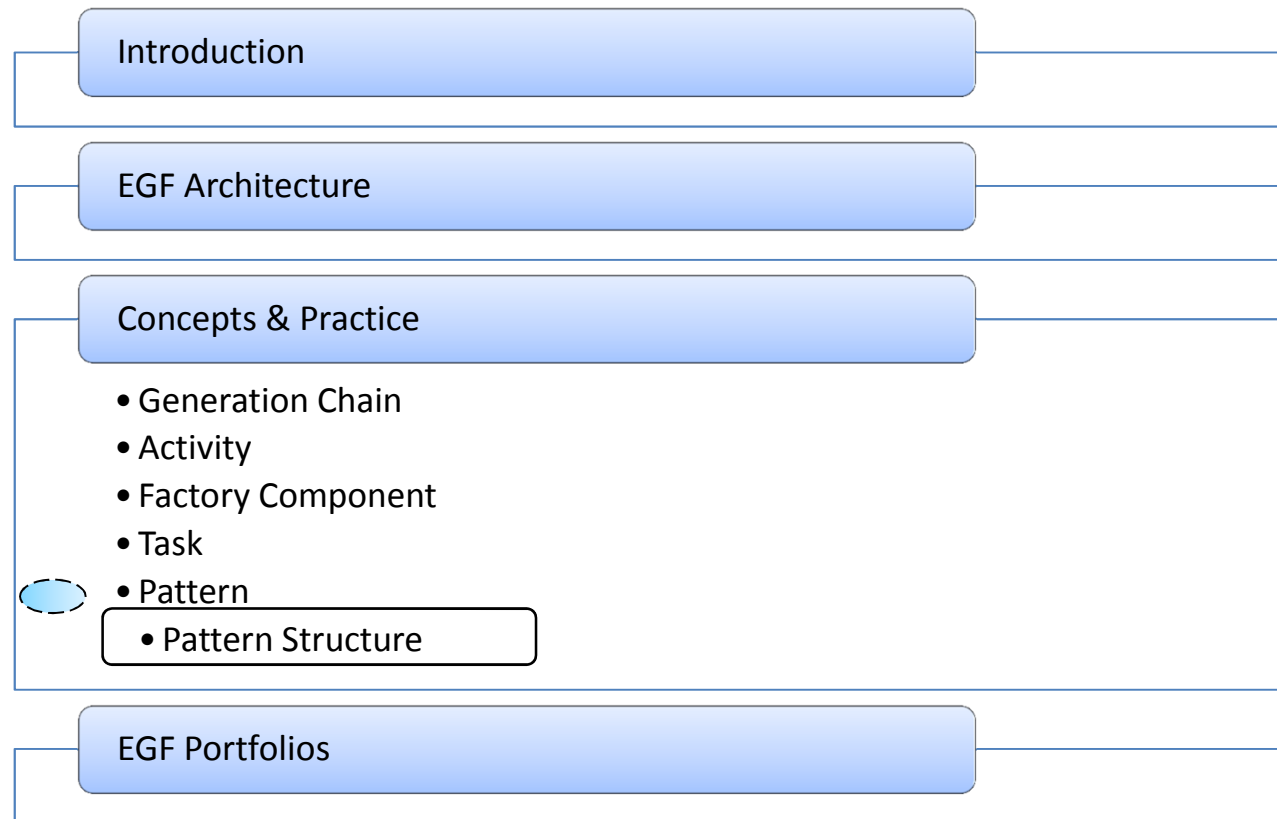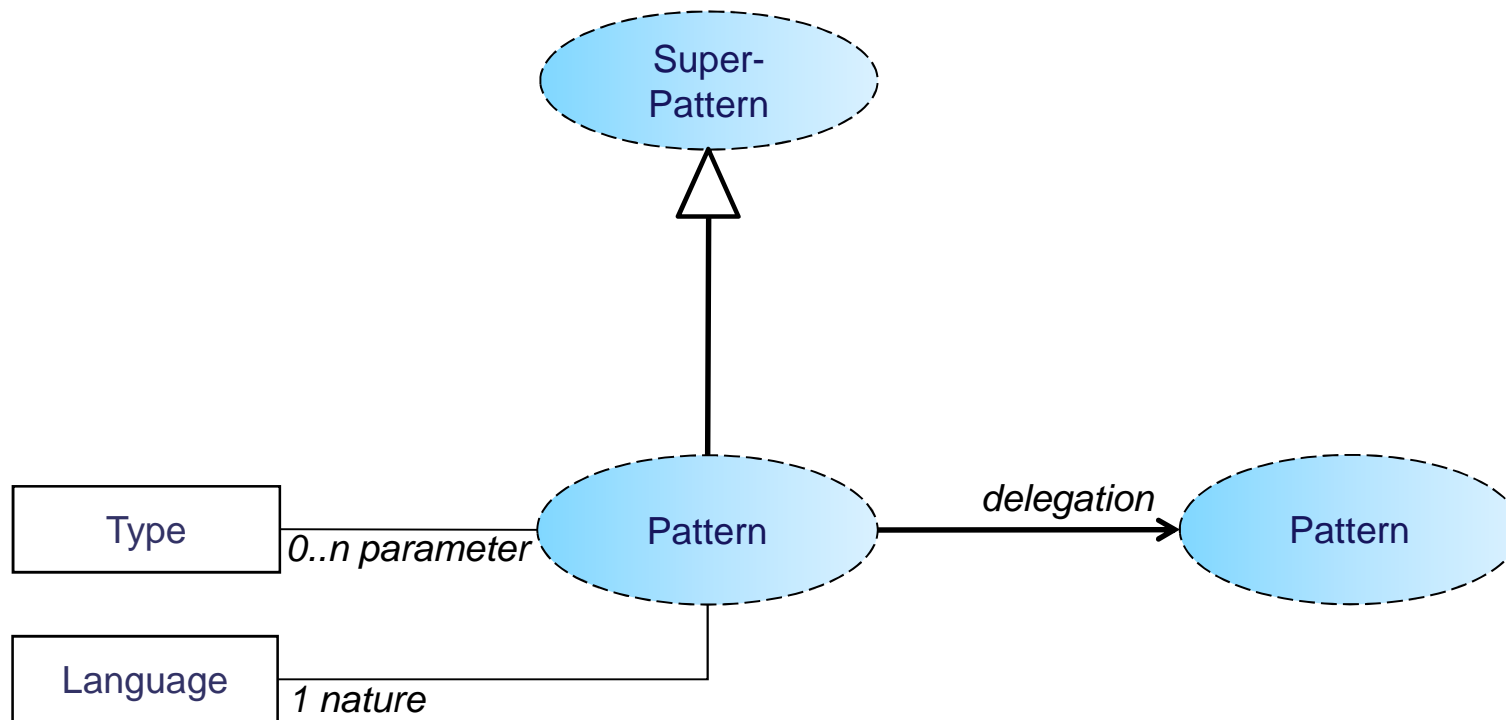
- ◆ **Parameter**: Type of a query record from a query applied over a resource (e.g., a class from an ecore model, a file of a file directory)

- ◆ **Nature:** Language used for the pattern implementation (e.g., Java, Jet for model-to-text)

Modèle presentation_epm version 1.0

THALES

Example

ECLassifier
Gen

ECore
Resource

EClass

parameter

EClass Gen

delegation

EStructural
Feature
Gen

Jet

nature
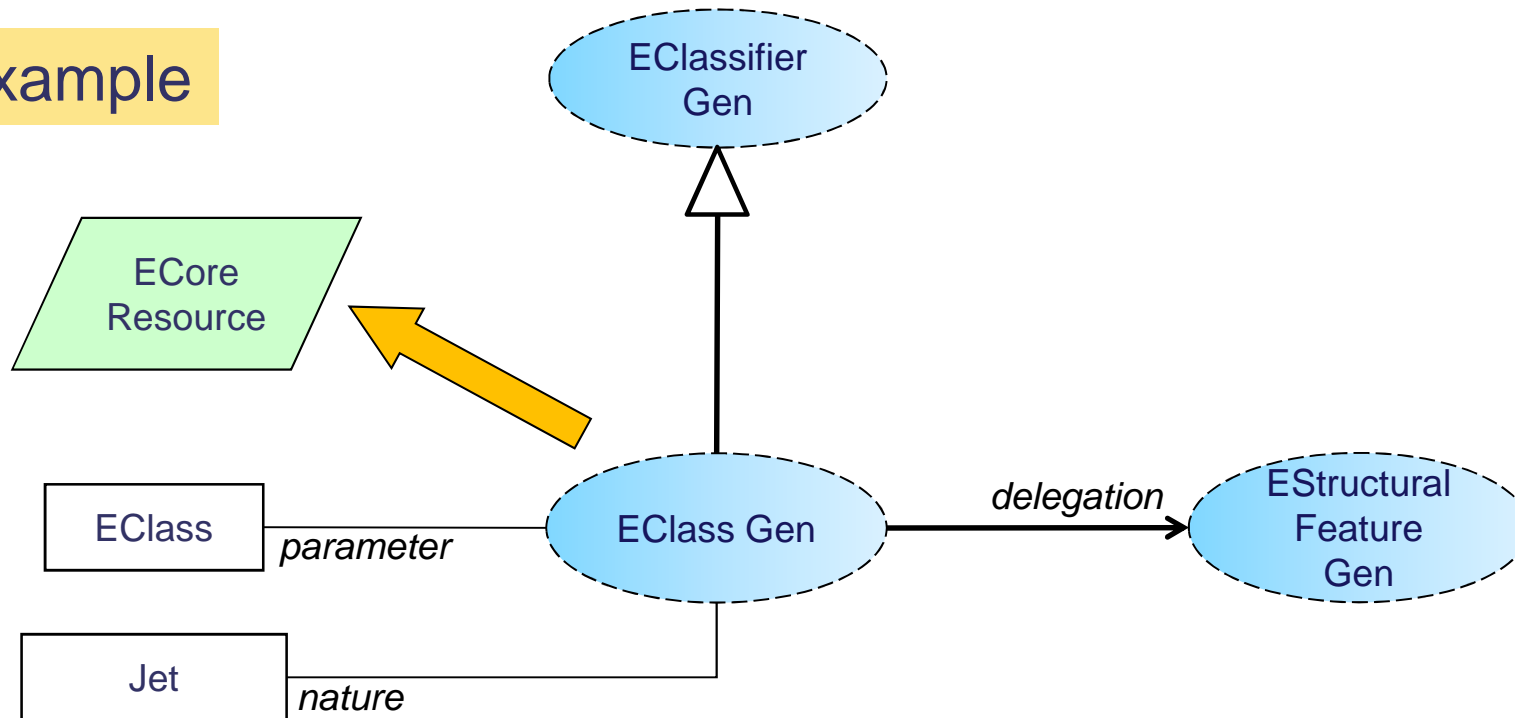
◆ The EClassGen pattern is applied onto a Ecore resource

◆ 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

Modèle presentation_epm version 1.0

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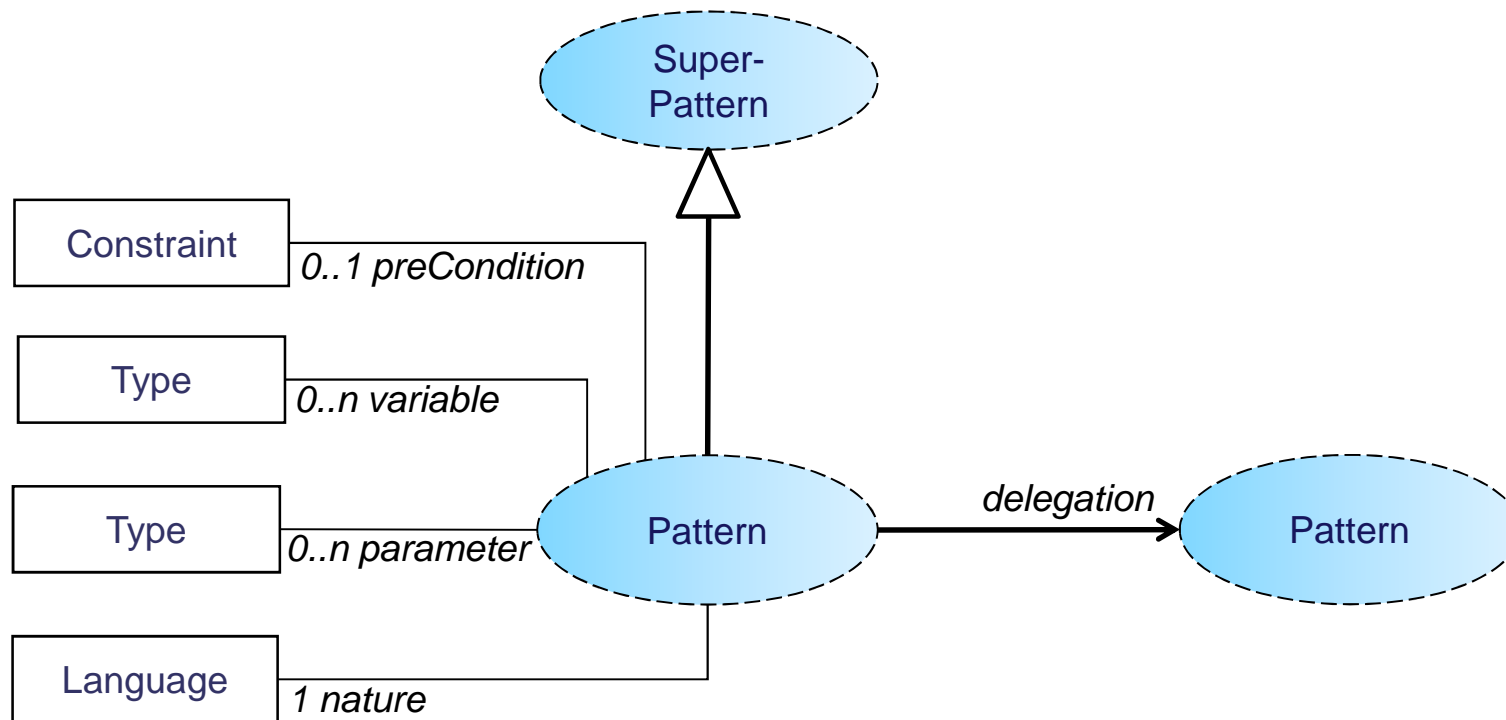
- ◆ **preCondition/Constraint:** constraint to be verified to be applied
- ◆ **variable/Type:** local variable declaration for the pattern implementation

Modèle presentation_epm version 1.0

THALES

Pattern Language

Specification

Inheritance

Choose the super pattern:

Super-pattern

Parent: No parent | Browse | | ✖ |

Pattern Nature

Select the kind of the pattern:

Type: | JetNature | ▼ |

Parameters

Define parameters for this pattern in the following section.

Query Parameter

Name	Type	Query
aClass	EClass	

➕

✏️

✖

⇧

⇩

| Overview | Specification | Implementation |

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

Introduction

EGF Architecture

Concepts & Practice

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

EGF Portfolios

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

THALES

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

Result

View [Optional]

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

Result

Modèle presentation_epm version 1.0

THALES

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

Modèle presentation_epm version 1.0

THALES

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

Modèle presentation_epm version 1.0

THALES

Introduction

EGF Architecture

Concepts & Practice

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

EGF Portfolios

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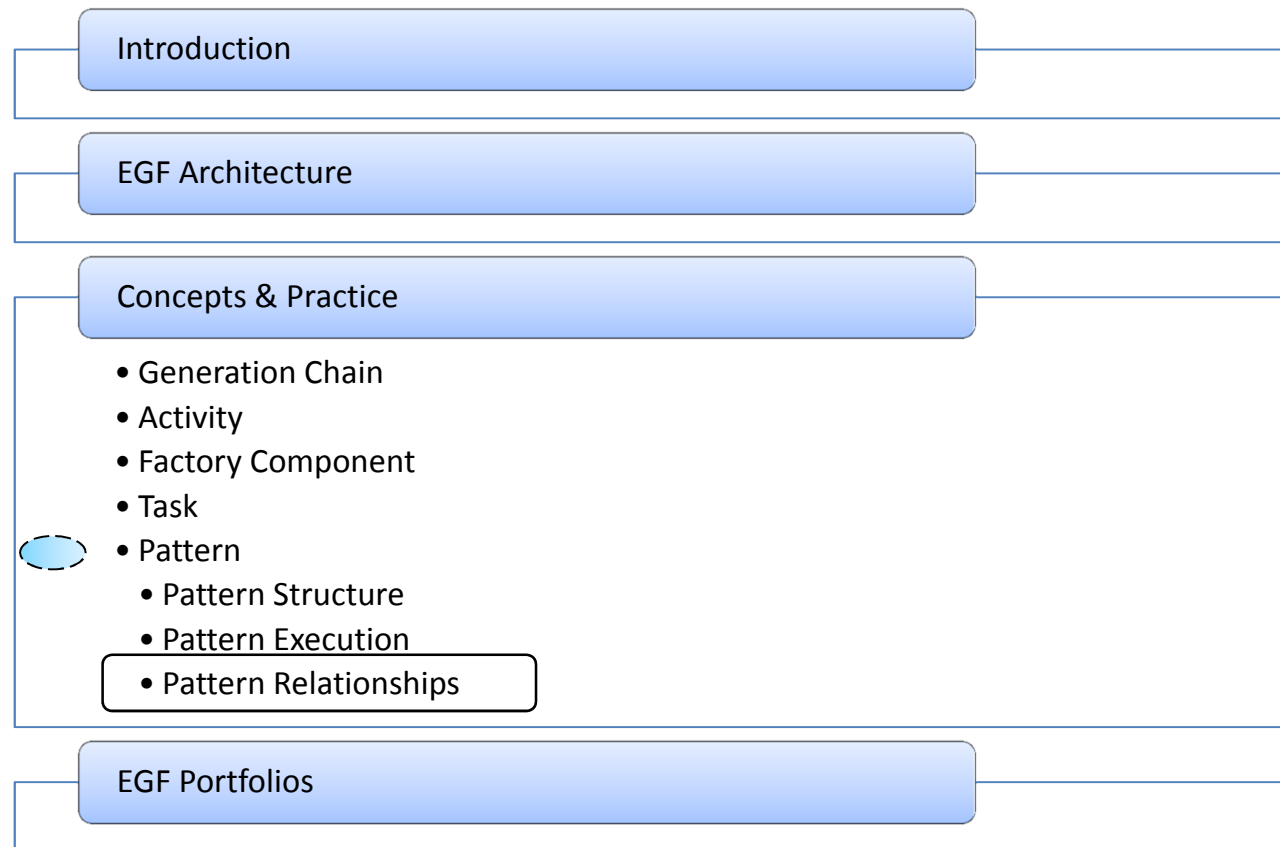
Patterns can be related together (e.g., pattern inheritance, pattern call)

The next slides present the different kinds of pattern relationships

Modèle presentation_epm version 1.0

THALES

Pattern Inheritance
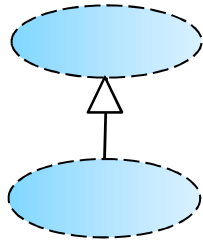
Pattern Delegation

delegates
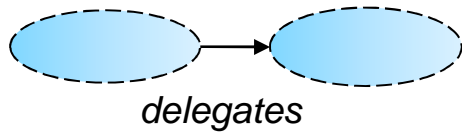
Pattern Injection

injects
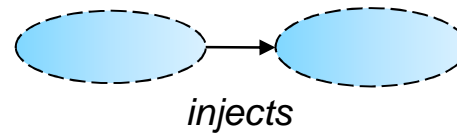
Pattern Callback

Pattern Substitution

substitutes

Pattern Merge

merges

Pattern Comparison

Modèle presentation_epm version 1.0

THALES
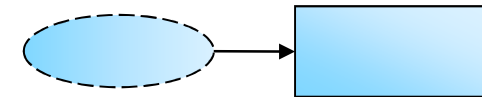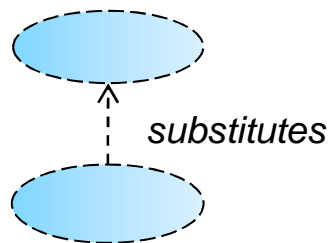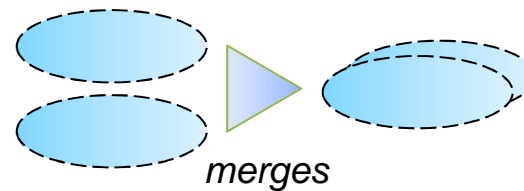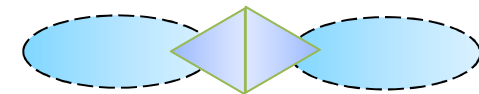
Pattern inheritance
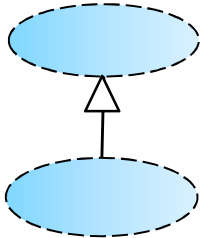
Case 1. Reuse of super-pattern methods

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

Example
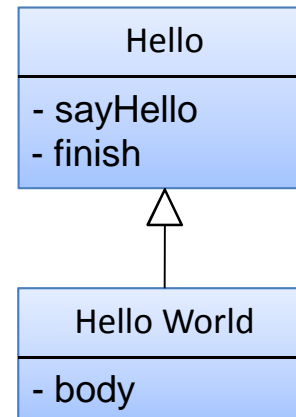
Orchestration of HelloWorld

THALES
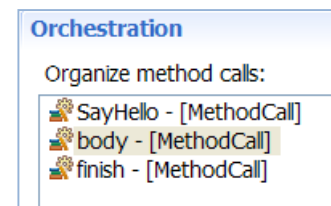
Pattern inheritance
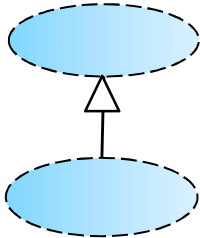
Case 2. Reuse of super-pattern orchestration

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

THALES
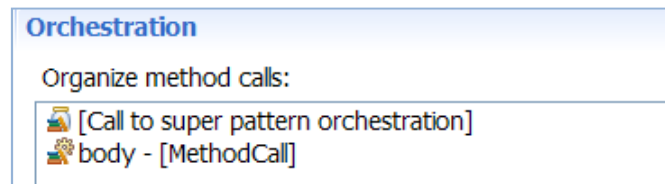
Pattern delegation

delegates
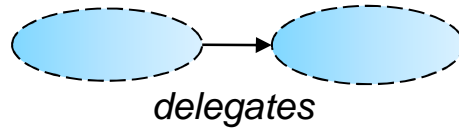
Case. For Problem decomposition & Reuse of pattern

- 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

Orchestration

Organize method calls:

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

Example 2

Modèle ...ation_epm version 1.0

THALES
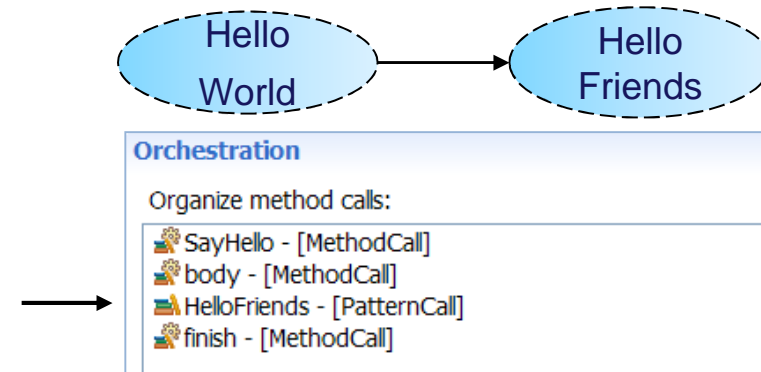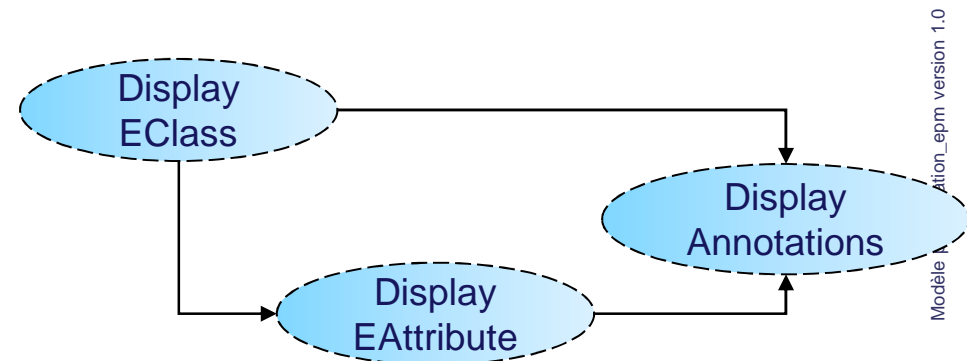
Pattern delegation
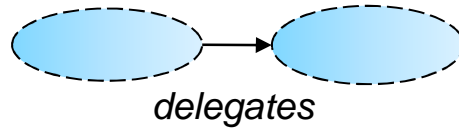
delegates

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

EClass
Report

Measure
EClass
Quality

Modèle presentation_epm version 1.0

THALES

Pattern injection
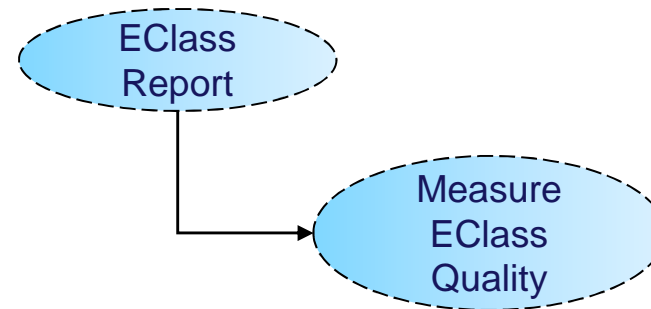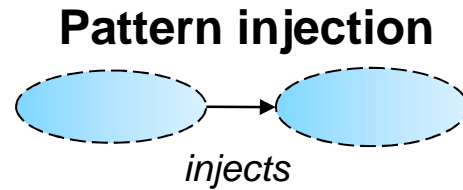
injects

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

Orchestration

Organize method calls:

body - [MethodCall]
setupVariable - [MethodCall]
ForInjectionPattern -> variable - [PatternInjectedCall]
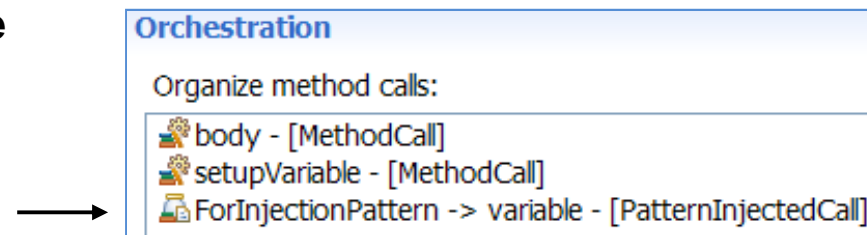
In this example, the "setupVariable" method sets the injection context

Modèle presentation_epm version 1.0

THALES
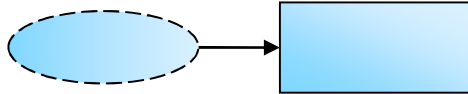
Pattern Callback

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]
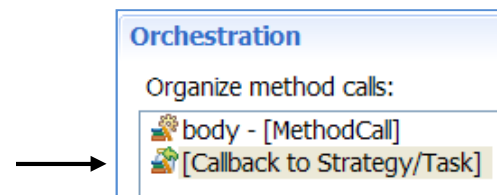
Specification of the Java Class in the production plan

> [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

THALES
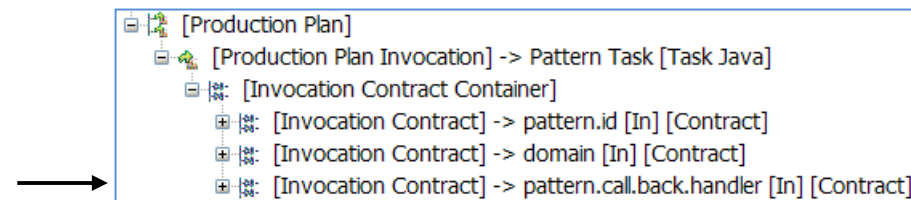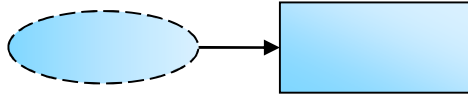
Pattern Callback

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]

THALES
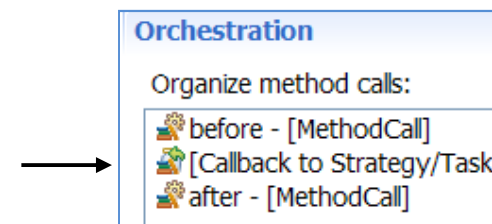
Pattern substitution
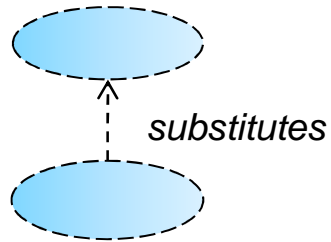
substitutes

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

THALES

Pattern Merge
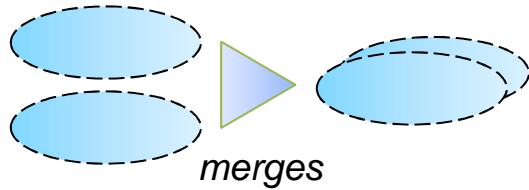
merges

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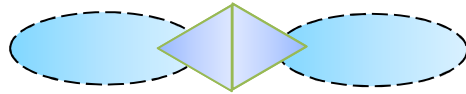
Case. Used during pattern edition – Face pattern evolution when pattern-based generation scales up

- 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.

Structure Compare

- 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

Text Compare

```
//Pattern comparison              //Pattern comparison

I am the super-pattern            I am the child pattern

// End of comparison              // End of comparison
```

THALES

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

Modèle presentation_epm version 1.0

THALES

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

Modèle presentation_epm version 1.0
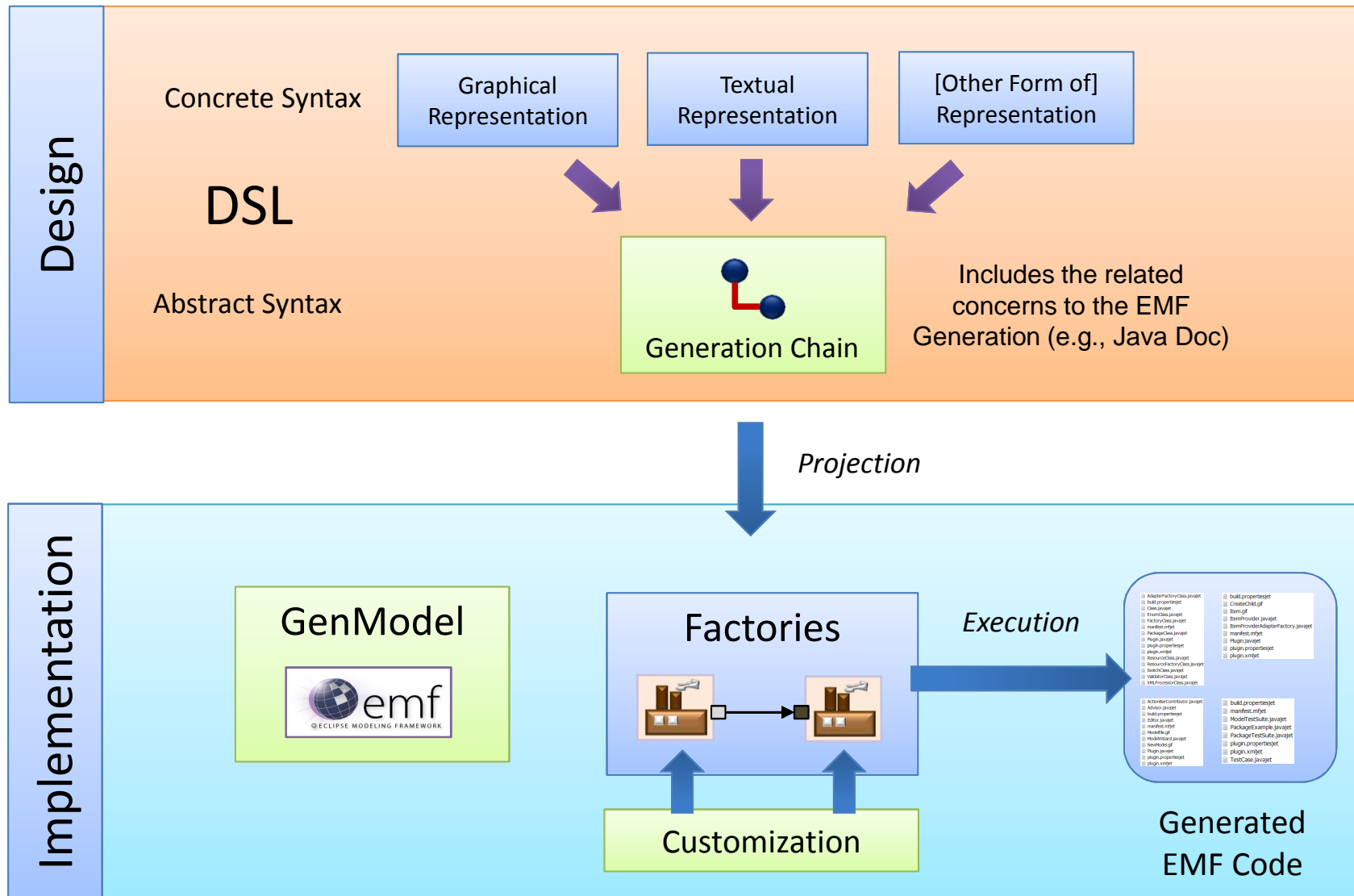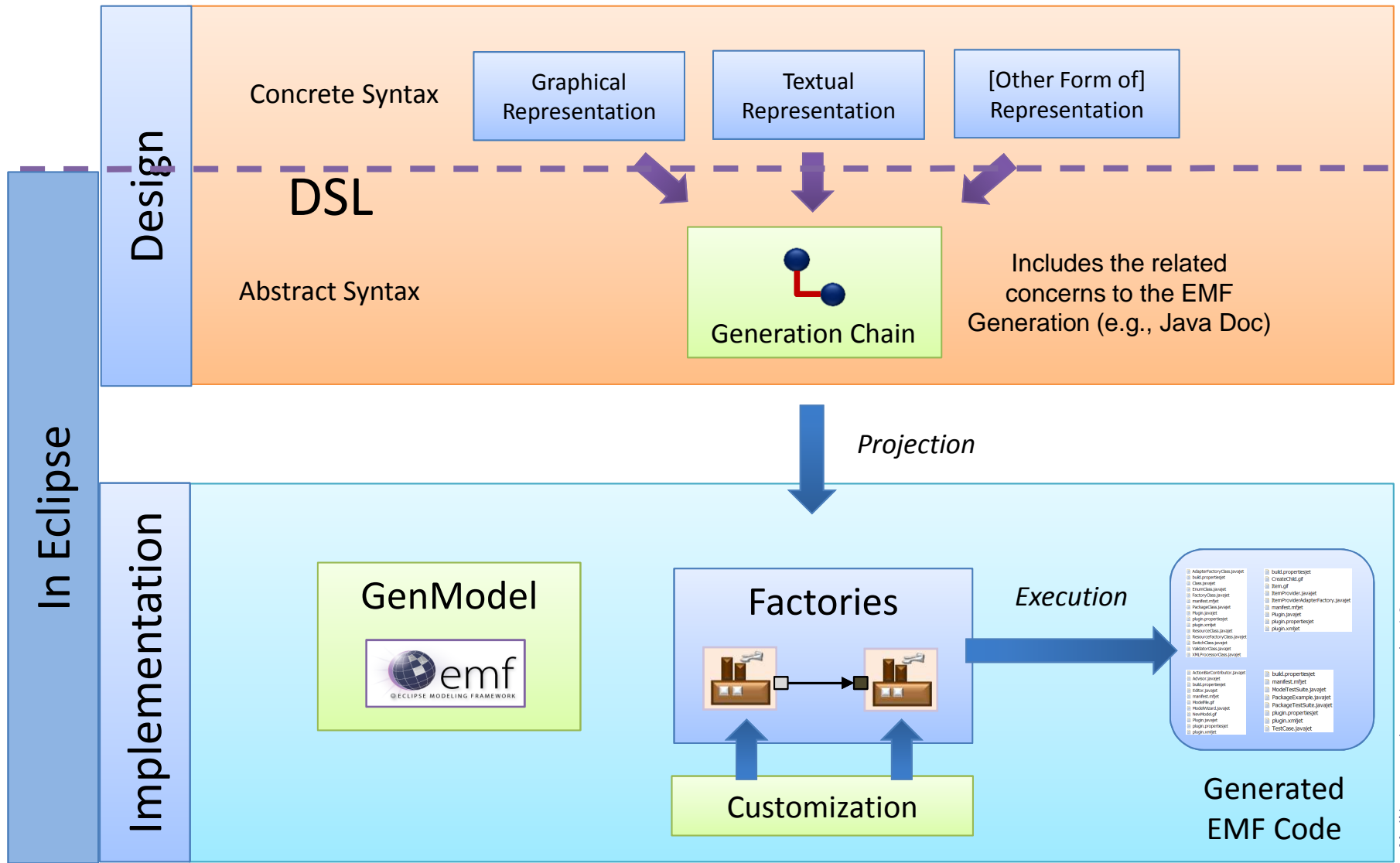
THALES

THALES

Several levels of Customization

EMF Generation Factories

EMF Generation

Reverse into patterns

EMF Generation

Team Working

Standard Portfolio

customization

Enterprise Portfolio

Team Portfolio

THALES
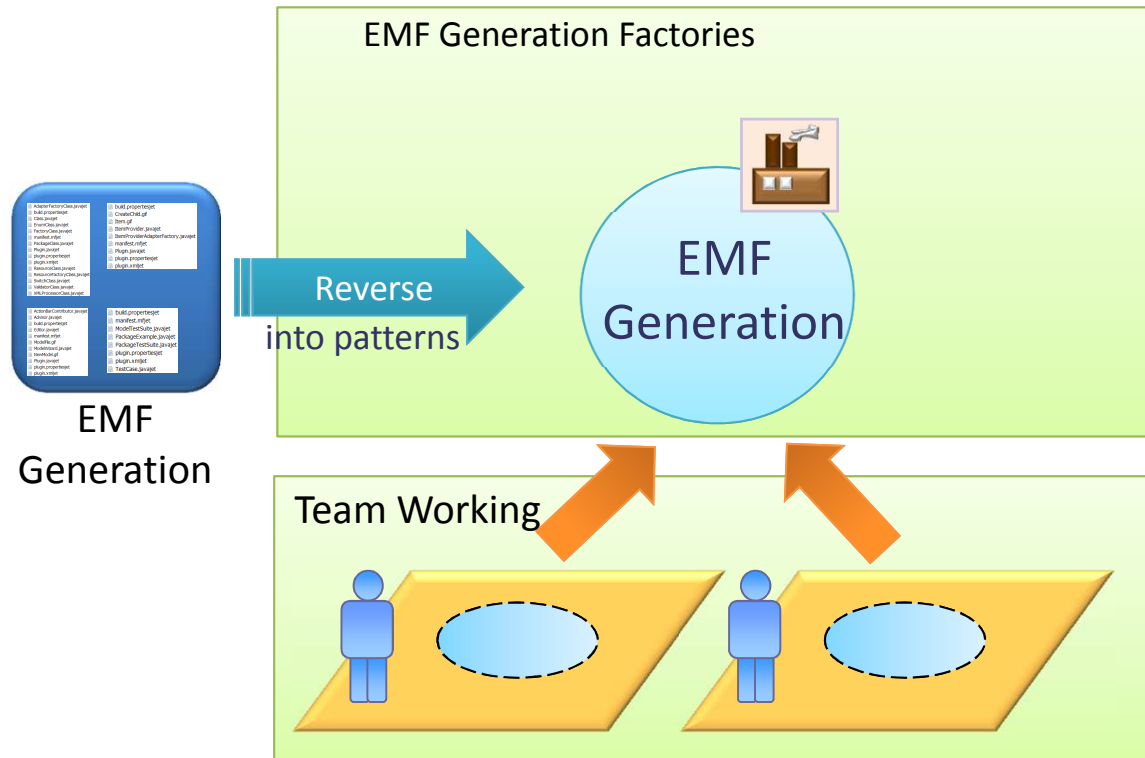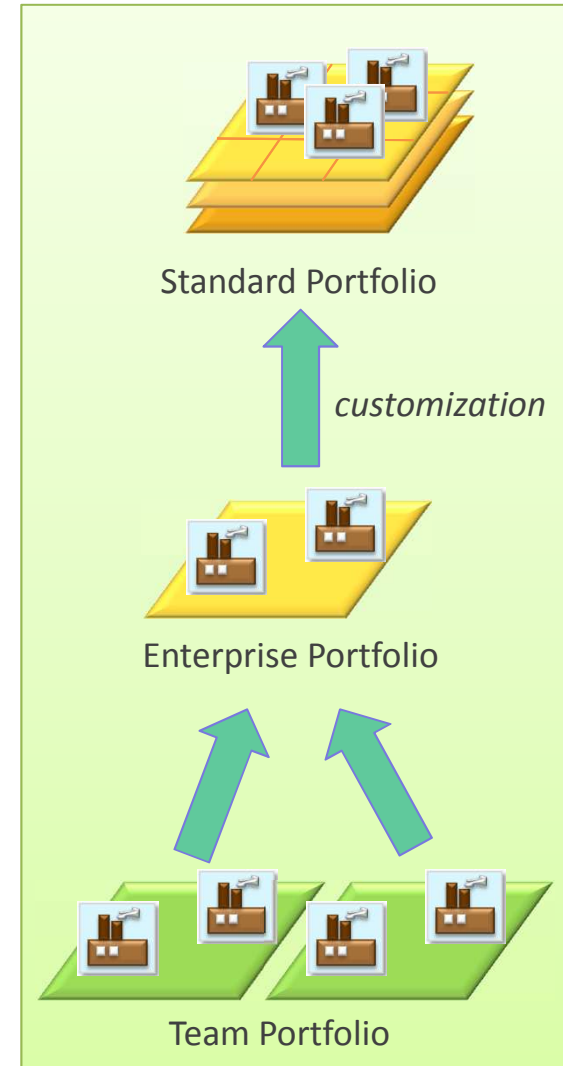
Exercices:

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

Modèle presentation_epm version 1.0

THALES

Introduction

EGF Architecture

Concepts & Practice

EGF Portfolios

- Enhancement of the EMF Generation
- Build Chain Portfolio

THALES

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

Modèle presentation_epm version 1.0

THALES

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

THALES

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

Modèle presentation_epm version 1.0

THALES

EGF

Eclipse Generation Factories

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

Modèle presentation_epm version 1.0

THALES