**Team**: 6, Mert Siginc, Michael Müller

**Aufgabenaufteilung**:

1. Wie gefordert alles allein.

**Quellenangaben**:

* <http://users.informatik.haw-hamburg.de/~klauck/verteiltesysteme.html>

**Bearbeitungszeitraum**:

**Aktueller Stand**:

Fertig

**Änderungen des Entwurfs**: Vorabgabe des Entwurfs & Nachgabe nicht möglich, deswegen keine Änderung.

**Entwurf**: <Entwurf nach den bekannten SE-Richtlinien und den Vorgaben gemäß Aufgabenstellung.>

Ergebnis des Prozesses des Definierens von Architektur, Komponenten, Schnittstellen, Abstrakten Datentypen und anderen Charakteristika eines Systems oder einer Komponente. Dient als einziges Dokument bei der Implementierung!

Anforderungsermittlung:

* (Nicht) Funktionale Anforderungen
* Anwendungsfälle
* Userstories

Modellierung:

* UML
* OO/funktionales Design
* Design & Architektur Pattern
* GRASP

Gesamte Kommunikation über ObjectBroker und sein Communicator. Somit jeder Socket pro ObjectBroker nur genau einmal und kann wiederverwendet werden.

Jedoch dann ggf. gegenseitiges Ausbremsen der Services / Clients da gewartet werden muss bis selber Socket wieder frei. Hätten alle einen eigenen Socket gings schnell und Deadlock ist kein Thema (wenn Service selbst nicht Statusbehaftet).

Reflection vs. Alternative

Ich habe hier Reflection verwendet, allerdings könnte man auch alle Services eine bestimmte „IMiddlewareService“ Schnittstelle implementieren lassen. In der es eine Funktion gibt die die angekommene Nachricht (siehe Nachrichtenformat) verwertet und prüft ob es die Methode in der Klasse gibt und sie dann benutzt (nach Extrahierung der Parameterwerte aus der Nachricht).

Dafür müsste man jedoch für jede bekannte Methode ein neuen If-Case oder Switch-Case bauen. Was immer noch eine Erweiterung ermöglicht, jedoch ist diese mit mehr Aufwand verbunden als die Reflection.

Die ist dafür allerdings etwas unsicherer, was man allerdings mit try-catch abfangen kann.

Nachrichtenformat:

Request: „Methodenname(ParameterEinsKlasse ParameterEinsWert,ParameterZweiKlasse ParameterZweiWert)“

Reply: „Antwort“

Z.b.:

* „resolve(String calculator)“
* „add(int 3,int 5)“
* „localhost:15000“