

Pflichtenheft

der Gruppe 04

zum Softwaretechnikpraktikum 2003

ShuttleTycoon [Go 04 it]

Auftraggeber: AG Softwaretechnik
Prof. Dr. Wilhelm Schäfer
Warburger Str. 100
33098 Paderborn

Auftragnehmer: Dürksen, Marcus
Feldkord, Stefan
Gawrych, Arkadius
Gebel, Alexander
Hannwacker, Dennis
Klein, Bernd Niklas
Meissner, Jochen
Niehaus, Henrik
Sülberg, Thilo
Vincenz, Axel

Betreuer: Stefan Sauer

Inhaltsverzeichnis

1.	Zielbestimmung	4
2.	Produkteinsatz	4
2.1.	Beschreibung des Problembereichs	5
2.2.	Glossar	6
3.	Produktfunktionen	7
3.1.	Use Case Diagramm „Analyse“	7
3.1.1.	Beschreibung zu: Durchschnittlichen Gewinn pro Auftrag auswerten	8
3.1.2.	Beschreibung zu: Kontostand auswerten	8
3.1.3.	Beschreibung zu: Kosten und Gewinn vergleichen	8
3.1.4.	Beschreibung zu: Aktuelle Auslastung auswerten	9
3.1.5.	Beschreibung zu: Durchschnittliche Auslastung auswerten	9
3.1.6.	Beschreibung zu: Rang in der Gesamtwertung auswerten	9
3.1.7.	Beschreibung zu: Verkehrsdichte auf den Strecken auswerten	10
3.1.8.	Beschreibung zu: Auftragshäufigkeit pro Startbahnhof auswerten	10
3.1.9.	Beschreibung zu: Datenquelle auswählen	10
3.1.10.	Beschreibung zu: Statistiken kombinieren	11
3.1.11.	Aktivitätendiagramm „Durchschnittliche Auslastung auswerten“	11
3.2.	Use Case Diagramm „Shuttlesteuerung“	12
3.2.1.	Beschreibung zu: Auftrag hinzufügen	12
3.2.2.	Beschreibung zu: Plan aktualisieren	13
3.2.3.	Beschreibung zu: Plan abarbeiten	13
3.2.4.	Beschreibung zu: Gehe zu Station	13
3.2.5.	Beschreibung zu: Repariere Shuttle	13
3.2.6.	Beschreibung zu: Belade Shuttle	14
3.2.7.	Beschreibung zu: Entlade Shuttle	14
3.2.8.	Beschreibung zu: Angebot abgeben	14
3.2.9.	Aktivitätendiagramm „Plan aktualisieren“	15
3.3.	Use Case Diagramm „Faktura“	16
3.3.1.	Beschreibung zu: Rechnung erstellen	16
3.3.2.	Beschreibung zu: Kontostand abfragen	16
3.3.3.	Beschreibung zu: Mahnprozess anstoßen	17
3.3.4.	Beschreibung zu: Kreditkartenzahlung abwickeln	17
3.3.5.	Aktivitätendiagramm „Faktura“	18
4.	Reengineering (Ist-Zustand)	19
4.1.	Der Simulations-Kernel	19
4.2.	Messages	21
4.3.	Topology	26
4.4.	RemoteObjects (RemoteObj)	27
4.5.	Kommunikation zwischen Kernel und Visualisierung (RMI)	28

5.	Grobentwurf (Soll-Zustand)	32
5.1.	Proxy	32
5.2.	Simulationsarchiv	32
5.3.	Analysemodul	33
5.4.	Visualisierungs-Plug-In	33
5.5.	Analyse-Fenster	35
5.6.	Toolbar	36
5.7.	Vorbereitung der Wegstrecken	36
5.8.	Angebotsberechnung	36
6.	Qualitätsmerkmale	37

1. Zielbestimmung

Der Auftraggeber stellt eine Simulationsumgebung zur Verfügung. Simuliert wird ein Schienennetz, auf dem sich kleine autonome Shuttle bewegen und um vom System erstellte Aufträge zur Passagierbeförderung konkurrieren. Dabei können Streckenteile ausfallen und Reparaturkosten bzw. Maut anfallen. Zudem müssen Strafen für nicht fristgerechtes Erledigen von Aufträgen gezahlt werden.

Ziel ist es, eine intelligente Steuerung für ein Shuttle zu entwickeln, das autonom auf einer vorgegebenen Strecke operiert. Es soll sich dabei gewinnorientiert verhalten.

Ein vorhandenes Programm zur Visualisierung des Schienennetzes soll erweitert werden. Dafür muss der Auftragnehmer ein Plugin für sein Shuttle implementieren, das eine dafür vorgesehene Schnittstelle erfüllt. Angezeigt werden Informationen wie Streckenplanung oder Aufträge mit Status.

Um das Shuttleverhalten analysieren zu können, soll ein Analysemodul entwickelt werden. Dieses Analysemodul stellt dann Statistiken zur Simulation und zum Shuttle zur Verfügung.

Anwender sind die Mitarbeiter des Auftraggebers, die sich mit der bereitgestellten Systemumgebung auskennen und wissen wie man eine Shuttlesteuerung in eine Simulation einbindet.

2. Produkteinsatz

Unter dem Produkteinsatz versteht man den direkten Problembereich, wo das zu entwickelnde System eingesetzt werden soll.

Als Motivation dient ein neues schienengebundenes Transportsystem auf Basis von autonom handelnden Shuttles, das im Rahmen einer Reihe neuer Forschungsprojekte und Einrichtungen an der Universität Paderborn entwickelt wird.

Die von jeder Gruppe entwickelte Lösung aus „intelligenter“ Shuttlesteuerung, Visualisierungskomponente und Analysemodul wird in ein zur Verfügung gestelltes System integriert und im Rahmen einer Simulation des gesamten Transportsystems evaluiert. Das Basis-System besteht aus einem Streckennetz, darauf operierenden Shuttles, auszuführenden Aufträgen und den Einnahmen bzw. Ausgaben eines Shuttles.

Für das Befahren einer Strecke zahlt der Shuttle eine Mautgebühr. Eine Wartung in einem Bahnhof kostet ihn zusätzliches Geld. Bei nicht ausgeführten Aufträgen muss der Shuttle eine Konventionalstrafe zahlen. Das Startkapital das ihm zu Beginn zugewiesen wird und Einnahmen aus erfolgreich ausgeführten Aufträgen stehen dem entgegen.

Das Ziel jedes Shuttles ist möglichst viel Geld einzunehmen. Ist ein Shuttle bankrott, so wird es im Bahnhof stillgelegt und scheidet aus.

2.2. Glossar

Analysemodul: Das Analysemodul soll verschiedene Statistiken präsentieren, die sich zum einen auf die Simulation, zum anderen auf Shuttles beziehen.

Aufnahmekapazität: Ein Shuttle hat ein begrenztes Fassungsvermögen, das bei der Beladung nicht überschritten werden kann.

Bearbeitungszeit: Entspricht der Ausführungszeit eines Auftrags. Beginnt mit der Vergabe eines Auftrags und legt den Termin fest, in dem ein Auftrag erfolgreich ausgeführt werden muss.

Konventionalstrafe: Wird ein Auftrag nicht termingerecht abgewickelt so wird eine Konventionalstrafe erhoben und vom jeweiligen Shuttle bezahlt.

Makler: Ein Makler bietet allen Shuttles den gleichen vom System neu erstellten Auftrag an. Er entscheidet auch über die Vergabe eines Auftrags, indem der bestbietende Shuttle von ihm den Zuschlag erhält.

Mautgebühr: Eine Mautgebühr wird von Shuttles nach dem Zurücklegen einer Verbindung im Bahnhof gezahlt. Die Kosten errechnen sich aus der Summe der Kosten der benutzten Gleisabschnitte dieser Verbindung.

Shuttle: Ein Shuttle ist ein autonom handelndes Transportmittel, das auf Schienen operiert. Hier soll es den Personenverkehr abwickeln.

Shuttlesteuerung: Bestimmt die Aktionen des Shuttles nach denen ein Shuttle operiert und die einer Strategie zu Grunde liegen.

Streckennetz: Besteht aus Bahnhöfen, Gleisabschnitten und Weichen. Auf dem Streckennetz verkehren die Shuttles.

Transportsystem: Ein System das auf Basis von schienengebundenen Shuttles den Transport von Gütern und Personen ermöglicht.

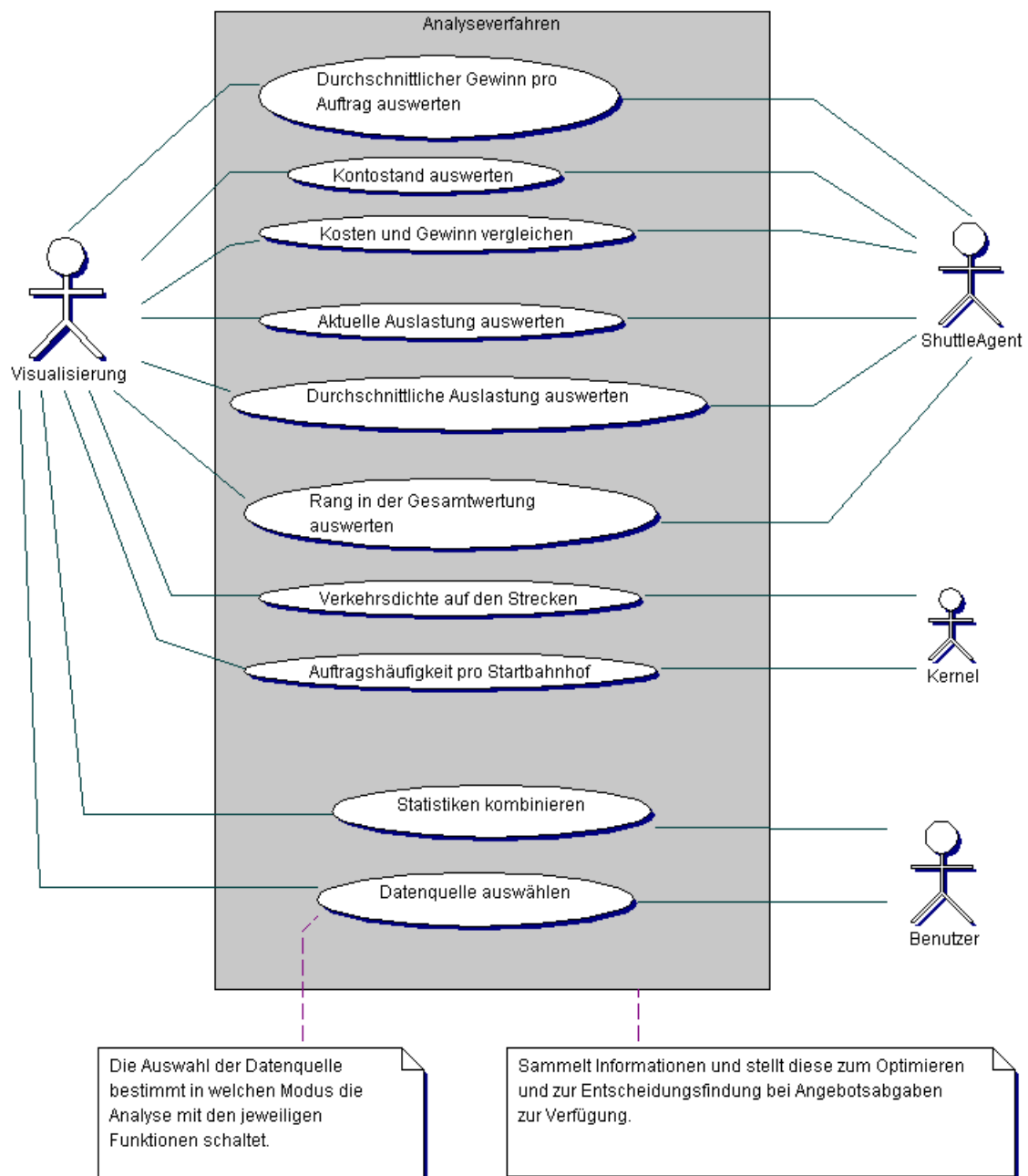
Verbindung: Ein Streckenzug bildet zwischen zwei Bahnhöfen eine Verbindung. Diese kann nur in einer Richtung befahren werden.

3. Produktfunktionen

In diesem Dokument werden die vom Produkt erwarteten Funktionalitäten beschrieben. Dabei haben wir die folgenden Systemgrenzen zugrunde gelegt:

- Analyse
- Shuttlesteuerung
- Faktura

3.1. Use Case Diagramm "Analyse"



Das Use Case Diagramm des Teilsystems Analyse umschließt das Analysemodul, in dem Daten aus verschiedenen Quellen gesammelt und ausgewertet werden. Der Benutzer kann über die Visualisierung des Analysemoduls (GUI) Statistiken abrufen und darstellen lassen. Aus diesem Grund ist der Benutzer nicht direkt, sondern über die Visualisierung mit den Aufrufen verbunden.

Grundsätzlich ist es dem User jederzeit möglich, aufgrund der vorhandenen Daten eine Statistik erstellen zu lassen. Sind noch keine Daten vorhanden, werden die nicht vorhandenen Variablen gleich null gesetzt.

3.1.1. **Beschreibung zu: Durchschnittlichen Gewinn pro Auftrag auswerten**

Charakterisierende Informationen

Ziel des Use Cases:	<i>Durchschnittlichen Gewinn ermitteln</i>
Umgebende Systemgrenze:	<i>Analyse</i>
Vorbedingung:	<i>Keine</i>
Nachbedingung bei erfolgreicher Ausführung:	<i>Durchschnittlicher Gewinn pro Auftrag ermittelt. Statistik wird ausgegeben.</i>
Beteiligte Nutzer:	<i>ShuttleAgent, Visualisierung</i>
Auslösendes Ereignis:	<i>Anfrage den Durchschnittlichen Gewinn zu ermitteln und auszugeben</i>

3.1.2. **Beschreibung zu: Kontostand auswerten**

Charakterisierende Informationen

Ziel des Use Cases:	<i>Kontostand verwalten</i>
Umgebende Systemgrenze:	<i>Analyse</i>
Vorbedingung:	<i>Keine</i>
Nachbedingung bei erfolgreicher Ausführung:	<i>Der Verlauf des Kontostandes ist verfügbar und wird ausgegeben</i>
Beteiligte Nutzer:	<i>ShuttleAgent, Visualisierung</i>
Auslösendes Ereignis:	<i>Anfrage den Verlauf des Kontostand zu ermitteln und auszugeben</i>

3.1.3. **Beschreibung zu: Kosten und Gewinn vergleichen**

Charakterisierende Informationen

Ziel des Use Cases:	<i>Gewinnermittlung</i>
Umgebende Systemgrenze:	<i>Analyse</i>
Vorbedingung:	<i>Keine</i>
Nachbedingung bei erfolgreicher Ausführung:	<i>Kosten-/Gewinnvergleich verfügbar und wird ausgegeben</i>
Beteiligte Nutzer:	<i>ShuttleAgent, Visualisierung</i>
Auslösendes Ereignis:	<i>Anfrage den Gewinn zu ermitteln und auszugeben</i>

3.1.4. Beschreibung zu: Aktuelle Auslastung auswerten

Charakterisierende Informationen

<i>Ziel des Use Cases:</i>	<i>Aktuelle Auslastung ermitteln</i>
<i>Umgebende Systemgrenze:</i>	<i>Analyse</i>
<i>Vorbedingung:</i>	<i>Keine</i>
<i>Nachbedingung bei erfolgreicher Ausführung:</i>	<i>Aktuelle Auslastung ist ermittelt und wird ausgegeben</i>
<i>Beteiligte Nutzer:</i>	<i>ShuttleAgent, Visualisierung</i>
<i>Auslösendes Ereignis:</i>	<i>Anfrage die Aktuelle Auslastung zu ermitteln und auszugeben</i>

3.1.5. Beschreibung zu: Durchschnittliche Auslastung auswerten

Charakterisierende Informationen

<i>Ziel des Use Cases:</i>	<i>Durchschnittliche Auslastung ermitteln</i>
<i>Umgebende Systemgrenze:</i>	<i>Analyse</i>
<i>Vorbedingung:</i>	<i>Keine</i>
<i>Nachbedingung bei erfolgreicher Ausführung:</i>	<i>Durchschnittliche Auslastung ist ermittelt und wird ausgegeben</i>
<i>Beteiligte Nutzer:</i>	<i>ShuttleAgent, Visualisierung</i>
<i>Auslösendes Ereignis:</i>	<i>Anfrage die Durchschnittliche Auslastung zu ermitteln und auszugeben</i>

3.1.6. Beschreibung zu: Rang in der Gesamtwertung auswerten

Charakterisierende Informationen

<i>Ziel des Use Cases:</i>	<i>Rang ermitteln</i>
<i>Umgebende Systemgrenze:</i>	<i>Analyse</i>
<i>Vorbedingung:</i>	<i>Keine</i>
<i>Nachbedingung bei erfolgreicher Ausführung:</i>	<i>Rang des Shuttle in der Gesamtwertung ist ermittelt und wird ausgegeben</i>
<i>Beteiligte Nutzer:</i>	<i>ShuttleAgent, Visualisierung</i>
<i>Auslösendes Ereignis:</i>	<i>Anfrage den Durchschnittlichen Gewinn zu ermitteln und auszugeben</i>

3.1.7. Beschreibung zu: Verkehrsdichte auf den Strecken auswerten

Charakterisierende Informationen

<i>Ziel des Use Cases:</i>	<i>Verkehrsdichte ermitteln</i>
<i>Umgebende Systemgrenze:</i>	<i>Analyse</i>
<i>Vorbedingung:</i>	<i>Keine</i>
<i>Nachbedingung bei erfolgreicher Ausführung:</i>	<i>Verkehrsdichte auf den Strecken ist ermittelt und wird ausgegeben</i>
<i>Beteiligte Nutzer:</i>	<i>Kernel, Visualisierung</i>
<i>Auslösendes Ereignis:</i>	<i>Anfrage die Verkehrsdichte zu ermitteln und auszugeben</i>

3.1.8. Beschreibung zu: Auftragshäufigkeit pro Startbahnhof auswerten

Charakterisierende Informationen

<i>Ziel des Use Cases:</i>	<i>Auftragshäufigkeit pro Startbahnhof ermitteln</i>
<i>Umgebende Systemgrenze:</i>	<i>Analyse</i>
<i>Vorbedingung:</i>	<i>Keine</i>
<i>Nachbedingung bei erfolgreicher Ausführung:</i>	<i>Auftragshäufigkeit pro Startbahnhof ist ermittelt und wird ausgegeben</i>
<i>Beteiligte Nutzer:</i>	<i>Kernel, Visualisierung</i>
<i>Auslösendes Ereignis:</i>	<i>Anfrage den Durchschnittlichen Gewinn zu ermitteln und auszugeben</i>

3.1.9. Beschreibung zu: Datenquelle auswählen

Charakterisierende Informationen

<i>Ziel des Use Cases:</i>	<i>Es wird die Herkunft der zu analysierenden Daten bestimmt</i>
<i>Umgebende Systemgrenze:</i>	<i>Analyse</i>
<i>Vorbedingung:</i>	<i>Keine</i>
<i>Nachbedingung bei erfolgreicher Ausführung:</i>	<i>Die Analyse arbeitet mit den Daten aus der gewählten Datenquelle</i>
<i>Beteiligte Nutzer:</i>	<i>Benutzer, Visualisierung</i>
<i>Auslösendes Ereignis:</i>	<i>Anfrage die Analyse auf einer Datenquelle auszuführen</i>

3.1.10. Beschreibung: Statistiken kombinieren

Charakterisierende Informationen

Ziel des Use Cases:	<i>Benutzer kann Statistiken miteinander kombinieren</i>
Umgebende Systemgrenze:	<i>Analyse</i>
Vorbedingung:	<i>Keine</i>
Nachbedingung bei erfolgreicher Ausführung:	<i>Liefert kombinierte Statistiken und stellt diese dar</i>
Beteiligte Nutzer:	<i>Benutzer, Visualisierung</i>
Auslösendes Ereignis:	<i>Anfrage des Benutzer von ihm ausgewählte Statistiken zu kombinieren</i>

3.1.11. Aktivitätendiagramm "Durchschnittliche Auslastung auswerten"

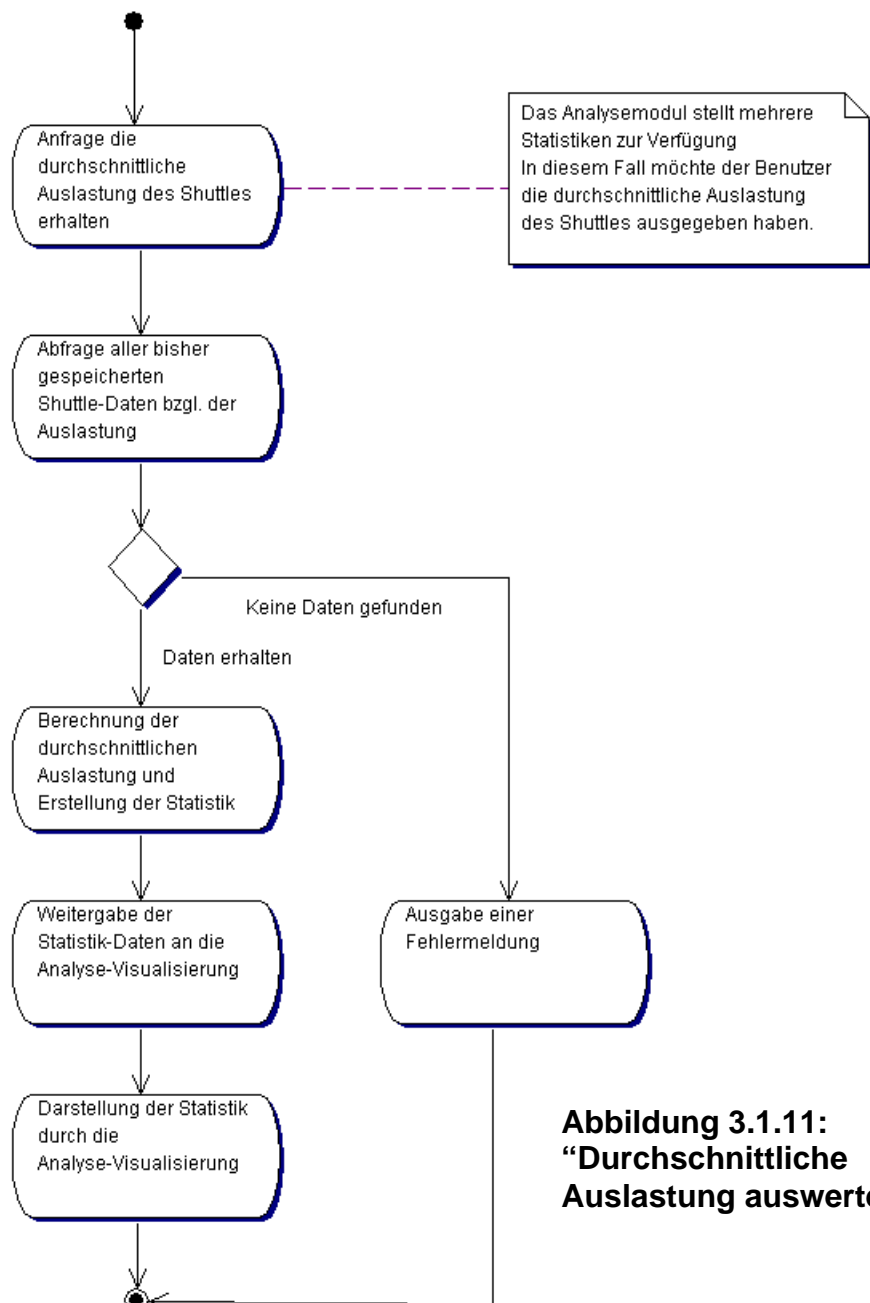
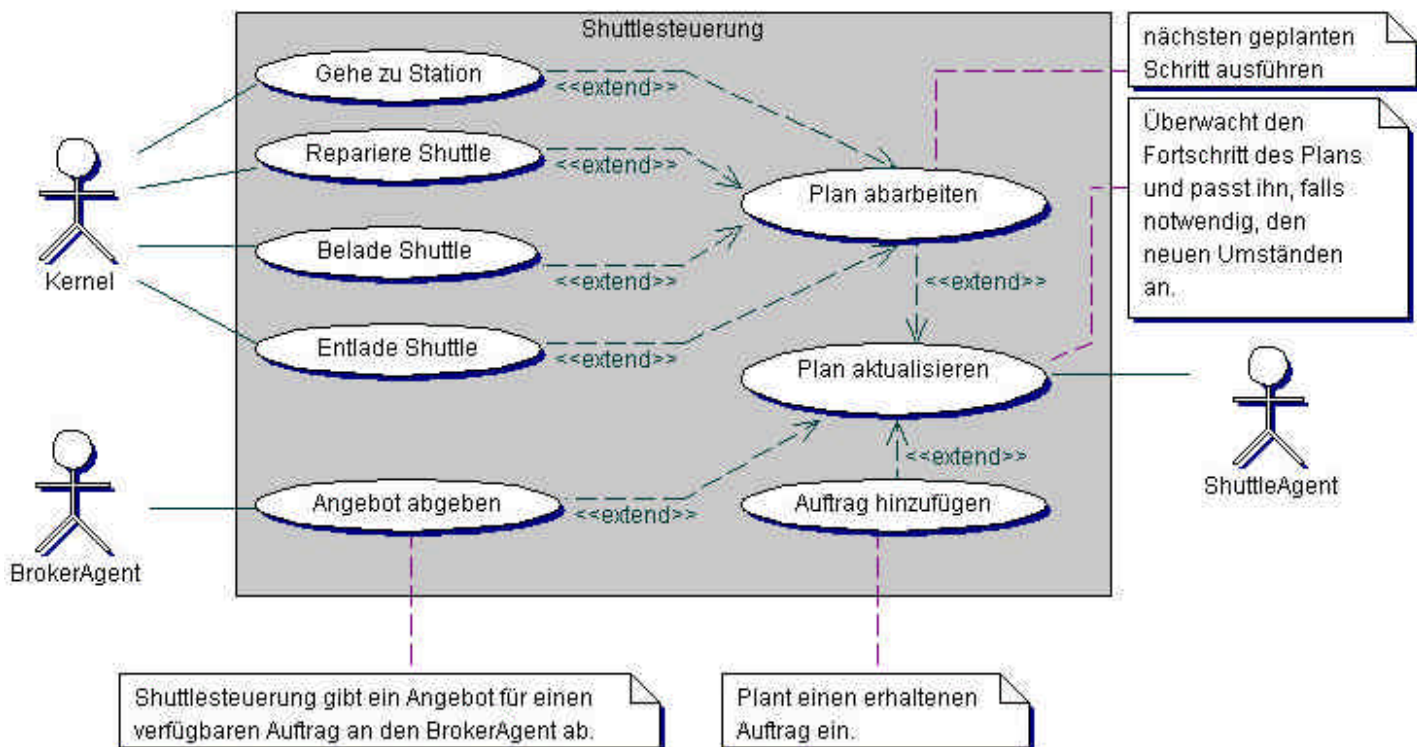


Abbildung 3.1.11:
"Durchschnittliche Auslastung auswerten"

Das Analysemodul hat die Anfrage erhalten, die durchschnittliche Auslastung des Shuttles auszugeben. Es lädt nun alle verfügbaren und für die Berechnung relevanten Daten. Sollte es dem Modul nicht möglich sein, Daten zu laden führt dies zu einer Fehlermeldung und zur Beendigung der Anfrage. Sollten Daten gefunden werden, berechnet die Analyse nun selbstständig für die Statistik alle notwendigen Parameter und versendet diese dann an seine eigene Visualisierung in der die Statistik dann visuell ausgegeben wird.

3.2. Use Case Diagramm "Shuttlesteuerung"



Das Use Case Diagramm der Shuttlesteuerung beschreibt die internen Shuttleroutinen. Die wichtigste Funktion "Plan aktualisieren" wird nachfolgend durch ein Aktivitäts-Diagramm (Abbildung 3.2.9) genauer erläutert.

3.2.1 Beschreibung zu: Auftrag hinzufügen

Charakterisierende Informationen

Ziel des Use Cases:	Erhaltenen Auftrag einplanen
Umgebende Systemgrenze:	Shuttlesteuerung
Vorbedingung:	Keine
Nachbedingung bei erfolgreicher Ausführung:	Auftrag eingeplant
Beteiligte Nutzer:	Keine
Auslösendes Ereignis:	Auftrag erhalten

3.2.2. Beschreibung zu: Plan aktualisieren

Charakterisierende Informationen

<i>Ziel des Use Cases:</i>	<i>Überwachung und Anpassung des Planes</i>
<i>Umgebende Systemgrenze:</i>	<i>Shuttlesteuerung</i>
<i>Vorbedingung:</i>	<i>Keine</i>
<i>Nachbedingung bei erfolgreicher Ausführung:</i>	<i>Plan wurde den Parametern entsprechend angepasst</i>
<i>Beteiligte Nutzer:</i>	<i>ShuttleAgent</i>
<i>Auslösendes Ereignis:</i>	<i>Plan fortgeschritten o. Änderung der Umstände</i>

3.2.3. Beschreibung zu: Plan abarbeiten

Charakterisierende Informationen

<i>Ziel des Use Cases:</i>	<i>Nächsten geplanten Schritt durchführen</i>
<i>Umgebende Systemgrenze:</i>	<i>Shuttlesteuerung</i>
<i>Vorbedingung:</i>	<i>Es existiert ein geplanter Schritt</i>
<i>Nachbedingung bei erfolgreicher Ausführung:</i>	<i>Der geplante Schritt wurde durchgeführt und aus dem Plan gelöscht, bzw. als abgearbeitet markiert</i>
<i>Beteiligte Nutzer:</i>	<i>Keine</i>
<i>Auslösendes Ereignis:</i>	<i>Vorheriger Schritt im Plan durchgeführt</i>

3.2.4. Beschreibung zu: Gehe zu Station

Charakterisierende Informationen

<i>Ziel des Use Cases:</i>	<i>Shuttle zu Station bewegen</i>
<i>Umgebende Systemgrenze:</i>	<i>Shuttlesteuerung</i>
<i>Vorbedingung:</i>	<i>Bewegung zu Station ist aktuell geplanter Schritt</i>
<i>Nachbedingung bei erfolgreicher Ausführung:</i>	<i>Shuttle hat Station erreicht</i>
<i>Beteiligte Nutzer:</i>	<i>Kernel</i>
<i>Auslösendes Ereignis:</i>	<i>Keine</i>

3.2.5. Beschreibung zu: Repariere Shuttle

Charakterisierende Informationen

<i>Ziel des Use Cases:</i>	<i>Reparieren des Shuttles</i>
<i>Umgebende Systemgrenze:</i>	<i>Shuttlesteuerung</i>
<i>Vorbedingung:</i>	<i>Reparieren des Shuttles ist aktuell geplanter Schritt und Shuttle ist in einer Station</i>
<i>Nachbedingung bei erfolgreicher Ausführung:</i>	<i>Shuttle repariert</i>
<i>Beteiligte Nutzer:</i>	<i>Kernel</i>
<i>Auslösendes Ereignis:</i>	<i>Keine</i>

3.2.6. Beschreibung zu: Belade Shuttle

Charakterisierende Informationen

Ziel des Use Cases:	<i>Beladen des Shuttles gemäß Auftrag</i>
Umgebende Systemgrenze:	<i>Shuttlesteuerung</i>
Vorbedingung:	<i>Beladen des Shuttles ist aktuell geplanter Schritt, Shuttle ist in richtiger Station und Shuttle hat genug freie Plätze</i>
Nachbedingung bei erfolgreicher Ausführung:	<i>Shuttle beladen</i>
Beteiligte Nutzer:	<i>Kernel</i>
Auslösendes Ereignis:	<i>Keine</i>

3.2.7. Beschreibung zu: Entlade Shuttle

Charakterisierende Informationen

Ziel des Use Cases:	<i>Entladen des Shuttles gemäß Auftrag</i>
Umgebende Systemgrenze:	<i>Shuttlesteuerung</i>
Vorbedingung:	<i>Entladen des Shuttles ist aktuell geplanter Schritt und Shuttle ist in richtiger Station</i>
Nachbedingung bei erfolgreicher Ausführung:	<i>Shuttle entladen</i>
Beteiligte Nutzer:	<i>Kernel</i>
Auslösendes Ereignis:	<i>Keine</i>

3.2.8. Beschreibung zu: Angebot abgeben

Charakterisierende Informationen

Ziel des Use Cases:	<i>Angebot für verfügbaren Auftrag abgeben</i>
Umgebende Systemgrenze:	<i>Shuttlesteuerung</i>
Vorbedingung:	<i>Auftrag positiv bewertet</i>
Nachbedingung bei erfolgreicher Ausführung:	<i>Angebot abgegeben</i>
Beteiligte Nutzer:	<i>BrokerAgent</i>
Auslösendes Ereignis:	<i>Auftrag verfügbar</i>

3.2.9. Aktivitätendiagramm “Plan aktualisieren”

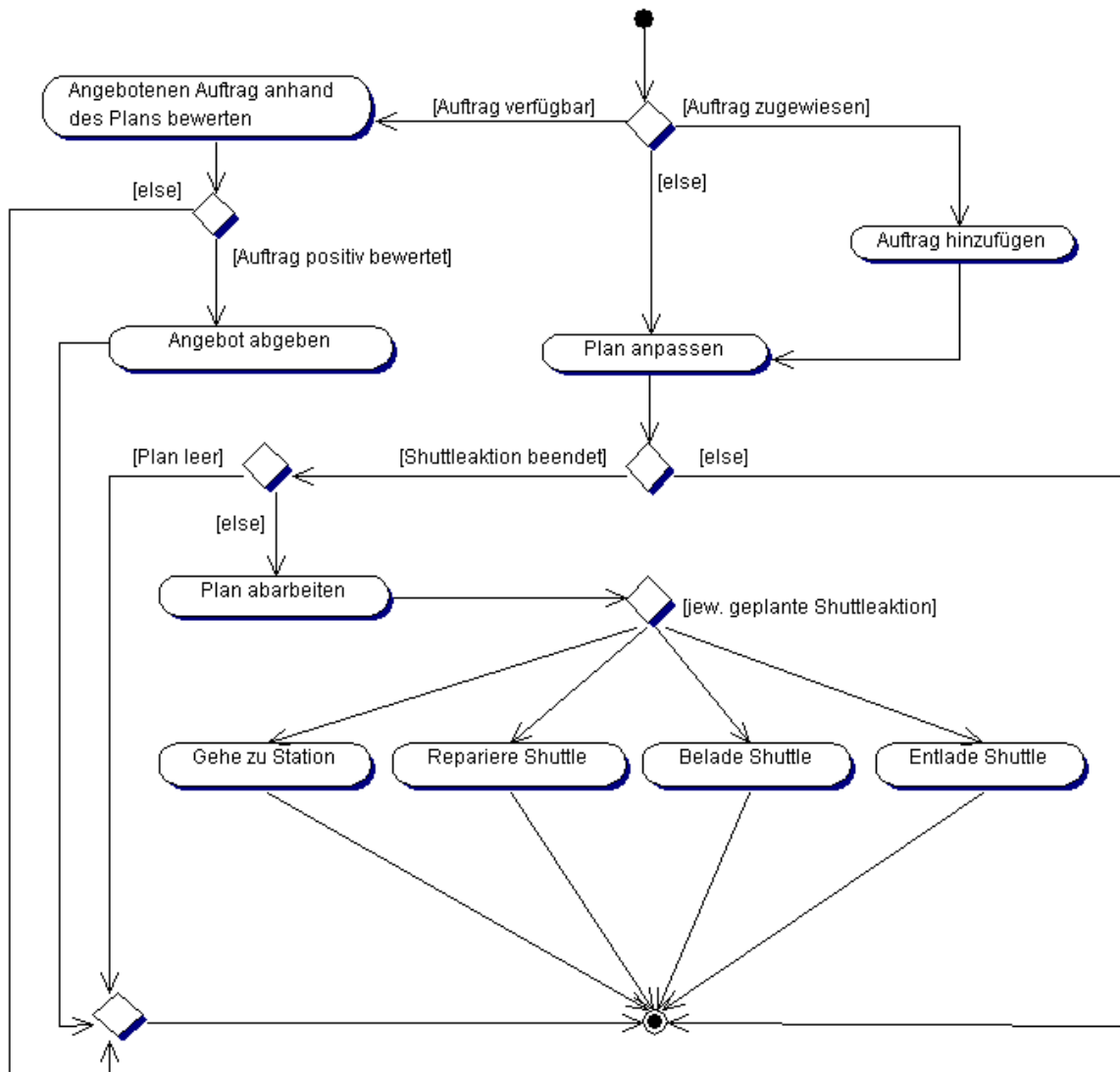


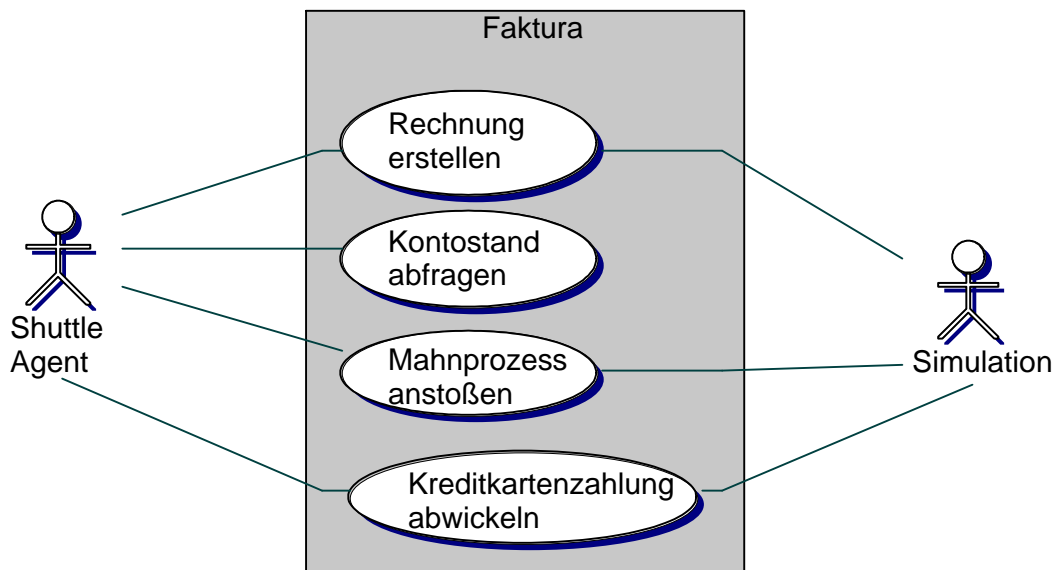
Abbildung 3.2.9: “Plan aktualisieren”

Der ShuttleAgent hat eine den Plan betreffende Meldung erhalten und will daraufhin den „Plan aktualisieren“. Zu unterscheiden sind drei Arten von Meldungen:

1. Auftrag verfügbar,
2. Auftrag zugewiesen,
3. Shuttleaktion ausgeführt.

Wenn ein Auftrag verfügbar ist, wird er anhand des Plans bewertet und – bei positivem Entscheid – ein Angebot dafür abgegeben. Wenn ein Auftrag zugewiesen wurde, wird er in die Liste der aktuell zu bearbeitenden Aufträge eingefügt und der Plan daraufhin angepasst. Meldungen, die den Fortschritt von Shuttleaktionen und damit des Plans anzeigen, stoßen bei erfolgreicher Ausführung einer Shuttleaktion den nächsten Schritt im Plan an.

3.3. Use Case Diagramm "Faktura"



Die Faktura beschreibt Prozess der Zahlungsabwicklung. "Mahnprozess anstoßen" und "Kreditkartenzahlung abwickeln" werden nach noch zu erfolgender Spezifizierung durch den Auftraggeber weiterentwickelt und existieren zur Zeit in Rohform.

3.3.1. Beschreibung zu: Rechnung erstellen

Charakterisierende Informationen

Ziel des Use Cases:	<i>Erstellung einer Rechnung</i>
Umgebende Systemgrenze:	<i>Faktura</i>
Vorbedingung:	<i>Auftrag ausgeführt, Rechnung als Zahlart gewählt</i>
Nachbedingung bei erfolgreicher Ausführung:	<i>Rechnung wurde erstellt und versandt</i>
Beteiligte Nutzer:	<i>ShuttleAgent, Kernel</i>
Auslösendes Ereignis:	<i>Anfrage eine Rechnung zu erstellen</i>

3.3.2. Beschreibung zu: Kontostand abfragen

Charakterisierende Informationen

Ziel des Use Cases:	<i>Aktuellen Kontostand ermitteln</i>
Umgebende Systemgrenze:	<i>Faktura</i>
Vorbedingung:	<i>Keine</i>
Nachbedingung bei erfolgreicher Ausführung:	<i>Aktueller Kontostand ist ermittelt</i>
Beteiligte Nutzer:	<i>ShuttleAgent</i>
Auslösendes Ereignis:	<i>Anfrage den aktuellen Kontostand zu ermitteln</i>

3.3.3. Beschreibung zu: Mahnprozess anstoßen

Charakterisierende Informationen

Ziel des Use Cases:	<i>Mahnung an Rechnungsempfänger senden</i>
Umgebende Systemgrenze:	<i>Faktura</i>
Vorbedingung:	<i>Rechnung verschickt, Zahlung nicht erfolgt</i>
Nachbedingung bei erfolgreicher Ausführung:	<i>Mahnung wurde versendet</i>
Beteiligte Nutzer:	<i>ShuttleAgent, Kernel</i>
Auslösendes Ereignis:	<i>Anfrage den Mahnprozess auszuführen</i>

3.3.4. Beschreibung zu: Kreditkartenzahlung abwickeln

Charakterisierende Informationen

Ziel des Use Cases:	<i>Zahlung per Kreditkarte abwickeln</i>
Umgebende Systemgrenze:	<i>Faktura</i>
Vorbedingung:	<i>Auftrag ausgeführt, Kreditkarte als Zahlverfahren gewählt</i>
Nachbedingung bei erfolgreicher Ausführung:	<i>Verfahren durchgeführt, Zahlung ist erfolgt</i>
Beteiligte Nutzer:	<i>ShuttleAgent, Kernel</i>
Auslösendes Ereignis:	<i>Anfrage den Kreditkartenabwicklungs-Prozess durchzuführen</i>

3.3.5. Aktivitätendiagramm „Faktura“

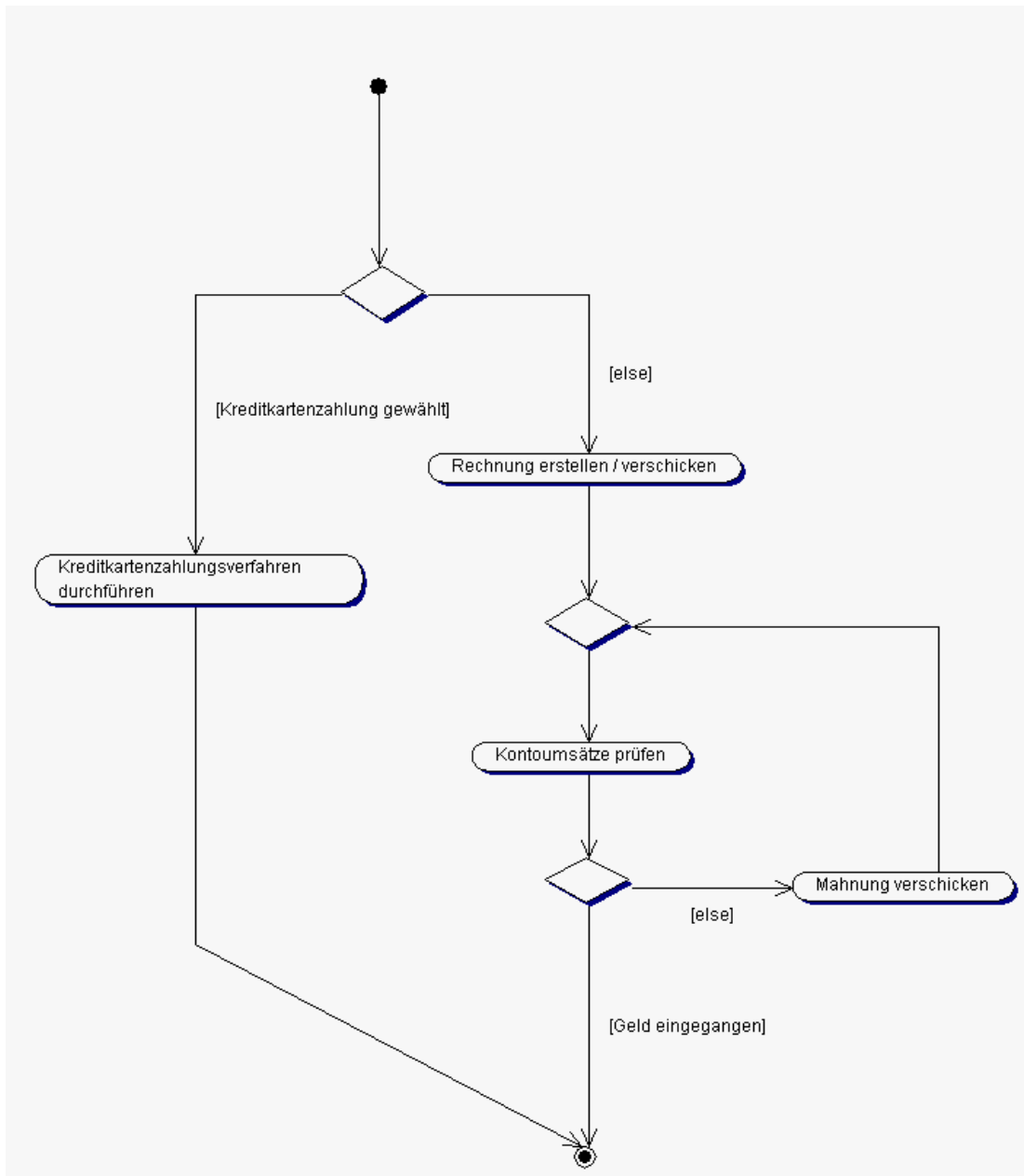


Abbildung 3.3.5: „Faktura“

In dem oben zu sehenden Aktivitätendiagramm kann man den geplanten Ablauf für die Faktura betrachten. Der Kunde hat die Möglichkeit zu wählen, ob er die Zahlung per Rechnung oder per Kreditkartenzahlung leisten möchte. Entscheidet sich der Kunde für die Kreditkartenzahlung, wird der Betrag von seiner Kreditkarte abgebucht und der Vorgang ist beendet. Entscheidet er sich allerdings für die Bezahlung per Rechnung, so wird vom Shuttle eine Rechnung für den Kunden erstellt und danach an ihn verschickt. Danach prüft das Shuttle, ob der zu entrichtende Betrag auf dem Konto eingegangen ist. Ist dies nicht der Fall, so wird vom Shuttle eine Mahnung an den Kunden verschickt. Dieser Vorgang wiederholt sich, bis der Kunde den Betrag letztendlich bezahlt hat. Sobald dies passiert ist, ist der Vorgang beendet. Das Diagramm wurde auf Basis der zur Verfügung stehenden Daten erstellt. Erweiterungen erfolgen nach Rücksprache mit dem Auftraggeber.

4. Reengineering (Ist-Zustand)

4.1. Der Simulations-Kernel

Ein Überblick über die wichtigsten Klassen und deren Funktionen.

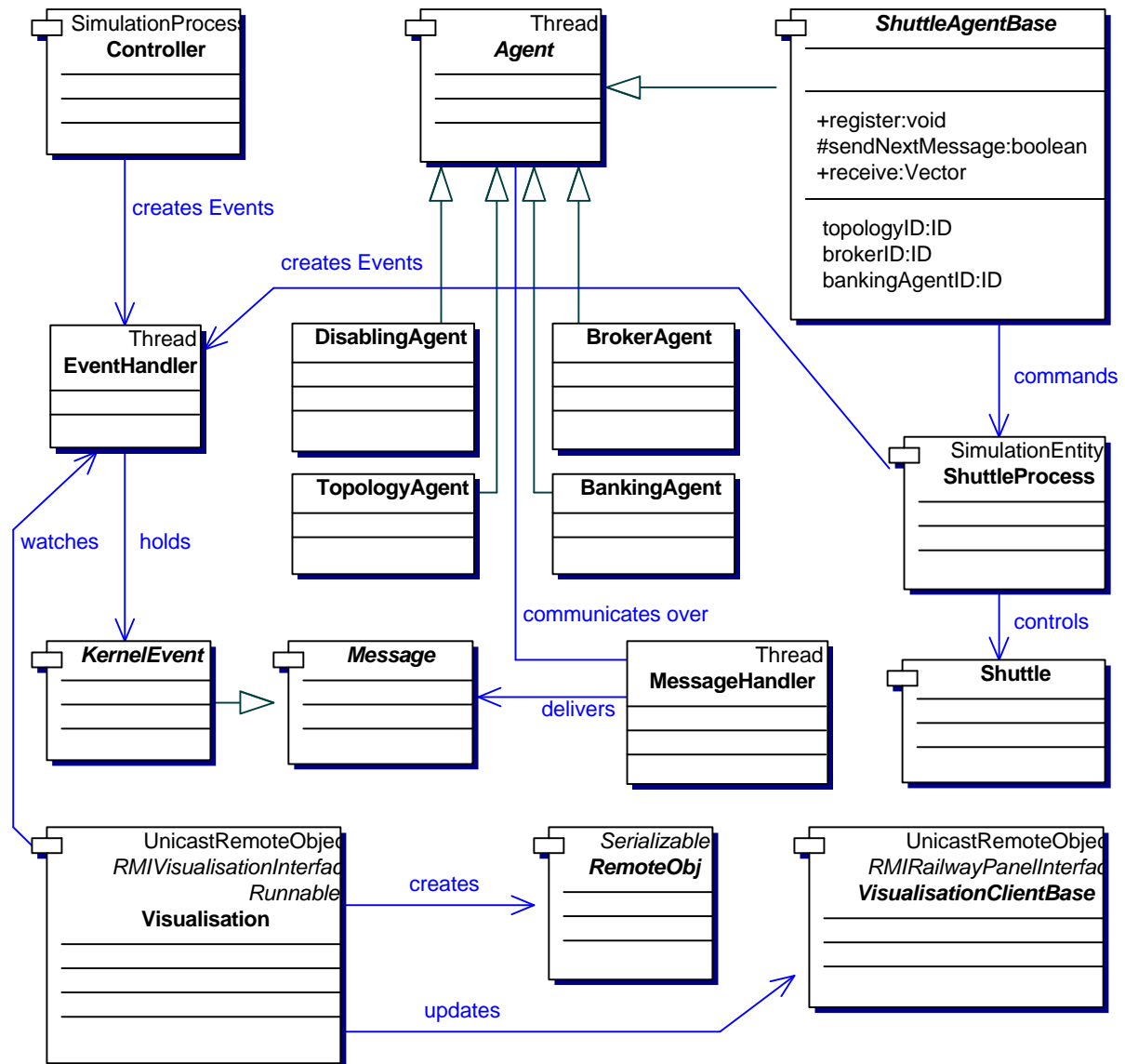


Abbildung 4.1.1: Klassendiagramm „Simulations-Kernel“

Controller

Der Controller initialisiert, steuert und überwacht die laufende Simulation. Auf Grund von Messages der Agents erzeugt er KernelEvents im EventHandler.

Agenten

Die Agenten kapseln jeweils Funktionsbereiche der Simulation:

- Der BrokerAgent ist für die Auftragsvergabe zuständig. Er erstellt Aufträge, bietet diese den Shuttles an und verteilt sie jeweils an den Shuttle mit dem niedrigsten Gebot.

- Der BankingAgent übernimmt die Zahlungsabwicklung.
- Der DisablingAgent ist dafür zuständig, Streckenteile temporär ausfallen zu lassen.
- Der TopologyAgent stellt die Topologie zur Verfügung.
- ShuttleAgentBase stellt die Schnittstelle zwischen dem Kernel und den realisierten Shuttlesteuerungen dar.

Agenten agieren, indem sie mithilfe von Messages über den MessageHandler kommunizieren.

MessageHandler

Der Messagehandler übernimmt die Verteilung von Messages der bei ihm registrierten Kommunikationspartner.

ShuttleAgentBase (ShuttleAgent)

Diese Klasse muss von der zu realisierenden Shuttlesteuerung erweitert werden. Sie befehligt mithilfe von Messages den ShuttleProcess, gibt Angebote an den BrokerAgent ab, kann die Topologie beim TopologyAgent abfragen oder mithilfe des BankingAgent Abrechnungen tätigen (siehe Sequenzdiagramm 4.2.2).

ShuttleProcess

Steuert die logische Repräsentation des Shuttles auf Grundlage der von ShuttleAgentBase verschickten Messages und erzeugt entsprechend KernelEvents im EventHandlerler.

EventHandlerler

Verwaltet die erzeugten KernelEvents und dokumentiert damit den bisherigen Simulationsverlauf.

Visualisation

Beobachtet den EventHandlerler, erstellt RemoteObj basierend auf den dort erzeugten KernelEvents, verschickt diese über RMI an die angedockten Visualisierungen (VisualisationClients) und hält sie damit über den Simulationsverlauf auf dem aktuellen Stand.

VisualisationClientBase

Diese Klasse stellt die Schnittstelle zwischen Kernel und Visualisierung dar und wird von der vorliegenden Visualisierung erweitert.

4.2. Messages

Messages (siehe Abbildung 4.2.1) sind das Mittel zur Kommunikation zwischen den Agents. Aus der Sicht vom ShuttleAgent gibt es dabei folgende Messages, die er empfangen bzw. verschicken kann:

Vom Shuttle zu sendende Messages:

sendable SimulationRequests	Anfragen vom Shuttle an die Simulation.
ActivateDeadManSwitch	Antwort auf "PleaseActivateDeadManSwitch" (siehe unten)
ShuttleCommand	Vaterklasse für alle vom Shuttle versendeten Steuerungskommandos.
MoveShuttle	Erbt von „ShuttleActionCommand“ und veranlasst die Simulation, das Shuttle zu bewegen.
GetTopology	Anfrage zur Topologie, erwartet als Antwort „TopologyInformation“.
MakeOffer	Dient zum Versenden von Angeboten
Other ShuttleCommands	Weitere Shuttle-Kommandos zur Steuerung von Transport und anderen Aufgaben.
Payment	Enthält später Nachrichten zu verschiedenen Zahlungsmethoden

Vom Shuttle zu empfangene Messages:

Receivable SimulationResponses	Enthält die Antworten auf „SimulationRequests“ (siehe oben)
PleaseActivateDeadManSwitch	Anfrage
ShuttleResponse	Vaterklasse für alle vom Shuttle empfangenen Steuerungskommandos
ShuttleMovingMessage	Rückmeldung über Shuttlebewegung
Other ShuttleResponses	Weitere Antworten auf die Anfragen aus „other ShuttleCommands“ (siehe oben)
TopologyInformation	Daten zur Topologie
AssignOrder	Nachricht, dass dieser Shuttle den Zuschlag bekommen hat
OrderAvailable	Nachricht, dass ein neuer Auftrag zur Vergabe bereitsteht

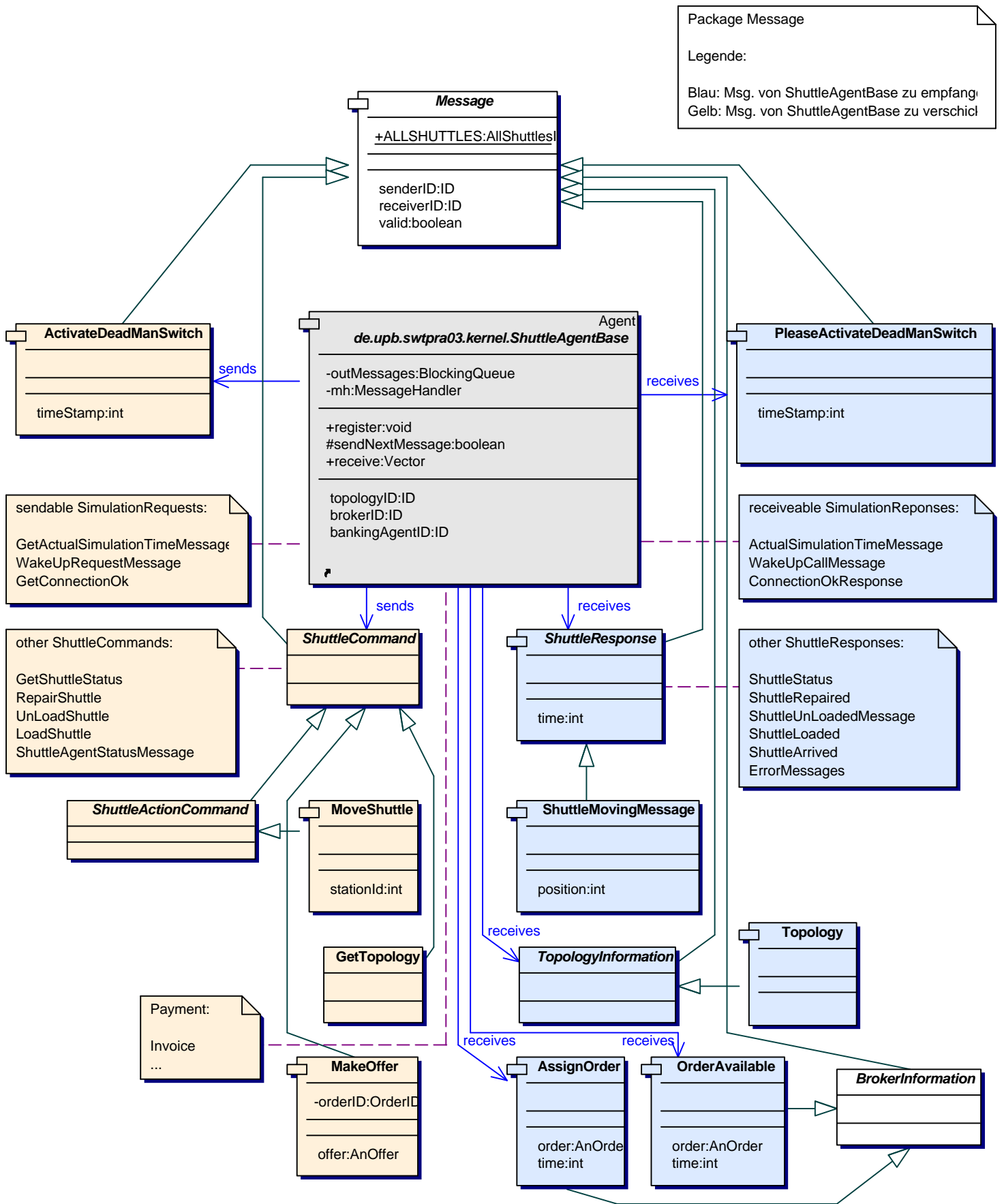
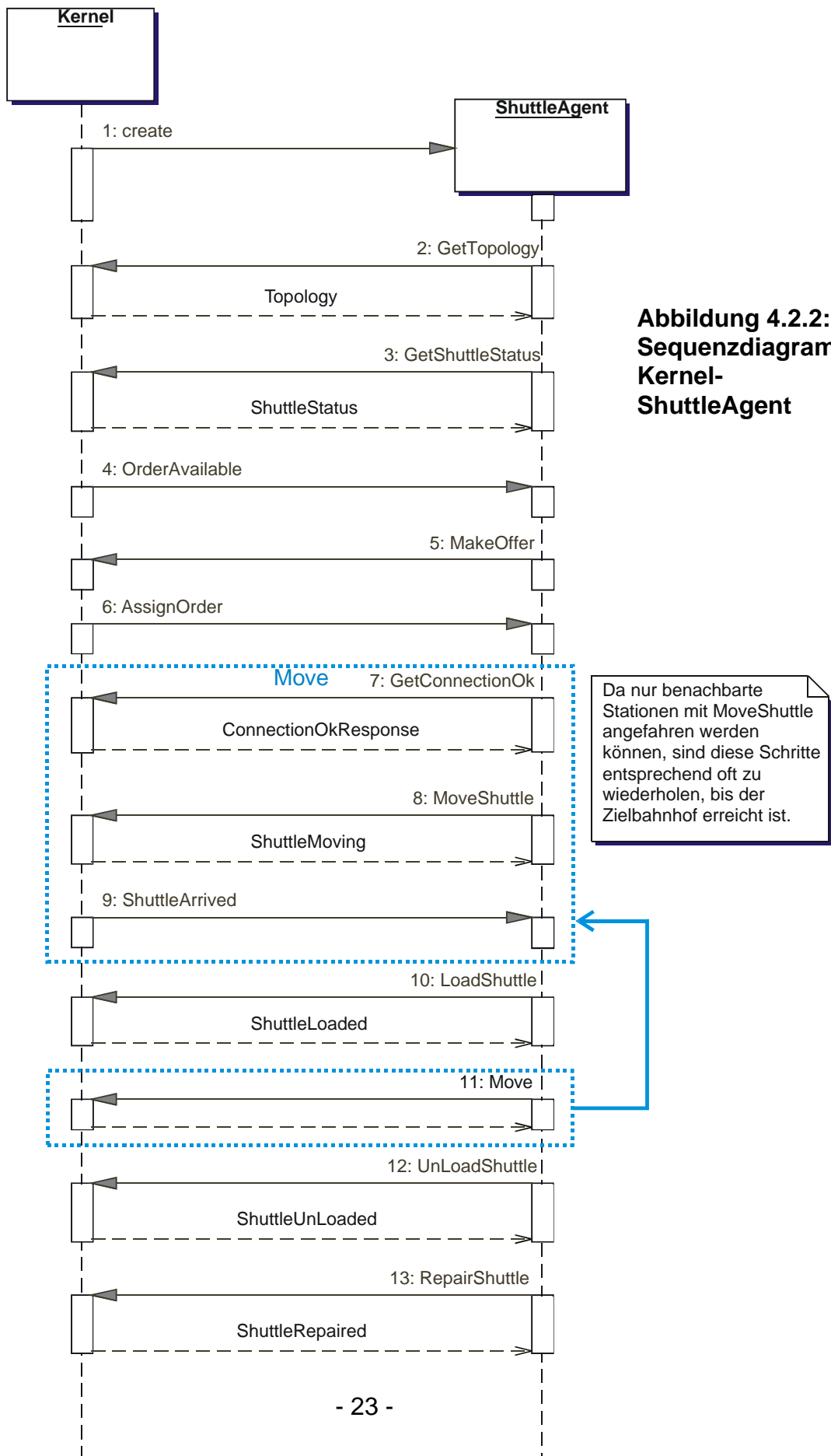


Abbildung 4.2.1: Klassen-Diagramm „Messages“

Standardszenario der Kommunikation zwischen Kernel und ShuttleAgent mithilfe von Messages:



Die Kommunikation zwischen Kernel und ShuttleAgent (siehe Abbildung 4.2.2) lässt sich dabei in folgende Phasen aufteilen:

Initialisierung:

Nachdem der ShuttleAgent zum Simulationsbeginn vom Kernel (Controller) erzeugt wurde, holt sich dieser den Streckenplan (Topology) und seinen aktuellen Status.

Auftragsverhandlung:

Nachdem der Kernel (BrokerAgent) dem ShuttleAgent einen Auftrag angeboten hat, gibt dieser ein Gebot für diesen Auftrag ab.

Auftragsabarbeitung:

Hat der ShuttleAgent den Zuschlag erhalten, kann er mit der Abarbeitung dieses Auftrags beginnen.

Verfolgung einer ShuttleActionCommand-Message

Messages vom Typ ShuttleActionCommand werden vom ShuttleAgent an den Kernel geschickt, um dem Shuttle einen Befehl zu erteilen (z.B.: MoveShuttle). Diese Messages müssen an den ShuttleProcess weitergeleitet werden, der dann diesen Befehl auf dem Shuttle ausführt.

Das folgende Sequenzdiagramm (siehe Abbildung 4.2.3) zeigt den Weg, den eine solche Message durch den Kernel nimmt, anhand des Beispiels der MoveShuttle-Message auf, die verschickt wird, um das Shuttle aus einer Station zu bewegen.

Erklärung zum Sequenzdiagramm (siehe Abbildung 4.2.3):

Der ShuttleAgent verschickt die MoveShuttle-Message über den MessageHandler an den Controller, der sie an den ShuttleProcess weiterleitet. Dieser ruft auf dem Shuttle die Methode moveShuttle(destinationStation) auf, um das Shuttle aus der Station zu bewegen. Bei erfolgreicher Ausführung wird vom ShuttleProcess eine MovingShuttleMessage über den MessageHandler an den ShuttleAgent geschickt.

Shuttle aus Station bewegen



Abbildung 4.2.3: Sequenzdiagramm „MoveShuttle“

4.3. Topology

Die TopologyData beschreibt das Streckennetz auf dem sich die Shuttles bewegen. Sie besteht aus TopologyDataObjects, die die Bahnhöfe (Station), Gleisabschnitte (Track) und Weichen (Switch) repräsentieren. Der TopologyAgent hält das TopologyDataObject für die an der Simulation beteiligten ShuttleAgents zum Abruf bereit.

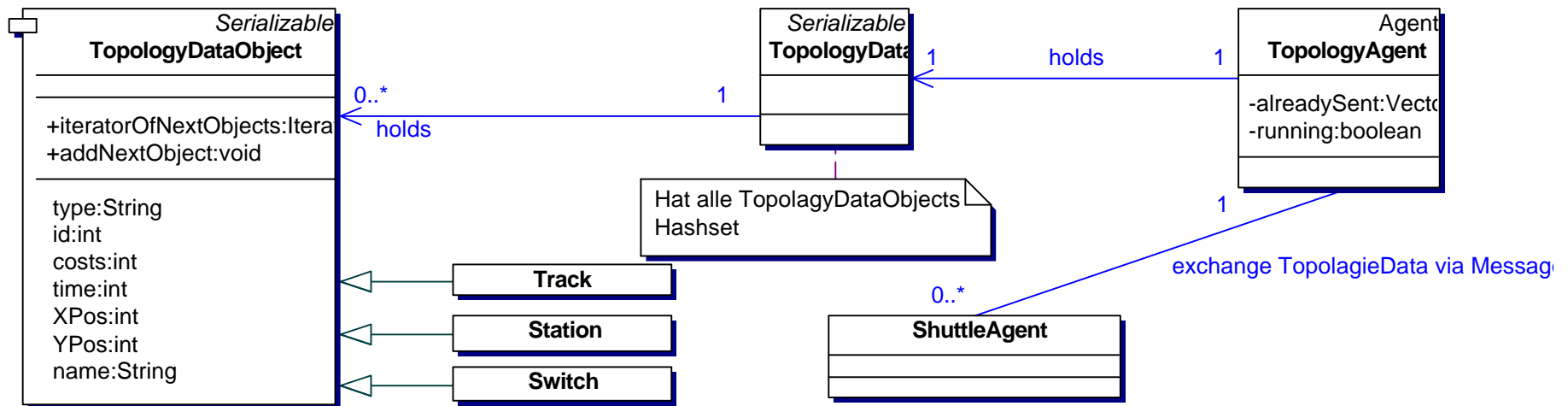


Abbildung 4.3.1: Klassendiagramm „Topology“

4.4. RemoteObjects (RemoteObj)

RemoteObjects werden vom Kernel an die Visualisierung geschickt, um diese über den Simulationsverlauf zu informieren. Die Visualisierung wiederum leitet einen Teil der empfangenen RemoteObj an das aktive Plugin weiter. Welche RemoteObj von PluginBase, der Schnittstelle zwischen Visualisierung und Plugin, zu empfangen sind, zeigt folgendes Diagramm:

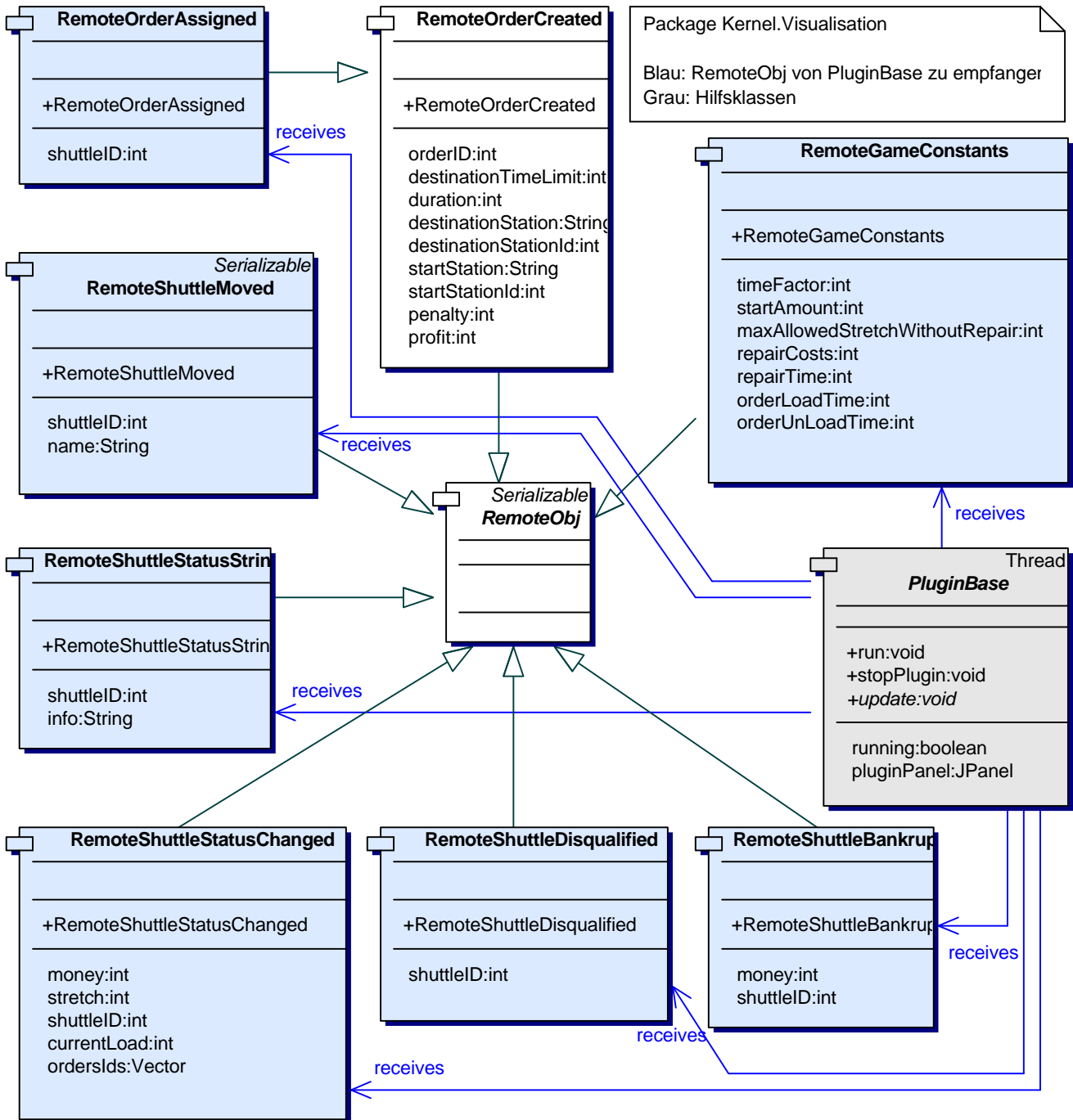


Abbildung 4.4.1: Klassendiagramm „RemoteObjects“

RemoteObj und ihre Funktionen

RemoteGameConstants	Übermittelt die globalen Simulationsparameter
RemoteShuttleMoved	Übermittelt die aktuelle Position des Shuttle
RemoteShuttleStatusString	Übermittelt eine Zeichenkette, die mit einer ShuttleAgnestStausMessage verschickt wurde
RemoteShuttleStatusChanged	Übermittelt den aktuellen Status des Shuttle
RemoteShuttleDisqualified	Wird gesendet, wenn ein Shuttle disqualifiziert wird
RemoteShuttleBankrupt	Wird gesendet, wenn ein Shuttle Bankrott ist
RemoteOrderAssigned	Dient der Übermittlung der Auftragsdaten nach erfolgter Vergabe

Nicht im Diagramm (siehe Abbildung 4.4.1) enthalten und von PluginBase nicht zu empfangen:

RemoteEndGame	Wird bei Beendigung der Simulation gesendet
RemoteTopologyData	Übermittelt die Topologie
RemoteOrderDelete	Übermittelt gelöschte Aufträge

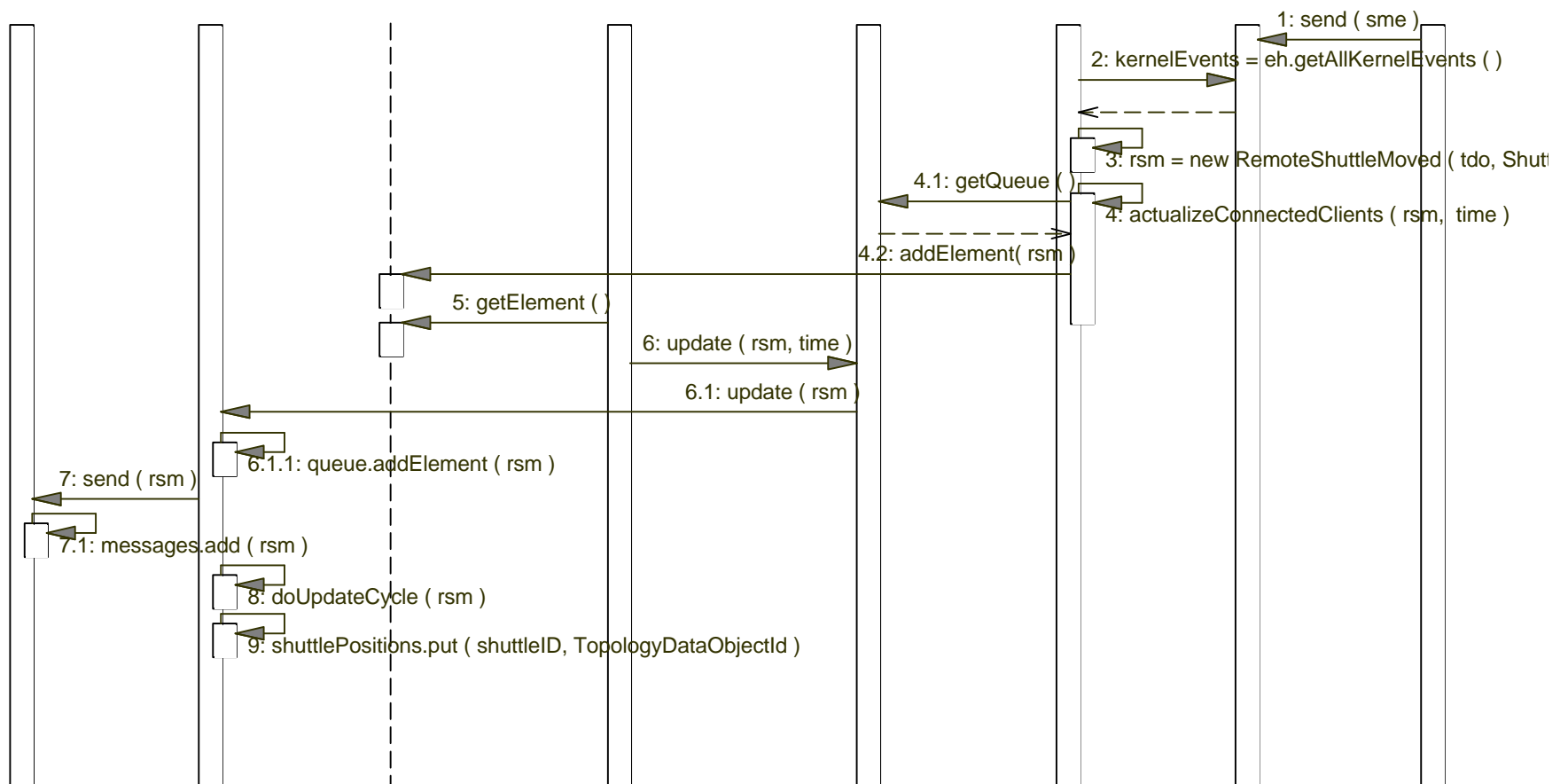
4.5. Kommunikation zwischen Kernel und Visualisierung

Darstellung anhand der Verfolgung eines ShuttleMovedEvents:
Sequenzdiagramm auf der nächsten Seite.

Vom ShuttleProcess wird das ShuttleMovedEvent (sme) an den EventHandlerer gesendet. Der Visualisation-Thread fragt in regelmäßigen Abständen alle KernelEvents ab und erzeugt je nach Event-Typ ein entsprechendes RemoteObject (hier: sme vom Typ ShuttleMovedEvent nach rsm vom Typ RemoteShuttleMoved).

Danach werden alle verbundenen Clients mit der Methode actualizeConnectedClients(rsm, time) aktualisiert. Dazu hinterlegt der Visualisation-Thread das rsm-Objekt in der RemoteQueue des Clients. Der RemoteMessageHandler holt das Objekt ab und informiert den VisualisationClient mit update(rsm,time).

Der VisualisationClient leitet das rsm-Objekt mit update(rsm) an LUpdateData weiter. LUpdateData, das alle notwendigen Informationen für die Visualisierung enthält, legt das Objekt in einer Schlange ab und sendet es zur Aktualisierung des aktiven Plugins an den PluginController. Mit doUpdateCycle(rsm) wird je nach Typ des RemoteObjects das Event weiterverarbeitet. In unserem Beispiel wird die Shuttle-Position angepasst.



RMI-Verbindungsaufbau zwischen Client und Server

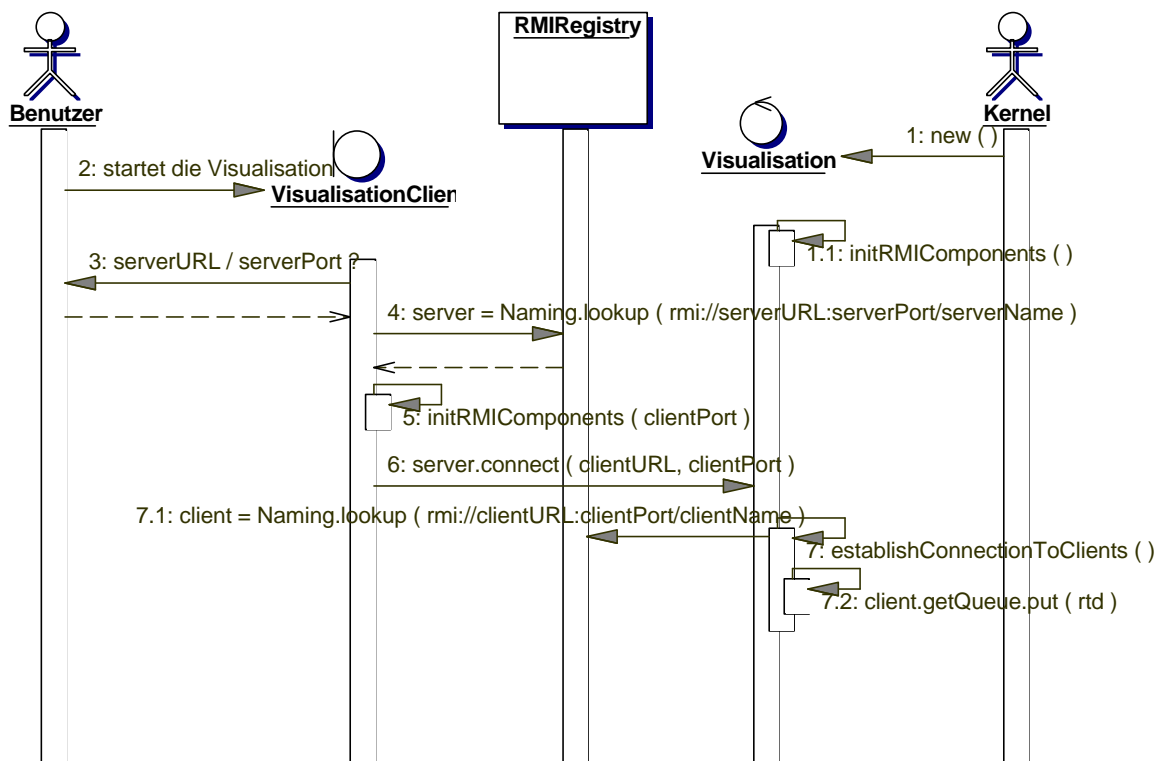
Die folgende Beschreibung bezieht sich auf die detaillierte Fassung des Sequenzdiagramms RMI-Verbindungsaufbau.

Der Kernel erzeugt einen neuen Visualisation-Thread, der in `initRMIComponents()` die RMI Registry für den Server mit der Portnummer 1099 erstellt und anschließend ein Objekt vom Typ `Visualisation` registriert, näheres dazu im detailliertem Sequenzdiagramm.

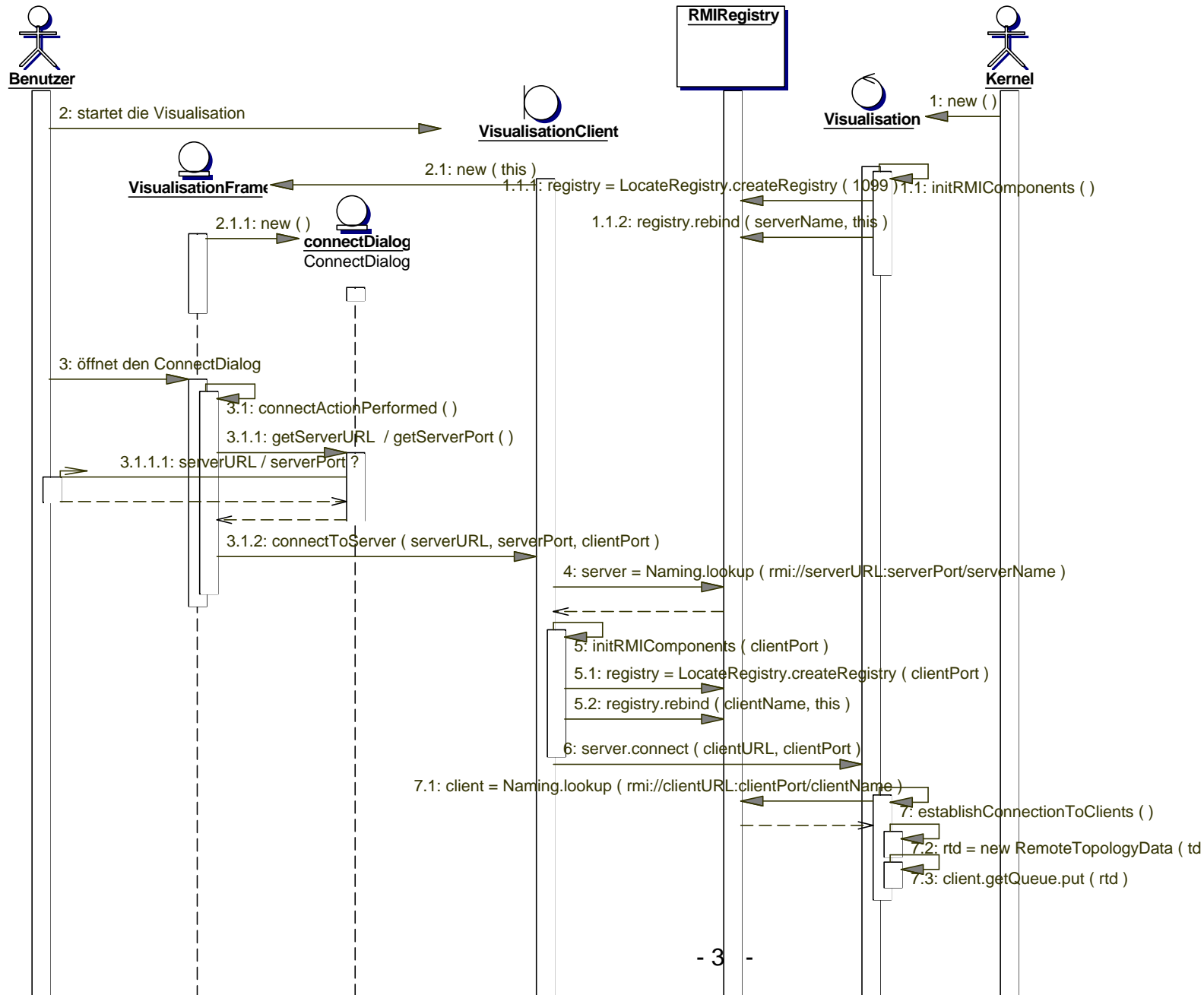
Der Benutzer startet die Visualisation und öffnet zum Verbinden den `ConnectDialog`. Nach Bestätigung der Eingaben ruft der `VisualisationFrame` auf dem `VisualisationClient` die Methode `connectToServer(serverURL,serverPort,clientPort)` auf und übergibt die Verbindungsdaten.

Mit der `lookup`-Methode bekommt der `VisualisationClient` eine Referenz (`server`) auf das `Visualisation`-Objekt, das danach wie ein lokales Objekt genutzt wird. Der `VisualisationClient` erstellt in `initRMIComponents()` seine RMI Registry, registriert ein `VisualisationClient`-Objekt und ruft zum Verbindungsaufbau die Methode `connect()` auf dem Server auf.

Daraufhin schlägt der Server in `establishConnectionToClients()` nach dem `VisualisationClient`-Objekt nach und bekommt eine Referenz. Auf das nun lokal bekannte Objekt wird mit `getQueue.put(rtd)` ein `RemoteTopologyData`-Objekt in die Schlange gelegt.



Detalliertes Sequenzdiagramm RMI-Verbindungsaufbau

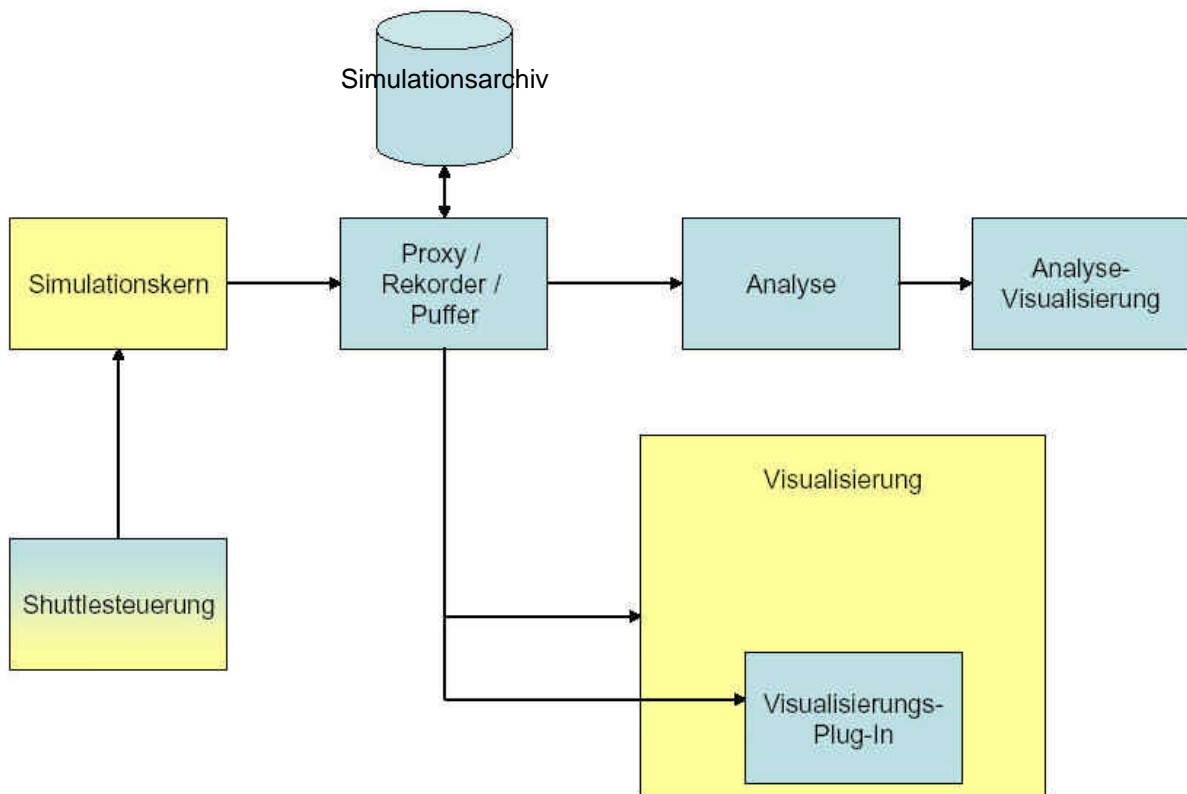


5. Grobentwurf (Soll-Zustand)

Dieser Abschnitt beschreibt die Aufteilung der zu entwickelnden Anwendung in Komponenten, deren Schnittstellen und Interaktion sowie die Verteilung der Komponenten auf Netzwerkknoten.

5.1. Proxy

Um die Module Analyse und Visualisierung auch offline mit aufgezeichneten Daten versorgen zu können, setzen wir einen Proxy ein. Dieser speichert den Datenfluss im Online-Modus und kann diese Daten auf Anforderung wiedergeben. Der Proxy arbeitet transparent, d.h. die Herkunft der Daten ist für die Analyse- und Visualisierungsprozesse nicht relevant.



5.2. Simulationsarchiv

Die Daten der Simulation werden in einem gemeinsamen Archiv gehalten und sind wie oben bereits erwähnt über einen Proxy sowohl von der Visualisierung als auch von der Analyse zugreifbar.

5.3. Analysemodul

Die Analyse stellt Daten für die Analyse-Visualisierung bereit. Als Datenquelle kommt der o.g. Proxy zum Einsatz, so dass auch aufgezeichnete Daten nachträglich analysiert werden können.

Das Analysemodul soll verschiedene Statistiken präsentieren, die sich zum einen auf die Simulation, zum anderen auf Shuttles beziehen.

5.4. Visualisierungs-Plug-In

Das Visualisierungs-Plug-In soll einen Überblick über das aktuell betrachtete Shuttle gegeben werden. Hierzu kann man die Karteikarten „Shuttle Data“ und „Order/Status“ auswählen.

Karteikarte „Shuttle Data“:

Source	Destination	costs	time to cover distance	passed	repair at destination
Hövelhof	New York	123	2	yes	yes
New York	Paris	1123	1	yes	no
Paris	Paderborn	1223	3	no	no
Paderborn	Dortmund	1253	4	no	yes

Diese Karteikarte gibt einen Überblick über den aktuellen Shuttle-Zustand sowie die weitere Planung. Angezeigt wird zum einen der Kontostand, zum anderen die Planung der erhaltenen Aufträge. Es werden die Start- und Zielstationen, die Kosten und die Zeit, die für diese Strecke benötigt werden, und eventuelle Reparaturen am Zielbahnhof angezeigt.

Karteikarte „Order / Status“:

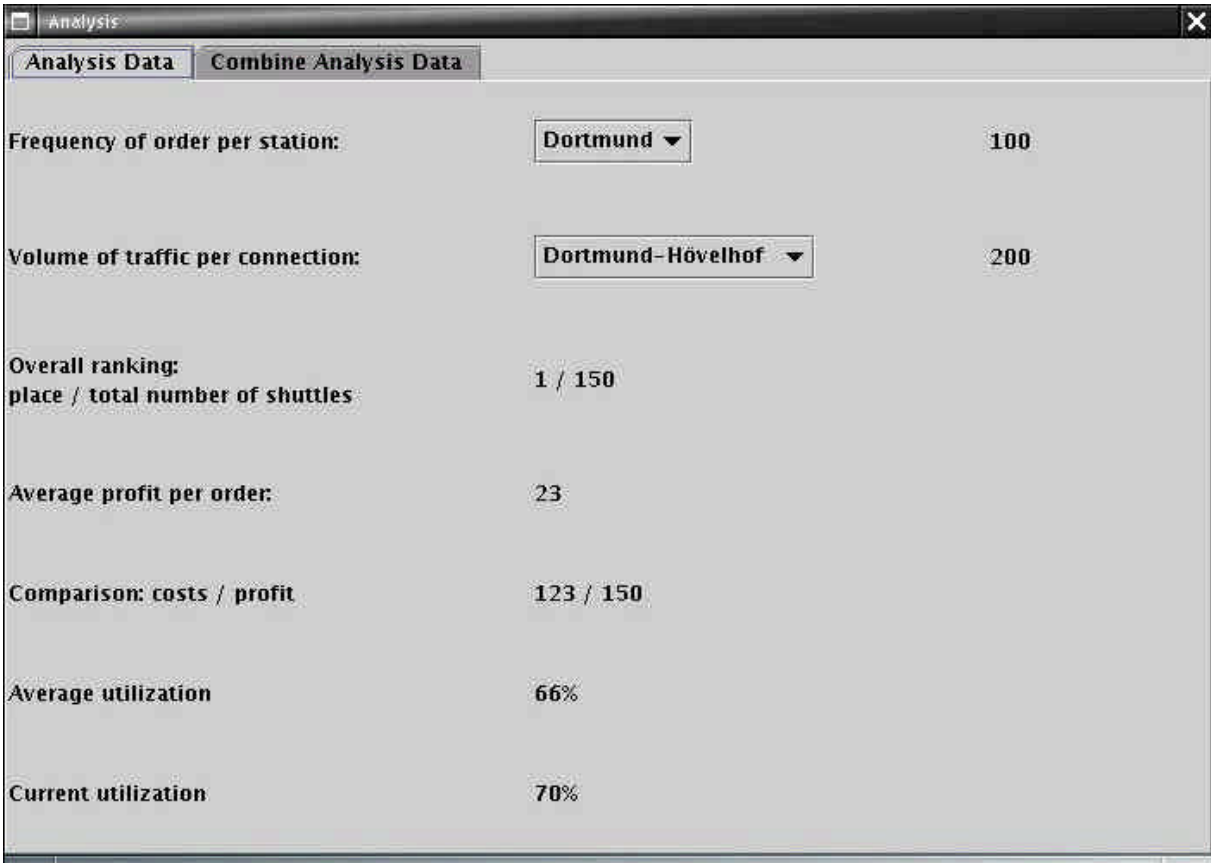
Shuttle Data		Order / Status			
ID	Status	Source	Destination	Persons	Money
1	active	Hövelhof	New York	4	100
2	done	Paderborn	Frankfurt	5	1250
3	done	Uni	Bahnhof	2	100
4	active	Hövelhof	Paris	2	300
5	done	Hövelhof	New York	10	100
6	done	Hövelhof	New York	10	100

In diese Karteikarte werden Informationen über die aktuelle Auftragslage angezeigt. Zu sehen sind der aktuelle Status (in arbeit/bereits bearbeitet), Start- und Zielbahnhof, die transportierten Personen und das verdiente Geld.

5.5. Analyse-Fenster

Im Analyse-Fenster werden die vom Analyse-Modul gesammelten Daten angezeigt. Zum einen gibt es die Karteikarte „Analysis Data“, zum anderen ist mit der Karteikarte „Combine Analysis Data“ eine Möglichkeit zur Kombination dieser Daten vorgesehen.

Karteikarte „Analysis Data“:



The screenshot shows a window titled 'Analysis' with two tabs: 'Analysis Data' (selected) and 'Combine Analysis Data'. The 'Analysis Data' tab displays the following information:

Frequency of order per station:	Dortmund ▼	100
Volume of traffic per connection:	Dortmund-Hövelhof ▼	200
Overall ranking: place / total number of shuttles	1 / 150	
Average profit per order:	23	
Comparison: costs / profit	123 / 150	
Average utilization	66%	
Current utilization	70%	

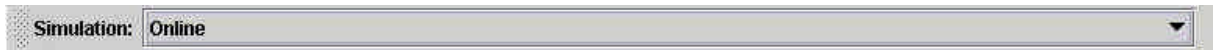
Die Analysis-Karteikarte zeigt statistische Daten zu Simulation und Shuttle an. Abgerufen werden kann die Auftragshäufigkeit pro Bahnhof sowie der durchschnittliche Verkehr auf einzelnen Streckenabschnitten. Zum Shuttle kann die aktuelle Platzierung in der Simulation, der durchschnittliche Profit pro Auftrag, ein Kosten/Gewinnvergleich sowie die durchschnittliche und aktuelle Auslastung angezeigt werden.

Karteikarte „Combine Analysis Data“

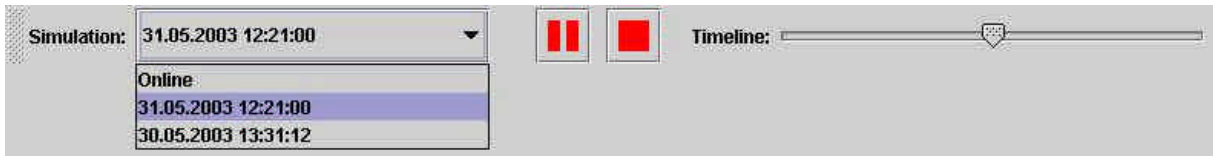
Diese Karteikarte ist für spätere Erweiterungen vorgesehen. Hier soll die Möglichkeit gegeben werden, die verschiedenen Daten miteinander zu kombinieren.

5.6. Toolbar

Online-Modus:



Offline-Modus:



Mit der Toolbar wird die Visualisierung gesteuert. Man hat die Wahl, ob man die Visualisierung im Online- oder im Offline-Modus betrachten möchte. Online-Modus bedeutet, dass man die Daten sieht, die gerade in diesem Moment aufkommen. Wenn man in den Offline-Modus schaltet, kann man sich die Visualisierung von vergangenen Zeitpunkten anschauen und analysieren sowie vor- und zurückspulen.

5.7. Vorberechnung der Wegstrecken

Um beim fahrenden Shuttle Ressourcen bei der Streckenfindung zu sparen, sollen günstige Wege zwischen den Bahnhöfen bereits im Vorhinein nebenläufig berechnet werden. Somit stehen diese Informationen dem Shuttle bei der Angebotsabgabe zur Verfügung, ohne sie jedes Mal neu berechnen zu müssen. Auch Streckensperrungen könnten hier berücksichtigt werden. Der Ressourcenverbrauch hierfür wird über die Auslastung des Shuttles bestimmt und soll somit möglichst minimal bleiben.

5.8. Angebotsberechnung

Zunächst wird über das Shuttle-Analysemodul entschieden, ob für den Transportauftrag ein Angebot abgegeben werden soll. Dabei muss zunächst geprüft werden, ob der Auftrag in der momentanen Situation machbar ist. Im Anschluss daran kann dann ein Angebot abgegeben werden, das alle auftretenden Kosten sowie mögliche Risiken berücksichtigt. Wenn also z.B. das Risiko einer Konventionalstrafe hoch ist, wird ein entsprechend höheres Angebot abgegeben.

6. Qualitätsmerkmale

Benutzbarkeitsanforderung (Typ USE)

Das VisualisierungsPlugin ist so zu gestalten, dass es von jedem Anwender auch ohne Vorkenntnisse einfach zu bedienen ist. Der Wechsel zwischen on- und offline-Modus soll problemlos funktionieren.

Die Statistiken und Diagramme im Analysemodul soll der Benutzer miteinander kombinieren können.

Wegfindung (Typ EFFIZIENZ)

Zur Ermittlung der optimierten Wegstrecke soll dem Shuttle ein modifizierter Shortest-Path-Algorithmus (z.B. Dijkstra) in Kombination mit bereits gesammelten Daten herangezogen werden. Dabei darf das System nie gänzlich ausgelastet werden. Nach Möglichkeit sind dabei Strecken bereits im Vorhinein zu berechnen und abzuspeichern.

Auftragsverhandlung/-planung (Typ EFFIZIENZ)

Bei der Auftragsverhandlung soll im Sinne möglichst großer Wirtschaftlichkeit erfolgen. Dabei sind die Angebote entsprechend der aktuellen Planung in Zusammenhang mit den zur Wegfindung sowie eventuell weiteren gesammelten Daten zu kombinieren und auszuwerten.

Wartbarkeitsanforderungen (Typ PFLEGE)

Der Quellcode ist mit Kommentaren in englischer Sprache zu versehen, um die Wartbarkeit zu erhöhen.

Portierbarkeitsanforderung (Typ PFLEGE)

Zur Wahrung der Portierbarkeit ist als Programmiersprache Java zu verwenden. Zudem sind die bereits definierten Schnittstellen beizubehalten.

Systemvoraussetzungen (Typ Effizienz)

Visualisierung/Analyse und Simulation jeweils:

- Dual-Pentium 3, 933 MHz
- 394 MB RAM
- Festplattenspeicher je nach gewünschtem Simulationsarchiv
- (2 Poolraum-Rechner)