**FACTORY COMPONENT**

Author: Benoît Langlois – benoit.langlois@thalesgroup.com
Version: 1.0

**DEFINITION**
A factory component is a composite activity. Structurally, it contains a set of viewpoints and a production plan. Behaviorally, it delegates its activities to other factory components or tasks.

**OBJECTIVES**
The objectives of a factory component are to:
- Provide reusable generation service.

The interests are to:
- Assemble and reuse generation activities.

**CONCERNS**
- The designer functionally assembles factory components and tasks.
- The designer executes factory components.
- The developer develops the generation aspects which require code (e.g., patterns).
Comprehension:

- A Factory Component is an Activity with:
  - A production plan as a workflow to assemble and invoke activities (i.e., Factory Components or Tasks).
  - A set of Viewpoints to describe aspects, such as domain declaration or patterns, used in the production plan.
- A Production Plan invokes a suite of Activities where each Activity Invocation provides a value to an identified Contract (e.g., parameter) of the invoked Activity. Then, a Factory Component is called with valued Contracts which can be reused later in Activity Invocations. The principle is the same for a Task which provides a valued Contract to a Task implementation. With the In/Out/InOut Contract mode, a parameter can have its value read or changed several times along the Production Plan.
EXAMPLES

The following figure shows a Pricer which computes an amount from a price and quantity, displays the computed amount, and returns the amount. The computation and display steps are each time delegated to a specific task.

The second example displays in a console the classes and attributes of the EGF Fcore model. The Factory Component contains a Viewpoint Container which declares the Fcore.ecore model as a Domain Viewpoint and Patterns in a Pattern Viewpoint. The Production Plan delegates the job to a “Domain Driven Pattern Strategy” Task which navigates over the Fcore model, applies the two patterns (i.e., classPattern and attributePattern), and reports the result to the console. The result is shown in the next figure.

Those examples are available in the Basic EGF Examples distributed with EGF.

Figure 2. Example of Pricer with a Factory Component
Figure 3. Example of model output with Patterns

Figure 4. Result in the console
**Domain Types**

In the Domain Viewpoint are declared a set of domains (aka resources), instances of domain types. The following table details the standard domain types supported by EGF. A domain is potentially any kind of structured data representation.

<table>
<thead>
<tr>
<th>Domain Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EMF Domain</td>
<td>EMF Ecore model</td>
</tr>
<tr>
<td>File System Domain</td>
<td>File and directories</td>
</tr>
<tr>
<td>Workspace Domain</td>
<td>Eclipse IFiles and IContainers</td>
</tr>
</tbody>
</table>

*Table 1. Standard Domains provided by EGF*
The section presents the process dimension from the designer and developer viewpoints.

**Designer viewpoint**

<table>
<thead>
<tr>
<th>Activity</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Create Factory Component</strong></td>
<td>The Designer creates a Factory Component in an fcore file.</td>
</tr>
<tr>
<td><strong>Edit Contract</strong></td>
<td>The Designer defines the Contracts (aka parameters) of the Factory Component.</td>
</tr>
<tr>
<td><strong>Edit Viewpoint</strong></td>
<td>The Designer edits a specific Viewpoint with its own formalism.</td>
</tr>
<tr>
<td><strong>Edit Production Plan</strong></td>
<td>The Designer edits the Production Plan in order to define the workflow of Contract values between the Factory Component Contracts and different Activity Invocations.</td>
</tr>
<tr>
<td><strong>Execute Factory Component</strong></td>
<td>The Designer executes the Factory Component.</td>
</tr>
</tbody>
</table>

*Table 2. Designer activities*

The Designer has to take care of two points:

- Factory Component naming: like in Java, a namespace in the Factory Component name may reflect a classification to organize Factory Components.
- Factory Component Contracts: they define the interface of the Factory Component.
Developer viewpoint

The Developer develops all the viewpoint artifacts such as Patterns.

Table 2. Developer activities