EGF Tutorial
EMF Generation Patterns

Benoît Langlois – Thales/EPM
Agenda

- Introduction
- Process
- Benefits of the Pattern Technique
- Best Practices for the EMF Generation
What is EGF?

- EGF (Eclipse Generation Factories) is an Eclipse open source project under the **EMFT project**

- **Purpose:** providing a **model-based generation framework**

- **Operational objectives:**
  - Supporting complex, large-scale and customizable generations
  - Promoting the constitution of generation portfolios in order to capitalize on generation solutions
  - Providing an extensible generation structure
The EMF generation is central for model-based developments,

But limits exist to the EMF generation today, e.g.:
- Some Jet files are monolithic, problem of readability due to the generation complexity
- Reuse / customization: problem of capitalization and scalability for large-scale applications with common and specific needs

Work realized with EGF:
- Transformation of the Jet files for the EMF generation into patterns

Added-value:
- Clarification of the EMF generation
- Taking profit from the pattern technique
Process Overview

EMF Generation Jet files

Reverse of EMF Generation

Activity in the scope of the EGF project

Standard EMF generation with patterns

Project activities

Customization of the EMF Generation with patterns

EMF generation variant

EMF Generation with EGF

Model API

Model
Reverse of the EMF Generation into Patterns

Jet files

Patterns organized in pattern libraries

Model

Edit

Editor

Tests
Reverse of EMF Generation

Jet files → Patterns organized in pattern libraries

Model

EGF: Eclipse Generation Factories – Thales Corporate Services/EPM

EGF Tutorial – EMF Generation Patterns 0.1.0 | © 2010 by Thales; made available under the EPL v1.0
Example of Decomposition

Jet code is distributed in the implementation part of the different patterns accordingly.
Agenda

- Introduction
- Process
- Benefits of the Pattern Technique
- Best Practices for the EMF Generation
**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:**
  - A pattern conforms to a language and is executable
  - The pattern specification reflects the external view (e.g., parameters), while pattern implementation reflects the internal view (e.g., methods)

- **Introduction to patterns:**
  - Tutorials: “EGF Tutorial”, “Reuse and Customization”
  - Examples: Pattern Use Cases 1 and 2, EMF generation use cases
Benefits for the EMF Generation

- **Pattern = generation unit**
  - Interest of the problem decomposition by pattern
- **Extensibility**
  - Interest of the pattern inheritance and pattern call mechanisms
  - Ability to change / extend patterns by a substitution mechanism
  - Ability to combine patterns written in different languages
- **Team management**
  - Possibility to have several contributors
- **Toward product lines**
  - Autonomy to create generation variants
Agenda

- Introduction
- Process
- Benefits of the Pattern Technique
- Best Practices for the EMF Generation
Best Practices with Patterns

• Next slides present best practices that can be introduced in the EMF generation

• Level of confidence:
  - ★★★ ★★ Tested
  - ★★★ ★ Experimented
  - ★★★★★ ★ Operational
Pattern Best Practices
**Pattern Adaptation**

Pattern methods: m1, m2, m3
Orchestration: m1, m3, m2

Pattern

Pattern methods: m1, m4
Orchestration: applies super-pattern orchestration, called pattern, m4

**Pattern Adaptation**: Reusing orchestration of the parent pattern and adding orchestration specificities (e.g., method polymorphism, pattern call)

**Example**: Redefinition of the getText pattern
Pattern Alternative: A super-pattern defines a prototype; a sub-pattern is applied when its condition is satisfied; the prototype is redefined. Possibility of exclusive / inclusive alternatives.

Example: Method contents depends on metamodel conditions
Pattern Merge: Merge of two pattern lists

Example: Combining standard and customized generations
Separation of Concerns: A standard generation delegates processing for a specific concern

Example: During an EMF generation, invocation of pattern for a text-to-text transformation based on an AST analysis in order to modify method content.
Bridge of Language: A pattern written in a language calls a pattern written in another language

Example: a Jet pattern calls a pattern in Java / in another model-to-text language
Portfolio Best Practices
Customization by Substitution: Extension of a pattern-based standard generation with patterns for customization

Example: Redefinition of the insert/override patterns
**Generation Façade**: A façade hides a standard generation and customization in the façade, and takes into account provided customizations.

**Example**: Creation of a standard EMF generation for a company / department.
Organizational Best Practices
**Generation Variation**: Teams isolate and apply different generations based on the same standard generation.

**Example**: Two teams extend differently the EMF generation.
Scenario of Customization in series

Portfolio

EMF Generation
Scenario of Customization in series

Portfolio Adaptation
Team #1

EMF Generation

Portfolio

customization
Scenario of Customization in series

Portfolio

customization

EMF Generation Variant #1

EMF Generation
Scenario of Customization in series

EMF Generation

Portfolio

customization

EMF Generation Variant #1

customization

Portfolio Adaptation Team #2

EGF: Eclipse Generation Factories – Thales Corporate Services/EPM

EGF Tutorial – EMF Generation Patterns 0.1.0 | © 2010 by Thales; made available under the EPL v1.0
Scenario of Customization in series

Portfolio

EMF Generation

EMF Generation Variant #1

customization

EMF Generation Variant #2

customization