2013

Remoting Patterns

Protokoll zur Arbeitsdurchführung

Patrick Mühl, Nanak Tattyrek
TGM
17.10.2013

Inhaltsangabe

Inhaltsangabe 2
Aufgabenstellung 3
Aufgabe01 - Remoting Patterns 3
Arbeitsaufteilung und Zeitaufzeichnung 4
UML Diagramme 5
Beschreibung der Applikation [1] 8
Kritik 9
Verbesserungsvorschläge 9
Quellen 9

Aufgabenstellung

Aufgabe01 - Remoting Patterns

Das Framework für Remoting Patterns finden sie unter dem Thema "Resources"!

Gruppenarbeit: 2 Mitglieder (Server/Client)

Analysieren Sie in einer Gruppe von 2 Leuten die mitgelieferte Implementation der verteilten LeelaApplikation. Identifizieren Sie dabei alle verwendeten Elemente der "Basic Remoting Patterns" und erstellen Sie UML-Klassendiagramme für die Pakete comm, comm.socket, comm.soap, evs2009 und evs2009.mapping

Schließen Sie die unfertigen Tests ab, und dokumentieren Sie etwaige Schwierigkeiten.

Was ist zu tun?

- UML Klassendiagramm
- Erweitern der Testfälle (mind. einen Testfall erweitern)
- Kritik und Verbesserungsvorschläge

Punkte (16):

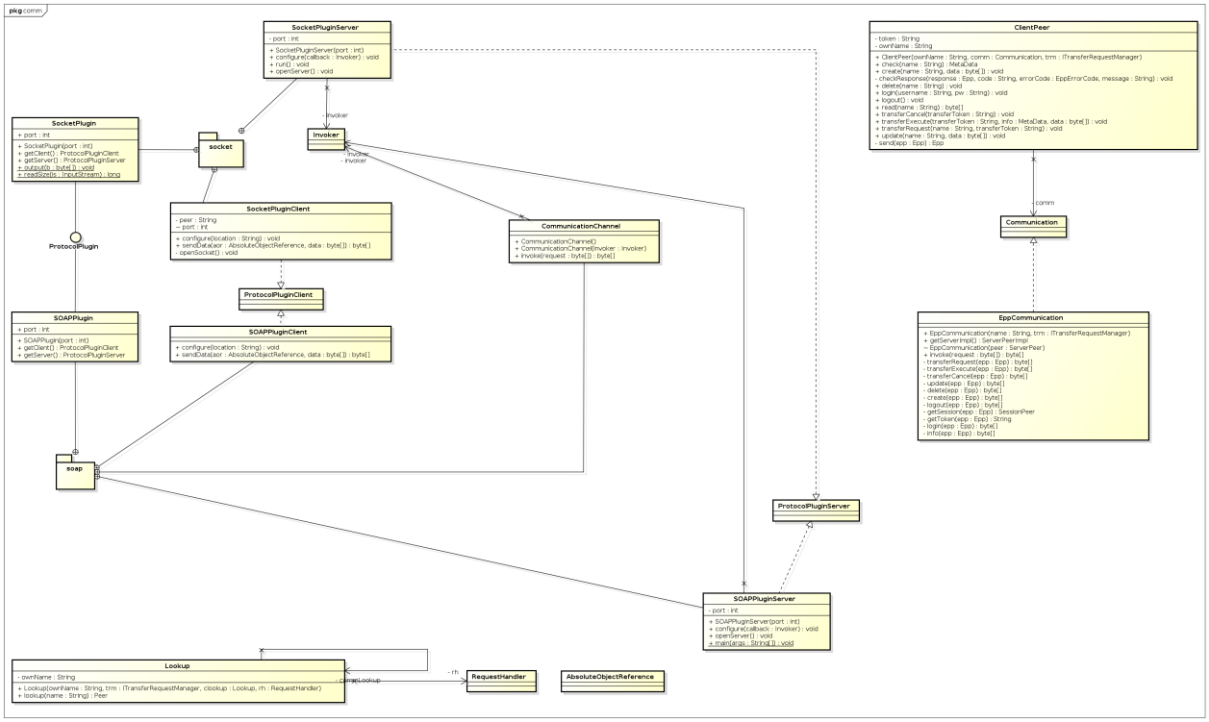
Identifikation von Basic Remoting Patterns ... 1Pkt
Beschreibung der Applikation ... 4Pkt
UML-Diagramme ... 2Pkt
Schreiben von neuem Testfall ... 3Pkt
konstruktive Verbesserungsvorschläge / Kritikpunkte ... 6Pkt

Arbeitsaufteilung und Zeitaufzeichnung

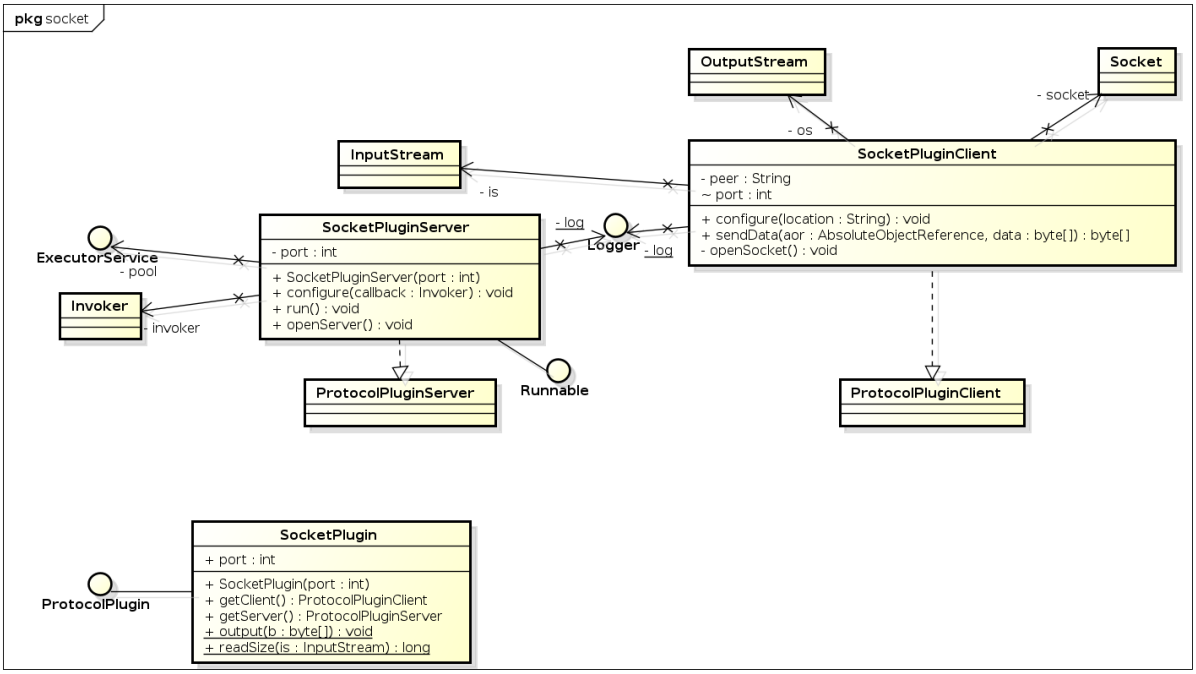
Subject	Patrick Mühl	Nanak Tattyrek	Geschätzte Zeit	Gesamte Zeit
Erweitern der Testfälle	x		2h	
Erstellen der UML Diagramme		x	2h	
Beschreibung der Applikation	x		3h	
Schreiben des Protokolls	x	x	1h	

UML Diagramme

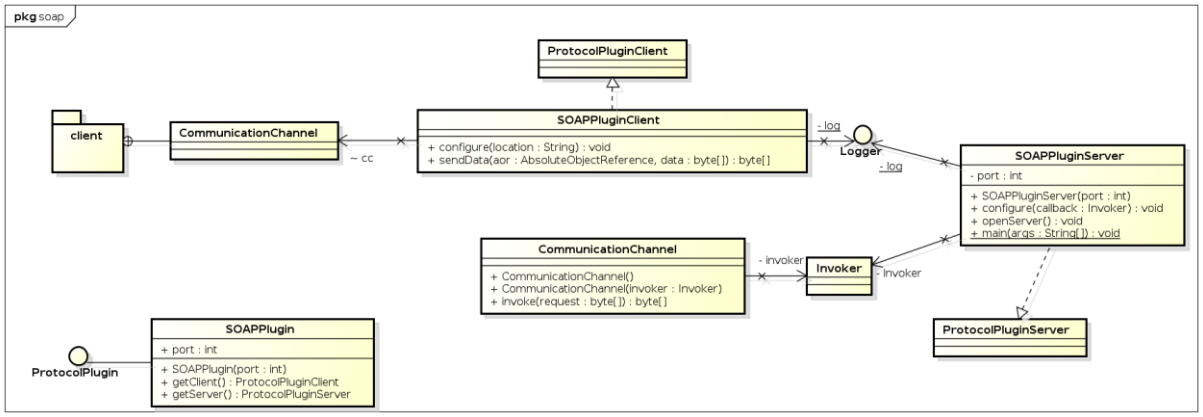
Package comm:



Package comm.socket

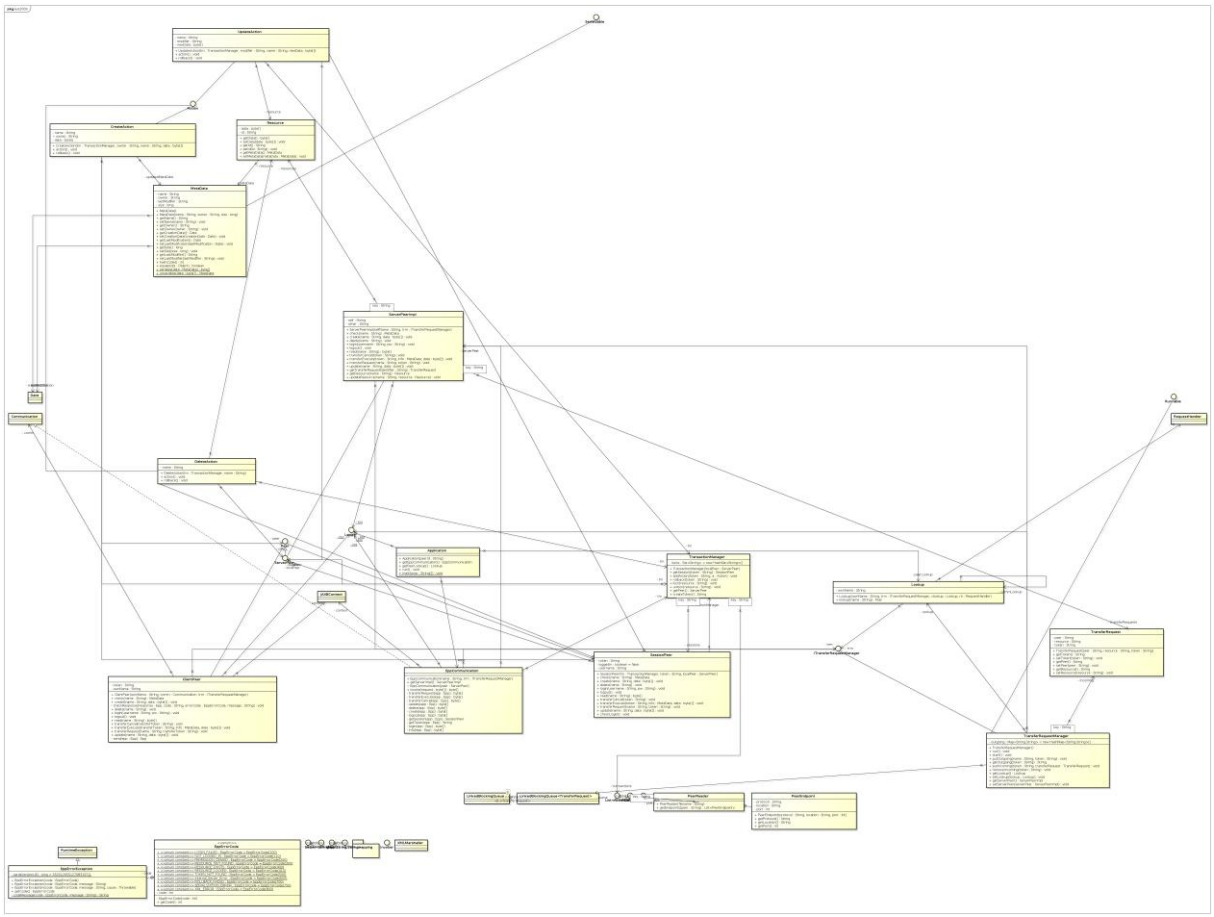


Package comm.soap

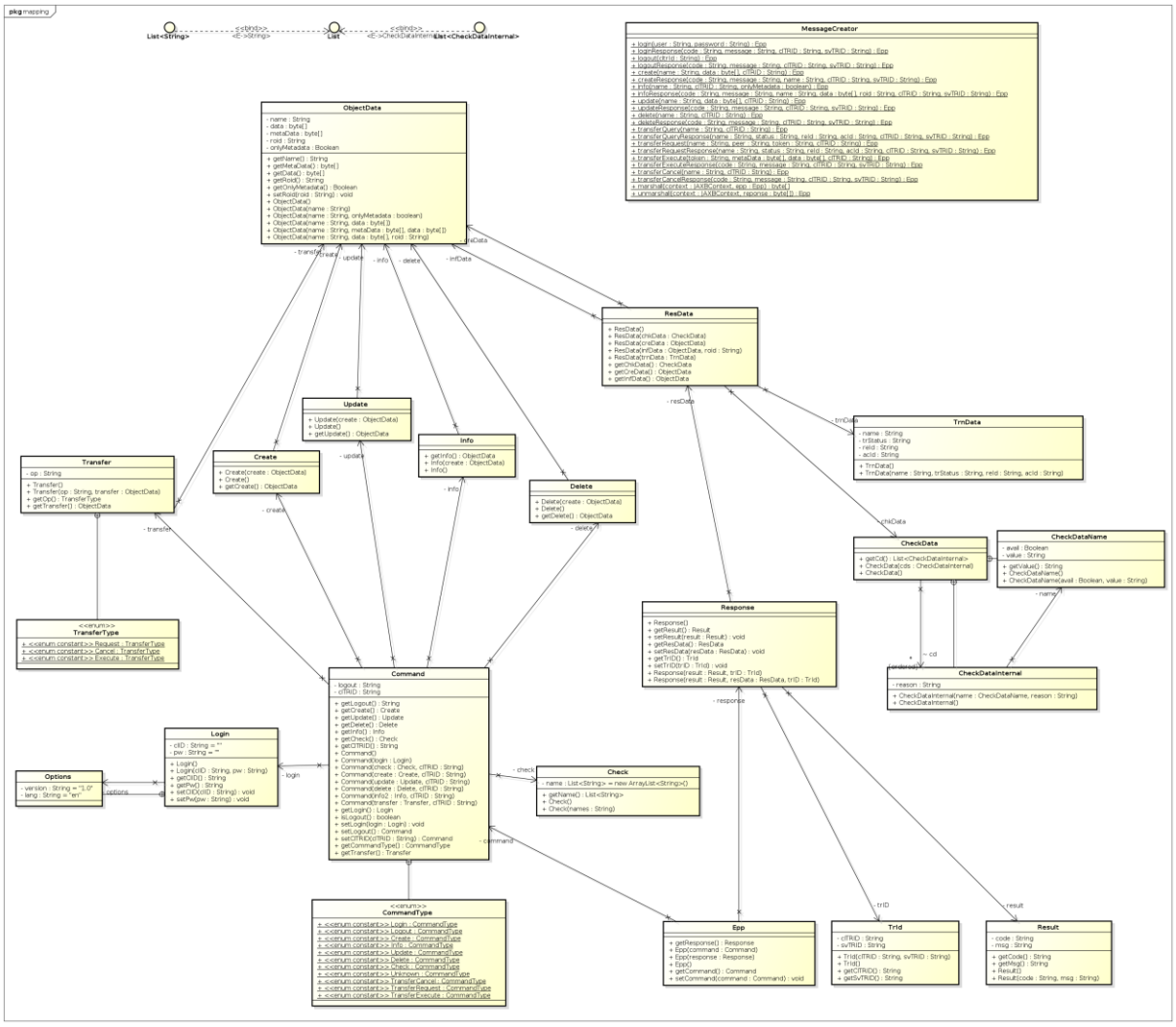


Package evs2009

Durch unangekündigte Vorverlegung der Abgabe konnte dieses Diagramm nicht ausreichend fertig gestellt werden. Von einem Punkteabzug diesbezüglich ist abzusehen.



Package comm.mapping



Beschreibung der Applikation [1]

Das zu verwendende Modell basiert auf dem Broker-Muster aus dem Buch Software Architektur[2]. *„Es soll vermieden werden, dass Aspekte verteilter Programmierung über den Code einer verteilten Applikation verstreut sind ... Durch die Verlagerung aller Kommunikationsaufgaben in einen Broker werden die Kommunikationsaufgaben eines verteilten Systems von dessen Applikationslogik getrennt. Der Broker verbirgt und steuert die Kommunikation zwischen den Objekten oder Komponenten des verteilten Systems. Auf der Klientenseite baut der Broker die verteilten Aufrufe zusammen und leitet sie danach an den Server weiter. Auf der Server-Seite nimmt der Broker die Anfrage entgegen und baut daraus einen Aufruf zusammen, den er dann auf einem Server-Objekt durchführt ... Der Broker übernimmt alle Details der verteilten Kommunikation, wie Verbindungsaufbau, Marshalling der Nachricht etc., und verbirgt diese Details – so weit wie möglich – vor dem Klienten und dem verteilten Objekt.“*

AbsoluteObjectReference hat notwendige Informationen eines Peers, wie Protokoll und Bestimmungsort. Das AOR wird vom Requestor verwendet.

Lookup liefert das AOR eines Peers zurück, das durch dessen Name indentifizierbar ist.

Requestor bietet ein dynamisches Interface zum Aufruf von Methoden über den RequestHandler an.

RequestorHandler arbeitet als Schnittstelle zwischen dem lokalen Peer und den Anfragen von entfernten Peers. Dabei nutzt der RequestHandler den Invoker für die einzelnen Server Instanzen.

Invoker bietet die Methode handleRequest(byte[]), welche eingehende Anfragen abarbeitet.

ProtocolPluginServer wird als Interface in den einzelnen Plugins implementiert und bearbeitet die eingehenden Aufrufe. Die einzelnen Protokolle werden beim Aufruf des Invokers instanziert und konfiguriert.

ProtocolPluginClient ist als Interface in den Protokollen als Schnittstelle nach außen vorgesehen. Durch das AOR wird der richtige Requestor ausgewählt und verwendet um eine Anfrage an einen entfernten Peer zu senden.

2 Software-Architektur; O.Vogel, I.Arnold, A.Chughtai, E.Ihler, U.Mehlig, Th.Neumann, M.Völter, U.Zdun; Spektrum2005

Kritik

- Der Programmcode ist nicht dokumentiert. Dies führt dazu das die Beschreibung der Applikation etwas erschwert wird.
- Sockets mit gleichem Namen werden 2- mal initialisiert. Das führt dazu das die Tests Fehler liefern.
- Die Klasse ApplicationTest liefert Fehler beim Test.

Verbesserungsvorschläge

- Code Kommentare vereinfachen das lesen des Programmcodes. Daher ist es sinnvoll des Code ausreichend zu dokumentieren.
- Sockets nicht gleich benennen und falls doch dann nicht initialisieren. Die Sockets wurden umbenannt (Am Namen des Sockets einfach eine 1 hinzugefügt)
- Da Maven in der default Einstellung alle Klassen die mit blaTest enden als Test ausführt wurde die Klasse ApplicationTest umbenannt um den Test hier nicht durchführen zu lassen.

Quellen

[1] Communication Framework
184.153 Design Methods for Distributed Systems swa028
BORKO Michael,GREIFENEDER Michael, MOTLIK Florian,