TECHNISCHE UNIVERSITÄT DRESDEN

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

SS2018 – Component-based Software Engineering

Implementing Component-Based Systems–Part I

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

Task 1 Transparency Problems

A transparency problem describes software concerns that should be transparent (invisible, hidden) when you write or deploy a component. This task repeats the different kinds of transparency problems.

a) What can be subject of secrets wrt. transparency problems of component-based systems?

Solution:
Content secrets are secrets that deal with the concrete implementation of a component. According to Szypersky, all context-dependencies should be specified in the interface.
Connection secrets are concerned with the connection and communication of components. They should be hidden in connectors.

b) What aspects of transparancy do you know? How are they aligned with the secret subjects?

Solution:

- *Content secrets*
  - **Language transparency**: interoperability of components using different programming languages
  - **Persistency transparency**: Hide whether server has persistent memory
  - **Lifetime transparency**: Hide whether server has to be started
- *Connection secrets*
  - **Location transparency**: Hiding distribution of programs
  - **Naming transparency**: naming of services Hiding, how a service is called
  - **Transactional transparency**: Hide whether server is embedded in parallel writes

c) What is language transparency and how can it be achieved?

Solution: Interoperability of components which is independent of the concrete programming language. An Example would be SOAP-Web Services (they use a standardized XML-based exchange protocol for communication).

d) Why is location transparency important? Give an example.

Solution: Location Transparency is the interoperability of components independent of the concrete location of the component (device running the component). When a component changes its location, the implementation of dependent components should not change. Many modern systems are distributed component-based systems. Especially for applications for Internet-of-Things (IoT), a multitude of heterogeneous, distributed devices should be integrated dynamically to form a context-dependent ad-hoc system of systems. In those systems, location transparency is one of the most important concepts.

Task 2 Open Services Gateway initiative (OSGi)

Open Services Gateway initiative (`OSGi`)[1] is a hardware-independent composition system for designing and executing modularized, component-based systems [1].

a) Is `OSGi` a composition system? Describe the component model, composition technique and composition language.

Solution: Partially, it encompasses most aspects required for composition systems.
Component Model: Bundles represent components that encapsulate a set of classes and packages, which provide and require services. Required services can specify contracts on the version of provided services. Bundles can be composed to micro service Bundles carry additional meta data.
Composition Technique:Automatically binding provided services/packages and required services/packages by name. Finding bundles in repository by name and meta data.
Composition Language: There is no actual composition language. Dependencies are specified declaratively.

b) Compare `OSGi` components to the definition of components by Szyperski et al. [2].

Solution:

- Unit of composition: **bundle**
- Specified interfaces: **provided and required services**
- Explicit context dependencies: **Environment and requirements** specified in manifest
- Independently deployed: **bundles with own life cycle**
- Third-party composition: **OSGi** is standardized and available for many platforms

c) Which transparency problems does `OSGi` address?

Solution:

- *Lifetime transparency*: bundle activation
- *Naming transparency* bundles are found by service/package name

[1] https://www.osgi.org/what-is-osgi

Task 3 Implementation of the Factory Automation application – Part 1

In the last exercise you designed a simple management application for factory automation for a 3D-printing service. In this task you will start to implement parts of your design in OSGi. OSGi is a mature and powerful composition system. We will use OSGi to implement parts of the factory automation use case, described in exercise 2. To get familiar with OSGi, you will install the required tools and work yourself through the listed tutorials. In this first part you are going to implement 3 components, such as the customer-, product-, and order-management. All components offer interfaces to add, remove and list customers, products and orders. You do not have to implement a front-end for your components. However, you must test their individual functionality.

Note: You can work in groups of up to five students.

a) Read and reconstruct the following tutorials:

- http://www.vogella.com/tutorials/OSGi/article.html
- http://www.vogella.com/tutorials/OSGiServices/article.html

b) Implement the customer manager component.
c) Implement the stock manager component.
d) Implement the order manager component.
e) Test your components.
f) Prepare a short presentation and demo (maximum 5 minutes)!

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

References

[1] Richard Hall, Karl Pauls, Stuart McCulloch, and David Savage. *OSGi in action: Creating modular applications in Java.* Manning Publications Co., 2011.

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