Introduction

Pattern Extensibility
Need of Reuse and Customization

- General needs:
  - Need #1: Ability to **reuse a portfolio**, where a portfolio is a consistent set of off-the-shelf components
  - Need #2: Ability to **customize** an off-the-shelf component in order to fit specific project’s expectations
  - Need #3: Ability to capitalize customized off-the-shelf components in order to reuse them as a **new portfolio**
Customization in the EGF Context

- **EGF Vocabulary:**
  - Portfolio = consistent set of factory components
  - Factory component = factory component parameters + viewpoints + activity workflow
  - Pattern = formalism to express systematic behavior (Java and Jet supported today) executed by a pattern activity

- **Means of Customization with EGF:**
  - Means #1: **Parameterization** of factory component
  - Means #2: **Composition**: Creation of new factory component from factory components through a specific activity workflow
  - Means #3: **Pattern extension**: Ability to extend patterns from a reused factory component portfolio without any intrusion
Creation of Customized Portfolios

Initial Portfolio
Team #1

Portfolio
Creation of Customized Portfolios

Capitalization

Portfolio
Creation of Customized Portfolios

Off-The-Shelf solution

Portfolio
Creation of Customized Portfolios

Portfolio

Off-The-Shelf solution

customization

Portfolio Adaptation
Team #2

Variants
Creation of Customized Portfolios

Portfolio

customization

Off-The-Shelf solution

Off-The-Shelf solution
Creation of Customized Portfolios

Off-The-Shelf solution

Portfolio

customization

Off-The-Shelf solution

customization

Portfolio Adaptation

Team #3

Variants
Creation of Customized Portfolios

Off-The-Shelf solution

Portfolio

customization

Off-The-Shelf solution

customization

Off-The-Shelf solution
Customization – Parameterization

Parameters

Invocation with parameters passing for contextualization
This factory component is the composition of activities (i.e., factory component, task) defined in an activity workflow.
Customization – Pattern Extension

**Effect:** The new factory component has the same behavior than the reused factory component except the pattern customization.
Agenda

Introduction

Pattern Extensibility
**Pattern – Definition**

- **Definition:**
  - Definition #1 – Rationale: A pattern is a solution to a recurrent problem
  - Definition #2 – Structural: A pattern is a formalism to express systematic behavior

- **Key points:**
  - Dissociating the specification (external view) from the implementation (internal view) of the behavior
  - Supporting multilingual patterns for the behavior implementation in order to use the best programming language for a given situation (e.g., programming language such as Java; M2T, M2M, T2M, T2T)
Pattern – Main Elements

- **Language for implementation**
- **Pattern methods**
- **Orchestration of the pattern methods**

**Specification**
- **Inheritance**
- **Pattern Nature**
- **Parameters**

**Implementation**
- **Methods**
- **Variables**

**Code**

```
Class <#aClass.getName#>
```
# Pattern – Relationships

<table>
<thead>
<tr>
<th>Relationship</th>
<th>Purpose</th>
<th>Defined in…</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inheritance</td>
<td>Inheriting properties from parent patterns</td>
<td>Specification View</td>
</tr>
<tr>
<td>Pattern Call</td>
<td>Behavioral delegation. In the orchestration, a pattern calls another pattern</td>
<td>Implementation View / Orchestration</td>
</tr>
<tr>
<td>Pattern Injection</td>
<td>Behavioral delegation. Determination of the call context at runtime</td>
<td>Implementation View / Orchestration</td>
</tr>
<tr>
<td>Multilingual Call</td>
<td>Property of the pattern call and pattern injection: a pattern implemented in a language calls a pattern implemented in another language</td>
<td>Implicit: depends on the pattern nature (implementation view) and the engine able to execute a pattern in a given language</td>
</tr>
<tr>
<td>Pattern Callback</td>
<td>Giving back the control to the pattern strategy which orchestrates the patterns to be applied over a resource (e.g., model)</td>
<td>Implementation View / Orchestration</td>
</tr>
<tr>
<td>Pattern Extension</td>
<td>Redefinition of pattern with a substitution mechanism</td>
<td>Activity invocation / substitution parameter values</td>
</tr>
</tbody>
</table>
Stage. Identification of Reused Factory Components

Initial Patterns

In the Reused Factory Component

Pattern Library contains P1

Pattern

P2
Mechanism of Pattern Substitution

Stage. Pattern Definition – Customization Time

In the Reused Factory Component

Initial Patterns

Pattern Library

contains

P1

P2

Pattern

In the Factory Component for Customization

Patterns for substitution

Pattern Library

contains

PA

PB

PC

Substitutions

P1

PA

PB

P2

PC

Stage. Pattern Definition – Customization Time
Mechanism of Pattern Substitution

Stage. At Runtime

**Initial Patterns**

In the Reused Factory Component

- Pattern Library contains \( P_1 \)
- \( P_1 \) contains \( P_A \) and \( P_B \)
- \( P_A \) contains \( P_C \)

Patterns for substitution

In the Factory Component for Customization

- Pattern Library contains \( P_A \)
- \( P_A \) contains \( P_B \) and \( P_C \)
- \( P_2 \) contains \( P_C \)
Reused Factory Component

This declaration means that this FC accepts substitution

Initial Patterns

Factory Component for Customization

Pattern for substitution

List of substitutions in an activity invocation of the reuse factory component

One substitution

Initial Patterns

Pattern for substitution

List of substitutions in an activity invocation of the reuse factory component

One substitution
Merge of Pattern Substitution

- When there is more than one substitution, a merge of pattern substitution is necessary.
- A merge of pattern substitution consists in merging two lists of pattern substitution.
- It is possible to merge pattern substitution in series.