THALES

A Quick Tour of EGF

http://www.eclipse.org/egf
http://wiki.eclipse.org/EGF

Benoît Langlois – Thales/EPM

What is EGF?

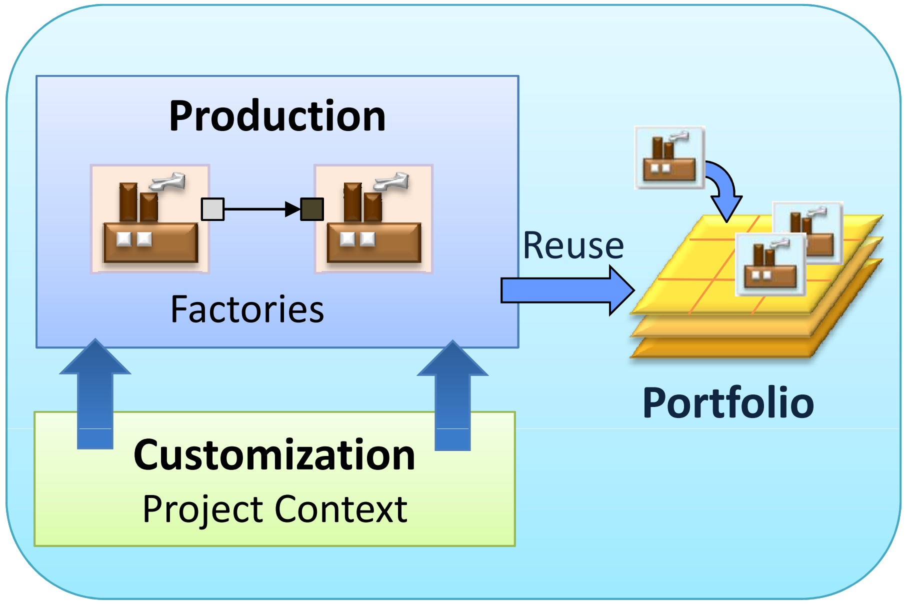
- EGF (Eclipse Generation Factories) is an Eclipse open source component project in incubation under the EMFT project.

- **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

THALES

EGF in a nutshell



Main Features:

- Software production with **factory components**
- **Reuse** of off-the-shelf factory components
- Development by **assembly**
- **Customization** in a specific context
- Capitalization with **executable patterns**

Types of Generation Units

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

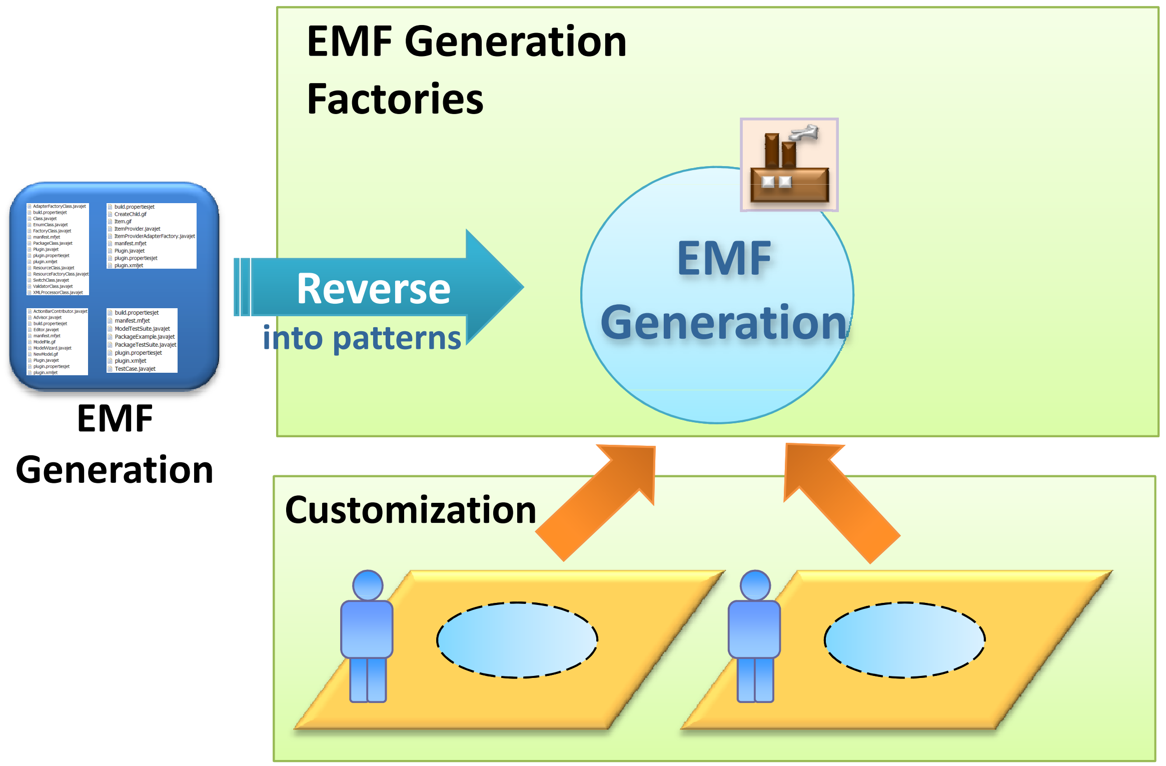
Generation Chain
High generation view to organize complex generations

EGF Pattern
Description of systematic behavior

Example: Enhanced EMF Generation

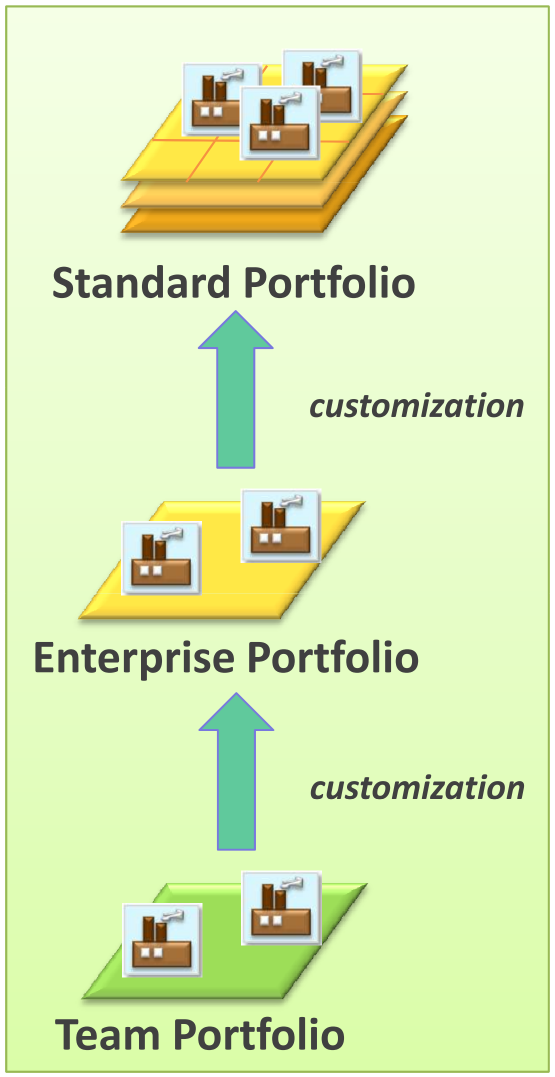
EMF Generation completely ensured by EGF

- Example of complex generation
- Interest of EMF customization
- Open customization between team members



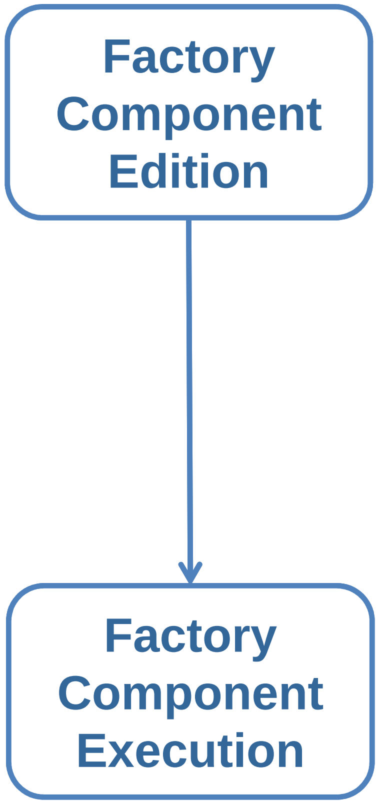
Enriching the standard generation
Accurate customization of the EMF generation

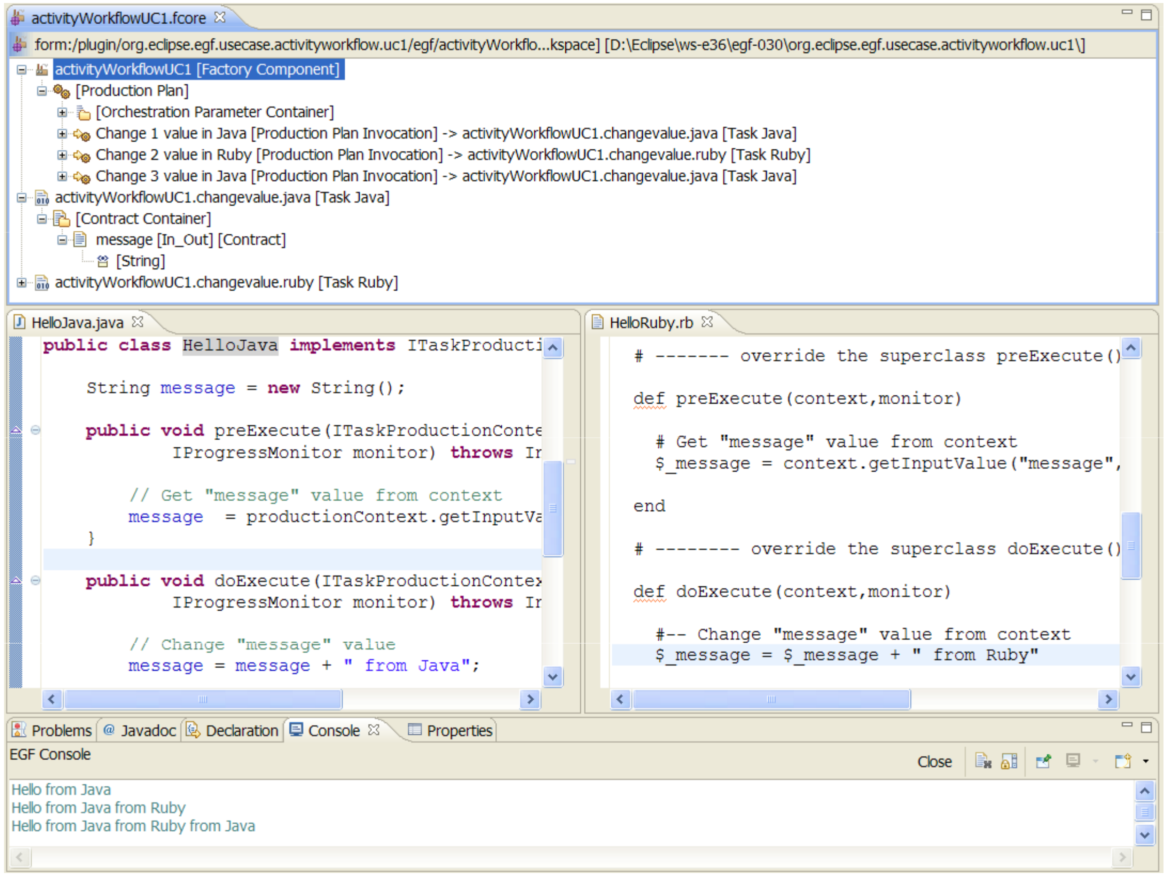
Several levels of Customization



THALES

Factory Component Lifecycle





```
form:/plugin/org.eclipse.egf.usecase.activityworkflow.uc1/egf/activityWorkflo...kspace] [D:\Eclipse\ws-e36\egf-030\org.eclipse.egf.usecase.activityworkflow.uc1\]
activityWorkflowUC1 [Factory Component]
    [Production Plan]
        [Orchestration Parameter Container]
        Change 1 value in Java [Production Plan Invocation] -> activityWorkflowUC1.changevalue.java [Task Java]
        Change 2 value in Ruby [Production Plan Invocation] -> activityWorkflowUC1.changevalue.ruby [Task Ruby]
        Change 3 value in Java [Production Plan Invocation] -> activityWorkflowUC1.changevalue.java [Task Java]
activityWorkflowUC1.changevalue.java [Task Java]
    [Contract Container]
        message [In_Out] [Contract]
            [String]
activityWorkflowUC1.changevalue.ruby [Task Ruby]
```

```java
public class HelloJava implements ITaskProducti

    String message = new String();

    public void preExecute(ITaskProductionConte
            IProgressMonitor monitor) throws Ir

        // Get "message" value from context
        message  = productionContext.getInputVa
    }

    public void doExecute(ITaskProductionContex
            IProgressMonitor monitor) throws Ir

        // Change "message" value
        message = message + " from Java";
```

```ruby
# ------- override the superclass preExecute()

def preExecute(context,monitor)

  # Get "message" value from context
  $_message = context.getInputValue("message",

end

# -------- override the superclass doExecute()

def doExecute(context,monitor)

  #-- Change "message" value from context
  $_message = $_message + " from Ruby"
```

```
Hello from Java
Hello from Java from Ruby
Hello from Java from Ruby from Java
```

- Management of the workspace / target platform / runtime environment
- Edition/Execution of the activity workflow (production plan)

THALES