4 Anhang

4.1 Erstellen einer Aggregationsfunktion in Odysseus

Im folgenden Abschnitt wird erklärt, wie eine neue Aggregationsfunktion in Odysseus angelegt werden kann. Voraussetzung dafür ist, dass die Developmentversion von Odysseus geklont wurde und bereits in Eclipse importiert wurde. Es darf außerdem nicht die Standardversion von Eclipse verwendet werden, sondern *Eclipse for RCP and RAP Developers*. Anschließend muss in Eclipse noch eine passende Target-Plattform gewählt werden, in diesem Fall wurde 'master-stable' verwendet.

Sind diese Schritte abgeschlossen, kann mit der Erstellung einer Aggregationsfunktion begonnen werden. Zur Veranschaulichung wird in dieser Anleitung eine Aggregationsfunktion mit dem Namen *ExampleAggregation* erstellt.

4.1.1 Plug-in Project

Im ersten Schritt muss ein neues Plug-in Project angelegt. Die Option dazu findet man unter File -> New -> Plugin Project.

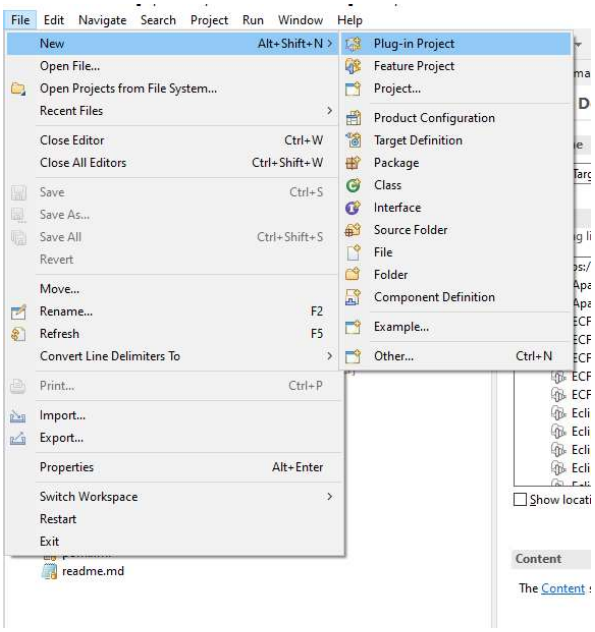

Abbildung 4.1: Anlegen eines neuen Plugin Projects

Anschließend öffnet sich ein Fenster, in dem einige Sachen festgelegt werden müssen (siehe Abb. 4.2).

Der Projektname wird im Regelfall so aufgebaut: de.uniol.inf.is.odysseus.server.aggregation. (Aggregationsname). Die genaue Angabe dieses Pfades ist dabei auch im weiteren Verlauf sehr wichtig, da

es bei unterschiedlicher Benennung schnell zu Problemen kommen kann.

Als Location des Projekts muss ein spezifischer Pfad eingegeben werden (Abb. 4.2). Der Pfad muss dabei auf einen Ordner namens 'aggregation.Aggregationsname' verweisen, der dafür vorher manuell im Ordner 'server' des Projekts angelegt werden muss.

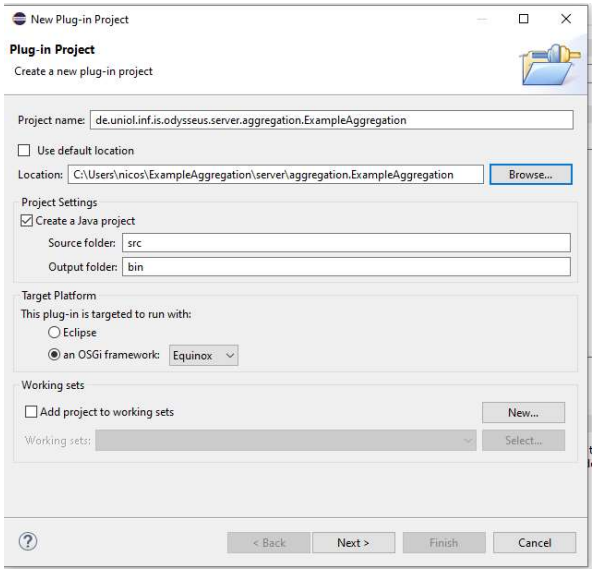

Abbildung 4.2: Angaben für Plugin Project

Als Letztes muss noch die Target Platform angepasst werden. Diese liegt standardmäßig auf Eclipse, für unsere Zwecke muss sie jedoch auf 'an OSGi framework' und 'standard' geändert werden.

Sind diese Schritte erledigt, kann man mit 'Next' zum nächsten Fenster übergehen. Dieses ermöglicht, ein bestimmtes Java Environment auszuwählen. Gibt es jedoch keine spezifischen Anforderungen diesbezüglich an das Plugin, kann ganz einfach kein Environment ausgewählt werden (siehe Abb. 4.3).

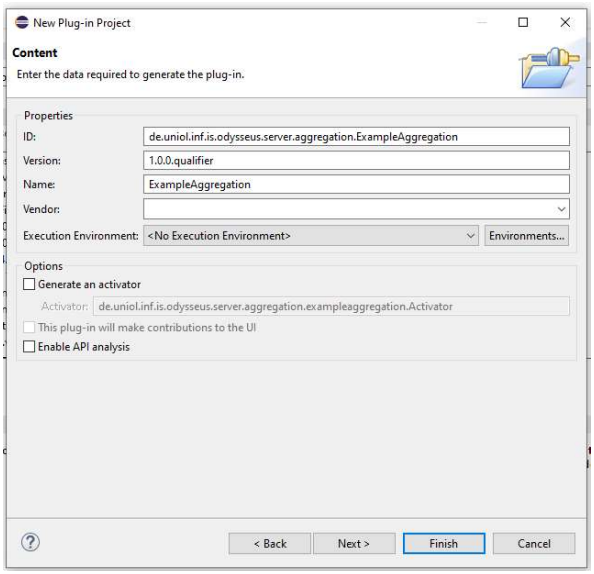

Abbildung 4.3: Angaben für Plugin Project - Seite 2

Anschließend kann man mit 'Finish' die Erstellung des Plugin Projects abschließen. Wenn alles geklappt hat, entsteht nun ein neues Projekt zwischen den bereits existierenden Projekten aus Odysseus mit dem Namen der Aggregation. In diesem Ordner befindet sich ein Unterordner, META-INF, welcher sich als nächstes angeschaut werden muss.

4.1.2 MANIFEST.MF

In diesem Ordner befindet sich eine MANIFEST.MF, in der verschiedene Plugin Dependencies für die Aggregation festgelegt werden müssen. Diese können entweder manuell in die Datei geschrieben werden (MANIFEST.MF -> Reiter MANIFEST.MF) oder im Reiter 'Dependencies' ganz einfach hinzugefügt werden (Abb. 4.4)

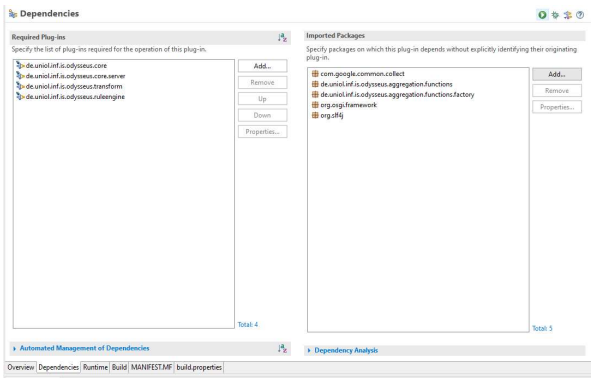

Abbildung 4.4: MANIFEST.MF - Dependencies

Im Reiter ’Dependencies’ gibt es nun ’Required Plugins’ und ’Imported Packages’. Diese können abhängig vom Plugin, welches erstellt werden soll, noch ergänzt werden, grundlegend müssen aber folgende Packages importiert werden:

Unter ’Required Plugins’:

- de.uniol.inf.is.odysseus.core
- de.uniol.inf.is.odysseus.core.server
- de.uniol.inf.is.odysseus.transform
- de.uniol.inf.is.odysseus.ruleengine

Und unter ’Imported Packages’:

- com.google.common.collect
- de.uniol.inf.is.odysseus.aggregation.functions
- de.uniol.inf.is.odysseus.aggregation.functions.factory
- org.osgi.framework
- org.slf4j

Diese werden durch den Import automatisch in die MANIFEST.MF hinzugefügt. Importiert man die Packages allerdings auf diesem Wege, werden sie mit einer bestimmten Version in die MANIFEST.MF importiert. Diese Versionsvorgaben können Fehler verursachen, und sollten daher manuell entfernt werden (siehe Abb. 4.5)

Zusätzlich zu diesen Packages müssen der MANIFEST.MF außerdem manuell noch folgende Zeilen hinzugefügt werden:

- Bundle-ActivationPolicy: lazy
- Service-Component: OSGI-INF/* oder OSGI-INF/AggregationFunctionAggregationsname.xml

Die Service-Komponente muss dabei importiert werden, um die Inhalte aus dem OSGI-INF Ordner zu laden, welcher im nächsten Schritt erstellt wird.

Vollständig sollte die MANIFEST.MF ungefähr so aussehen wie Abb. 4.5.

```
Manifest-Version: 1.0
Bundle-ManifestVersion: 2
Bundle-Name: ExampleAggregation
Bundle-SymbolicName: de.uniol.inf.is.odysseus.server.aggregation.ExampleAggregation
Bundle-Version: 1.0.0.qualifier
Automatic-Module-Name: de.uniol.inf.is.odysseus.server.aggregation.ExampleAggregation
Service-Component: OSGI-INF/AggregationFunctionExampleAggregation.xml
Require-Bundle: de.uniol.inf.is.odysseus.core,
 de.uniol.inf.is.odysseus.core.server,
 de.uniol.inf.is.odysseus.transform,
 de.uniol.inf.is.odysseus.ruleengine
Import-Package: com.google.common.collect,
 de.uniol.inf.is.odysseus.aggregation.functions,
 de.uniol.inf.is.odysseus.aggregation.functions.factory,
 org.osgi.framework,
 org.slf4j
Bundle-ActivationPolicy: lazy

```

Abbildung 4.5: vollständige MANIFEST.MF

4.1.3 OSGI-INF

In diesem Ordner wird die Komponente definiert. Der Ordner wird nicht automatisch erstellt, weswegen man in innerhalb des Plugin Projects manuell erstellen muss (siehe Abb. 4.6)

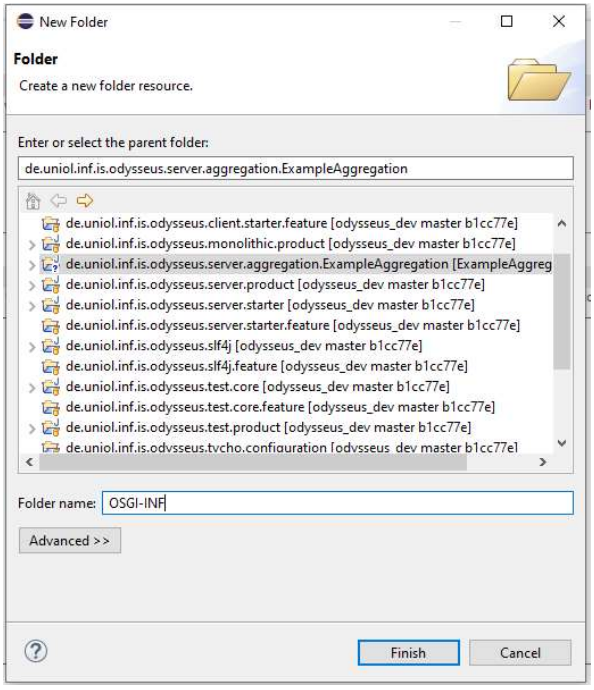

Abbildung 4.6: Anlegung des OSGI-INF Ordners

Nach der Erstellung muss der Ordner zu den ’Build Properties’ hinzugefügt werden. Dafür öffnet man die Datei ’build.properties’ und den Reiter ’Build’. Dort sollte der OSGI-INF Ordner aufgeführt sein. Dieser, sowie der ’bin’-Ordner, sollten hier markiert werden. Dadurch werden sie zu den ’Build Properties’ hinzugefügt (siehe Abb. 4.7).

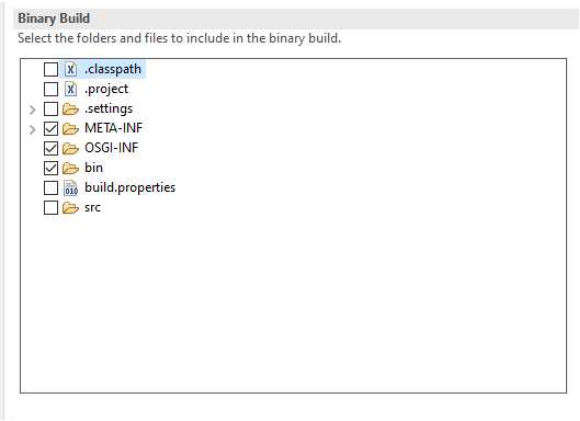

Abbildung 4.7: Anlegung des OSGI-INF Ordners

4.1.4 Feature Project

Nachdem das Plugin Project fertiggestellt wurde, muss zusätzlich ein Feature angelegt werden. Dies funktioniert genau wie beim Plugin Project unter File -> New -> Feature Project.

Nachdem dies ausgewählt wurde, öffnet sich erneut ein Fenster, ähnlich wie beim Plugin (siehe Abb. 4.8.

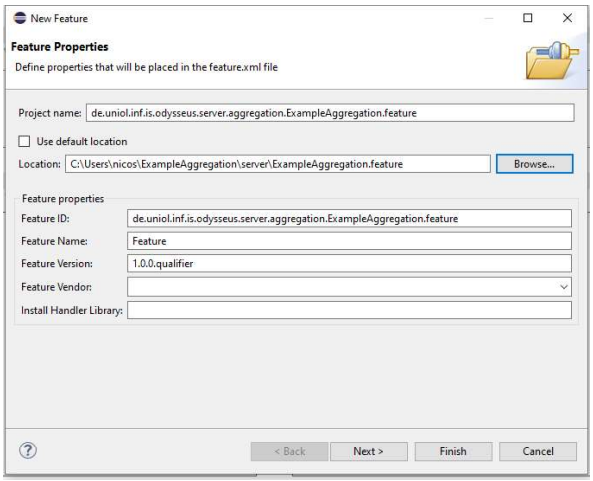

Abbildung 4.8: Angaben für das Feature

Der Projektname sollte hier genauso lauten wie der des Plugins, lediglich mit einem '.feature' am Ende. Um zu prüfen ob der Pfad übereinstimmt, kann man den Pfad ohne das '.feature' eingeben. Erscheint dann eine Meldung, dass dieses Projekt bereits existiert, ist der Pfad korrekt angegeben.

Auch für das Feature muss unter 'Location' ein neuer Ordner unter 'server' im Verzeichnis von Odysseus angelegt werden. Der Name dieses Ordners muss 'Aggregationsname.feature' lauten.

Anschließend gelangt man mit 'Next' zum nächsten Fenster (siehe Abb. 4.9). In diesem Fenster müssen zwei Plugins hinzugefügt werden:

- de.uniol.inf.is.odysseus.aggregation
- de.uniol.inf.is.odysseus.aggregation.Aggregationsname (das zuvor erstellte Plugin)

Wenn man nach 'aggregation' sucht, sollten beide dieser Plugins angezeigt, und mit einem Haken versehen werden, um sie hinzuzufügen.

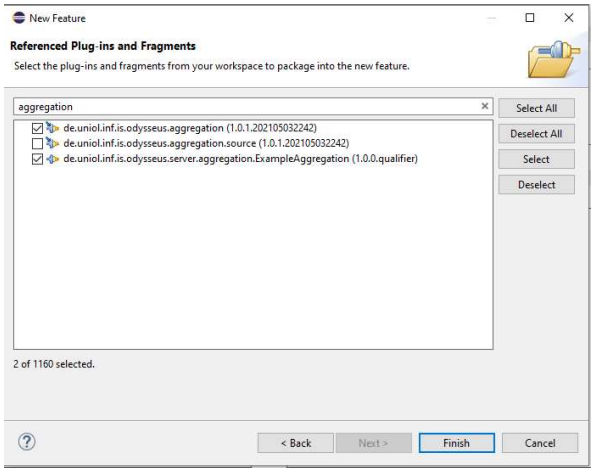

Abbildung 4.9: Angaben für das Feature - Seite 2

4.1.5 Implementierung der Aggregationsfunktion

Als nächstes folgt die tatsächliche Implementierung der Funktion. Im anfangs erstellten Ordner des Projekts gibt es einen 'src' Ordner, in dem die Klassen implementiert werden können. Dafür muss zunächst ein Package erstellt werden, bei Welchem wieder auf die richtige Benennung geachtet werden muss. Es muss genau den gleichen Namen tragen wie das Bundle, dies geschieht im Normalfall jedoch automatisch (siehe Abb. 4.10).

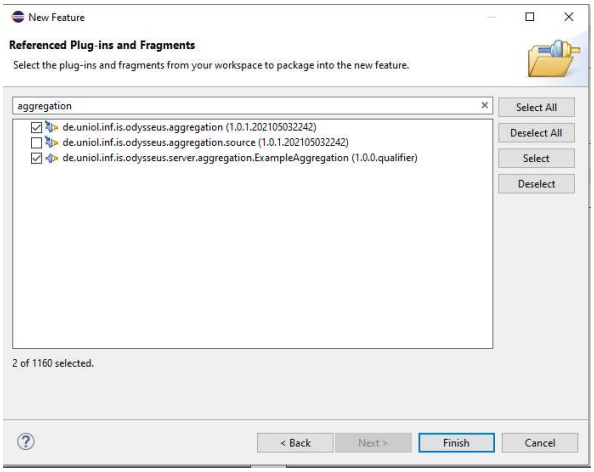

Abbildung 4.10: Erstellen des Packages für die Implementierung

In diesem Package kann dann die Klasse angelegt werden. Zu Demonstrationszwecken kopiert die hier erstellte *ExampleAggregation* lediglich den Inhalt von der in Odysseus bereits existierenden Funktion 'Count'. Sollten beim Import von verschiedenen Modulen, insbesondere aus Odysseus, noch zu Errors führen, müssen diese noch zur MANIFEST.MF hinzugefügt werden. Dafür geht man wieder in den Reiter 'Dependencies' und fügt die zu importierenden Packages manuell hinzu. //

Ansonsten gilt es bei der Implementierung lediglich noch, besonders darauf zu achten, der Funktion ihren Namen zu geben, um sie später in Odysseus auch aufrufen zu können. Dies macht man ganz simpel mit der Methode *getFunctionName()* (siehe Abb. 4.11). Diese fügt man einfach in die Aggregationsfunktion ein. Der Name muss dabei nicht unbedingt genauso lauten wie der Name der Klasse, dies wird aber in den meisten Fällen wahrscheinlich der Fall sein.

```
@Override
public String getFunctionName() {
    return "ExampleAggregation";
}
```

Abbildung 4.11: Zuweisung eines Namens für den Aufruf in Odysseus

4.1.6 Erstellung der Komponente

Nachdem eine Aggregationsfunktion erstellt wurde, muss diese noch als Komponente definiert werden, um in Odysseus aufgerufen werden zu können. Dafür wird der OSGI-INF Ordner benötigt, der zuvor erstellt wurde. Um die Aggregationsfnktion als Komponente zu definieren, geht man in dem OSGI-INF Ordner auf New -> Other..

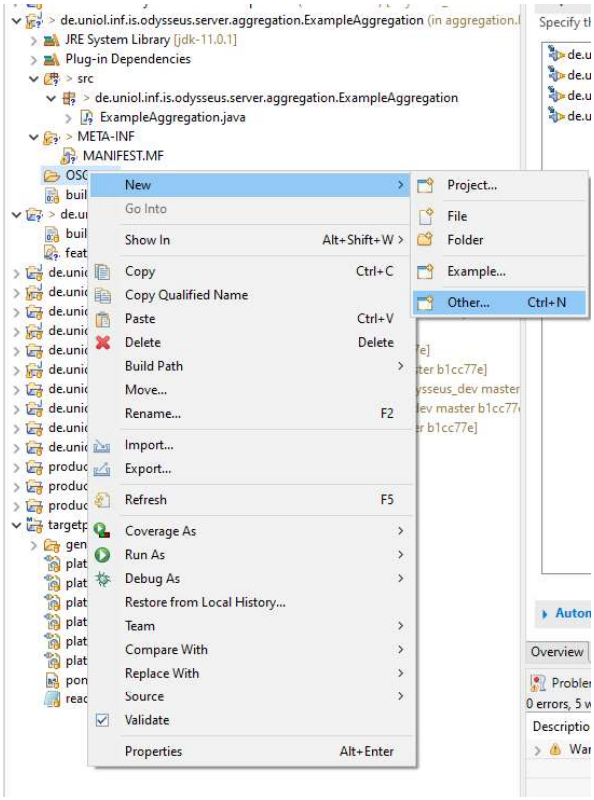

Abbildung 4.12: Auswahl zur Komponentendefinition

Dadurch öffnet sich ein neues Menü. Unter dem Menüpunkt 'Plug-in Dependencies' befindet sich dann die Option zur 'Component Definition'.

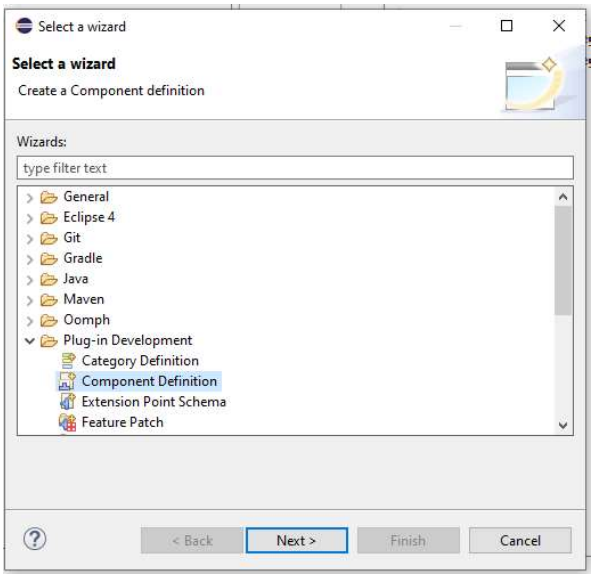

Abbildung 4.13: Auswahl zur Komponentendefinition

In der Komponentendefinition muss erneut sehr auf die Benennung der einzelnen Elemente geachtet werden. Es gibt dort vier Eingaben:

Zunächst wird der parent folder gefordert, welcher im Normalfall schon richtig eingetragen ist. Als Zweites soll ein File Name angegeben werden. Aggregationsfunktionen sollten für die Übersichtlichkeit immer mit 'AggregationFunction' beginnen, und anschließend den Namen der Funktion enden. Im Beispiel wäre der Name daher 'AggregationFunctionExampleAggregation.xml'. Der dritte Punkt ist der Name der Komponente selbst, welcher wieder mit dem Namen des Bundles und des Features übereinstimmen sollte (siehe Abb. 4.14 für die Bennenung im Beispiel). Als Letztes wird noch der Pfad zur Javaklasse benötigt, in der die Funktion implementiert wurde. Dieser kann dort manuell eingegeben werden oder mit 'Browse...' gesucht werden.

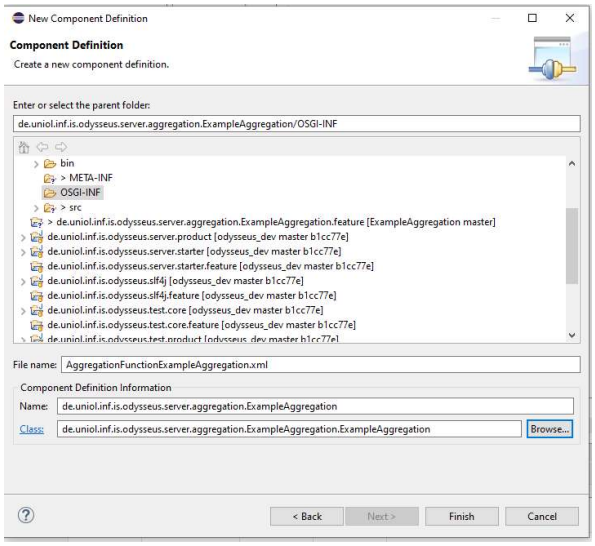

Abbildung 4.14: Eintragungen zur Komponentendefinition

Sind alle Felder ausgefüllt und die Namen übereinstimmend mit den zuvor erstellten Projekten, kann die Komponente nun erstellt werden. Ist dies geschehen, entsteht im OSGI-INF Ordner die soeben definierte XML-Datei. In dieser muss unter dem Reiter 'Services' anschließend das IAggregation-FunctinFactory Interface unter 'Provided Services' hinzugefügt werden (siehe Abb. **??**).

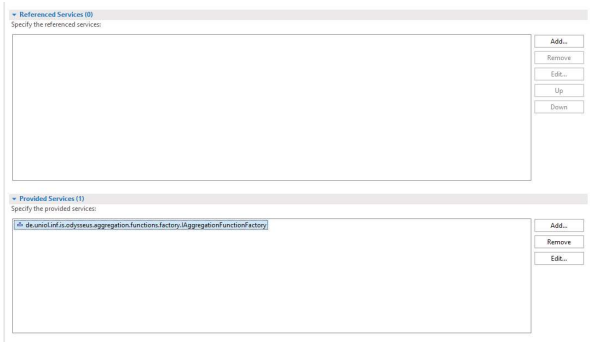

Abbildung 4.15: Provided Services in der Komponente

4.1.7 Abschluss

Im letzten Schritt müssen nur noch zwei Abhängigkeiten zu bereits bestehenden Projekten in Odysseus hinzugefügt werden. Als erstes muss im Projekt 'product.monolithic.feature' in der 'feature.xml' im Reiter 'Included Features' das Feature 'product.server.feature' hinzugefügt werden.

Anschließend muss noch in eben diesem 'product.server.feature', ebenfalls in der feature.xml im Reiter 'Included Features' das von uns selbst erstellte Feature hinzugefügt werden. Dies heißt im Beispiel 'de.uniol.inf.is.odysseus.aggregation.ExampleAggregation.feature'.

Sobald dies erledigt ist, ist die erstellte Aggregationsfunktion in Odysseus verfügbar. Um Odysseus nun zu starten, benutzt man das Projekt 'de.uniol.inf.is.odysseus.monolithic,product'. Von dort aus kann Odysseus als Eclipse Application gestartet werden. Wenn dies keine Fehler verursacht, kann die Aggregationsfunktion in Odysseus getestet und verwendet werden.