

# Realisierung von Service-Varianten und zustandsbehafteten Services auf Basis von OSGi

Timo Kehrer   Sven Wenzel   Maik Schmidt

Universität Siegen, Praktische Informatik  
{kehrer, wenzel, mschmidt}@informatik.uni-siegen.de

OSGi Users'-Forum Treffen 2009  
27.10.2009 @ Eclipse Summit in Ludwigsburg

## Outline

- 1 **OSGi-Einsatzkontext: "SiDiff"**
  - Applikationsdomäne
  - Problemmotivation
- 2 **Warum OSGi?**
  - Architektur
  - Technologische Umsetzung
- 3 **Realisierungs-Probleme**
  - Einführendes Beispiel
  - Zustandsbehaftete Services
  - Service-Varianten
- 4 **Lösungsansätze und technische Umsetzung**
  - Kapselung der OSGi Service-Schicht
  - Zustandsbehaftete Services: "ProvideableService"
  - Service-Varianten: "ConfigurableService"
  - Kombination: "ConfigurableProvideableService"

## Outline

- 1 **OSGi-Einsatzkontext: "SiDiff"**
  - Applikationsdomäne
  - Problemmotivation
- 2 **Warum OSGi?**
  - Architektur
  - Technologische Umsetzung
- 3 **Realisierungs-Probleme**
  - Einführendes Beispiel
  - Zustandsbehaftete Services
  - Service-Varianten
- 4 **Lösungsansätze und technische Umsetzung**
  - Kapselung der OSGi Service-Schicht
  - Zustandsbehaftete Services: "ProvideableService"
  - Service-Varianten: "ConfigurableService"
  - Kombination: "ConfigurableProvideableService"

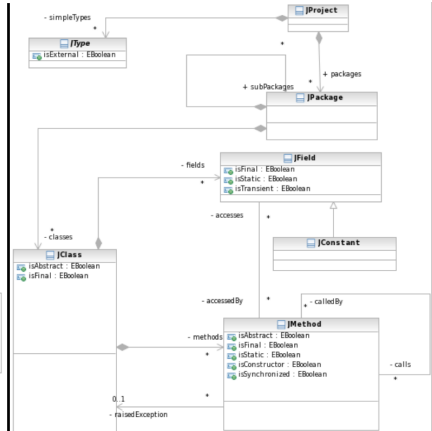
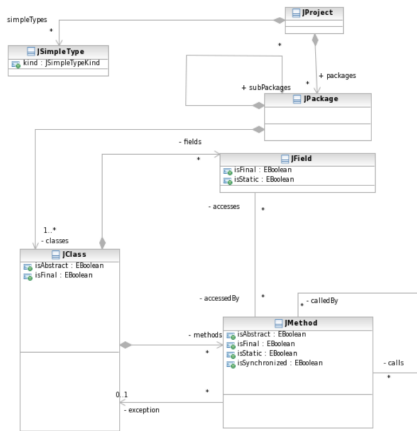
## Versionen und Varianten technischer Dokumente

- Technische Dokumente mit grafischer Notation gewinnen zunehmend an Bedeutung, bspw.
  - CAD-Dokumente
  - UML-Modelle
  - Matlab/Simulink-Modelle
  - etc.
- Komplexe Dokumente werden in der Regel im Team bearbeitet und in Repositories gespeichert
- Dokumente existieren in mehreren Versionen (Revisionen oder Varianten)
- Daraus resultieren eine Reihe praktischer Probleme:
  - Konfigurationsmanagement (Diff/Merge)
  - Analyse von Dokumenthistorien
  - Clone Detection
  - etc.

## Versionen und Varianten technischer Dokumente

- Technische Dokumente mit grafischer Notation gewinnen zunehmend an Bedeutung, bspw.
  - CAD-Dokumente
  - UML-Modelle
  - Matlab/Simulink-Modelle
  - etc.
- Komplexe Dokumente werden in der Regel im Team bearbeitet und in Repositories gespeichert
- Dokumente existieren in mehreren Versionen (Revisionen oder Varianten)
- Daraus resultieren eine Reihe praktischer Probleme:
  - Konfigurationsmanagement (Diff/Merge)
  - Analyse von Dokumenthistorien
  - Clone Detection
  - etc.

# Textuelle Dokumente vs. grafische/graphbasierte Dokumente



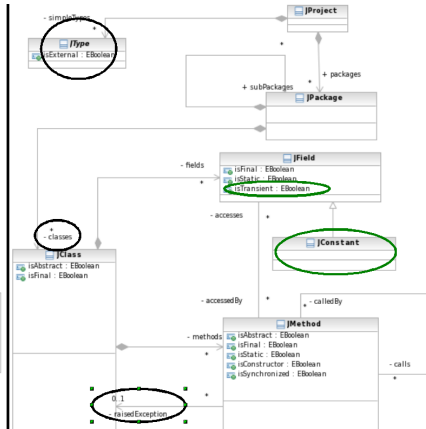
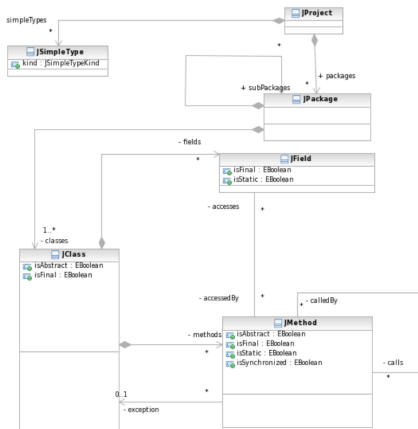
# Differenzen auf Basis der textuellen Repräsentation

<pre> 671 &lt;styles xmi:type="umlnotation:UMLShapeStyle" xmi: 672 &lt;layoutConstraint xmi:type="notation:Bounds" xmi: 673 &lt;/children&gt; 674 &lt;children xmi:id="rCWXYLYUEd6DuZJKbi7Quw" elemen 675 &lt;children xmi:id="rCZasLYUEd6DuZJKbi7Quw" type= 676 &lt;layoutConstraint xmi:type="notation:Size" xmi: 677 &lt;/children&gt; 678 &lt;children xmi:id="rCaBwLYUEd6DuZJKbi7Quw" type= 679 &lt;children xmi:id="rCaBwbyUEd6DuZJKbi7Quw" type= 680 &lt;children xmi:id="rCao0LYUEd6DuZJKbi7Quw" type= 681 &lt;children xmi:id="rCao0byUEd6DuZJKbi7Quw" type= 682 &lt;styles xmi:type="umlnotation:UMLListCompartm 683 &lt;/children&gt; 684 &lt;children xmi:id="rCbP4LYUEd6DuZJKbi7Quw" type= 685 &lt;styles xmi:type="umlnotation:UMLListCompartm 686 &lt;/children&gt; 687 &lt;children xmi:id="rCb28LYUEd6DuZJKbi7Quw" visi 688 &lt;styles xmi:type="umlnotation:UMLListCompartm 689 &lt;/children&gt; 690 &lt;children xmi:id="rCceALYUEd6DuZJKbi7Quw" visi 691 &lt;styles xmi:type="umlnotation:UMLShapeCompart 692 &lt;/children&gt; 693 &lt;styles xmi:type="umlnotation:UMLShapeStyle" xmi: 694 &lt;layoutConstraint xmi:type="notation:Bounds" xmi: 695 &lt;/children&gt; 696 &lt;styles xmi:type="umlnotation:UMLDiagramStyle" xmi: 697 &lt;element xsi:nil="true"/&gt; 698 &lt;edges xmi:id="cqA6cGVhEd6yV7V4a3bq1A" element=" 699 &lt;children xmi:id="cqA6c2VhEd6yV7V4a3bq1A" type= 700 &lt;children xmi:id="cqA6dVhEd6yV7V4a3bq1A" ty </pre>	<pre> 687 &lt;children xmi:id="TG1JEGYREd6-LLTu1P_2sg" \ 688 &lt;styles xmi:type="umlnotation:UMLListComp 689 &lt;/children&gt; 690 &lt;children xmi:id="TG1JEmYREd6-LLTu1P_2sg" \ 691 &lt;styles xmi:type="umlnotation:UMLShapeComp 692 &lt;/children&gt; 693 &lt;styles xmi:type="umlnotation:UMLShapeStyle" 694 &lt;layoutConstraint xmi:type="notation:Bounds" 695 &lt;/children&gt; 696 &lt;children xmi:id="py6JELjAEd6NGa6eM2hxPA" ele 697 &lt;children xmi:id="py_BkLjAEd6NGa6eM2hxPA" 1 698 &lt;layoutConstraint xmi:type="notation:Size" 699 &lt;/children&gt; 700 &lt;children xmi:id="pzBd0LjAEd6NGa6eM2hxPA" 1 701 &lt;children xmi:id="pzCE4LjAEd6NGa6eM2hxPA" 1 702 &lt;children xmi:id="pzC8LjAEd6NGa6eM2hxPA" 1 703 &lt;children xmi:id="pzGWULjAEd6NGa6eM2hxPA" 1 704 &lt;styles xmi:type="umlnotation:UMLListComp 705 &lt;/children&gt; 706 &lt;children xmi:id="pz69YLjAEd6NGa6eM2hxPA" 1 707 &lt;styles xmi:type="umlnotation:UMLListComp 708 &lt;/children&gt; 709 &lt;children xmi:id="pzJZoLjAEd6NGa6eM2hxPA" \ 710 &lt;styles xmi:type="umlnotation:UMLListComp 711 &lt;/children&gt; 712 &lt;children xmi:id="pzKwLjAEd6NGa6eM2hxPA" \ 713 &lt;styles xmi:type="umlnotation:UMLShapeComp 714 &lt;/children&gt; 715 &lt;styles xmi:type="umlnotation:UMLShapeStyle" 716 &lt;layoutConstraint xmi:type="notation:Bounds" </pre>
----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

<pre> 3786ute&gt; 3787te xmi:id="G4lKYGoVEd6QyWjtn_MHMg" name="usedByFragments" 3788nt xmi:id="nqwKgLUEd6zPpBabTgdIQ" annotatedElement="G4 3789COMING,@IGNOREDIFF&lt;/body&gt; 3790ent&gt; 3832 3833d="lvNLUgVjEd6yV7V4a3bq1A" name="exception" visibility="f 3834e="uml:LiteralUnlimitedNatural" xmi:id="lvPagGVjEd6yV7V4a 3835e="uml:LiteralInteger" xmi:id="lv0ZcGvJEd6yV7V4a3bq1A"/&gt; </pre>	<pre> 3818ute&gt; 3819te xmi:id="fuz5QGYQEd6-LLTu1P_2sg" name="isTransient" 3820type="uml:PrimitiveType" href="//UML_LIBRARIES/ 3821ute&gt; 3822te xmi:id="G4lKYGoVEd6QyWjtn_MHMg" name="usedByFragme 3823nt xmi:id="nqwKgLUEd6zPpBabTgdIQ" annotatedElement=" 3824 3825d="lvNLUgVjEd6yV7V4a3bq1A" name="raisedException" vis 3827e="uml:LiteralUnlimitedNatural" xmi:id="lvPagGVjEd6yV 3828e="uml:LiteralInteger" xmi:id="lv0ZcGvJEd6yV7V4a3bq1A </pre>
----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

# Konzeptuelle Differenzen





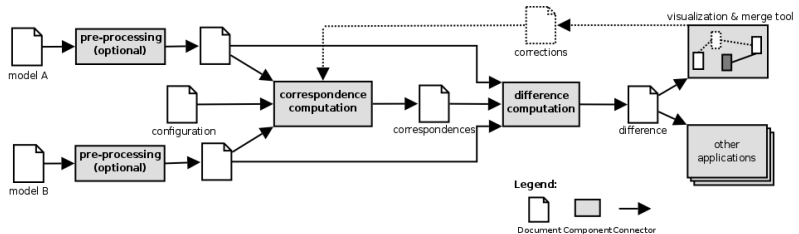
## Fazit

- Text-orientierte Werkzeuge ungeeignet für technische Dokumente mit grafischer Notation
- Einsatz grafischer Dokumente und Modelle rasant zunehmend, bspw. im Zuge moderner Entwicklungsparadigmen wie der modellbasierten Software-Entwicklung (MDA, MDD, MDSD, etc.)
- Daher Bedarf für hoch optimierte Verfahren und Werkzeuge für Modelle und grafische Dokumente
- → **SiDiff-Projekt** ([www.sidiff.org](http://www.sidiff.org))

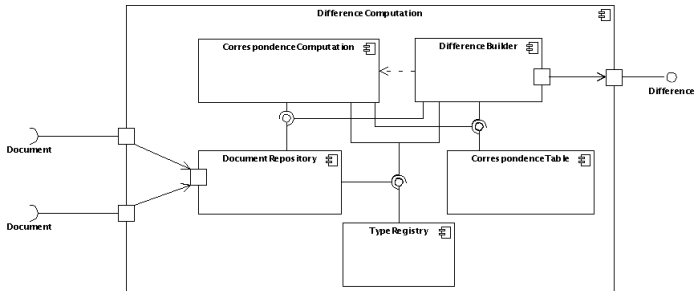
## Outline

- 1 **OSGi-Einsatzkontext: "SiDiff"**
  - Applikationsdomäne
  - Problemmotivation
- 2 **Warum OSGi?**
  - Architektur
  - Technologische Umsetzung
- 3 **Realisierungs-Probleme**
  - Einführendes Beispiel
  - Zustandsbehaftete Services
  - Service-Varianten
- 4 **Lösungsansätze und technische Umsetzung**
  - Kapselung der OSGi Service-Schicht
  - Zustandsbehaftete Services: "ProvideableService"
  - Service-Varianten: "ConfigurableService"
  - Kombination: "ConfigurableProvideableService"

## Workflow (Hier: Diff-Prozess)



## Komponenten und Services (Hier: Diff-Applikation, vereinfacht)



## Gründe für den Einsatz von OSGi

### Architektonische Gründe

- Flexibilität, Austauschbarkeit
- Lose Kopplung von klar definierten Komponenten/Services

### Organisatorische Gründe

"Echtes" Geheimhaltungsprinzip auf Komponentenebene:

- Einhaltung von Konventionen weitestgehend durch das Framework sichergestellt
- Effektive Maßnahme um der Degeneration der Architektur entgegen zu wirken

### Pragmatische Gründe

- Umsetzung des Service-Konzepts auf Basis von Java
- Nahtlose Integration in die Eclipse IDE

## Gründe für den Einsatz von OSGi

### Architektonische Gründe

- Flexibilität, Austauschbarkeit
- Lose Kopplung von klar definierten Komponenten/Services

### Organisatorische Gründe

“Echtes” Geheimhaltungsprinzip auf Komponentenebene:

- Einhaltung von Konventionen weitestgehend durch das Framework sichergestellt
- Effektive Maßnahme um der Degeneration der Architektur entgegen zu wirken

### Pragmatische Gründe

- Umsetzung des Service-Konzepts auf Basis von Java
- Nahtlose Integration in die Eclipse IDE

## Gründe für den Einsatz von OSGi

### Architektonische Gründe

- Flexibilität, Austauschbarkeit
- Lose Kopplung von klar definierten Komponenten/Services

### Organisatorische Gründe

"Echtes" Geheimhaltungsprinzip auf Komponentenebene:

- Einhaltung von Konventionen weitestgehend durch das Framework sichergestellt
- Effektive Maßnahme um der Degeneration der Architektur entgegen zu wirken

### Pragmatische Gründe

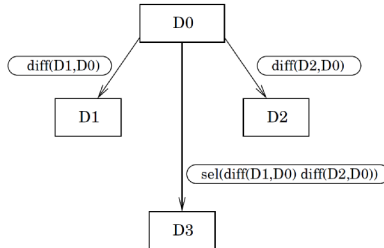
- Umsetzung des Service-Konzepts auf Basis von Java
- Nahtlose Integration in die Eclipse IDE

# Outline

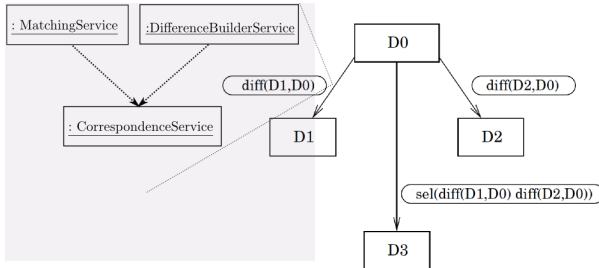
- 1 **OSGi-Einsatzkontext: "SiDiff"**
  - Applikationsdomäne
  - Problemmotivation
- 2 **Warum OSGi?**
  - Architektur
  - Technologische Umsetzung
- 3 **Realisierungs-Probleme**
  - Einführendes Beispiel
  - Zustandsbehaftete Services
  - Service-Varianten
- 4 **Lösungsansätze und technische Umsetzung**
  - Kapselung der OSGi Service-Schicht
  - Zustandsbehaftete Services: "ProvideableService"
  - Service-Varianten: "ConfigurableService"
  - Kombination: "ConfigurableProvideableService"



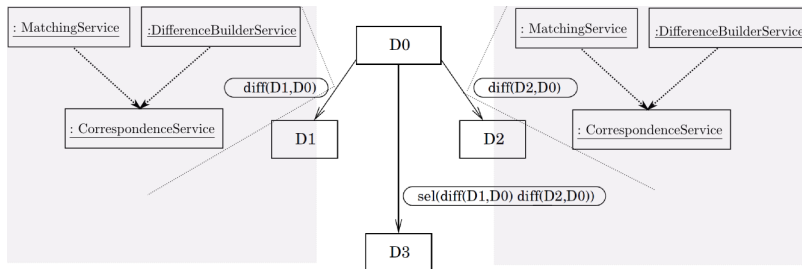
## 3-Wege-Vergleich/-Mischen



## 3-Wege-Vergleich/-Mischen



## 3-Wege-Vergleich/-Mischen



## Nutzung von Services in verschiedenen Kontexten

- Es kann oft vorkommen, dass ein Service gleichzeitig in verschiedenen Kontexten genutzt werden soll.
- Problematisch ist dies für zustandsbehaftete Services
- Hier kann nicht genau eine Service-Instanz beim OSGi-Framework registriert werden.
- Ein erster Lösungsansatz ist die OSGi Service Factory

# Die OSGi Service Factory und deren Grenzen

## OSGi Service Factory

- Die Service Factory ist Bestandteil des OSGi Standards.
- Kann anstatt des eigentlichen Services beim OSGi-Framework registriert werden.
- Die eigentliche Service-Instanz wird bei der Anforderung des Services transparent über die Factory erzeugt.

## Grenzen der OSGi Service Factory

- Die erzeugten Service-Instanzen werden vom OSGi-Framework **pro anfragendem Bundle** gecached.
- Szenarien, in denen ein Bundle verschiedene Instanzen eines Services erhalten sollen sind damit nicht realisierbar.

## Die OSGi Service Factory und deren Grenzen

### OSGi Service Factory

- Die Service Factory ist Bestandteil des OSGi Standards.
- Kann anstatt des eigentlichen Services beim OSGi-Framework registriert werden.
- Die eigentliche Service-Instanz wird bei der Anforderung des Services transparent über die Factory erzeugt.

### Grenzen der OSGi Service Factory

- Die erzeugten Service-Instanzen werden vom OSGi-Framework **pro anfragendem Bundle** gecached.
- Szenarien, in denen ein Bundle verschiedene Instanzen eines Services erhalten sollen sind damit nicht realisierbar.

## Konfigurierbare Services und Service-Varianten

- Oftmals benötigen wir Services, die z.B. für einen bestimmten Dokumenttyp konfiguriert werden.
  - Ein Beispiel ist der ähnlichkeits-basierte Matching-Service, welcher durch eine Dokumenttyp-spezifische Heuristik konfiguriert wird.
- Zudem sollte die Koexistenz von mehreren Service-Instanzen, welche verschieden konfiguriert sind unterstützt werden.
  - So zum Beispiel beim Einsatz von SiDiff im Rahmen einer Entwicklungsumgebung, welche mehrere Dokumenttypen unterstützt.

## Konfigurierbare Services und Service-Varianten

- Oftmals benötigen wir Services, die z.B. für einen bestimmten Dokumenttyp konfiguriert werden.
  - Ein Beispiel ist der ähnlichkeits-basierte Matching-Service, welcher durch eine Dokumenttyp-spezifische Heuristik konfiguriert wird.
- Zudem sollte die Koexistenz von mehreren Service-Instanzen, welche verschieden konfiguriert sind unterstützt werden.
  - So zum Beispiel beim Einsatz von SiDiff im Rahmen einer Entwicklungsumgebung, welche mehrere Dokumenttypen unterstützt.



## Outline










- 1 **OSGi-Einsatzkontext: "SiDiff"**
  - Applikationsdomäne
  - Problem motivation
- 2 **Warum OSGi?**
  - Architektur
  - Technologische Umsetzung
- 3 **Realisierungs-Probleme**
  - Einführendes Beispiel
  - Zustandsbehaftete Services
  - Service-Varianten
- 4 **Lösungsansätze und technische Umsetzung**
  - Kapselung der OSGi Service-Schicht
  - Zustandsbehaftete Services: "ProvideableService"
  - Service-Varianten: "ConfigurableService"
  - Kombination: "ConfigurableProvideableService"

## Outline

- 1 **OSGi-Einsatzkontext: "SiDiff"**
  - Applikationsdomäne
  - Problem motivation
- 2 **Warum OSGi?**
  - Architektur
  - Technologische Umsetzung
- 3 **Realisierungs-Probleme**
  - Einführendes Beispiel
  - Zustandsbehaftete Services
  - Service-Varianten
- 4 **Lösungsansätze und technische Umsetzung**
  - Kapselung der OSGi Service-Schicht
  - Zustandsbehaftete Services: "ProvideableService"
  - Service-Varianten: "ConfigurableService"
  - Kombination: "ConfigurableProvideableService"

## Der Service-Helper

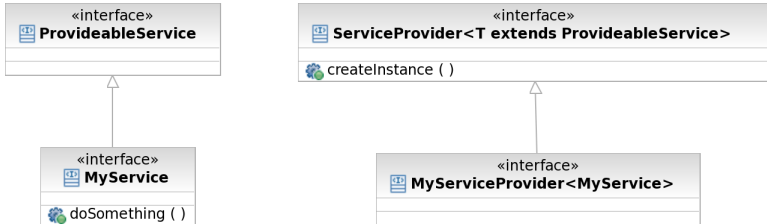
- Keine direkte Kommunikation mit dem OSGi-Framework über den OSGi-BundleContext
- Service-Helper kapselt die Zugriffe auf den Service-Layer des OSGi-Frameworks
- Allgemeine Schnittstelle für Aufgaben wie bspw. die Registrierung oder das Anfordern von Services

 <b>ServiceHelper</b>	
	<u>registerService ( )</u>
	<u>getService ( )</u>
	<u>registerServiceProvider ( )</u>
	<u>registerServiceConfigurator ( )</u>
	<u>configureInstance ( )</u>
	<u>unregisterInstances ( )</u>
	<u>unregisterVariantInstance ( )</u>
	<u>isAvailable ( )</u>

## Outline

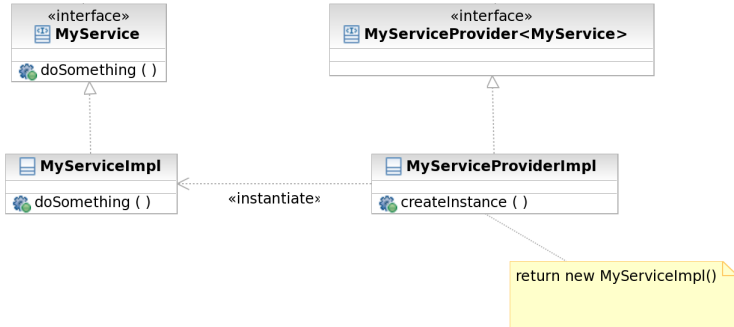
- 1 **OSGi-Einsatzkontext: "SiDiff"**
  - Applikationsdomäne
  - Problem motivation
- 2 **Warum OSGi?**
  - Architektur
  - Technologische Umsetzung
- 3 **Realisierungs-Probleme**
  - Einführendes Beispiel
  - Zustandsbehaftete Services
  - Service-Varianten
- 4 **Lösungsansätze und technische Umsetzung**
  - Kapselung der OSGi Service-Schicht
  - **Zustandsbehaftete Services: "ProvideableService"**
  - Service-Varianten: "ConfigurableService"
  - Kombination: "ConfigurableProvideableService"

## Definition der Service-Schnittstelle



- “Leeres” Interface
- Dient als “Marker” im automatischen Instanziierungsprozess
- Heißt wie die Service-Schnittstelle trägt das Suffix `Provider`.
- Liegt im selben Paket wie die Service-Schnittstelle.
- Erbt vom Interface `ServiceProvider`, getypd auf die Service-Schnittstelle.

## Implementierung des Services



- Implementierung des Dienstes
- Erzeugung einer Instanz des Dienstes

## Bekanntmachen der Service-Implementierung

```
public class Activator implements BundleActivator {  
  
    public void start(BundleContext context) throws Exception {  
        ServiceHelper.registerServiceProvider(  
            context,  
            MyServiceProvider.class,  
            new MyServiceProviderImpl(),  
            null,                               // DocType  
            null);                               // Variant  
    }  
  
    public void stop(BundleContext context) throws Exception {  
    }  
  
}
```

- Registriert wird der `ServiceProvider`
- Die Service-Implementierung selbst muss nicht registriert werden
- Sie wird über den `ServiceProvider` bereitgestellt

## Verwenden des Services

```
MyService ms = ServiceHelper.getService(  
    context,           \\ BundleContext  
    MyService.class,   \\ ServiceInterface  
    null,              \\ DocumentType  
    null);             \\ Variant  
ms.doSomething();
```

- Der Zugriff auf den Provider und die Instanzierung der eigentlichen Service-Implementierung erfolgt transparent.
- Der `ServiceHelper` prüft, ob der angeforderte Service ein `ProvideableService` ist.
- Wenn ja, wird beim OSGi-Framework ein entsprechend benannter `ServiceProvider` (hier `MyServiceProvider`) gesucht und dessen `createInstance()`-Methode aufgerufen.



## Outline

- 1 **OSGi-Einsatzkontext: "SiDiff"**
  - Applikationsdomäne
  - Problem motivation
- 2 **Warum OSGi?**
  - Architektur
  - Technologische Umsetzung
- 3 **Realisierungs-Probleme**
  - Einführendes Beispiel
  - Zustandsbehaftete Services
  - Service-Varianten
- 4 **Lösungsansätze und technische Umsetzung**
  - Kapselung der OSGi Service-Schicht
  - Zustandsbehaftete Services: "ProvideableService"
  - **Service-Varianten: "ConfigurableService"**
  - Kombination: "ConfigurableProvideableService"

## Definition der Service-Schnittstelle

- Konfigurierbare Services müssen das Interface `ConfigurableService` erweitern.

```
public interface ConfigurableService {  
    public String configure(Object... configData);  
    public void deconfigure();  
    public Dictionary<String, String> getProperties();  
}
```

- Die `configure()`-Methode wird aufgerufen, um den Service mit beliebigen Daten (`Object...`) zu konfigurieren.
- Der Rückgabewert der `configure()`-Methode ist der Dokumenttyp für den die Konfiguration geeignet ist. Da sich der Typ i.d.R. aus den Konfigurationsdaten ableitet, kann er sinnvollerweise nur hier ermittelt werden.

## Implementierung des Services

- Implementierung unterscheidet sich von "einfachen Services" nur durch die zusätzlichen Methoden des `ConfigurableService`
- damit wird der Service-Instanz die Konfiguration übergeben übergeben wird.

```
public class MyConfigurableServiceImpl implements MyConfigurableService

    String docType;

    public String configure(Object... configData) {
        // ...
        return docType;
    }

    public void deconfigure() {
    }

    public Dictionary<String, String> getProperties() {
        return null;
    }
}
```

## Bekanntmachen der Service-Implementierung

- Registrierung der Schnittstelle und der konkreten Implementierung des konfigurierbaren Services beim ServiceHelper

```
public class Activator implements BundleActivator {  
  
    public void start(BundleContext context) throws Exception {  
        ServiceHelper.registerServiceConfigurator(  
            context ,  
            MyConfigurableService.class ,  
            MyConfigurableServiceImpl.class );  
    }  
  
    public void stop(BundleContext context) throws Exception {  
    }  
}
```

## Konfiguration des Services

### Methode des ServiceHelpers:

```
public static void configureInstance(  
    BundleContext context,  
    Class<?> interfaceClass,  
    String docType,  
    String variant,  
    Object... configData)
```

### Aufrufbeispiel:

```
ServiceHelper.configureInstance(  
    context,  
    MyConfigurableService.class,  
    getEPackage().getNsURI(),  
    "SIMPLE",  
    "config.xml");
```

- Anschließend steht der Service in konfigurierter Form zur Verfügung.
- Die eigentliche Konfiguration erfolgt transparent durch den ServiceHelper.

## Verwenden des Services

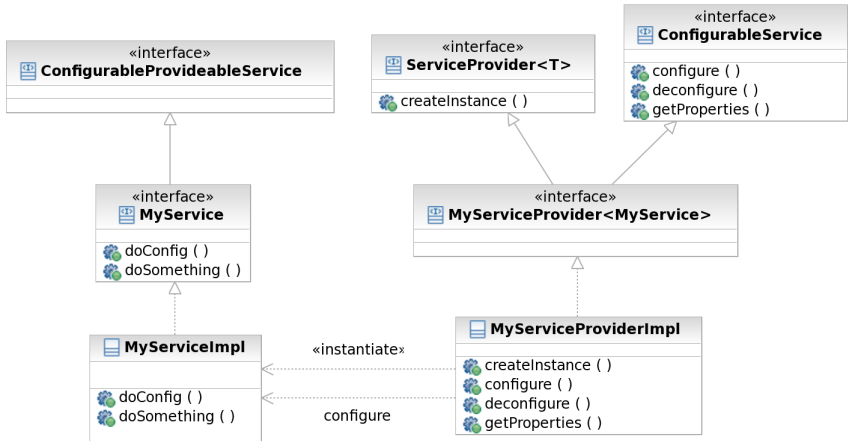
- Um einen fertig konfigurierten Service zu verwenden, wird dieser wie alle anderen Services über den `ServiceHelper` angefordert.
- Hier müssen jedoch der Dokumenttyp oder der Dokumenttyp und die Variante als Parameter mitgegeben werden, z.B.:

```
MyService ms = ServiceHelper.getService(  
    context ,  
    MyConfigurableService.class ,  
    eObj.eClass().getEPackage().getNsURI() ,  
    "SIMPLE" );
```

## Outline

- 1 **OSGi-Einsatzkontext: "SiDiff"**
  - Applikationsdomäne
  - Problem motivation
- 2 **Warum OSGi?**
  - Architektur
  - Technologische Umsetzung
- 3 **Realisierungs-Probleme**
  - Einführendes Beispiel
  - Zustandsbehaftete Services
  - Service-Varianten
- 4 **Lösungsansätze und technische Umsetzung**
  - Kapselung der OSGi Service-Schicht
  - Zustandsbehaftete Services: "ProvideableService"
  - Service-Varianten: "ConfigurableService"
  - Kombination: "ConfigurableProvideableService"

## Konfigurierbare und zustandsbehaftete Services





## Zusammenfassung

- SiDiff als Werkzeugkasten zum Bau von Differenz- und Mischwerkzeugen für Modelle
- Realisierung einer Service-orientierten Architektur auf Basis von OSGi
- Umsetzung der SiDiff-spezifischen Anforderungen durch eine auf dem OSGi Standard aufbauende Service-Schicht
  - Kapselung des OSGi-Frameworks
  - Unterstützung von zustandsbehafteten Services
  - Unterstützung von Service-Varianten
  - Unterstützung der Kombination beider Service-Arten

**Vielen Dank für Ihre Aufmerksamkeit**

Fragen?



[www.sidiff.org](http://www.sidiff.org)