Entwurf VS – Aufgabe 4

Team (Teamnummer, Teammitglieder)	- **2** - **Lukas Lühr & Florian Stäps**

Aufgabenaufteilung	- Lukas Lühr
	  o Compiler
	  o Kommunikationsmodul
	- Florian Stäps
	  o Planung
	  o Namensdienst

Quellenangaben	- Aufgabenstellung - Vorlesungsfolien von Herrn Klauck - Bereitgestelltes Material von Herrn Schulz
Bearbeitungszeitraum	- ca. 21 Stunden - + 4,4 Stunden
Aktueller Stand	- Alles fertig
Änderung des Entwurfs	- Protokoll wurde angepasst, um Exceptions ohne Text Übertragen zu können (siehe Protokoll Spezifikation).
Entwurf	- Siehe folgendes Dokument

1 INHALTSVERZEICHNIS

1 AUFGABENSTELLUNG

Im Rahmen der vierten Aufgabe ist eine Middleware, mit dazu gehörigen Namensdienst zu entwickeln und zu implementieren.
Die Programmiersprache ist in dieser Aufgabe mit Java vorgeschrieben. Allerdings sind alle Details, bis auf den groben Aufbau, uns frei überlassen. Deshalb muss diese Lösung nicht mit der, der anderen Gruppe zusammenarbeiten können.

Die Middleware soll hier von ganz einfachen Programmen genutzt werden. Wir haben uns für einen Taschenrechner mit Addition und Subtraktion entschieden. Dabei kann ein Taschenrechner nur jeweils eine Rechenoperation und soll die jeweils andere vom anderen Taschenrechner über die Middleware nutzen.
Zur Middleware gehört unter anderem ein Compiler, der aus einer idl – Datei entsprechende Java – Dateien erstellen sollen, die für den weiteren Programmablauf benötigt werden. Der Compiler erstellt Schnittstellenklassen, von denen für die weitere Benutzung, die Klassen des Anwenders abgeleitet werden.
Des Weiteren gehört zur Middleware ein Namensdienst dazu. Dieser soll später auf einem separaten Rechner laufen und soll es ermöglichen, sich bei diesem zu registrieren und eine Namensauflösung vorzunehmen.

2 TECHNISCHE PLANUNG

Im weiteren Verlauf gehen wir auf unsere Überlegungen zur Umsetzung der Aufgabe vier ein. Neben erklärenden Texten, sind auch UML – Diagramme enthalten.

2.1 NAMENSDIENST

Der Namensdienst ist kein eigentlicher Teil der Middleware, er ist ein Objekt, welches über diese bereitgestellt wird. Dabei erfüllt er die Funktionalität einer Map. Es können Objekte unter einem Namen dort registriert werden, und ein Name kann zu einer oben beschriebenen Objekt Referenz aufgelöst werden.

2.1.1 Referenzaufbau

Die Referenz eines Objektes setzt sich aus dem Rechner, auf dem das Objekt läuft, dem Port über dem die Middleware des Rechners erreichbar ist und einem Identifizierungsmerkmal zusammen. Wobei das Identifizierungsmerkmal eine frei wählbare Zeichenkette ist.

Eine möglicher Objekt Referenz währe:

Lab33:6000/NameService

Welche ein Objekt namens NameService auf dem Rechner lab33 mit der Middleware unter Port 6000 erreichbar beschreibt.

2.2 MIDDLEWARE

2.2.1 ObjectBroker

Ist für den Benutzer das zentrale Einstiegsobjekt der Middleware, um mit den entfernen Objekte zu interagieren.
Dazu nutzt der ObjectBroker das Kommunikationsmodul, um Nachrichten zu verschicken und Antworten auszuwerten. Damit der Benutzer Zugriff auf die Referenzen der entfernten Objekte bekommt, stellt der ObjectBroker dem Benutzer ein Objekt des Namensdienstes zur Verfügung.

Der ObjectBroker führt eine Liste aller Lokalen Objekte, und ihrer Namen.

2.2.1.1 Entfernter Aufruf

Wird meistens durch ein Stellvertreter Objekt ausgelöst. Mittels des Kommunikationsmodules wird, falls noch nicht geschehen, eine Verbindung zum entfernten Rechner aufgebaut, und ein3 2ie im Protokoll spezifizierter Methoden Aufruf gestartet.

Anschließend, wird wie im Protokoll spezifiziert, auf eine Antwort gewartet, welche dann entsprechend zurückgegeben wird.

2.2.1.2 Asynchroner Aufruf

Technisch ist auch ein asynchroner Aufruf möglich, dieser umgeht das Abwarten einer Antwort.

Die Antwort wird von der Middleware zwischengespeichert, und kann später abgefragt werden.

Diese Art des Aufrufes wird nicht durch stellvertreter Objekte verwendet.

2.2.2 Kommunikationsmodul

Ist für die interne Kommunikation zwischen mehreren Modulen zuständig.
Für diese Kommunikation untereinander, wurde ein entsprechendes Request / Replay Protokoll entwickelt.

2.2.2.1 Protokoll

Unser Protokoll ist Byteorientiert und orientiert sich an der Type-Length-Value Struktur. Strings werden als UTF-8 kodiert.

2.2.2.1.1 Schlüsselwörter

Name	Wert(32bit)	Verwendung
Key_Request	0	Signalisiert, dass es sich bei einem Datenpaket um eine Request handelt
Key_Response	1	Signalisiert, dass es sich bei einem Datenpaket um eine Response handelt
Int_Type	1	Kennzeichnet einen integer Wert (Response/Request)
Double_Type	2	Kennzeichnet einen Double Wert(Response/Request)
String_Type	3	Kennzeichnet einen String Wert (Response/Request)
Exception_Type	4	Kennzeichnet eine Exception (Response)

2.2.2.1.2 Datenformate

Für Daten welche übertragen werden, wie etwa Parameter oder Rückgabewerte, gelten folgende Formatierungen:

<table>
<tr><td>Type (32bit)</td><td colspan="2">Value(Variable)</td></tr>
<tr><td>Int_Type</td><td colspan="2">Int(32bit)</td></tr>
<tr><td>Double_Type</td><td colspan="2">Double(64bit)</td></tr>
<tr><td>String_Type</td><td>Length(int 32bit)</td><td>ByteArray(Length)</td></tr>
<tr><td>Exception_Type</td><td>Exception class
(String Data)</td><td>Exception Message
(String Data)
Sollte kein Text Für die Exeption vorhanden sein wird als länge des Strings -1 angegeben und Keine Daten für diesen übertragen.</td></tr>
</table>

2.2.2.1.3 Request

Ein Request hat immer eine, pro sende Station, eindeutige Nummer. Diese wird für eine Antwort verwendet, des Weiteren wird ebenfalls der Name des Objektes kodiert.

Int(32bit)	Int(32bit)	stringData	stringData	Int(32Bit)	ParamCount *Data
Key_Request	RequestNo	ObjektName	MethodeName	ParamCount	Methode Args

Nach dem Absenden einer solchen Nachricht, an eine Station, wird ein gewisser Timeout, welcher konfigurierbar ist, auf eine Antwort in Form eines Respons gewartet. Wurde nach dem Timeout keine Antwort erhalten, wird der Aufrufende über ein potentielles Problem benachrichtigt.

2.2.2.1.4 Response

Eine Antwort, welche nur als Antwort auf einen Request gesendet werden darf, hat folgende Kodierung

Int(32bt)	Int(32bit)	Daten
Key_Response	RequestNo	RückgabeDaten

Der Response kann über die RequestNo dem zugehörigen Request zugeordnet werden. Hierüber sind auch asynchrone Aufrufe möglich.

Der Response kann über eine Beliebige Verbindung an die Station gesendet werden.

2.2.3 Encoder / Decoder

Ist für die Kodierung und Dekodierung von Response bzw. Request Nachrichten zuständig.

2.2.4 Compiler

Der Compiler erzeugt, aus in den idl - Dateien spezifizierten Klassenbeschreibungen, Schnittstellenklassen. Diese können dann vom Anwender genutzt werden, um mit der Middleware zu interagieren. Darunter auch eine abstrakte Klasse, die die Schnittstellen des Objektes spezifiziert. (Interface), sowie die lokale Stellvertreterklasse.

2.2.4.1 NarrowCast

Die narrow Cast Methode wird bereits in der abstrakten Schnittstellenspezifikationsklasse implementiert. Es wird ein String, die Referenz auf das Objekt, in eine Stellvertreterklasse umgewandelt.

2.2.4.2 Stellvertreterklasse

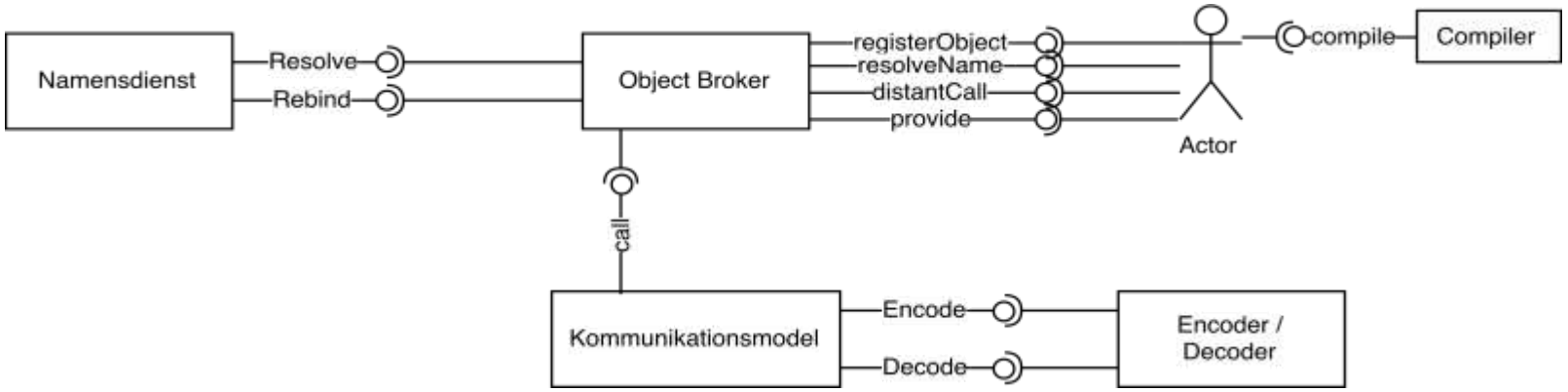
Die Stellvertreterklasse ist ebenfalls automatisch generiert.

Sie interagiert mit der Schnittstelle des Objekt Brokers, um entfernte Aufrufe zu tätigen, hierbei sind nur synchrone aufrufe vorgesehen.

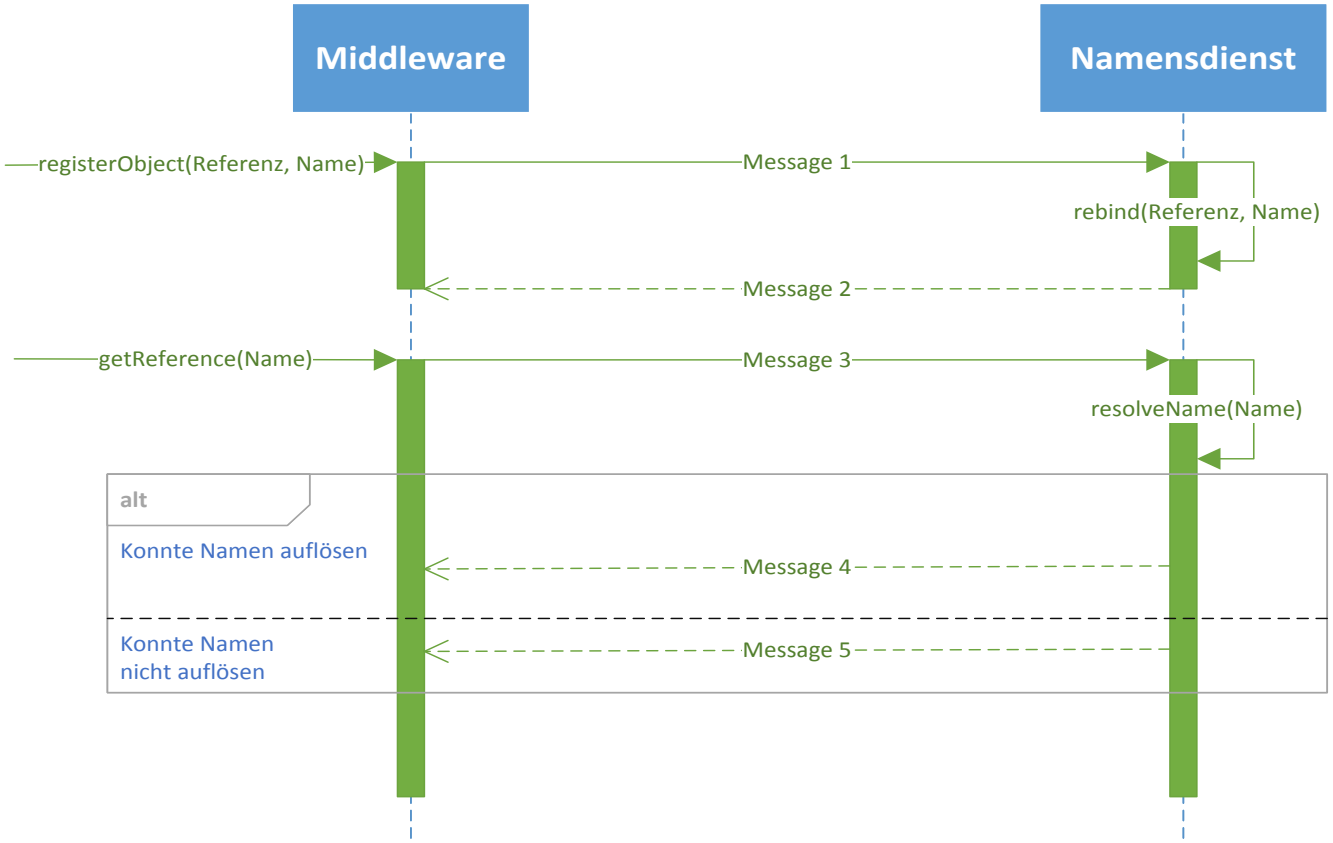
Eine Exception, welche ggf. von dem entfernten Objekt geworfen wurde, wird an dieser stelle erneut geworfen.

2.3 DIAGRAMME

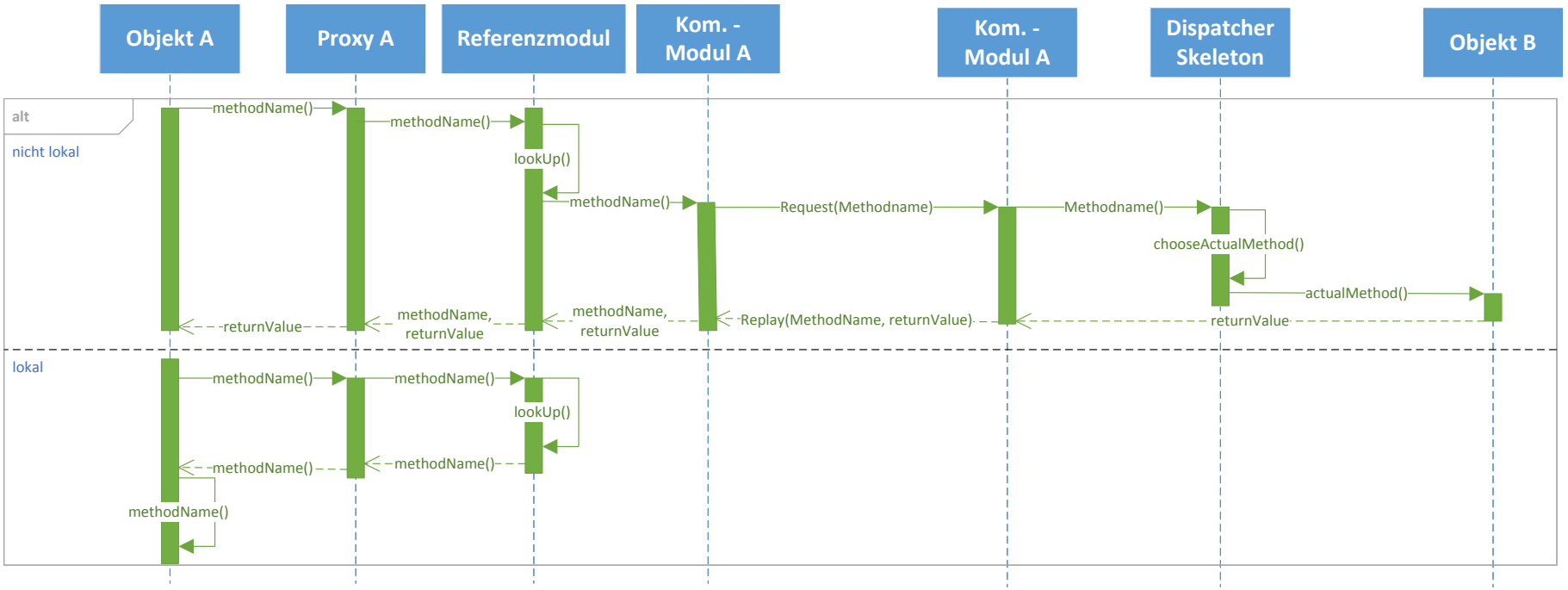
2.3.1 Schnittstellendiagramm



2.3.2 Kommunikation zwischen Middleware und Namensdienst

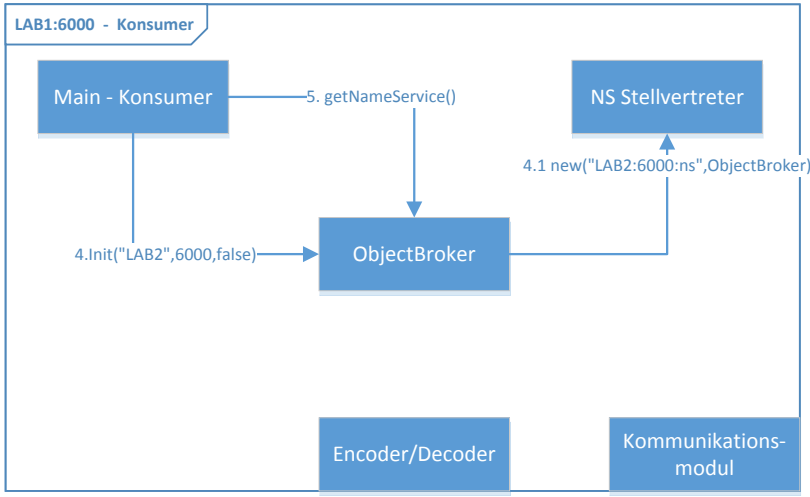


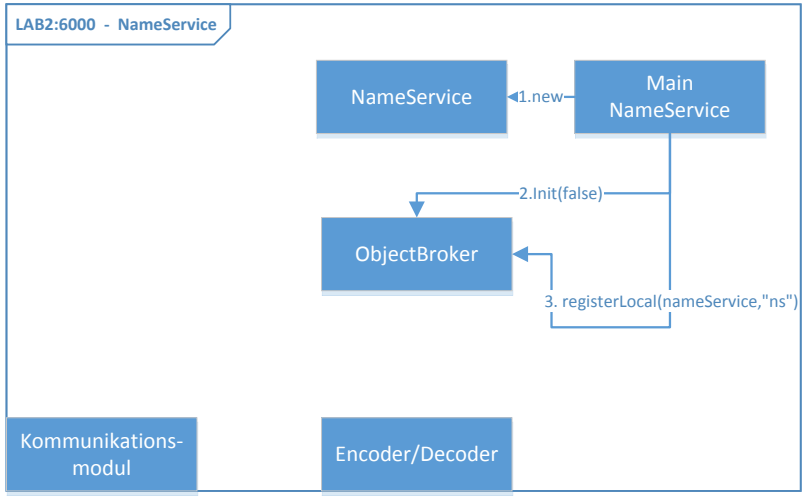
2.3.3 Genereller Ablauf

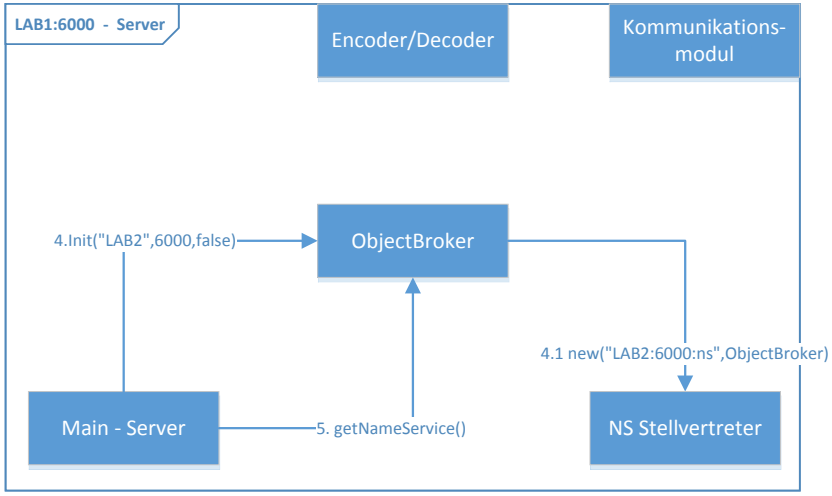


2.3.4 Kollaborationsdiagramme

2.3.4.1 Initialisierung des Systems





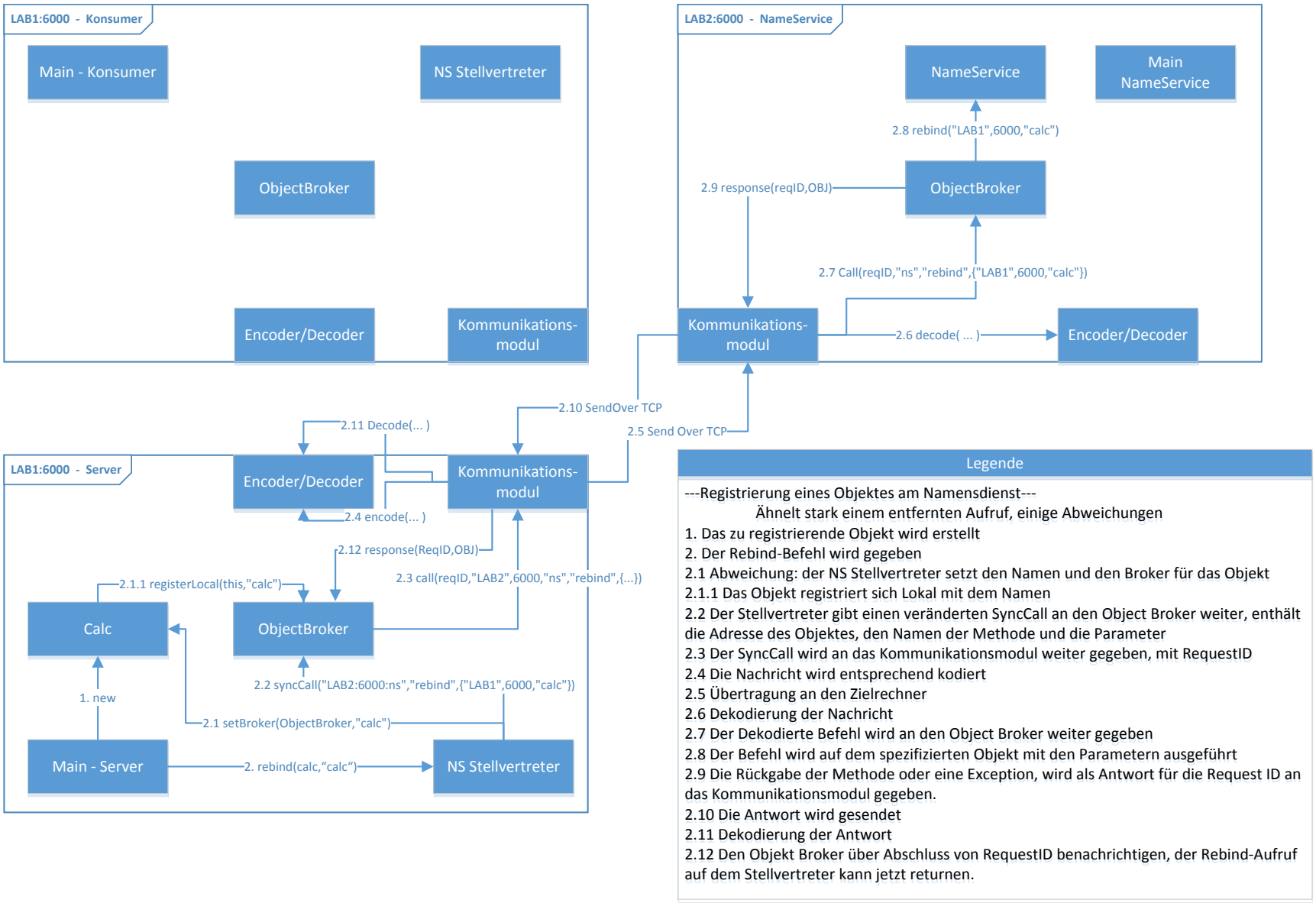


Legende

---Initialisierung des Name Services---
1. Der Nameservice wird erzeugt
2. Der Objekt Broker wird ohne Referenz auf einen Name Service initialisiert
3. Der Name Service wird nur lokal registriert

---Initialisierung eines nicht Name Services---
4. der Objekt Broker wird mit Referenz auf einen Name Service initialisiert
4.1 Der Objekt Broker erstellt einen lokalen NS Stellvertreter
5. Der Nutzer holt sich den Name Service Stellvertreter

2.3.4.2 *Registrierung am Namensdienst*

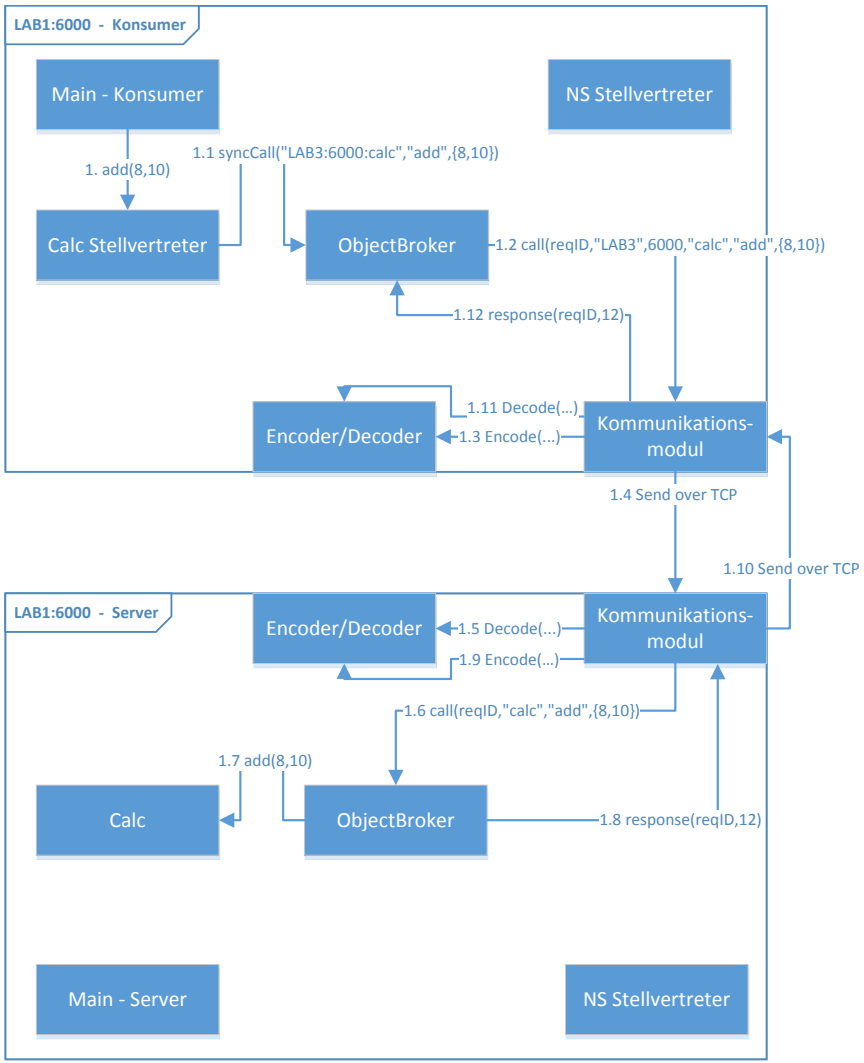


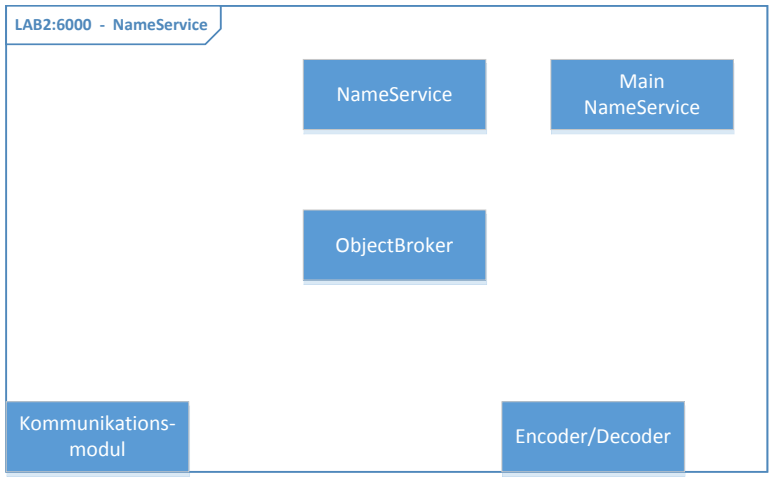
Legende

---Registrierung eines Objektes am Namensdienst---
Ähnelt stark einem entfernten Aufruf, einige Abweichungen

1. Das zu registrierende Objekt wird erstellt
2. Der Rebind-Befehl wird gegeben
2.1 Abweichung: der NS Stellvertreter setzt den Namen und den Broker für das Objekt
2.1.1 Das Objekt registriert sich Lokal mit dem Namen
2.2 Der Stellvertreter gibt einen veränderten SyncCall an den Object Broker weiter, enthält die Adresse des Objektes, den Namen der Methode und die Parameter
2.3 Der SyncCall wird an das Kommunikationsmodul weiter gegeben, mit RequestID
2.4 Die Nachricht wird entsprechend kodiert
2.5 Übertragung an den Zielrechner
2.6 Dekodierung der Nachricht
2.7 Der Dekodierte Befehl wird an den Object Broker weiter gegeben
2.8 Der Befehl wird auf dem spezifizierten Objekt mit den Parametern ausgeführt
2.9 Die Rückgabe der Methode oder eine Exception, wird als Antwort für die Request ID an das Kommunikationsmodul gegeben.
2.10 Die Antwort wird gesendet
2.11 Dekodierung der Antwort
2.12 Den Objekt Broker über Abschluss von RequestID benachrichtigen, der Rebind-Aufruf auf dem Stellvertreter kann jetzt returnen.

2.3.4.3 *Entfernter Methoden Aufruf*





Legende

---Einfacher entfernter Methodenaufruf, Namensauflösung funktioniert auch so---
1.Methodenaufruf am dem Stellvertreter
1.1 synchroner entfernter Aufruf auf dem vertretenem Objekt mit den Parametern und der Methode.
1.2 Übertragung des Aufrufes an das Kommunikationsmodul, um Request ID erweitert
1.3 Codierung für die Übertragung
1.4 Übertragung zur Zielstation
1.5 Dekodierung zur Bearbeitung
1.6 Weitergabe des Aufrufes an den Broker
1.7 Aufruf auf dem eigentlichen Objekt
1.8 Der Broker informiert über Abschluss der Bearbeitung
1.9 Kodierung der Antwortnachricht
1.10 Übertragung zum Anfragensteller
1.11 Dekodierung für Bearbeitung
1.12 Die Antwort für den Request REQID wird an den Broker Gegeben, 1.1 und somit 1. Returnen

2.3.5 Klassendiagramm

Zeigt die einzelnen Schnittstellen untereinander.