Representing Formal Component Models in OSGi

Marco Müller, Moritz Balz, Michael Goedicke

Specification of Software Systems

Institute for Computer Science and Business Inforamtion Systems

University of Duisburg-Essen, Essen, Germany

{marco.mueller, moritz.balz, michael.goedicke}@s3.uni-due.de

Part of the work reported herein was funded by the German Federal Ministry of Education and Research under the grant number 03FPB00320.

Motivation

- Formally founded component models
  - Verification of characteristics, specification of interaction, simulation

- Current industrially used programming languages, platforms and frameworks
  - Module semantics provided by frameworks
  - Lack the expressiveness of formally founded component models

→ **How to change existing module frameworks to achieve the expressiveness of formally founded component models?**

Outline

1. Examine features of formal component models
2. Examine features of an industrially used framework
   - **OSGi Service Platform:**
     - Widely-used module system
     - Based on Java
3. Compare the feature sets and propose changes to OSGi

Component Models

- Component models with different foci

component interaction	Allen and Garlan
message flow	Cox and Song
data abstraction and concurrency	Pi (Cramer et al.)
dynamic architectures	Magee et al., SOFA 2.0 (Bures et al.)
modeling and prediction of quality requriements	KLAPER (Grassi et al.), Palladio (Reussner et al.), ROBOCOP

Desirable Features of Component Models

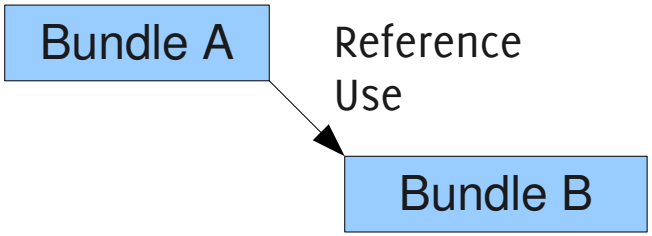
- Composition
  - Composite components are constructed of subcomponents
- Provided Interfaces
  - Defined by a set of operations, callable by the component context
- Dependencies
  - Description of expected functionality
  - Required data types: e.g. in common platform (Darwin) or in component description (Pi)
  - Shared interfaces vs. context independence

Desirable Features of Component Models

- Instantiation
  - Named instances of components
- Interactions
  - Connectors mapping required to provided services
  - Call sequences, concurrency (e.g. Path Expressions in Pi or Protocol in Palladio)
- Assembly
  - Separate stage in development
  - Instantiate and interconnect existing components

OSGi Service Platform – A very brief introduction

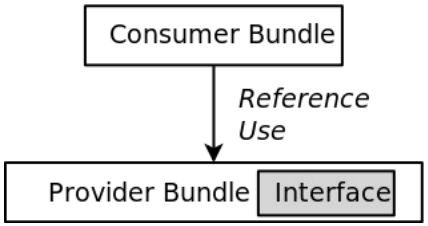
- Bundle: Set of resources (compiled classes and data), usually in an archive – the **module** in OSGi
  - Meta Information (MANIFEST.MF): e.g. ID, name, version
  - Relations to other bundles: provided packages, referenced packages/bundles

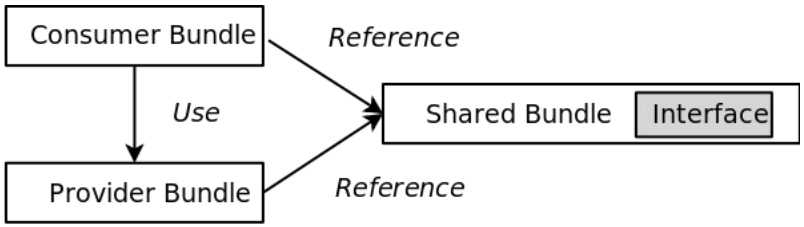


- Service components:
  - Register Java Objects at a platform wide service registry under the name of a Java Interface
  - Requiring bundles get these service objects and use them

Evaluation of the Desired Features in OSGi

- Composition
  - No composition
- Provided Interfaces
  - Provided service components
  - Data types need to be shared by both bundles
- Dependencies
  - Referencing: Bundle name, package name, service components
  - Tight coupling:



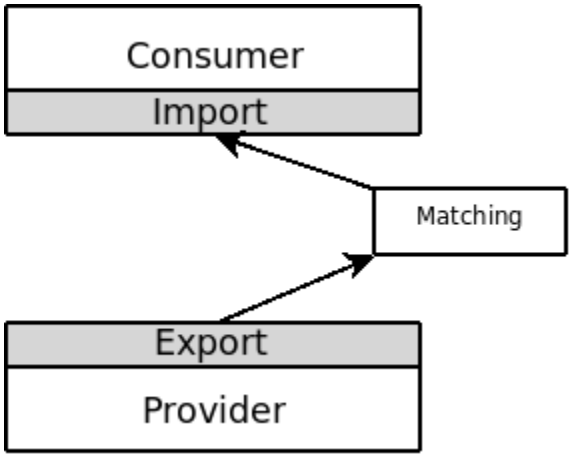


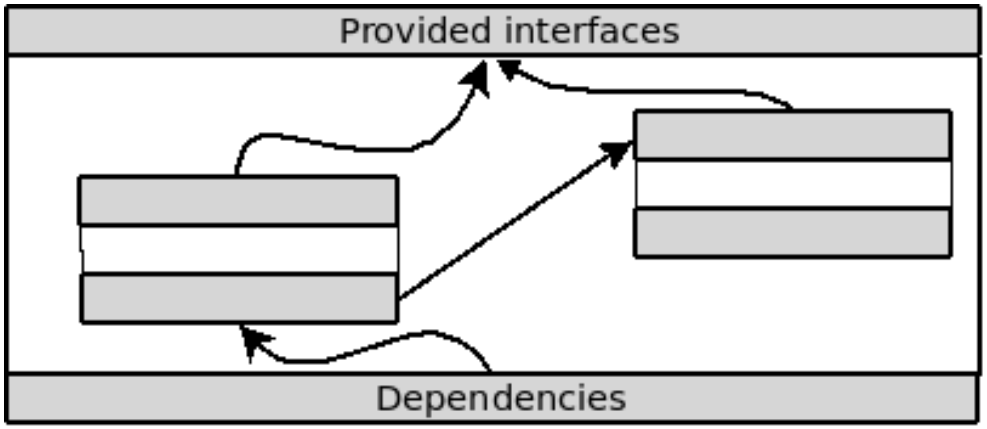
Evaluation of the Desired Features in OSGi

- Instantiation
  - Bundles cannot be instantiated
  - Service component instances registered under different names
- Interactions
  - No constraints
  - Java method calls
- Assembly
  - Bundles automatically assembled at run time
  - Services assembled programatically or descriptively

Proposal for Getting the Desired Features

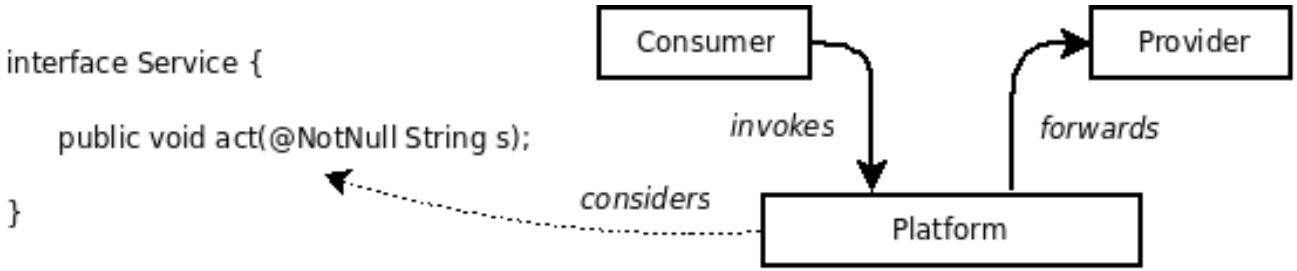
- Dependencies
  - context-independence
  - self-containdness
- Composition
  - hidden subcomponents
  - not externally observable





Proposal for Getting the Desired Features

- Interaction
  - constraints on methods
  - call sequences
  - concurrency constraints



- Assembly
  - verifying interconnection before deployment

Related Work

- Spring DM: Spring Beans represented as OSGi Services
  - Components do not differ from OSGi Service Layer
- JEE Session Beans: Modules in Java Enterprise Edition
  - No context independence (bound to type)
- Beanome: component model for OSGi
  - No context independence (shared interfaces), outdated

Conclusion

- Industrially used component frameworks lack the expressiveness of formally founded component models
- **How to change existing module frameworks to achieve the expressiveness of formally founded component models?**
- Proposal for
  - loose coupling through required and provided interfaces
  - composite components
  - interaction modeling using e.g. path expressions
- Plan to implement the proposal and consider more frameworks and formal component models

Thank you for listening

Questions?