TECHNISCHE UNIVERSITÄT DRESDEN

Faculty of Computer Science Institute of Software and Multimedia Technology, Software Technology Group

SS2018 – Component-based Software Engineering

Composition Systems and Metamodeling

Professor: Prof. Dr. Uwe Aßmann
Tutor: Dr.-Ing. Thomas Kühn

Task 1 Composition Systems Basics

Composition systems simplify the development of large systems by focusing on loose coupling, separation of concerns, reuse, reducibility and standardization. This task repeats the terminology and the fundamental concepts of composition systems.

a) Clemens Szyperski provided one of the well-established definitions of a component [2]. How did he define a component? Try to summarize the key features of a component in your own words!

Solution: *"A software component is a unit of composition with contractually specified interfaces and explicit context dependencies only. A software component can be deployed independently and is subject to third-party composition."* – [2]

- Unit of composition (can be composed with other components)
- Contractually specified interfaces (explicit interface descriptions that act as a contract)
- Explicit context dependencies only (all dependencies are explicitly described)
- Can be deployed independently (components do not rely on concrete other components)
- Are subject to third-party composition (Can be reused by others)

b) What are the three elements of composition systems? Try to explaint each part in your own words!

Solution:
Component Model: A component model describes the shape of components. It describes what components are, how they are structured, what features they have and how their ports look like.
Composition Technique: The composition technique describes how components can be composed and decomposed.
Composition Language: The composition language is a language for defining composition programs.

c) Is the UNIX Pipes and Filters approach a composition system? Explain the three parts!

Solution: *Yes.* The components are filters with 3 standardized untyped ports (`stdin`, `stdout`, `stderr`). The connectors are pipes `|`. The composition technique is to compose filters by connecting the `stdout` or `stderr` port with the `stdin` port of another filter. The composition language is a shell script.

d) Can LEGO(TM) be considered as composition system?[1] Explain your conclusion.

Solution: *Yes.* The components are LEGO(TM) bricks with provided (the studs on top) and required ports (the holes on the bottom). They can be composed by plugging a stud of one brick in the holes of other bricks. The way LEGO(TM) describes its instructions (visually) can be considered a composition language.

[1] http://www.lego.com/en-us/default.aspx

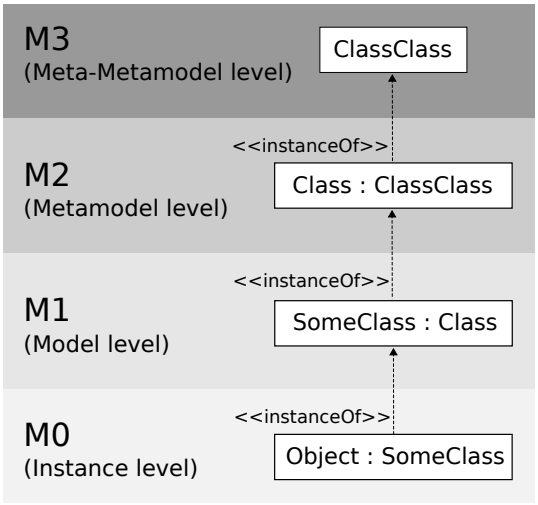

Figure 1: Meta-Pyramid of MOF

Task 2 Metamodeling Basics

Metamodeling is one central discipline in today's software engineering landscape. Especially for safety-critical systems, models are used to formally describe the structure and behavior of systems. Thus, certain properties can be proven. But not only for safety-critical systems, models are used. For example, one important engineering tool today are *Domain Specific Languages* (DSLs). This task repeats the basic terminology and concepts of metamodeling.

a) Explain the terms *Model*, *Metamodel* and *Metametamodel*. What are the relations across those elements?

Solution:
Model: A model is an abstraction of some Original. Usually, models reflect a part of the real or virtual world, with a certain degree of abstraction.
Metamodel: A metamodel describes types of model elements and their relations.
Metametamodel: A metametamodel describes the types of metamodels and their relations.

b) The *Meta Object Facility* (MOF)[2] standard of the OMG emphasizes 4 layers of metamodeling (M3 - M0). In the lecture, we called this Meta-Pyramid. How are the terms of task a) aligned with those layers?

Solution: As shown in Figure 1
M3: Metametamodels
M2: Metamodels
M1: Models
M0: Instances

[2] http://www.omg.org/mof/

Listing 1: Example instance of the DSL.

```
customer John Doe {
    date of birth : 01.01.1980
    sex : male
    primary address : Sample Street 1 , 12345 Samplecity
}
```

Listing 2: Example language specification for the customer DSL.

```
Customer := 'customer' name = String '{'
  'data of birth' ':' dateOfBirth = DateString
  'sex' ':' sex = Sex
  'primary address' ':' primaryAddress = String
'}'
Sex := 'male' | 'female'
```

c) Why is there no 5^{th} layer M4?

Solution: The metametamodel can be used to describe itself. Therefore, a layer M4 is not needed.

d) What is a *Domain-Specific-Language* (DSL)? Give an example.

Solution: A DSL is a language in Software Engineering specifically designed for a certain domain. A DSL uses concepts and keywords from this domain, and thus, can be used by domain experts which are non-programmers.

Listing 1 illustrates a language for describing customers of a company. The corresponding grammar is specified in Listing 2.

e) How can your example be aligned with the Meta-Pyramid?

Solution:
M2: Language specification (and language metamodel).
M1: Example instance.

f) Explain the terms *reflection*, *introspection* and *meta-object protocol* (MOP).

Solution:
Reflection is the ability of a model to reason about and change its own metamodel.
Introspection is Read-Only reflection.
MOP is the implementation of a language semantics, using the language itself.

g) What happens when the MOP is changed?

Solution: The semantics of the language is changed.

Task 3 Metamodeling in Practice

We are going to design a composition system for classical components (according to Clements Szyperski [2]) utilizing the *Eclipse Modeling Framework* (EMF) [1] and `xText` language workbench. Components have a name, a set of attributes and a set of provided and required interfaces. Attributes have a name and a types (represented as a String). An interfaces has a name and a set of methods, whereas each method has a name, a return type (represented as a String) and a set of parameters. Like attributes, a parameter has a name and a type. Based on this model, compositions can be described as an instantiation and composition of components by connecting required and provided ports of the same type.

a) Download the latest version of the Eclipse Modeling Tools[3] and install the latest version of `xText`.[4]
b) Create an EMF-Metamodel (Ecore Model) for the composition system.
c) Create a DSL using `xText` that lets you design components and compose them in a composition language
d) Optionally, use the `xText` validation engine to validate rules that must be enforced in the static semantics of the language.[5]

Solution: The solution can be downloaded from the website.

References

[1] Dave Steinberg, Frank Budinsky, Ed Merks, and Marcelo Paternostro. *EMF: eclipse modeling framework*. Pearson Education, 2008.

[2] Clemens Szyperski, Jan Bosch, and Wolfgang Weck. Component-oriented programming. In *European Conference on Object-Oriented Programming*, pages 184–192. Springer, 1999.

[3] https://eclipse.org/downloads/packages/eclipse-modeling-tools/oxygen3a
[4] https://www.eclipse.org/Xtext/download.html
[5] Look at the `xText` help section for further information. 5