TECHNISCHE UNIVERSITÄT DRESDEN

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

SS2019 – Component-based Software Engineering

Designing Component-Based Systems

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

Task 1 Components

Components are the central elements of component-based systems. This tasks reiterates the terminology and the fundamental concept of components.

a) Malcolm Douglas McIlroy was talking about *Components Off the Shelf* (COTS), already in 1969 [2]. What is the main idea of *COTS* and why would this be beneficial?

Solution: The idea of *Components Off the Shelf* in software engineering is based on the standardized components in production. When the components comply with a common standard wrt. to their shape, function and interfaces, then no individualization is necessary. Furthermore, different variants of different vendors are exchangeable. That way, software systems can be developed by composing existing components from various vendors.

b) What is a facet and what is a facet classification? Give an example.

Solution: Facets are orthogonal (independent) dimensions of a model. Every facet represents an independent classification of an entity. Fig. 1 illustrates the facet classification of Lego(TM) products from shop.[1]

c) What is a component repository, a component market and a component trader?

Solution:
Component Repository is an environment where components can be distributed and obtained.
Component Market is a potentially commercial public component repository with a component trader.
Component Trader is an engine for describing, finding and obtaining components. The description/search can be based on various properties (functionality, contracts, protocols etc.).

[1] https://shop.lego.com

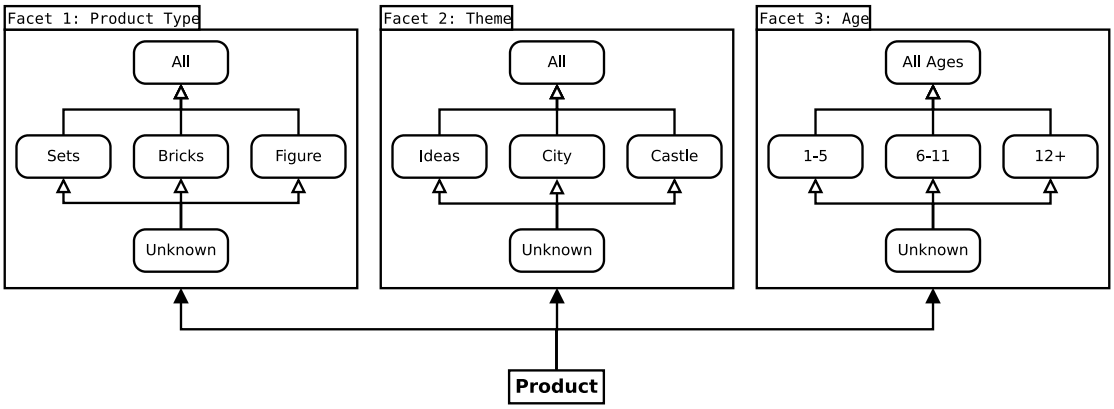

Figure 1: Example facet classification of Lego(TM) products.

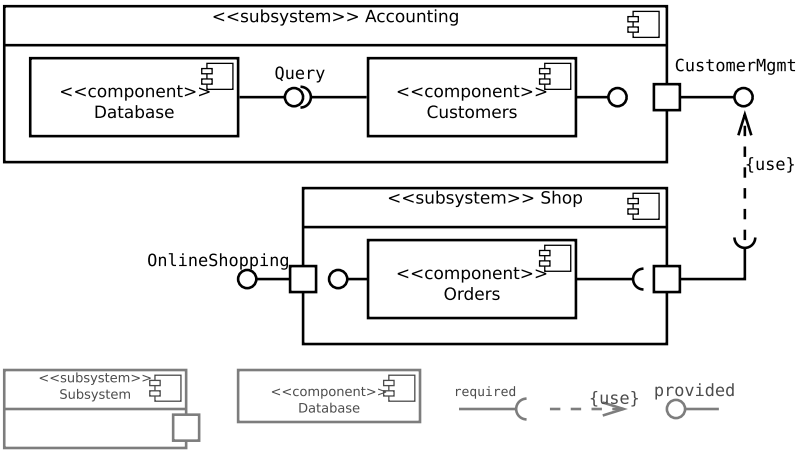

Figure 2: Example notation of UML component diagrams.

d) What are the key modeling elements of UML Components? What is the graphical notation? Provide an example.

Solution: UML Components have a name, provided and required ports (interfaces) and may have more internal components (hierarchical). The graphical notation and an example is illustrated in Fig. 2.

e) Why should big component-based systems be structured hierarchically?

Solution: From a construction perspective, *devide-and-conquer* is a well established approach to deal with complex problems. From an analytical perspective, hierarchical decomposition helps to understand abstracted, less complex models and iteratively get more detailed information.

Task 2 Cheesman/Daniels Process

The **Cheesman/Daniels process** [1] helps to identify UML-Components, by stepwise refinement, starting with a requirements specifications.

a) What is a domain model and why is it necessary?

Solution: The domain model (business concept model) describes elements of the (business) domain, their properties and their relationships. It is necessary to capture all entities of the domain, model their relationships to make sure, that the domain and its terminology is understood. It is used as a communication artifact with the stakeholders.

b) What is a business component, according to Cheesman and Daniels?

Solution: A business component consists of business objects (atomic components) and other business components (hierarchical).

c) What should be visible from a component? How is that related to the *Information Hiding Principle* [3]?

Solution: Only implementation independent information should be visible (interfaces). Implementations may change, interfaces are more stable. Furthmore, components with the same interfaces can be exchanged without changing any code.

d) How is the *Cheesman/Daniels Process* related to technologies, such as CORBA, EJB, and Android?

Solution: The Cheesman/Daniels Process is a process for developoing (large) component based systems. CORBA, EJB, and Android are composition systems that can be used as an implementation technology.

Task 3 Factory Automation

This task will be used as a basis for the following exercises. You can either solve the task alone or form groups of up to five students.

Note: The solutions have to presented in the next exercise.

You are supposed to develop a component-based management system for factory automation. The company you are developing the system for, provides customizable 3D-printing services. Customers must first have registered and must be approved. Afterwards, they can upload 3D-printing jobs. The company provides multiple different types of 3D printers and different types of finishing procedures. Depending on the concrete order of an individual customer, a process is generated (e.g., print, finish, paint). In such a process, multiple different machines (e.g., 3D printers, painting machines) are involved. The company also uses mobile robotic platforms to move parts between the individual machines. Furthermore, the machines are equipped with robot arms to move the parts between the machine and the mobile robot. The robots act autonomously, but receive their tasks from a central process management system. After the production process finished, the product is transported to a central store and shipped to the customer. The customer also receives an invoice. Invoices are managed by a central invoicing system.

a) Design the application following the Cheesman/Daniels process. Create the required models, i.e., *Domain Model*, *Use-Case* model, *Business Type Model*, *Business Object Interface Model*, *Component Specification*, and *Component Architecture* (cf. lecture).
b) Create **one** PDF Document containing your models in a readable format.

Solution: A possible solution can be downloaded from the website.

References

[1] John Cheesman and John Daniels. *UML Components: a Simple Process for Specifying Component-Based Software.* Addison-Wesley, Longman, Amsterdam, 2001. ISBN 201-70851-5.

[2] M Douglas McIlroy. Mass-produced software components. In J. M. Buxton, Peter Naur, and Brian Randell, editors, *Proceedings of the 1st International Conference on Software Engineering, Garmisch Pattenkirchen, Germany*, pages 88–98. NATO Science Committee, 1968.

[3] David Lorge Parnas, Paul C Clements, and David M Weiss. The modular structure of complex systems. In *Proceedings of the 7th international conference on Software engineering*, pages 408–417. IEEE Press, 1984.