The issue

• The M2T transformation generally combines:
  ▸ The logic of text generation from a model,
  ▸ Text transformations collateral to M2T transformation,
  ▸ The logic of storage of the generation result.

• The objective is to dissociate those three concerns
Need of Post-Processing and Reporting

- **Needs of Post-Processing:**
  - Need #1: Ability to realize text transformations following the M2T, independently of the logic of M2T transformation, such as text reformatting.
  - Need #2: Ability to transform text with a context independent of the M2T context.

- **Needs of Reporting:**
  - Need #3: Ability to dissociate the logic of M2T storage from the logic of M2T transformation (e.g., storage in one or several files, at one or another location).

- **Common Needs:**
  - Need #4: Ability to consider the post-processor (managing the post-processing) and the reporter (managing the reporting) as parameters of the M2T transformation, in order to reuse the M2T transformation in different ways.
Steps of the Text Generation Activity

1. **Resource**
   - Heart of the generation, this step consists in transforming a model into text with patterns.

2. **M2T Transformation**
   - This step consists in applying a text transformation independent of the logic of M2T transformation.

3. **Post-Processing**
   - This step consists in realizing the final reporting operations and storing the generation result.

4. **Reporting**

---

 EGFR: Eclipse Generation Factories – Thales Global Services

EGF Tutorial | © 2012 by Thales; made available under the EPL v1.0
Parameterization

- Resource
- M2T Transformation
  - Choice of the M2T language and associated engine, transformation strategy, resource visitor
- Post-Processing
  - Choice of the post-processor
- Reporting
  - Choice of the reporter
- Generation Result
Process for Reporting a M2T Transformation

1. For each Pattern
   - Pattern execution
     - Generated text from one pattern
   - PatternOutputProcessor. applyOnLoopResult()
     - Post-processed text
   - PatternExecutionReporter. loopFinished()
     - (optional) Reporting by pattern

2. End of Generation
   - PatternOutputProcessor. applyOnExecutionResult()
     - Post-processed text
   - PatternExecutionReporter. executionFinished()
     - (optional) Reporting of the complete generation

Generated and post-processed text for all patterns

M2T Transformation

Post-Processing

Reporting
Organization of the Report

• The report is incrementally created during the generation

• Internal structure of the report
  ▸ The report structure is built during the pattern execution
  ▸ The report structure is organized as a composite pattern
    ▹ A Container reflects the pattern call orchestration
    ▹ A DataLeaf contains the generated and post-processed text
  ▸ The report tree corresponds to a syntactical tree for reporting

• Modifying and Building up the final text
  ▸ Each DataLeaf is accessible by navigating over the report tree; for identification, each Container is associated to its source pattern
  ▸ The final text, transmitted to the reporter, is built by navigating over the Containers of the report tree and concatenating the DataLeaf texts of the Containers
• The report tree is not the concern of the reporter, which has no visibility on

• LoopFinished environment:
  ▶ There is one loop by tuple of resource elements matching the pattern parameters. Ex: each class of an ecore model instance which matches an EClass pattern parameter.
  ▶ parameterValues: key/value of each pattern parameter value
  ▶ Output: result of pattern execution for one tuple; this output is already post-processed.

• executionFinished environment:
  ▶ Output: final result of the execution of all the patterns; this output is already post-processed.