Design Patterns and Frameworks	**Exercise Sheet No. 9**
Dipl.-Inf. Steffen Zschaler	Software Engineering Group
INF 2097	Institute for Software and Multimedia Technology
`http://st.inf.tu-dresden.de/teaching/dpf`	Department of Computer Science
	Technische Universität Dresden
	01062 Dresden

Formal Models of Design Patterns II

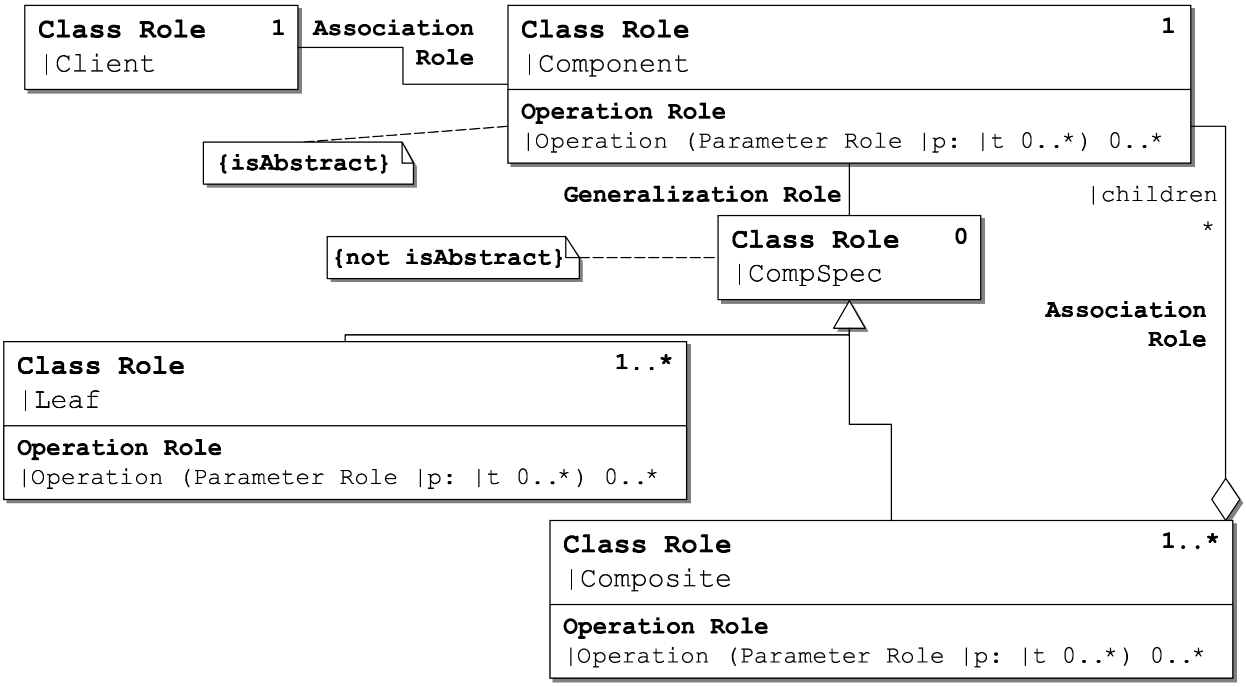
Task 1: Composite in RBML

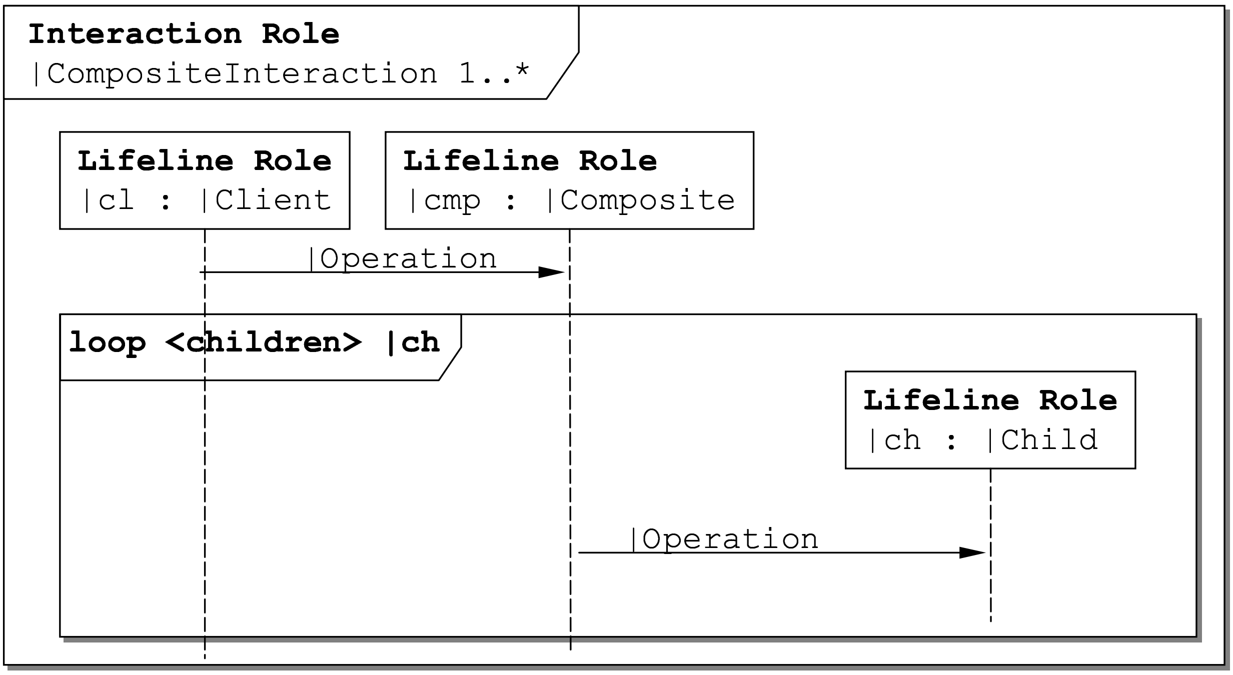
Read and understand [1]. This paper presents RBML, a UML-like notation for representing design patterns. The formal backing of this notation (representing the design patterns' role models as extension of the meta-model) allows for formal treatment of design patterns in actual models.

1a) **Task:**

Use RBML to represent the Composite design pattern.

Solution:





1b) **Task:**

Go back to task sheet 3 on extensibility patterns and look at your solution for task 1a). Use the RBML techniques presented in [1] to show that this is indeed a realisation of COMPOSITE.

Solution: *Unfortunately, solution hint is not available.*

Bibliography

1. Robert France, Dae-Kyoo Kim, Sudipto Ghosh, Eungee Song, *A UML-Based Pattern Specification Technique.* IEEE Transactions on Software Engineering, Vol 30, number 3, pp 193-206, March 2004. *This paper is available online at the IEEE digital library by visiting http://ieeexplore.ieee.org/Xplore/DynWel.jsp and searching for it by title. You should have access to the digital library from any computer in the domain of the Computer Science Department.*

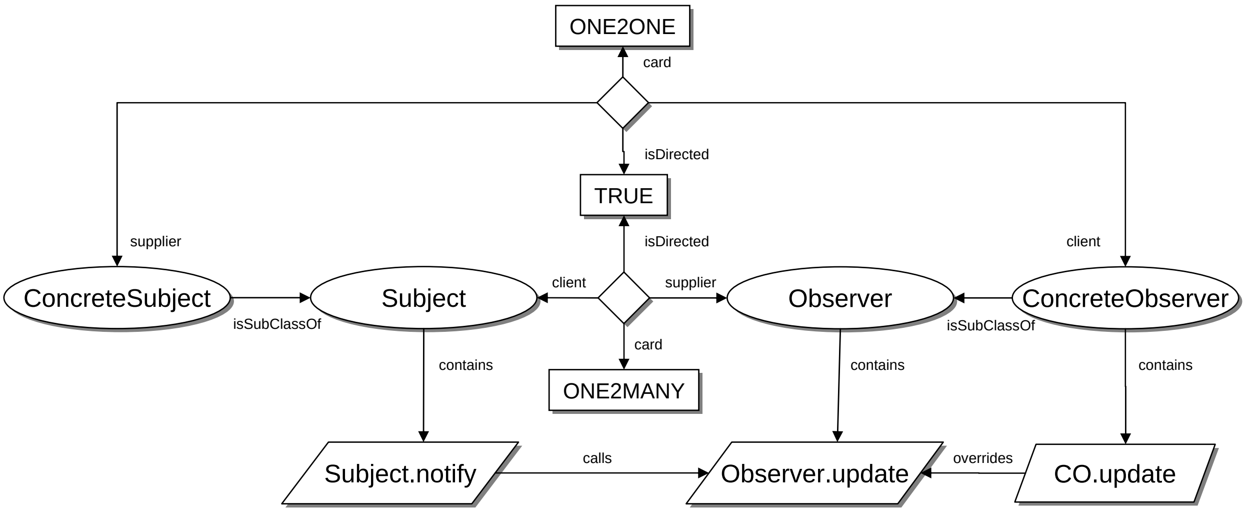
Task 2: OWL Observant

Read and understand [1]. This presents an approach that uses Semantic Web technology (in particular ontologies) to model design patterns. An ontology can be viewed (grossly simplifying) as a special kind of class diagram modelling concepts and their relations. An ontology, thus, provides vocabulary allowing to talk about a specific domain.

2a) **Task:**

Use the technology from [1] to model the OBSERVER design pattern.

Solution: The following is a graphical representation of the resulting ontology. To simplify matters we have left out all properties relating to concepts from ODOL (as specified in `wop.rdf`). Instead, we have used ellipses to denote class templates, diamonds to denote association templates, parallelograms for method templates and rectangles for data-type values.



Bibliography

1. Jens Dietrich and Chris Elgar. *A Formal Description of Design Patterns Using OWL.* In Proc. 2005 Australian Software Engineering Conference (ASWEC'05), IEEE Press, 2005.

Task 3: ρ-calculus and Template Class

Read and understand [1]. This presents a formal approach to writing down design patterns, which allows reasoning about the patterns. To understand it properly, you will also need to read, and understand the intuition of, [2], in particular Chapter 5, and Sections 6.1, 6.2, 6.6, 7.1, 7.2, 8.1, 8.5. Don't worry, though, it's fun reading!

3a) **Task:**

Use the ρ-calculus to model the TEMPLATE HOOK meta-pattern. TEMPLATE HOOK essentially proclaims that there is a template method which invokes (i.e., depends on) a hook method. The two methods may be in the same class or they may not. TEMPLATE HOOK thus is essentially a role model.

Solution: We use the following very straight-forward specification:

$$\frac{\begin{array}{c} Template <: [opT : T] \\ Hook <: [opA : A] \\ t : Template \\ h : Hook \\ t.opT \ll_m h.opA \end{array}}{TemplateHook(t, h, opT, opA, Template, Hook)}$$

3b) **Task:**

Now use the ρ-calculus to model the TEMPLATE CLASS design pattern. In this pattern there are two distinct classes, one for a template method and one for a hook method.

Solution:

$$\frac{\begin{array}{c} Template <: [opT : T] \\ Hook <: [opA : A] \\ \neg Hook <: Template \\ \neg Template <: Hook \\ t : Template \\ h : Hook \\ t.opT \ll_m h.opA \end{array}}{TemplateClass(t, h, opT, opA)}$$

3c) **Task:**

Can you represent Template Class using Template Hook? What does this mean?

Solution:

$$\frac{\begin{array}{c} TemplateHook(t, h, opT, opA, Template, Hook) \\ \neg Hook <: Template \\ \neg Template <: Hook \end{array}}{TemplateClass(t, h, opT, opA)}$$

This formula means that the Template Class pattern is a specialisation of the Template Hook pattern, where template and hook have been allocated to distinct classes.

Bibliography

1. Jason McC. Smith, and David Stotts. *Elemental Design Patterns – A Link Between Architecture and Object Semantics.* Technical Report TR02-011, March 2002, Dept. of Computer Science, Univ. of North Carolina, Chapel Hill.
2. Martin Abadi, and Luca Cardelli. *A Theory of Objects.* Monographs in Computer Science, Springer, 1996. *There are 3 exemplars in the library.*

Task 4: Discussion: Formal Representation of Design Patterns

From your experience with the pattern formalisations looked at so far, what are the benefits and drawbacks of attempts at formalising design patterns?

Solution: The main points to be discussed here are: Ambiguity, Relations between Patterns, Automation and Tool Support, Difficulty, Lack of Variation in formally specified patterns, . . .

An interesting discussion occurs in http://www.eden-study.org/precise_and_formal/faq.htm .