

ACTIVITY

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

Version: 1.0

DEFINITION

An activity is an executable transformation unit with a contract.

OBJECTIVES

The objectives of an activity are to:


- Define a set of data and behaviors common to any EGF generation unit, and more largely to any EGF transformation unit.

The interests:

- An activity is the super-class of any transformation unit. It is not visible by the EGF users. It provides the common and essential structure and mechanisms to the units which derived from it, such as task and factory component.

CONCERNS

In the following table, the user represents a designer as well as a developer.

 User	<ul style="list-style-type: none">• The user manages fcore files which contains a set of activities• The user manages specific activities• The user executes activities
---	---

STRUCTURE

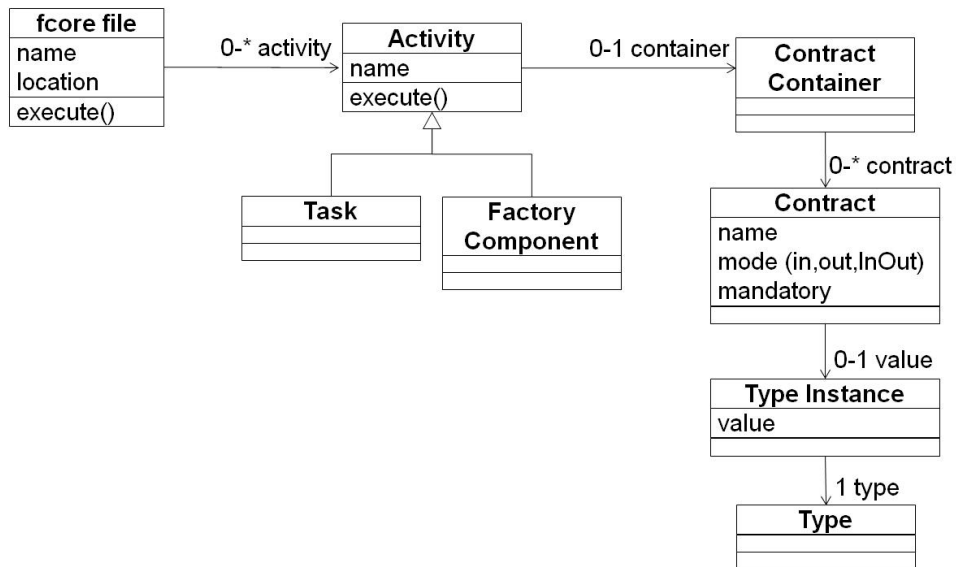


Figure 1. Activity Structure

Comprehension

- An fcore file contains a set of activities
- An activities contains a set of contracts (aka parameters)
- A contract has a passing mode (In, Out, InOut), is mandatory or not
- When a contract has a value, it is an instance type
- fcore files and activities are executable

Types

EGF proposes by default a set of common types (e.g., String, Integer, Long, Short), offers the possibility to reference an existing type (e.g., a Java Class), and to enrich the default EGF default types.

Contract

A contract defines an activity contract (aka activity parameter). A contract is typed. A type instance is used to define a default value. A contract is mandatory or not, and has a passing mode (In, Out, InOut).

EXAMPLE

In the following figure, the `FC_Hello.fcore` file contains two activities: the `FC_Hello` factory component and the `HelloT1` task. `FC_Hello` has a string contract, `message`; `HelloT1` also has a string contract, `message`. The red point on the contract means that there is a default value. In the two cases, the contract passing mode is the input mode.

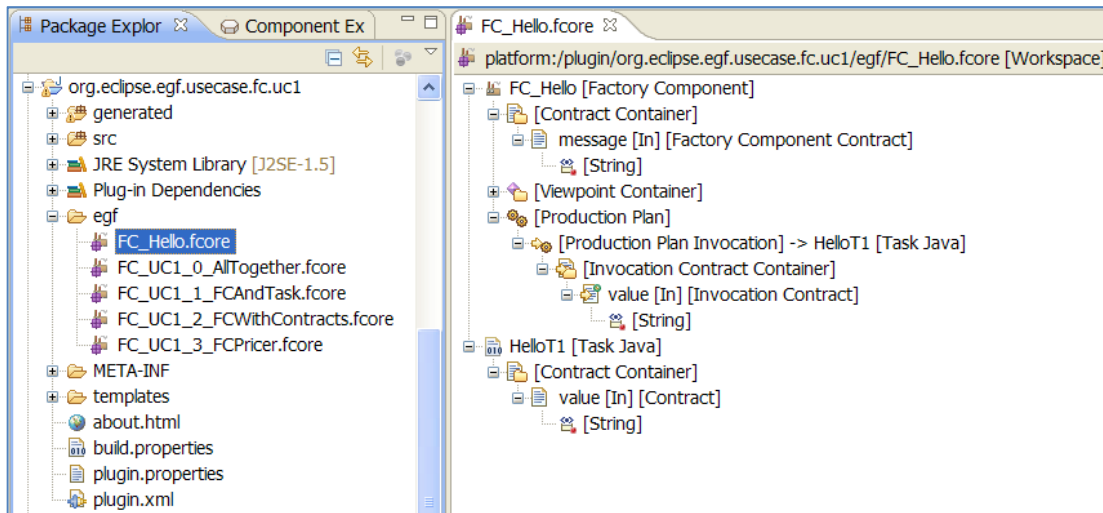


Figure 2. Instance of Generation Chain

BEHAVIOR

For efficiency, an activity which is edited can be immediately executed.

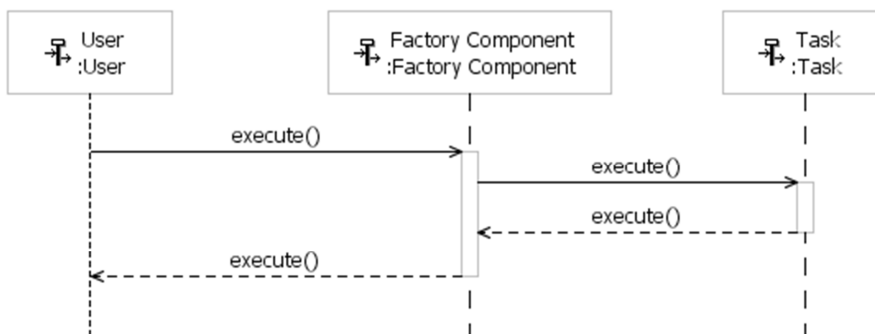


Figure 3. Execution of Activities

This sequence diagram depicts a typical sequence of activity execution:

- A user executes a factory component which is an activity
- This factory component executes a task, another activity

Next figure depicts the case where the user executes activities from an fcore file. When there are several activities in the same fcore file, the user has to select the activity to be executed.

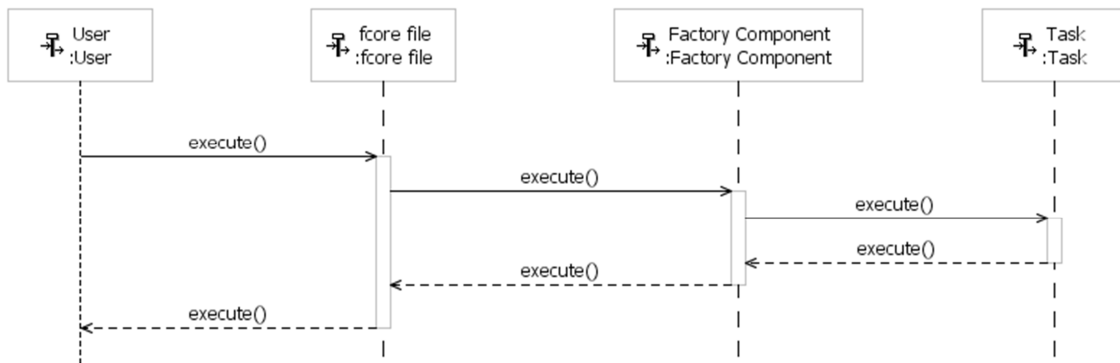


Figure 4. Execution of Activities

Dynamic execution

Dynamic execution enables to execute an activity while it is edited:

- Technically, as depicted in the following figure, an activity is validated before its execution. When valid, it is executed; otherwise a report lists the detected warning and errors.
- From the engineering viewpoint, for reasons of efficiency, development and test can be led simultaneously.

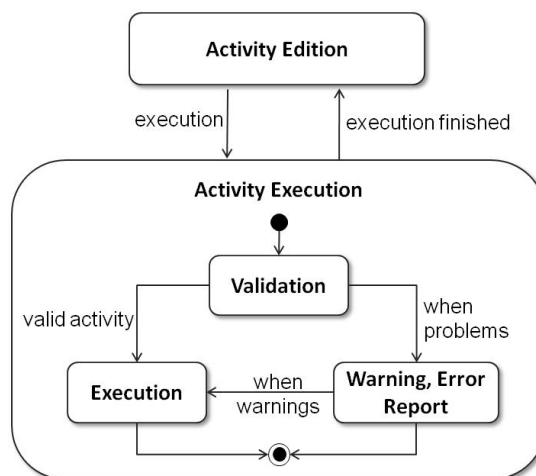


Figure 5. Activity Execution Lifecycle

PROCESS

An activity is an abstract element specialized in specific transformation units, such a factory component. The section presents the common process to any activity.

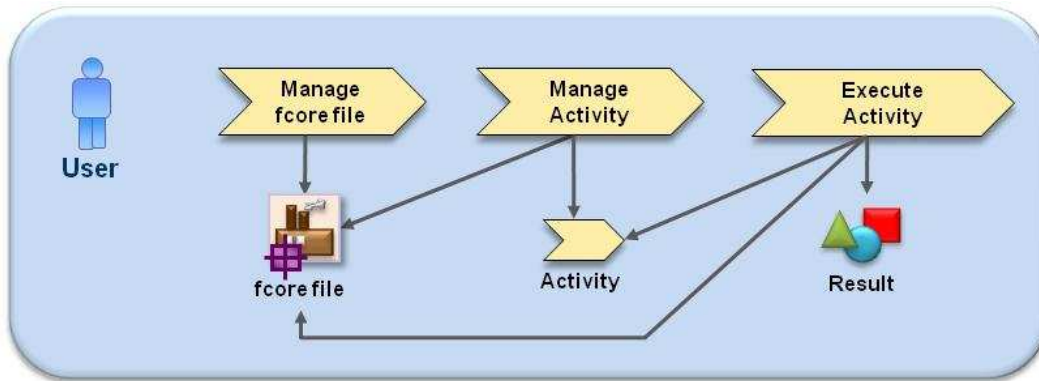


Figure 6. Activity Process

<i>Manage fcore file</i>	The user creates / renames / deletes fcore files which contain activities (e.g., tasks).
<i>Manage Activity</i>	The user creates / edits / moves / copies/ deletes a specific activity. An activity has its own edition mode and lifecycle.
<i>Execute Activity</i>	The user executes activities either directly from an activity or from an fcore file. This principle is also true in headless mode.

Table 1. Description of the process activities