TECHNISCHE UNIVERSITÄT DRESDEN

Institute for Software- und Multimedia-Technology Software Technology Group

emftext

concrete syntax mapper

Why create textual syntax for models?

Readability: Custom text syntax is much easier to read than XML/XMI. It can utilise special characters to express the meaning of model elements.

Diff/Merge/VCS: Comparing and diffing XML (the default representation for EMF models) often produces useless results. Even small changes made to a model result in large diffs. Using custom text syntax for models can support this central development activity.

Evolution: Whenever metamodels change, existing model instances may not be readable anymore. Text syntax allows to extend the file format for models and to preserve backward compatibility.

Tool autonomy: Text can be edited on every OS with a simple text editor. XML or binary files are hard to edit without sophisticated tools. If those tools can not read existing files users are doomed. Text can always be edited.

Quick model instantiation: The default HUTN syntax that can be fully automatically generated by EMFText from a given metamodel instantly allows to create model instances. In contrast to the EMF default tree editor, text can be created very fast. Model instances can thus be obtained quickly.

Why create models from text?

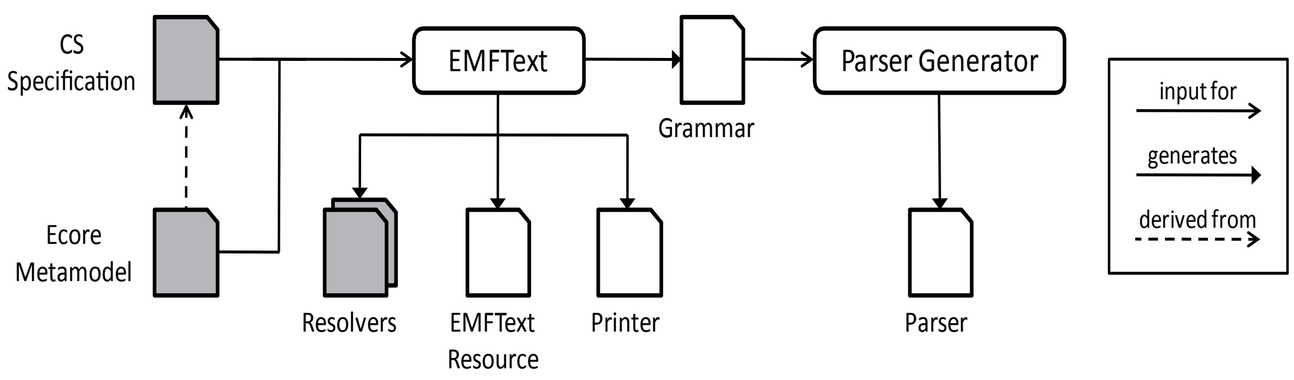
Tool reuse: There are many tools that operate on models and in particular on Ecore models. Deriving a model from text allows to use these tools. One can transform documents (e.g., using ATL) or analyze them (e.g., with OCL).

Know-how reuse: The aforementioned tools employ different languages to operate on models. Treating text documents as models allows to apply these languages to all kinds of documents. The know-how of developers can be applied to a much broader field. Once a developer knows, e.g., OCL, she can query all kinds of model-based documents.

Explicit representation of structure: To transform text to models, a metamodel that specifies the elements contained in the text is needed. This metamodel explicitly describes the structure of text documents. Meta information about the documents is no longer hidden in parsers (or other source code).

Tracing software artifacts: To trace requirements from initial documents down to the running code, links between artifacts must be established. Creating models from text allows to trace semantic units (model elements) instead of single characters. Text (e.g., program code) can be changed without loosing track of the trace relations between artifacts.

Approach



Features

- Automatic generation of default syntaxes
- Simple and precise syntax specification
- Modular syntax specification
- ANT support
- Default reference resolving mechanisms
- Comprehensive syntax analysis
- Outline View
- Code Completion
- Customizable Syntax Highlighting
- More than 35 example languages (including Java)

TreeModel.quml

```
m TreeModel {
  p BasicTreePackage {
    c Node {
      a parent : Node 0..1
      a children : Node 0..-1

      o addChild (p newChild : Node)
      o removeChild (p childToRemove : Node)
    }

    c NodeSet {
      a nodes : Node  0..-1
    }
  }
}
```

Outline

```
<Model> TreeModel
  <Package> BasicTreePackage
    <Class> Node
      <Property> parent : Node [0..1]
      <Property> children : Node [0..*]
      <Operation> addChild (newChild : Node)
      <Operation> removeChild (childToRemove : N
    <Class> NodeSet
      <Property> nodes : Node [0..*]
```

Example UML Model in Textual Syntax

feature.cs

```
SYNTAXDEF feature
FOR <http://www.tudresden.de/feature>
START FeatureModel

OPTIONS {
    tokenspace = "1";
    resourcePluginID = "org.emftext.language.feature.resource.feature";
    basePackage = "org.emftext.language.feature.resource.feature";
}

TOKENS {
    DEFINE COMMENT$'//'(~('\n'|'\r'))*$;
    DEFINE INTEGER$('-')?('1'..'9')('0'..'9')*|'0'$;
    DEFINE FLOAT$('-')?(('1'..'9') ('0'..'9')* | '0') '.' ('0'..'9')+ $;
}

RULES {
    FeatureModel ::= "FeatureModel"  name['"','"']
                     ( "{" "constraints"  ( constraints ";")? "}" )?
                     root;
    Constraint   ::= "Constraint" language[] ":" expression['"','"'];
    Feature      ::= "Feature" name['"','"']
                     ("{" minCardinality[INTEGER] ".." maxCardinality[INTEGER] "}")?
                     ("[" attributes* "]")?
                     ( groups* )?  ;
    Group        ::= "Group" ("{" minCardinality[INTEGER] ".." maxCardinality[INTEGER] "}" )?
                     ("{" childFeatures* "}")?;
    Attribute    ::= type[] name[] #1 "=" value['"','"'] !2;
}
```

Textual Syntax Definition for Feature Models

Contact:

TU Dresden - Institute for Software- and Multimedia-Technology
Software Technology Group
01062 Dresden

Tel.: +49 351 463 - 38463
http://st.inf.tu-dresden.de

EMFText Mailinglist:
emftext-users@mail-st.inf.tu-dresden.de

Web:
http://www.emftext.org
http://www.emftext.org/index.php/EMFText_Contact

MODELPLEX feasiPLe MOSTPROJECT