A megamodel for Object/XML mapping

Jean-Marie Favre[1] Ralf Lämmel[2]
Andrei Varanovich[2]
[1] Université Joseph Fourier, Grenoble, France
[2] Software Languages Team, Universität Koblenz-Landau, Germany

September 14, 2013

Abstract

This is an emerging megamodel for Object/XML mapping technologies. See the related megamodeling paper for details: `http://softlang.uni-koblenz.de/mega/`. This is a web-published note. Please cite with the date shown on this paper and the aforementioned URL.

Contents

1 Basic structure the technology

{− *This is the basic module explaining the component structure and inter−language and technology dependencies for an Object/XML mapping technology.* −}

megamodel *capabilities/Object_XML_mapping/technology .*

variable Technology *OxMapper . −− the name for the technology*
variable Library *OxLibrary . −− the assumed support library*
variable Language *XmlSchemas . −− the targeted XML type system, e.g., XSD*
variable Language *OoLanguage . −− the targeted OO language*
variable Language *MappableOo . −− the OO subset used in mapping*
Language *XML . −− XML as serialization format*
variable Language *ObjectGraphs . −− object graphs in memory*

OxLibrary ***partOf*** *OxMapper .*
OxMapper ***dependsOn*** *XmlSchemas .*
OxMapper ***dependsOn*** *OoLanguage .*
MappableOo ***subsetOf*** *OoLanguage .*
MappableOo ***partOf*** *OxMapper .*
OxMapper ***dependsOn*** *XML .*
OxMapper ***dependsOn*** *ObjectGraphs .*

2 Type-level basics

{− *This module covers the essential type−level aspects of Object/XML mapping. In particular, the types in the spaces XMLware and the objectware are pointed as they correspond to each other. We also make explicit the existence of a problem−specific language underlying the involved type−level artifacts. We think of this language as being abstract in the sense of not fixing it here as being a set of XML trees or object graphs.* −}

megamodel *capabilities/Object_XML_mapping/types .*

include *capabilities/Object_XML_mapping/technology .*

variable File+ *xmlTypes* ***elementOf*** *XmlSchemas .*
variable File+ *ooTypes* ***elementOf*** *MappableOo .*
local Language *problemLanguage .*

xmlTypes ***correspondsTo*** *ooTypes .*
xmlTypes ***definitionOf*** *problemLanguage .*
ooTypes ***definitionOf*** *problemLanguage .*

3 De-serialization

{− *This module describes the value−level aspect of specifically de−serialization. (There is a related module for specifically serialization. At the heart of this description is a function deserialize, which is somehow derived by the mapper from the XML types (or the corresponding OO types) such that deserialization may map an XML document to an object graph with the suitable correspondence and conformance relationships.* −}

megamodel *capabilities/Object_XML_mapping/deserialization .*

include *capabilities/Object_XML_mapping/values .*

local Function *deserialize : XML → ObjectGraphs .*
variable File *xmlInputDoc* ***elementOf*** *XML .*
variable ObjectGraph *intialObj* ***elementOf*** *ObjectGraphs .*

deserialize ***dependsOn*** *OxMapper .*
deserialize ***dependsOn*** *xmlTypes .*
xmlInputDoc ***conformsTo*** *xmlTypes .*
finalObj ***conformsTo*** *ooTypes .*
finalObj ***correspondsTo*** *xmlInputDoc .*
deserialize(xmlInputDoc) ↦ *intialObj .*

4 Serialization

{− *This module describes the value−level aspect of specifically serialization. (There is a related module for specifically de−serialization. At the heart of this description is a function serialize, which is somehow derived by the mapper from the XML types (or the corresponding OO types) such that serialization may map an object graph to an XML document with the suitable correspondence and conformance relationships.* −}

megamodel *capabilities/Object_XML_mapping/serialization .*

include *capabilities/Object_XML_mapping/values .*

local Function *serialize : ObjectGraphs* → *XML .*
variable File *xmlOutputDoc* ***elementOf*** *XML .*
variable ObjectGraph *finalObj* ***elementOf*** *ObjectGraphs .*

serialize ***dependsOn*** *OxMapper .*
serialize ***dependsOn*** *xmlTypes .*
xmlOutputDoc ***conformsTo*** *xmlTypes .*
finalObj ***conformsTo*** *ooTypes .*
finalObj ***correspondsTo*** *xmlOutputDoc .*
serialize(finalObj) ↦ *xmlOutputDoc .*

5 Code generation

{− *Arguably, the typical Object/XML mapping technology comes with a code generation component to derive OO types from XML types. Conceptually, though, this component is optional, which is why it is only introduced now.* −}

megamodel *capabilities/Object_XML_mapping/generation .*

include *capabilities/Object_XML_mapping/types .*

local Function *generator : XmlSchemas → MappableOo .*

generator ***partOf*** *OxMapper.*
generator(xmlTypes) ↦ ooTypes .

6 Annotation

{− *Object/XML mapping may be customized by some annotation mechanism such that schema −derived or authored OO types and their members are annotated with hints regarding de −/serialization.* −}

megamodel *capabilities/Object_XML_mapping/annotation .*

include *capabilities/Object_XML_mapping/serialization .*
include *capabilities/Object_XML_mapping/deserialization .*

variable Language *Annotation* ***partOf*** *OoLanguage .*
variable Language *OxAnnotation* ***subsetOf*** *Annotation .*
variable Fragment+ *anno* ***partOf*** *ooTypes .*

OxAnnotation ***partOf*** *OxMapper .*
deserialize ***dependsOn*** *anno .*
serialize ***dependsOn*** *anno .*

7 Configuration

{− *Configuration of Object/XML mapping may be achieved by designated configuration files. An additional or alternative form of configuration may be achieved by annotations, as discussed elsewhere.* −}

megamodel *capabilities/Object_XML_mapping/configuration .*

include *capabilities/Object_XML_mapping/values .*

variable Language *OxConfiguration .*
variable File *config* ***elementOf*** *OxConfiguration .*

OxConfiguration ***partOf*** *OxMapper .*
deserialize ***dependsOn*** *config .*
serialize ***dependsOn*** *config .*

8 Reading OO functionality

{− *De−serialization was already discussed in a designated module, but the derivation of the corresponding function for de−serialization was not yet explained. To this end, we need to assume actual program code which makes use of appropriate library functionality to issue de−serialization. (There is a similar module for serialization.)* −}

megamodel *capabilities/Object_XML_mapping/read .*

include *capabilities/Object_XML_mapping/deserialization .*

variable File *problemProgram* ***elementOf*** *OoLanguage .*
variable Fragment *deserialization* ***partOf*** *problemProgram .*

problemProgram ***dependsOn*** *ooTypes .*
deserialization ***realizationOf*** *deserialize .*
deserialization ***dependsOn*** *OxLibrary .*

9 Writing OO functionality

{− *Serialization was already discussed in a designated module, but the derivation of the corresponding function for serialization was not yet explained. To this end, we need to assume actual program code which makes use of appropriate library functionality to issue serialization. (There is a similar module for de−serialization.)* −}

megamodel *capabilities/Object_XML_mapping/write* .

include *capabilities/Object_XML_mapping/serialization* .

variable File *problemProgram* ***elementOf*** *OoLanguage* .
variable Fragment *serialization* ***partOf*** *problemProgram* .

problemProgram ***dependsOn*** *ooTypes* .
serialization ***realizationOf*** *serialize* .
serialization ***dependsOn*** *OxLibrary* .