

Checking of the Component Substitutability in OSGi

Jaroslav BAUML

jbauml@students.zcu.cz

Vedoucí práce: Přemysl Brada

Content

- What you can look forward to in the next 15 minutes.
-

- Background

- OSGi – component model
- Subtype-based component comparison

- My diploma thesis

- Problem
- Analysis
- Solution – implementation dependent
 - Apache Felix
 - Knopflerfish

- Future work

Background – OSGi

= Open Services Gateway initiative

- Light-weight **component** & **service** model – Java based
- Component is called **bundle**
- Various dependencies between bundles (import/export)
 - Java Package
 - Bundle
 - Service
- Industrial standard
- Implementations:
 - **Apache Felix**
 - Equinox
 - **Knopflerfish...**
- Practical applications:
 - Eclipse Platform (Equinox)
 - Application servers GlassFish and JOnAS
 - Communicator SIP (Apache Felix)

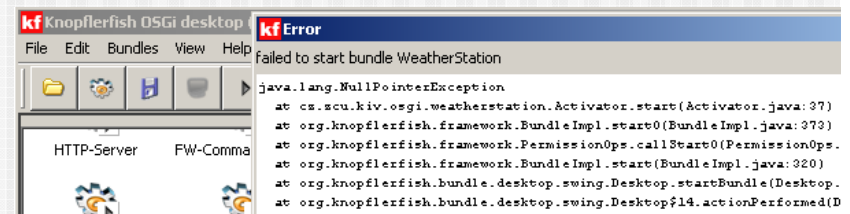
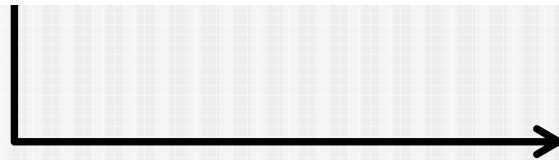


The problem

- When an update is not compatible, component is updated, but program fails.
 - When updating bundle in OSGi framework, no checking of compatibility is executed.
 - What can occur when an update is **incompatible**
 - Right versioning -> Application is **not** resolved and started
 - Wrong versioning -> Application is resolved, but probably **will crash!**

```
Interface Logger (v1)
void log(String msg)
void log(int level, String msg)
int getItemCount()
```

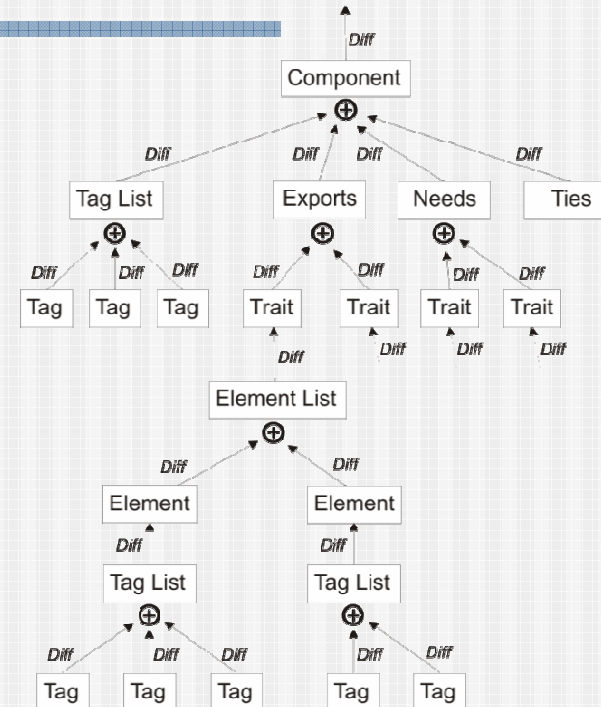
```
Interface Logger (v2)
void log(String msg)
long getItemCount()
```



Solution: Use **subtype based** component comparison and extend update process in OSGi with this comparison

Subtype Based Component Comparison

- Based on ENT meta-model
1. Create representation of components
 2. Comparing of the representations
 - **Input:** ENT representations of two components
 - Relevant parts are compared using **subtyping** rules
 - = The rules for comparing various metatypes
 - **Output:** comparison result

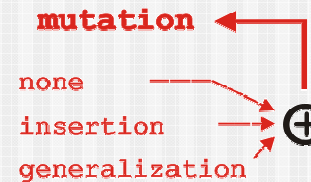


```

Interface Logger (v1)
void log(String msg)
int getItemCount()
    
```

```

Interface Logger (v2)
void log(String msg)
void log(int level, String msg)
long getItemCount()
    
```



Implementation issues

- Let's discuss these issues
-

1. How to extend the update mechanism?
 - No public API in OSGi for extending
2. How to get representation of components?
 - Installed – No public API
 - Updating – Public API exists

? Ad 1.

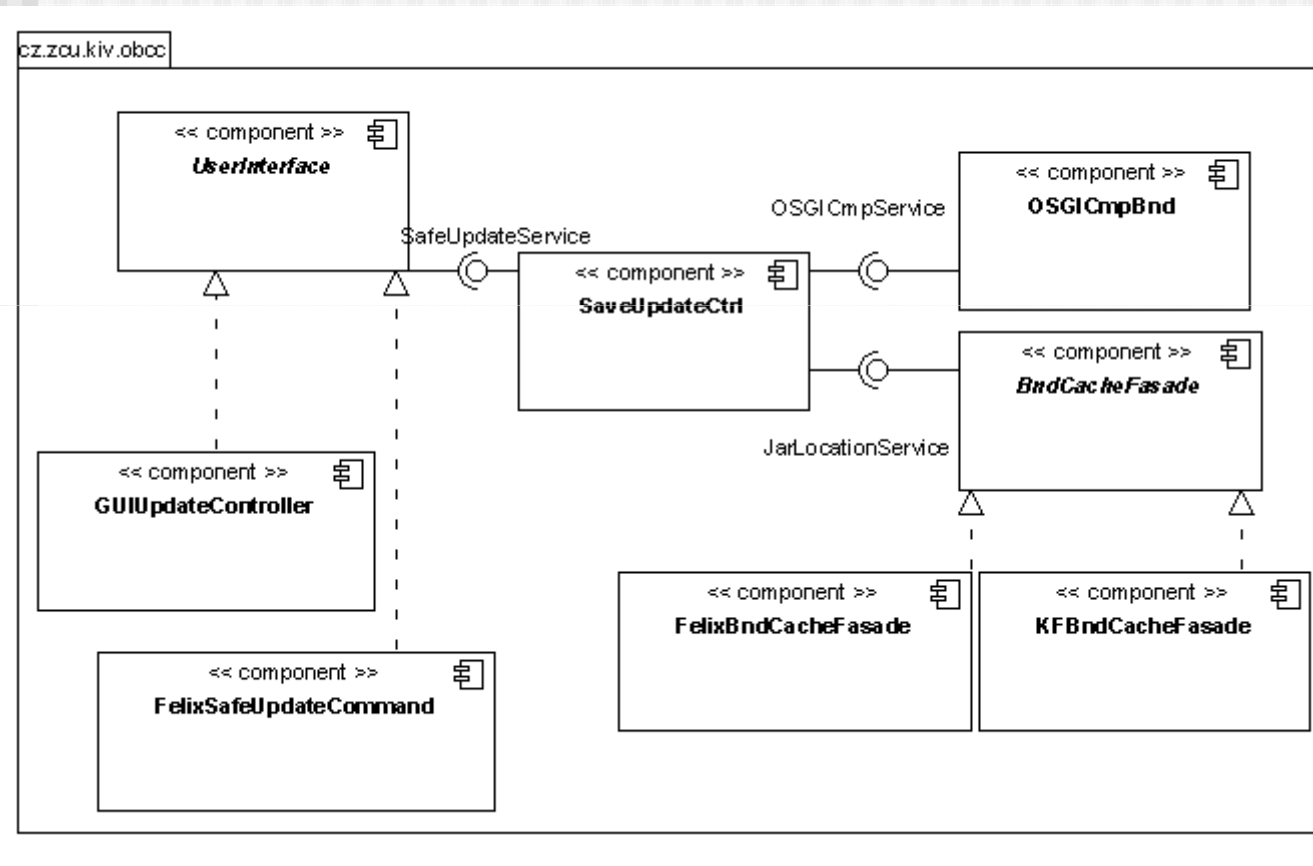
- To use own command for safe update.

? Ad 2.

- To use facade API for selected implementation of OSGi
- To enable portability for other implementations

Architecture (1)

- Component Based Architecture & Services

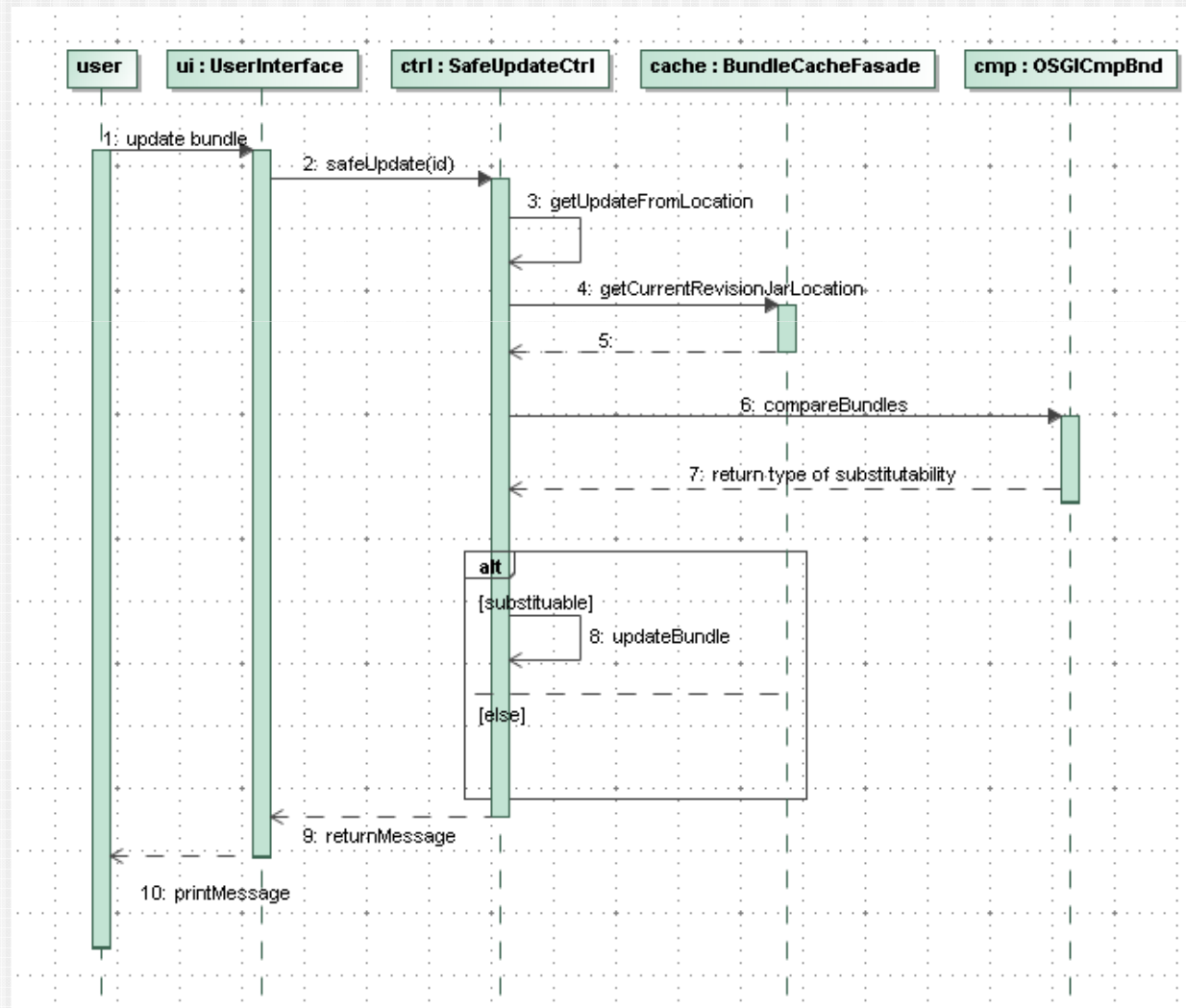


Services:

- OSGICmpService
- JarLocationService
- SafeUpdateService

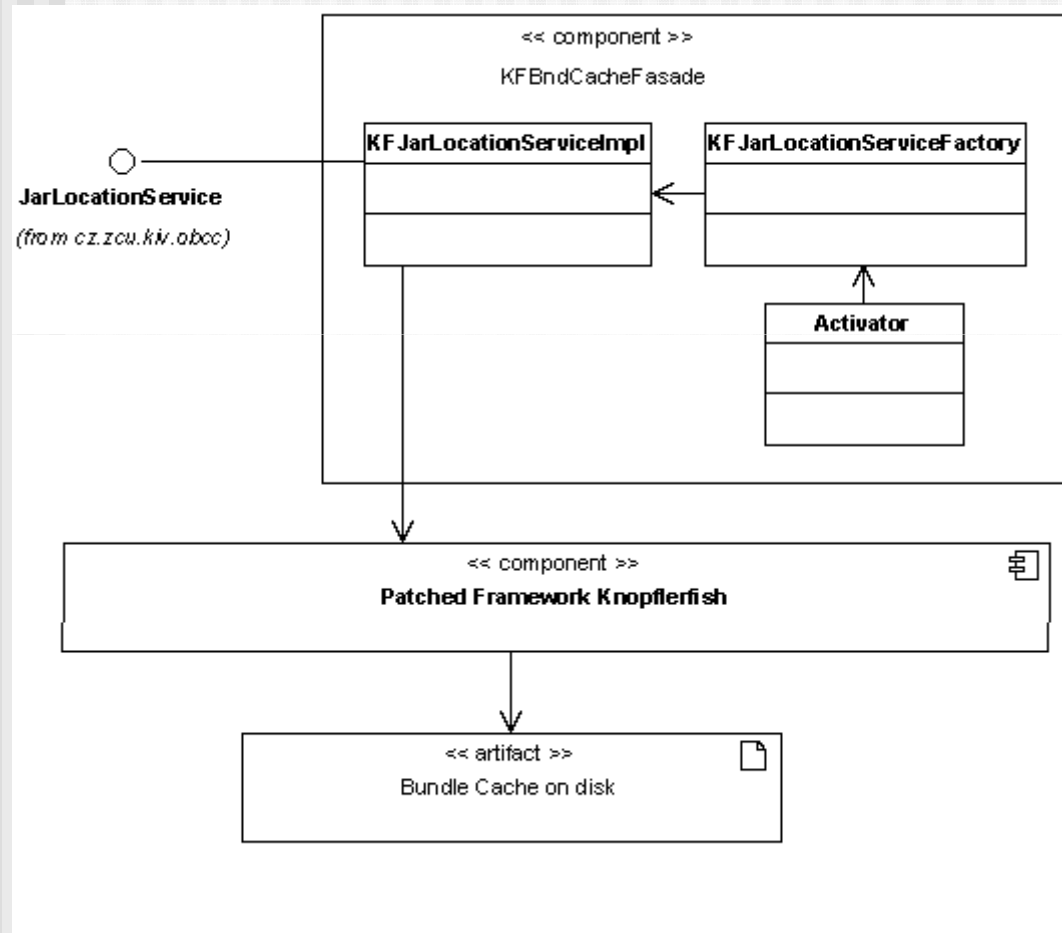
Architecture (2)

- Workflow on the component level



Knopflerfish

- How to implement access to bundle cache



- Bundle cache **not** documented
- **No way** of running framework programmatically
- The only way is to use **internal API**

Apache Felix

- How to implement access to bundle cache

1. Bundle cache on disk is **well** documented
2. **Easy way** of running framework programmatically

Chosen approach:

1. Direct access to bundle cache

Bundle cache is “public”

Easy and applicable for existing installation of Felix

```
~/.felix/  
  example/  
    bundle4/  
      bundle.id  
      bundle.location  
      bundle.startlevel  
      bundle.state  
      data/  
      version0.0/  
        bundle.jar  
        embedded/  
          embedded.jar  
        lib/  
          libfoo.so  
      revision.location
```

Future work

- Implementations for other OSGi frameworks
- The tool is usefull for single component update, but what about set of bundles to update.
 - > Not only subtype substitutability
- Automated tests for this kind of problems

Conclusion

Results:

- Tool for safe update of bundles in OSGi
- Supported implementations:
Knopflerfish v 2, Apache Felix v 1
- Tool is extensible
- Tool tested on representative sets of problematic components

Thank you