Interoperabilität zwischen Simulationsmodellen auf Basis der High Level Architecture

T. Schulze, G. Lantzsch, Technische Universität Dresden

U. Klein, S. Straßburger, Otto-von-Guericke-Universität Magdeburg

***Abstract**: Die High Level Architecture (HLA) gibt dem Gebiet der verteilten, interoperablen Simulation neue Impulse. Der Beitrag beschreibt eine der ersten zivilen Umsetzungen aus dem Bereich der Logistik. Am Fallbeispiel von kooperierenden Speditionsmodellen in der Sprache SLX wird auf die Kopplung von Simulationstools an HLA eingegangen und verschiedene Stufen der Modellvernetzung sowie die Modelle selbst beschrieben. Abschließend werden die ersten Erfahrungen zusammengefaßt.*

1 Motivation

Für unterschiedliche Applikationsgebiete wurden in den letzten Jahren leistungsstarke spezifische oder allgemein verwendbare Simulationswerkzeuge entwickelt. Beispielhaft sind die Systeme SIMPLE++, CREATE!, ARENA und FACTOR/AIM aufgeführt. Die Nutzung derartiger Systeme erlaubt die Modellierung der nachzubildenden Strukturen mit dem notwendig hohen Grad an Detaillierung.

Gegenwärtig ist ein Trend zu komplexen Anforderungen an die Simulationsmodelle zu erkennen. Zur Absicherung dieser Anforderungen werden komplexe Modelle entwickelt. Diese komplexen Modelle bestehen aus heterogenen oder homogenen Komponenten. Ein Fahrsimulator besteht aus heterogenen Komponenten, wie z.B. dem Echtzeitsystem für die Fahrdynamik, der Umgebungskomponente für Straßen- und Verkehrsbedingungen und einer Visualisierungskomponente. Der Distributionssimulator eines globalen Unternehmens setzt sich überwiegend aus homogenen Komponenten zusammen, die über die unterschiedlichen Standorte verteilt sind und aufgrund ihres lokalen Wissens die benötigten Informationen bereitstellen.

Für die Lösung dieser komplexen Aufgabenstellungen bieten sich zwei unterschiedliche Möglichkeiten an: Erstens die Erstellung von großen monolithischen Simulationsmodellen, die für jede veränderte Aufgabenstellung neu modelliert oder angepaßt werden müssen, und zweitens die Verwendung von modularen, wiederverwendbaren Komponenten, die entsprechend der Aufgabe zusammengesetzt werden.

Die Modularisierung ist ein bekanntes Prinzip in der Softwaretechnologie. Anwendungsmöglichkeiten in der Simulationswelt sind durch die Schlagworte **verteilte Simulation**, **Wiederverwendbarkeit** von Simulationsmodellen,

Interoperabilität zwischen Simulationsmodellen und **Simulationsföderationen mit Infrastruktur** charakterisiert.

Unter **verteilter Simulation** wird eine zeitlich parallele Nutzung von gegebenenfalls heterogenen Simulationsmodellen auf unterschiedlichen Rechnern verstanden. Komplexe Modelle werden in eigenständige Simulationsmodelle aufgeteilt, wobei jedes selbständige Modell seinen eigenen Speicherbereich, seinen eigenen Prozessor und sein eigenes Zeitmanagement (z.B. seine eigenen Ereignislisten) besitzt. Das Hauptziel bei dieser Form der verteilten Simulation liegt in der parallelen Nutzung von verteilten Simulationsmodellen, die nicht zu einem monolithischen Modell zusammengefaßt werden können oder sollen [Mehl94].

Simulationsmodelle werden oft als „single purpose“ Modelle entwickelt. Dem Entwickler ist es nicht möglich, alle späteren Anforderungen an die Nutzung des Modells vorherzusehen. Aber es ist möglich, das Modell so zu entwickeln, daß es im Zusammenspiel mit anderen Modellen neue und andere Nutzungsziele unterstützt. Die **Wiederverwendbarkeit** von Simulationsmodellen senkt dabei beträchtlich den Aufwand bei neuen Simulationsprojekten.

Die an der verteilten Simulation beteiligten Modelle müssen zusätzlich interoperabel sein. Unter dem Begriff **Interoperabilität** wird in diesem Zusammenhang die Fähigkeit von Simulationsmodellen verstanden, mit anderen Simulationsmodellen Daten auszutauschen, die empfangenen Daten für das eigene Verhalten im Simulationsmodell zu nutzen sowie den lokalen Zeitfortschritt zu koordinieren.

Die Grundlagen für den Datenaustausch sind eine entsprechende **Infrastruktur** und definierte Schnittstellen.

Die High Level Architecture (HLA) ist ein Framework zur verteilten Simulation. Dieser Architekturentwurf wurde von dem amerikanischen Verteidigungsministerium (DoD) aufbauend auf den Erfahrungen mit dem ALSP-Protokoll (Aggregate Level Simulation Protocol) und der DIS-Technologie (Distributed Interactive Simulation) entwickelt [DoD97], [KlStr97], [SISO97].

Der zweite Abschnitt dieses Beitrages gibt einen Überblick über die Grundprinzipien von HLA. An einem Prototyp für ein komplexes Logistikmodell aus homogenen Komponenten wird die Anwendung der HLA-Prinzipien vorgestellt. Hierzu wird im dritten Abschnitt das entsprechende Fallbeispiel beschrieben. In diesem Beitrag wird sich auf die Problematik der Interoperabilität zwischen den Komponenten beschränkt; daher werden im vierten Abschnitt die verwendeten Interoperabilitätsgrade erläutert. Der fünfte Abschnitt zeigt exemplarisch die Schritte bei der Entwicklung von HLA konformen Simulationsmodellen. Im sechsten Abschnitt wird ein Ausblick auf weitere notwendige Arbeiten gegeben.

2 Grundprinzipien von HLA

2.1 Konzepte

Die High Level Architecture ist nicht nur ein Standard für verteilte Simulation, sondern zielt auch auf die Unterstützung der Interoperabilität und der Wiederverwendbarkeit von verschiedenen Arten von Programmen ab, die nicht notwendigerweise Simulationen sein müssen. In einer verteilten Simulation gemäß der High Level Architecture schließen sich mehrere Teilnehmer (Simulationen, Beobachter, Steuerkomponenten, etc.), die in der HLA-Terminologie *Federates* (Föderierte) genannt werden, zu einer *Federation* (Föderation) zusammen. Ähnlich dem CORBA zugrunde liegenden Prinzip wird eine einheitliche Schnittstelle (*HLA Interface*) definiert, welche die Teilnehmer eines gemeinsamen Simulationslaufs (*Federates*) aufzuweisen haben, um in Kontakt mit einer *Runtime Infrastructure* (RTI) zu treten, welche Basis-, Koordinations- und Kommunikationsdienste zur Laufzeit bereitstellt (Bild 1).

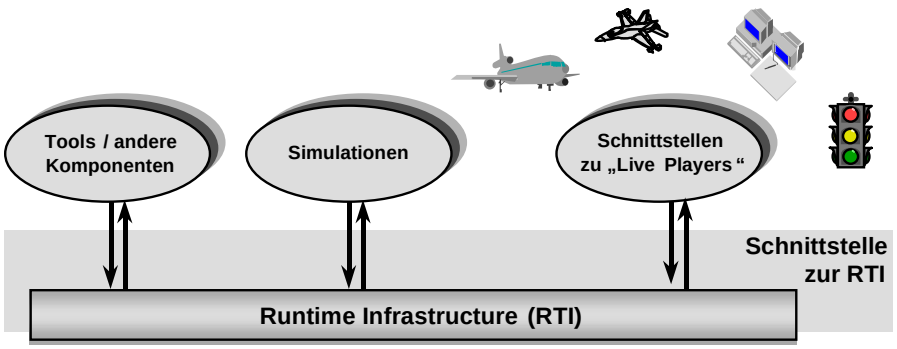

Bild 1 ***Aufbau der High Level Architecture***

Eine Federation kann als Vertrag der beteiligten Federates darüber angesehen werden, was jeder Teilnehmer zur Simulation beitragen kann oder will und was von anderen Teilnehmern erwartet wird. Zur Dokumentation eines solchen Vertrages und der Leistungen bzw. Fähigkeiten der einzelnen Teilnehmer wird in der High Level Architecture eine Objektsichtweise auf die Teilnehmer und die Gesamtsimulation angewendet, die in einer definierten Art und Weise zu dokumentieren ist. Die konkrete Umsetzung einer Federation (des Vertrages) wird in der HLA-Terminologie als *Federation Execution* bezeichnet und steht für den eigentlichen Prozeß der Ausführung der verteilten Simulation.

Ein weiteres entscheidendes Merkmal von HLA, welches in Verbindung mit der beabsichtigten Abstraktion von Simulationsfunktionalität und zugrundeliegenden Basisdiensten steht, ist die Transparenz des Zeitmanagements. Dies besagt, daß es für ein Federate nicht von Bedeutung ist, welches Zeitmanagement andere an einer Federation beteiligte Federates benutzen. So können in einer Federation z.B. optimistisch und konservativ synchronisierte, echtzeit-, zeitschrittgesteuerte und weitere, nicht in diese Kategorien passende Federates teilnehmen, solange sich die Federates der HLA-Zeitmanagementfunktionalität bedienen.

2.2 Komponenten der HLA

Die HLA an sich wird durch die folgenden 3 Komponenten definiert:

2.1.1 HLA Regeln (Rules)

Die *HLA Rules* sind zehn Verhaltensregeln für Federates und Federations. Die ersten fünf Regeln gelten für Federations und legen z.B. fest, daß diese in einer definierten Weise (gemäß Object Model Template) zu dokumentieren sind (Rule 1) und alle Datenrepräsentation einer Federation in den Federates zu erfolgen hat (Rule 2). Sie beinhalten außerdem weitere Laufzeitregeln für Federation Executions.

2.1.2 Object Model Template (OMT)

Die in der HLA angewandte Objektsichtweise schlägt sich nieder in den Objektmodellen, die die statischen und dynamischen Eigenschaften beschreiben (wobei dynamische Eigenschaften in der HLA Interactions genannt werden). Für die Beschreibung der Objektmodelle ist eine bestimmte Form, die im *Object Model Template* (OMT) festgelegt ist, vorgeschrieben. Die Objektmodelle der Federates (*Simulation Object Model*, SOM) und der Federation (*Federation Object Model*, FOM) genügen dem OMT, welches zusammen mit dem HLA *Data Interchange Format* (DIF) den Einsatz von Softwaretools erleichtert.

2.1.3 HLA Interface Spezifikation

Die Schnittstelle zwischen Federates und der RTI wird durch die HLA Interface Specification definiert, deren Funktionen in sechs Managementbereiche aufgeteilt werden:

- Federation Management: Funktionen zum Erzeugen und Entfernen von Federation Executions sowie dem Ein- und Austritt von Federates; Steuermöglichkeiten für das Anhalten, Speichern und Wiederherstellen des Federationzustandes.
- Declaration Management: Funktionen zur Nutzung des Publish- / Subscribe-Mechanismus für Elemente des Objektmodells (Interessenmanagement).
- Object Management: Funktionen zur Objektmanipulation: das Entdecken, Erzeugen und Löschen von Objekten, zum Nachrichtenaustausch über geänderte Attributwerte sowie Kontrolle über die Reihenfolge und Zuverlässigkeit der Nachrichtenübermittlung.
- Ownership Management: Funktionen zur Übertragung des Besitzes von Objektattributen.
- Time Management: Das Zeitmanagement von HLA erlaubt es, Federates mit unterschiedlichen Zeitregimen transparent zusammenarbeiten zu lassen.
- Data Distribution Management: Die hier verfügbaren Funktionen erlauben mittels *Routing Spaces* die dynamische Definition von publizierten und abonnierten Daten anhand der Objektattributswertebereiche.

2.3 Software-Architektur

Die RTI-Software in der vom amerikanischen Defense Modeling and Simulation Office (DMSO) bereitgestellten Version (z.Zt. Version 1.0 Release 3) setzt sich aus den folgenden Komponenten zusammen:

- dem RTI Execution Prozeß (RTIExec): dieser Prozeß ist die Grundlage der RTI und etabliert auf einem beliebigen Hostrechner einen *well known service*. RTIExec ist für die eindeutige Namensvergabe und Steuerung der FedEx-Prozesse zuständig. Federates melden sich bei diesem Prozeß an und erfahren durch ihn, ob und gegebenenfalls wo (auf welchem Rechner) der FedEx-Prozeß für die gewünschte Federation Execution residiert.
- dem Federation Execution Prozeß (FedEx): für jede Federation Execution existiert ein vom ersten Federate einer Federation Execution gestarteter FedEx-Prozeß, bei dem sich während seiner Laufzeit Federates an- bzw. abmelden können. Er verwaltet den gesamten Datenaustausch zwischen den Federates einer Federation Execution. Das erste einer Federation Execution beitretende Federate erfährt von der RTI, daß noch kein FedEx-Prozeß gestartet wurde, und übernimmt diese Aufgabe. Der FedEx-Prozeß residiert immer auf dem Rechner des ersten beitretenden Federates, ein Fakt, der für die Performance einer Federation Execution durchaus zu beachten ist. Jedes weitere, einer Federation Execution beitretende Federate erfährt beim Anmelden bei der RTI den Ort (Hostrechner) des FedEx-Prozesses und stellt alle subsequenten Anfragen direkt an diesen Prozeß.
- eine RTI-Bibliothek (Library), die potentielle Federates zur Ankopplung an die RTI nutzen müssen. Diese Bibliothek ist der „lokale“ Teil der RTI-Software und muß auf jedem Rechner, auf dem ein Federate gestartet werden soll, vorhanden sein.

In Bild 2 wird die oben beschriebene Softwarearchitektur dargestellt.

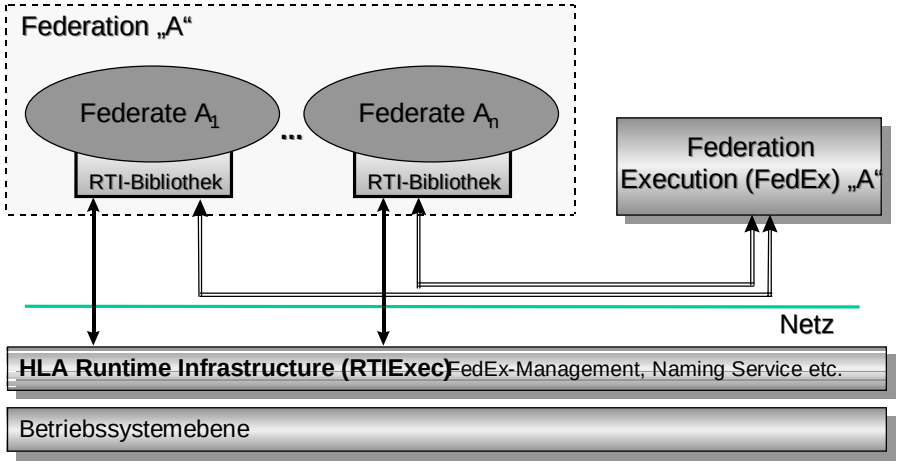

Bild 2: Softwaretechnische Sicht auf eine Federation Execution (RTI 1.0)

Es sei hier angemerkt, daß diese Software-Architektur spezifisch für die bisher von DMSO bereitgestellten RTI-Versionen ist. Da die HLA nicht die Softwarearchitektur, sondern eine Interface Specification standardisiert, sind andere Implementierungen möglich.

3 Logistisches Fallbeispiel

Das Fallbeispiel geht von folgender Situation aus: In einer Region agieren mehrere unterschiedliche eigenständige Speditionen. Jede Spedition hat ihr bestimmtes Liefergebiet und einen eigenen Fuhrpark. Jedes Fahrzeug kann nur einen Auftrag zur Auslieferung vornehmen. Ein Auftrag besteht wiederum aus einem Paket, wobei die Zieladressen gleichmäßig über das Liefergebiet verteilt sind. Die Zwischenankunftszeit im Auftragsstrom ist exponentialverteilt. Nach der Auslieferung eines Paketes kehren die Fahrzeuge wieder zu ihrem Heimatstandort zurück. Hier wird je nach dem zu verwendenden Grad der Interoperabilität zwischen zwei Strategien unterschieden:

1. die Fahrzeuge fahren leer zurück.
2. die Fahrzeuge fragen bei der örtlichen Spedition nach einem Fahrauftrag nach, der auf ihrem Rückweg liegt.

Für dieses Fallbeispiel wurden unterschiedliche Simulationsmodelle entworfen. Generelle Eingangsparameter einer jeden Spedition sind:

- Lokation der Spedition
- Anzahl der Fahrzeuge
- Mittelwert der exponentialverteilten Zwischenankunftszeit
- Koordinaten des Liefergebietes
- Mittlere Fahrgeschwindigkeit der Fahrzeuge
- Zeitpunkt zum Abbruch der Simulation

Die Zeiteinheit für die Simulationsmodelle ist Minuten. Nach jeweils 24 Stunden Simulationszeit wird von jeder Spedition ein Report generiert. Dieser Report enthält Ergebnisgrößen der Simulation.

Die Modellierung erfolgt in mehreren Varianten. Die Varianten berücksichtigen unterschiedliche Interoperabilitätsgrade zwischen den beteiligten Speditionen. Alle Modelle wurden mit dem System SLX [Henr97] erstellt und simuliert. Durch die an der Universität Magdeburg entwickelte DLL-Bibliothek [Stra98] zur Kopplung der RTI an SLX ist die Möglichkeit gegeben, SLX-Modelle unter Verwendung der HLA-Prinzipien zu koppeln.

4 Verwendete Interoperabilitätsstufen

4.1 Beschreibung der Stufen

Zur Untersuchung der HLA-basierten Interoperabilität wurden verschiedene Stufen definiert. Die Intensität der Interoperabilität wächst mit der Stufennummer. Die folgenden Tabelle beschreibt die verwendeten Stufen.

Stufe	Erläuterung	RTI-Dienste
0	Jede Spedition operiert ohne Beziehungen zu anderen Speditionen. Eine Verteilung der Simulationsmodelle ist nicht erforderlich.	Konventionelle, monolithische Simulation
1	Jede Spedition operiert ohne Beziehungen zu anderen Speditionen. Es wird eine Federation aufgebaut, in der sich die Federates nur an- und nach dem Simulationslauf wieder abmelden.	Federation Management
2	Jede Spedition berichtet mit einem eigenen Report nach einem für alle Speditionen gültigen Reportrhythmus. Die Federates müssen sich zur Reportausgabe synchronisieren.	Federation Management Time Management
3	Ein Reportfederate wird in die Federation aufgenommen. Dieses Federate erstellt nach jeder Reportperiode einen Federationreport von allen an der Federation angemeldeten Speditionen und wird hierzu mit den notwendigen Daten von den anderen Federates über Interaktionen versorgt.	Federation Management Time Management Declaration Management Object Management
4	Die Speditionen tauschen während der Simulation aktiv Informationen untereinander aus. Nach dem Entladen eines Containers wird bei einer auf dem Rückfahrweg liegenden anderen Spedition nach einem Auftrag in Richtung des Heimatstandortes gefragt.	Federation Management Time Management Declaration Management Object Management

Tabelle 1:Stufen der Interoperabilität der Beispielmodelle

4.2 Stufe 0

In dieser Stufe operiert jede Spedition ohne Beziehungen zu anderen Speditionen. Für jede beteiligte Spedition wird eine Instanz der Klasse Spedition generiert. Alle Instanzen interagieren zusammen in einem monolithischen Modell. Diese Modellierungsform entspricht der klassischen Vorgehensweise und wurde als Vergleichsstufe eingeführt.

4.3 Stufe 1

Diese Stufe ist durch die Verwendung einer Federation charakterisiert. Jede Spedition existiert als eigenständiger Prozeß, wobei die einzelnen Modelle über die Schnittstelle zum Aufruf der RTI-Funktionen verfügen.

Jede Spedition agiert unabhängig voneinander. Diese Stufe wurde eingeführt, um die An- und Abmeldevorgänge von SLX-Federates zu testen.

4.4 Stufe 2

In der zweiten Stufe berichtet jede Spedition mit einem eigenen Report nach einem für alle Speditionen gültigen Reportrhythmus. Die Speditionen sollen nun nach jedem Tag einen Zwischenbericht über die bisherige Tätigkeit geben. Der Tag stellt dabei die gültigen Reportperiode dar.

Mit der Ausführung des täglichen Reports einer Spedition muß diese solange warten, bis alle an der Federation beteiligten Federates diesen Zeitfortschritt erreicht haben. Der lokale Zeitfortschritt einer Spedition muß mit dem der anderen Speditionen synchronisiert werden. Es müssen also andere Federates mit der Simulation und dem Report eines weiteren Tages warten, solange die lokale Zeit einer Spedition noch nicht den vorgegebenen Zeitpunkt zur Ausführung des Reports erreicht hat. Eine Federate kann also von anderen im Zeitfortschritt behindert werden und darf andere in deren Zeitfortschritt behindern. Im RTI-Zeitmanagement werden die Parameter **Time Constrained** für das Warten auf andere und **Time Regulating** für das Beeinflussen anderer im Zeitfortschritt gesetzt. Als Lookahead wird die Zeitspanne bezeichnet, in der ein Federate -ausgehend von seiner aktuellen lokalen Zeit- keinem anderen Federate Ereignisse (hier: Interaktionen) sendet. In dieser Stufe wird für den Lookahead ein Wert von 1440, d.h. einer Reportperiode, eingetragen.

Für die verteilte Simulation ergeben sich hier zwei Probleme.

1. Federates können sich zu unterschiedlichen realen Zeitpunkten anmelden. Ein einzelnes Federate kann somit seine Simulation durchführen, ohne im Zeitfortschritt von einem anderen Federate behindert zu werden. Dieses andere Federate meldet sich erst nach dem Abschluß der Simulation an. Dies ist nicht wünschenswert, da die Speditionen miteinander interagieren sollen.
2. Ein weiteres Federate meldet sich während der laufenden Simulation eines anderen Federates an. Das alte Federate besitzt die lokale Zeit $t_1>0$, während das neue Federate mit der lokalen Zeit $t_2=0$ beginnen will.

Die Lösung der Probleme könnte einem weiteren Federate als Synchronisationsmanager, der externes Wissen über die zu beteiligenden Federates besitzt, übertragen werden. Diese Manager könnte z.B. durch Interaktionen den Simulationsstart anzeigen. Gegen einen Manager spricht jedoch die HLA-Philosophie, daß Federates prinzipiell eigenständige Teilnehmer sind und ihre An- und Abmeldung dynamisch erfolgen kann.

In der Stufe 2 wurde der Ansatz eines Steuerungsfederates daher nicht weiter verfolgt. Eine genutzte Möglichkeit des Simulationsstarts war es, manuell zu synchronisieren, indem durch externe Absprachen mit dem Start solange gewartet wird, bis sich alle gewünschten Speditionen angemeldet haben.

Die lokale Simulationszeit beginnt in einer Federation immer bei Null. Meldet sich ein Federate bei einer bereits laufenden Federation an, so muß das neue Federate zeitlich eingebunden werden. Da das Federate beim Anmelden die aus seiner Sicht gültige Zeit der Federation als seine initiale Zeit zugewiesen bekommt, stellt die zugewiesene Zeit für das Federate den logischen Zeitpunkt Null dar.

Ist eine Spedition beispielsweise schon einige Tage fortgeschritten, so bekommt eine sich dann neu anmeldende Spedition diesen Wert als eigenen Startzeitpunkt geliefert und muß seine logische Zeit ohne irgendwelche weiteren Aktivitäten von Null auf die aktuelle Federationszeit stellen, um beginnend von diesem Zeitpunkt aus an der verteilten Simulationszeit teilzunehmen. Dieser Ansatz wurde als zweite Möglichkeit zur Durchführung einer Federation Execution getestet.

4.5 Stufe 3

In dieser Stufe wird ein Reportfederate wird in die Federation aufgenommen. Dieses Federate erstellt nach jeder Reportperiode einen Federationreport. Hierzu wird dieses Federate von allen an der Federation angemeldeten Speditionen mit den notwendigen Daten versorgt. Diese Übertragung erfolgt mittels Interaktionen.

Zur Organisation des Zeitmanagements wird für dieses Federate die Eigenschaft „Time Constrained“ gesetzt. Somit ist gewährleistet, daß alle Reports am Ende eines Tages zusammenhängend ausgegeben werden.

Das Reportfederate hat für sich zu entscheiden, wann eine Federation Execution als beendet angesehen werden kann, um sich dann zurückzuziehen. In der aktuellen Version überprüft das Federate dafür, ob mehrere Reportperioden ohne Aktivität auftraten. Ein besserer, in einer zukünftigen Version zu verfolgender Ansatz ist es, die von HLA bzw. der RTI bereitgestellten Informationen des Management Object Model (MOM) zu nutzen, da hierüber z.B. Informationen über die Anzahl der Federates in einer Federation Execution zu erfahren sind.

4.6 Stufe 4

In dieser Ausbaustufe wird eine vollständige zeitliche Synchronisation erforderlich, da nach Ablieferung einer Ladung durch einen LKW am Zielort ggf. eine Anfrage nach einem Transportauftrag für den Rückweg an die am Ort ansässige Spedition gefragt wird. Diese Anfrage wird als Interaktion modelliert und dient als Basiselement zur Implementation von Kooperationen auf Basis unterschiedlicher Entscheidungsstrategien zwischen einzelnen Speditionen. Diese interessanteste Stufe der Zusammenarbeit wurde zunächst im Hinblick auf die technische Koordination von homogenen, auf einem gemeinsamen Netz kooperierenden Simulationen ausgewertet, weshalb zur Zeit lediglich eine einfache Kooperationslogik implementiert wurde.

5 Entwicklung der HLA-basierten Simulationsmodelle

Dieser Abschnitt faßt die Entwicklung der HLA-Objekt- und Simulationsmodelle zusammen. Zur Dokumentation der HLA-Simulationsmodelle sind für das logistische Fallbeispiel die Objektmodelle für die Federation (FOM) und die Federates (SOM) zu entwickeln. Die hierfür zu benutzenden Object Model Templates bieten hierzu Tabellen für Struktur- und Inhaltsinformation der Klassen, Interaktionen und Objektattribute, welche auch Angaben zur Publikations- (P) und Abonnierfähigkeit (S) enthalten. Im Fallbeispiel hat diese Tabelle lediglich die zwei Objektklassen „carriage“ und „observer“. Erstere ist publishable und subscribable, der Beobachter dagegen nichts von beidem.

Class1
carriage (PS)
observer

Tabelle 2: Object Class Structure Table (FOM)

Weitere Klassen oder Unterklassen gibt es im Beispiel nicht. Verwendete Begriffe sind in zusätzlichen Tabellen (Lexika) zu definieren:

Term	Definition
carriage	Objekt, welches eine Spedition der realen Welt simuliert
observer	Beobachter, sammelt Informationen der anderen Speditionen

Tabelle 3: Object Class Definitions (FOM)

Für die Interaktionen gibt es die ‘Object Interaction Table’, in der für das Beispiel die report_interaction deklariert wird, die die von den Speditionen versendeten Reports realisiert. Diese bereits umfangreichere Tabelle enthält neben den Interaktionsparametern (den Reportdaten) noch die erzeugende (carriage) und empfangende (observer) Klasse. Öffentliche Attribute von Objekten und Interaktionen sind mit ihren Eigenschaften in der ‘Attribute/Parameter Table’ zu dokumentieren. Dazu dienen Spalten wie ‘Updateable/Reflectable’, in der festgelegt wird, ob das Attribute veröffentlichbar/abonnierbar ist.

Object/ Interaction	**Attribute/ Parameter**	**Datatype**	**Cardi-nality**	**Units**	**Reso-lution**
Carriage	name	string	1	N/A	N/A
	location_carriage	location	1	N/A	N/A
	...				

Accu-racy	**Accuracy Condition**	**Update Type**	**Transferable/ Acceptable**	**Updateable/ Reflectable**
Perfect	always	Conditional	N/A	UR
Perfect	always	Conditional	N/A	UR
	...			

Tabelle 4: Ausschnitt aus der ‘Attribute/Parameter Table’ (SOM - carriage)

Location ist in der ‘Complex Datatype Table’ definiert und enthält die Adresse der Spedition.

Aus diesen Tabellen ist das *Federation Execution Data*-File erstellbar. In diesem sind die FOM-Objekt- und Interaktionsklassenhierarchien verzeichnet, die während der Laufzeit der Federation benötigt werden

6 Ausblick

Auf Basis eines einfachen Speditionsfallbeispieles konnten die wichtigsten Eigenschaften der HLA genutzt und untersucht werden. Wie auch für heterogen besetzte Federations ist die Realisierung verteilter Simulationen mit homogenen Komponenten auf Basis von HLA gut darstellbar. Die nächsten Schritte beinhalten die Abbildung der physikalisch-rechtlichen Übergabe von Objekten mittels des HLA Ownership Managements sowie die Implementation verschiedener Flottenmanagement- und Kooperationsstrategien.

Referenzen

[DoD97] Defense Modeling and Simulation Office (DMSO). *High Level Architecture Homepage*. URL http://hla.dmso.mil/.

[Henr97] Henriksen, J. O. *SLX and Proof Animation: Improved Integration Between Simulation and Animation*. Proceedings of the Simulation and Animation Conference Magdeburg '97. SCS European Publishing House San Diego/Erlangen/Ghent/Budapest 1997, pp. 287-294.

[KlStr97] Klein, U. S. Straßburger. *Die High Level Architecture: Anforderungen an interoperable und wiederverwendbare Simulationen am Beispiel von Verkehrs- und Infrastruktursimulationen*. 11. Symposium Simulationstechnik ASIM 97, 11.11.-14.11.1997, Dortmund.

[Mehl94] Mehl, H. *Methoden verteilter Simulation.* Vieweg Verlag, Braunschweig, 1994.

[SISO97] Simulation Interoperability Standards Organisation (SISO). *SISO and Simulation Interoperability Workshop Homepage*. URL http://siso.sc.ist.ucf.edu.

[Stra98] Straßburger, S. *Integration klassischer Simulationstools in die High Level Architecture*. Diplomarbeit, Fakultät für Informatik, Otto-von-Guericke-Universität Magdeburg. 1998.