HLA-basierte Kopplung der Simulationssysteme SIMPLEX III und SLX

Gunter Lantzsch
Institut für Informationssysteme
Technische Universität Dresden
gl1@irz301.inf.tu-dresden.de

Steffen Straßburger
Institut für Simulation und Grafik
Otto-von-Guericke-Universität Magdeburg
strassbu@isg.cs.uni-magdeburg.de

Christoph Urban
Lehrstuhl für OR und Systemtheorie
Universität Passau
urban@fmi.uni-passau.de

Abstrakt

Der Beitrag erläutert Konzepte zur HLA-Erweiterung von klassischen stand-alone Simulationssystemen. Es wird gezeigt, wie SIMPLEX III und SLX als zwei grundsätzlich verschiedene Simulationssysteme mittels des HLA-Standards gekoppelt werden können. Als Anwendungsbeispiel wird das Modell Abfüllanlage beschrieben, das aus einem zeitkontinuierlichen und einem zeitdiskreten Modell besteht. Beide Modelle arbeiten mit Hilfe des HLA-Standards zusammen. Hierdurch wird die Möglichkeit eröffnet, Teilmodelle von denjenigen Simulationssystemen bearbeiten zu lassen, welche hierfür am geeignetsten sind.

1 Einleitung

Die High Level Architecture for Modeling and Simulation (HLA) des amerikanischen Defense Modeling and Simulation Office (DMSO) bietet einen Ansatz, verschiedenartige Simulations-, Animations- und Supporttools miteinander zu vernetzen bzw. zu koppeln. Obwohl HLA ursprünglich unter dem Aspekt der Kopplung militärischer Trainingssimulatoren entwickelt wurde, bieten sich auch im zivilen Bereich vielfältige Möglichkeiten, die allgemeinen Forderungen nach Wiederverwendbarkeit und Interoperabilität von Simulationsmodellen mit Hilfe von HLA umzusetzen [1].

Ähnlich dem CORBA zugrundeliegenden Prinzip wird unter HLA eine einheitliche Schnittstelle (*HLA Interface*) definiert, welche die Teilnehmer eines gemeinsamen Simulationslaufes (*Federates*) aufzuweisen haben, um in Kontakt mit der *Runtime Infrastructure* (RTI) zu treten, welche Basis-, Koordinations- und Kommunikations-dienste zur Laufzeit bereitstellt. Eine *Federation* kann als ein Vertrag zur Durchführung eines Simulationslaufes (*Federation Execution*) zwischen den Federates angesehen werden, in dem die Einzelheiten und Objektmodelle der Federates (*Simulation Object Model*) und der Federation (*Federation Object Model*) festgelegt sind. Diese Informationen sind in einer vorgegebenen Form zu dokumentieren (*Object Model Template*).

Ein Vorteil der Architektur ist die Offenheit für verschiedene Arten von Federates. Neben Simulationssystemen können sowohl Softwarebausteine, wie z.B. Datenbanken und Beobachter, als auch reale Elemente, wie Sensoren und andere Hardware, bei Einhaltung der HLA-Spielregeln bei Federation Executions mitwirken. Damit sind durch die Nutzung

der HLA zur Kopplung von Simulationssystemen die einzelnen Teilmodelle ohne weiteres durch andere Federates ersetzbar, sofern diese sich an den festgelegten Vertrag halten. Es spielt dabei keine Rolle, ob ein zu ersetzendes Federate auf einem anderen Simulationssystem neu implementiert wird. Ein Teilmodell läßt sich damit auf einem beliebigen Simulationssystem implementieren, sofern dieses eine HLA-Anbindung und die Fähigkeit, das Teilmodell modellieren zu können, besitzt.

Demonstriert wird die Kopplung u.a. anhand eines Modells einer verfahrenstechnischen Abfüllanlage für Fässer. Dieses Modell enthält neben der Logistik auch einen kontinuierlichen Prozeß und zeigt damit, daß unter HLA selbst unterschiedlichste Simulationssysteme miteinander gekoppelt werden können. So ist SLX ein sehr schnelles Simulationssystem für diskrete Simulationsmodelle, während unter SIMPLEX neben diskreten auch kontinuierliche Modelle ausführbar sind. Ebenfalls erfolgt die Modellspezifikation und die HLA-Anbindung in beiden Systemen aus Nutzersicht auf völlig verschiedene Arten. Darauf wird in den folgenden Kapiteln eingegangen.

2 Konzepte zur HLA-Erweiterung von Simulationssystemen

Um Mitglied (Federate) in einer verteilten Simulation gemäß des HLA-Standards (Federation) werden zu können, muß ein Simulationssystem auf das HLA Application Programming Interface (API) zugreifen. Das API steht für typische Programmiersprachen wie C++, Java oder ADA zur Verfügung. Da es für den Simulationsentwickler wünschenswert ist, sein Modell auf einer höheren Ebene zu entwickeln, z.B. in einer Modellbeschreibungssprache wie der *Model Description Language (MDL)* von SIMPLEX oder einer Simulationssprache wie SLX, ist es notwendig, HLA-Schnittstellen in die Simulationssysteme zu integrieren [2]. HLA-Schnittstellen für Simulationssysteme können prinzipiell nach zwei Gesichtspunkten bzw. Fragestellungen kategorisiert werden:

a) Wie ist die HLA-Schnittstelle technisch realisiert ?

b) Wie präsentiert sich die HLA-Schnittstelle dem Modellentwickler ?

Die erste Fragestellung stellt ein klassisches programmierungstechnisches Problem dar, für das bereits verschiedene Möglichkeiten identifiziert wurden (siehe u.a. [3,10]). So kann die HLA-Anbindung entweder

- durch Erweiterung des Quellcodes des Simulationssystems an sich,
- durch Nutzung einer externen Programmierschnittstelle des Simulationssystems,
- durch Erweiterung eines vom Simulationssystem erzeugten Zwischencodes oder
- durch Nutzung eines Gateway-Programms

erfolgen.

Die zweite Fragestellung, d.h. wie sich die HLA-Schnittstelle dem Nutzer präsentiert, wurde bisher vernachlässigt und soll daher in diesem Beitrag schwerpunktmäßig diskutiert werden.

Grundsätzlich bieten sich die folgenden zwei prinzipiellen Möglichkeiten:

1. HLA-Anbindung durch Bereitstellung eines Simulator-spezifischen HLA-APIs

   In dieser Variante würde dem Nutzer die HLA-Schnittstelle in einer an die Spezifika des Simulationssystems (z.B. Datentypen, Aufrufkonventionen, etc.) angepaßten Form präsentiert. Der Modellentwickler müßte sein Simulationsmodell um HLA-Fähigkeiten erweitern (z.B. durch externe Funktionsaufrufe). Das Mapping vom HLA-API in das Simulator-spezifische API könnte in einer Zwischenschicht erfolgen, die zwischen Funktionsaufrufen des Simulators und HLA-Funktionsaufrufen übersetzt. Die Zwischenschicht, die z.B. in Form einer Wrapper-Bibliothek implementiert werden könnte, würde die teilweise recht aufwendigen Programmierarbeiten mit dem eigentlichen HLA API in eine vom Simulationsmodell komfortabel nutzbare Form verpacken [3]. Somit wäre es, wie am Simulationssystem SLX demonstriert, möglich, bestehende monolithische Simulationsmodelle um HLA-Fähigkeiten zu erweitern und als Federate in einer HLA-Federation mitwirken zu lassen [4]. Ein wesentlicher Nachteil dieser Lösung ist, daß am Simulationsmodell Erweiterungen bzgl. der HLA-Funktionalität vorgenommen werden müssen und somit ein bereits bestehendes Modell nur unter erhöhtem Aufwand an einer verteilten Simulation unter HLA teilnehmen kann.

2. Verbergen des HLA-APIs durch automatischen Aufruf vom Simulationssystem aus

   Die andere grundsätzliche Möglichkeit für eine HLA-Schnittstelle aus Nutzersicht besteht darin, die HLA-Schnittstelle unter der Oberfläche des Simulationsprogramms zu halten und zu fordern, daß die Beschreibung eines (Teil-)Modells unabhängig von dem Fakt sei, ob es systemintern mit anderen Teilmodellen kombiniert wird oder als eigenständiges Federate im Sinne von HLA auftritt und mit anderen HLA-Federates interagiert. Der größte Vorteil dieser Variante liegt in der Wiederverwendbarkeit bereits bestehender Simulationsmodelle. Die Aufgabe des Simulationssystems ist es dann, sämtliche Kopplungsaufgaben (Synchronisation, Datenaustausch, Reagieren auf externe Ereignisse) unter der Oberfläche durchzuführen. Die Möglichkeit der HLA-basierten Kopplung mit anderen Modellen würde somit eine Eigenschaft des Simulationssystems selbst. Aus Sicht des Modellentwicklers wäre diese Variante ideal, da er sich bis auf die Erstellung der HLA-Objektmodelle nicht mit HLA zu befassen hätte. Inwieweit sich dieses Idealbild umsetzen läßt, und wo Kompromisse notwendig oder wünschenswert sind, wird am Beispiel des Simulationssystems SIMPLEX III dargestellt [11].

Beide der vorgestellten Lösungen haben ihre Vor- und Nachteile und somit ihre Berechtigung. Die erste Lösung (Mapping-Ansatz) bietet sehr flexible Möglichkeiten zur Nutzung aller technischen Möglichkeiten von HLA. Durch die Tatsache, daß die HLA-Funktionalität nicht verborgen bleibt, sondern dem Anwender auf eine relativ komfortable Art zugänglich gemacht wird, läßt sich dieser Ansatz auch hervorragend als Lehrmittel zum Kennenlernen von HLA einsetzten. Da die Kopplung ein Teil des Modells ist, *muß* sich der Modellierer jedoch selbst um einige für HLA notwendige Details kümmern (Synchronisation, Datenaustausch, Reaktion auf externe Ereignisse). Dies ist gegenüber der zweiten Lösung ein wesentlicher Nachteil, da detaillierte HLA-Kenntnisse vorhanden sein müssen. Eine Nutzung von bestehenden Modellen ist nur mit HLA-Erweiterungen möglich.

Die zweite Lösung besticht durch ihre Nutzerfreundlichkeit, da kein „Hinzulernen“ oder Umdenken bei der Modellerstellung notwendig ist. Es ist für den Anwender fast nicht notwendig, Kenntnisse über HLA zu haben, da die Kopplungsfähigkeit ein Teil des Simulationssystems ist. Um HLA-kompatible Modelle erstellen zu können, muß der Modellierer lediglich zu jedem Modell ein Simulation Object Model gemäß des HLA *Object Model Templates* erstellen. Die Auseinandersetzung mit der API der HLA entfällt für ihn vollständig, da dieses direkt vom Simulationswerkzeug angesprochen wird.

Ein gewisser Nachteil der Lösung ist, daß für eine automatische Lösung, die für alle Modelle gültig ist, einige Einschränkungen bezüglich der Nutzbarkeit aller von HLA gebotenen Möglichkeiten in Kauf zu nehmen sind. Dies ist jedoch in Anbetracht des gebotenen Komforts durchaus vertretbar.

Die Vorteile aus beiden Lösungen könnten jedoch auch durch Nutzung von Hybridformen kombiniert werden. Dabei wird eine automatische Lösung durch vom Modellierer explizit aufzurufende HLA-Funktionsaufrufe ergänzt. Eine solche Technik könnte alle von der HLA angebotenen Möglichkeiten nutzen und zugleich minimalen Aufwand für den Modellierer gewährleisten.

Es sei hier angemerkt, daß es unabhängig von der Tatsache, welcher Ansatz zur HLA-Erweiterung eines Simulationssystems gewählt wird, möglich ist, diese miteinander über HLA zu koppeln.

3 Kopplung von SIMPLEX III und SLX

Das HLA-Interface für SLX, welches ausführlich in [5] vorgestellt wurde, basiert auf dem oben erwähnten Mapping-Ansatz. Die praktische Machbarkeit des zweiten vorgestellten Ansatzes wurde anhand des Simulationssystems SIMPLEX III (u.a. wegen der Verfügbarkeit des Quellcodes) untersucht [11]. Dieser Beitrag konzentriert sich auf die Darstellung des SIMPLEX III - Ansatzes.

3.1 HLA-Interface für SIMPLEX III

Datenaustausch

In SIMPLEX können Modelle durch hierarchische Zusammenschaltung (Abbildung 1) von einzelnen Teilmodellen erzeugt werden [6,8,9]. Diese Teilmodelle werden durch Basiskomponenten (*Basic Components*) bzw. *High Level Components* repräsentiert und sind auch unabhängig von einer Verschaltung als selbständige Modelle ausführbar. Eine High Level Component stellt bereits einen Verbund von Teilmodellen dar und definiert die Zusammenschaltung mehrerer Komponenten durch die Festlegung, welche Sensorvariablen einer Komponente lesend auf die einer anderen Komponente zugreifen kann (Glas-Box Prinzip).

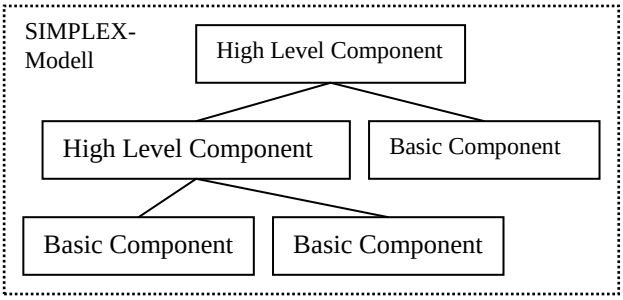

Abbildung 1: Mögliche Struktur eines SIMPLEX-Modells

Die oberste High Level Component repräsentiert innerhalb von SIMPLEX das gesamte SIMPLEX-Modell. Von ihr ausgehend kann innerhalb des Simulationsprogramms auf alle Modellvariablen zugegriffen werden. Dieser Fakt wurde für die HLA-Anbindung von SIMPLEX-Modellen genutzt. In einer neuen *HLA Component* wird das Mapping von SIMPLEX-Variablen aus Basiskomponenten oder auch mobilen Komponenten auf HLA-Strukturen (Objekte und Interaktionen) spezifiziert. Mobile Komponenten sind eine weitere Komponentenart in SIMPLEX und stellen bewegliche Einheiten dar.

Die eigentliche Modellspezifikation in der SIMPLEX-eigenen Model Description Language ändert sich hierfür nicht. Dies erhöht in einem hohen Grade die Wiederverwendbarkeit von Modellen, was eines der Hauptanliegen von HLA ist.

Somit kann ein Modell wahlweise in einer HLA-Federation als Federate auftreten oder wie gewohnt als reines SIMPLEX-Modell ausgeführt werden. Soll das Modell als Federate arbeiten, so muß die oberste High Level Component auf die darüber liegende HLA Component zugreifen (Abbildung 2). Diese liest und schreibt lediglich die Modellvariablen.

Der Modellierer muß sich bei dieser Lösung nur noch damit auseinandersetzen, welches Mapping von SIMPLEX-Elementen mit welchen Elementen der HLA-Strukturen erfolgen soll. Dabei muß er sich an den Vertrag halten, der von der Federation von allen

beitretenden Federates gefordert wird. Dieser Vertrag beinhaltet aber nur die erlaubten HLA-Strukturen. Die Definition des Mappings wird davon nicht berührt und bleibt in der Verantwortung des Modellierers.

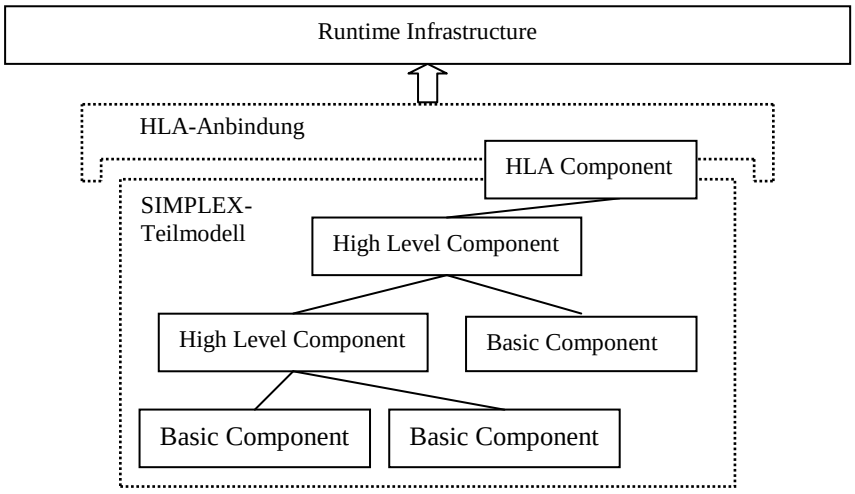

Abbildung 2: HLA-Anbindung über eine neue Komponentenart

Am Referenzbeispiel des Ampelmodells (siehe Kapitel 4.1) wird in der Abbildung 3 das Mapping zwischen Modellgrößen aus SIMPLEX III und HLA-Daten demonstriert. Bei jeder Übertragung von Daten zwischen HLA und SIMPLEX III wird dabei eine Konvertierung der Datentypen und ein Mapping der Bezeichner durchgeführt. Damit erscheint eine Modellgröße in SIMPLEX III unter HLA als ein in der HLA Component beliebig spezifizierbares Objektattribut bzw. Interaktionsparameter. Zusätzlich wird in der HLA-Component der Datentyp des Objektattributs bzw. Interaktionsparameters festgelegt. Gleichermaßen erfolgt das Mapping eines HLA-Objektattributs bzw. HLA–Interaktionsparameters mit einem beliebigem Datentyp auf die in der HLA Component spezifizierte Modellgröße von SIMPLEX III.

Von der Festlegung, wann die Datenkommunikation erfolgen muß, wird der Modellierer völlig entlastet. Diese Aufgabe wird vom Simulationssystem übernommen. Dabei wird der Wert einer zu übermittelnden Variable automatisch immer dann übertragen, wenn sich dieser ändert. Damit kann der Verlauf des Variablenwertes beim Empfänger vollständig reproduziert werden.

Ein Nachteil dieser Variante kann sein, daß nicht jede Änderung beim Empfänger tatsächlich benötigt wird und es dadurch zu unnötiger Netzkommunikation kommt. Dies sollte der Modellierer beachten und gegebenenfalls eine weitere Variable definieren, welche nur dann auf den Wert der ursprünglichen Variable gesetzt wird, wenn deren Wert zu anderen Federates übertragen werden soll. Solche Modelländerungen sind bei

ausreichender Performance der verteilten Simulation unnötig, ansonsten jedoch i.d.R. überschaubar mit wenigen Codezeilen durchführbar.

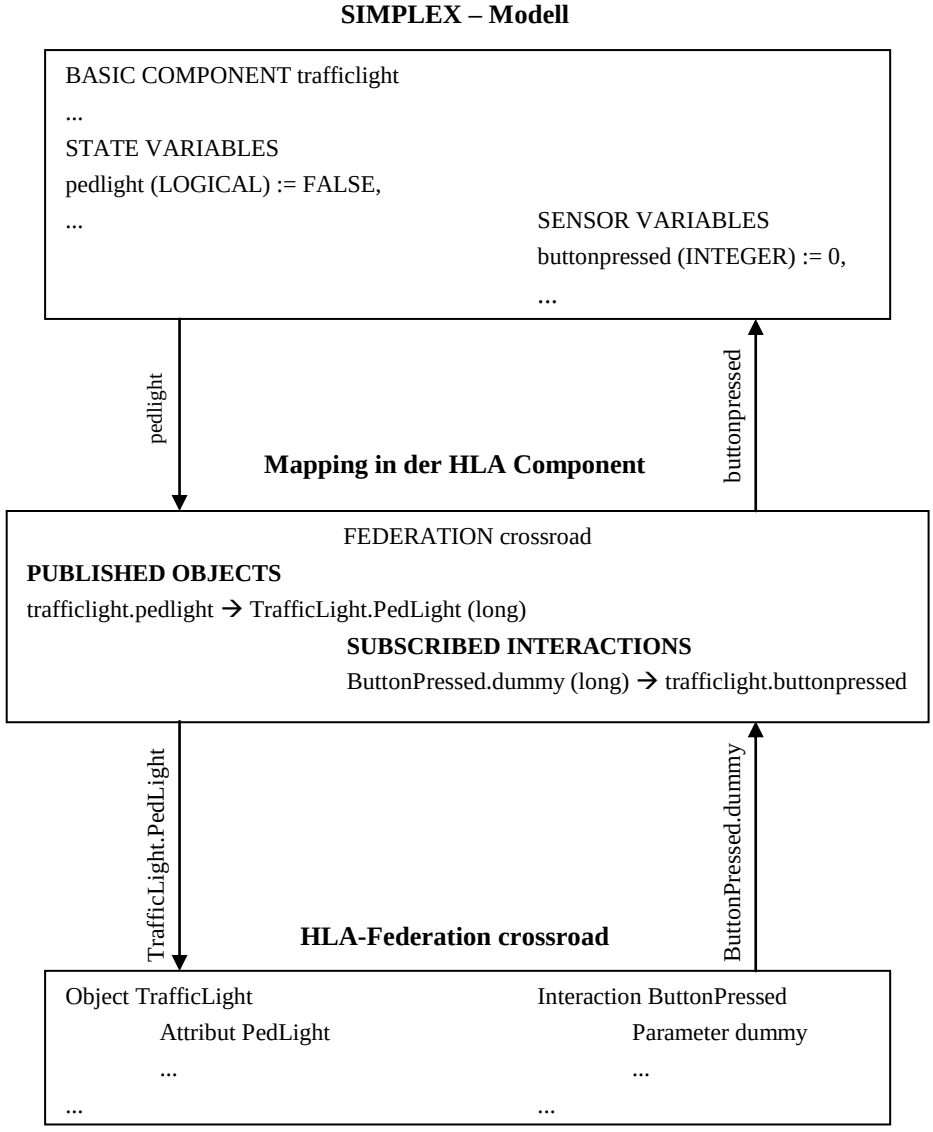

Abbildung 3: Mapping zwischen Modellgrößen und HLA-Daten

Nicht automatisiert werden kann jedoch die Erstellung des Simulation Object Model und des Federation Object Model. Letzteres muß nur einmal für die gesamte verteilte Simulation entworfen werden, ersteres jedoch individuell zu jedem Federate. Zur Unterstützung dieses Prozesses existieren bereits Werkzeuge, die lediglich Grundkenntnisse über die in HLA verfügbaren Datenstrukturen voraussetzen.

Synchronisation

Da es sich bei HLA um eine Architektur zur verteilten Simulation handelt, ist neben dem Datenaustausch mit anderen Federates für eine HLA-Anbindung die Synchronisation mit anderen Federates von Bedeutung. Grundsätzlich werden von HLA Basisfunktionen bereitgestellt, mit denen sich die verschiedenen Synchronisationsmechanismen (konservativ, optimistisch, Echtzeit) implementieren lassen. HLA schreibt dabei nicht vor, daß alle Federates ein gemeinsames Verfahren zur Synchronisation nutzen müssen. Es können somit prinzipiell z.B. konservativ synchronisierte Federates mit optimistisch arbeitenden Federates interagieren.

Da es sich bei SIMPLEX, wie auch bei SLX um ein klassisches stand-alone Simulationssystem handelt, wurde für die Kopplung vorerst ein konservatives Synchronisationsprotokoll mit Lookahead realisiert. Ein Lookahead ist nur aus dem Modell an sich ableitbar, deshalb muß für eine allgemeingültige automatisierte Lösung mit einem Lookahead von Null gearbeitet werden. Die für SIMPLEX III implementierte Lösung sieht daher z.Zt. vor, für jedes interne Ereignis gemäß den HLA-Regeln den Zeitfortschritt zu beantragen, bis zum gewährten Zeitpunkt fortzuschreiten, dann entweder mögliche externe Ereignisse zu bearbeiten oder das nächste interne Ereignis abzuarbeiten. Hierfür wurde die Hauptereignisschleife des SIMPLEX-Laufzeitsystems (siehe [6]) um die entsprechende Funktionalität erweitert (Abbildung 4).

Für die in SIMPLEX ebenfalls möglichen kombinierten oder rein kontinuierlichen Modelle stellt sich die Problematik wesentlich komplizierter dar. Das HLA-Synchronisationskonzept sieht das Beantragen des Zeitfortschritts *vor* dem eigentlichen Fortschreiten vor. Gerade dies ist bei kontinuierlichen Systemen i.d.R. nicht möglich, da sich erst durch das Lösen der Gleichungen an sich feststellen läßt, ob ein für die Simulation relevantes Ereignis vorliegt. Da das Lösen der Differentialgleichung durch Fortschreiten mit relativ kleinen Schritten erfolgt, ist es aus Perfomancegründen höchst ungünstig, für jeden dieser Schritte den Zeitfortschritt zu beantragen. Im Falle von eng gekoppelten Federates, bei denen ein Federate direkt von den Ausgangsgrößen der Differentialgleichungen in einem anderen Federate abhängt, läßt sich diese Variante nicht vermeiden. Daher sind eng gekoppelte Systeme wenn möglich zu vermeiden.

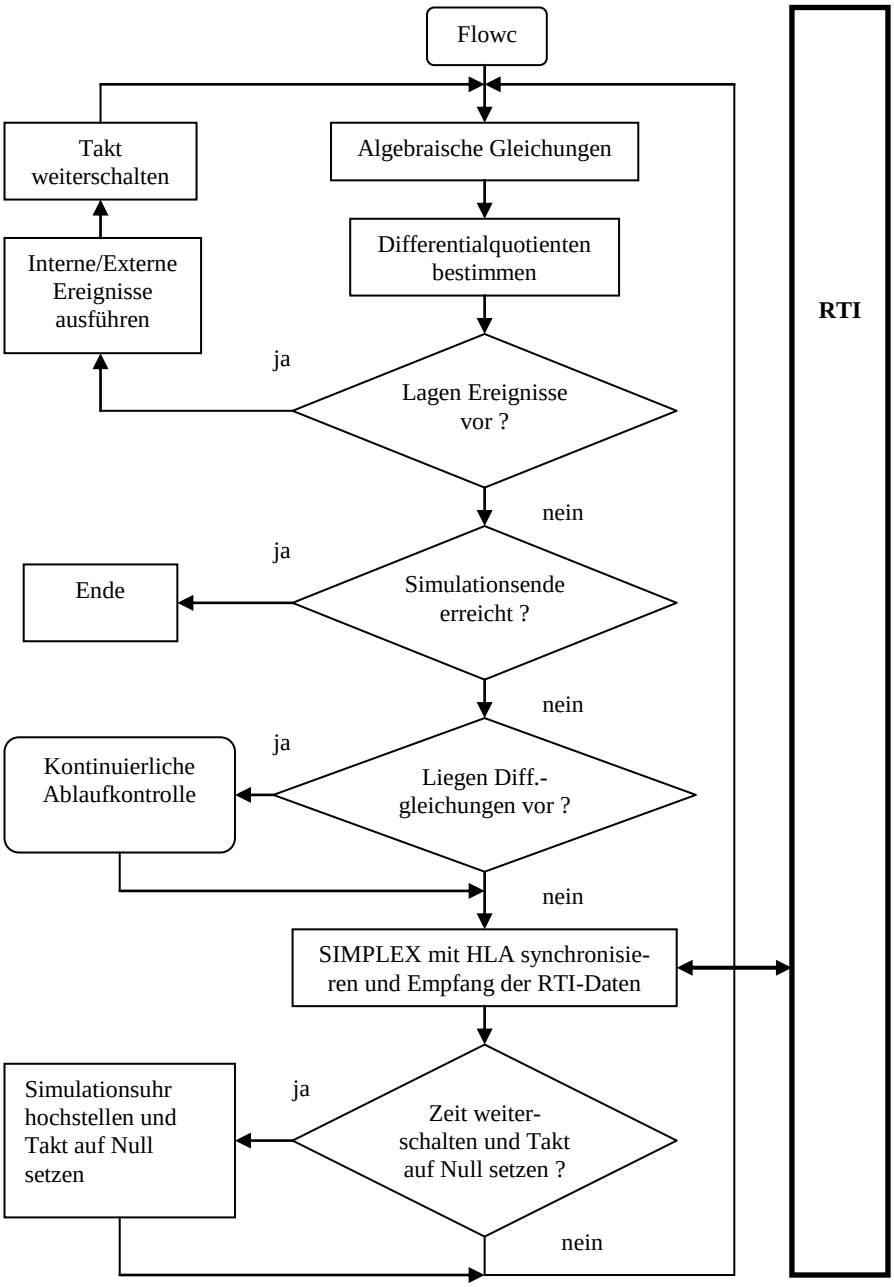

Abbildung 4: Integration der zeitlichen Synchronisation mit der RTI in das Laufzeitsystem von SIMPLEX III

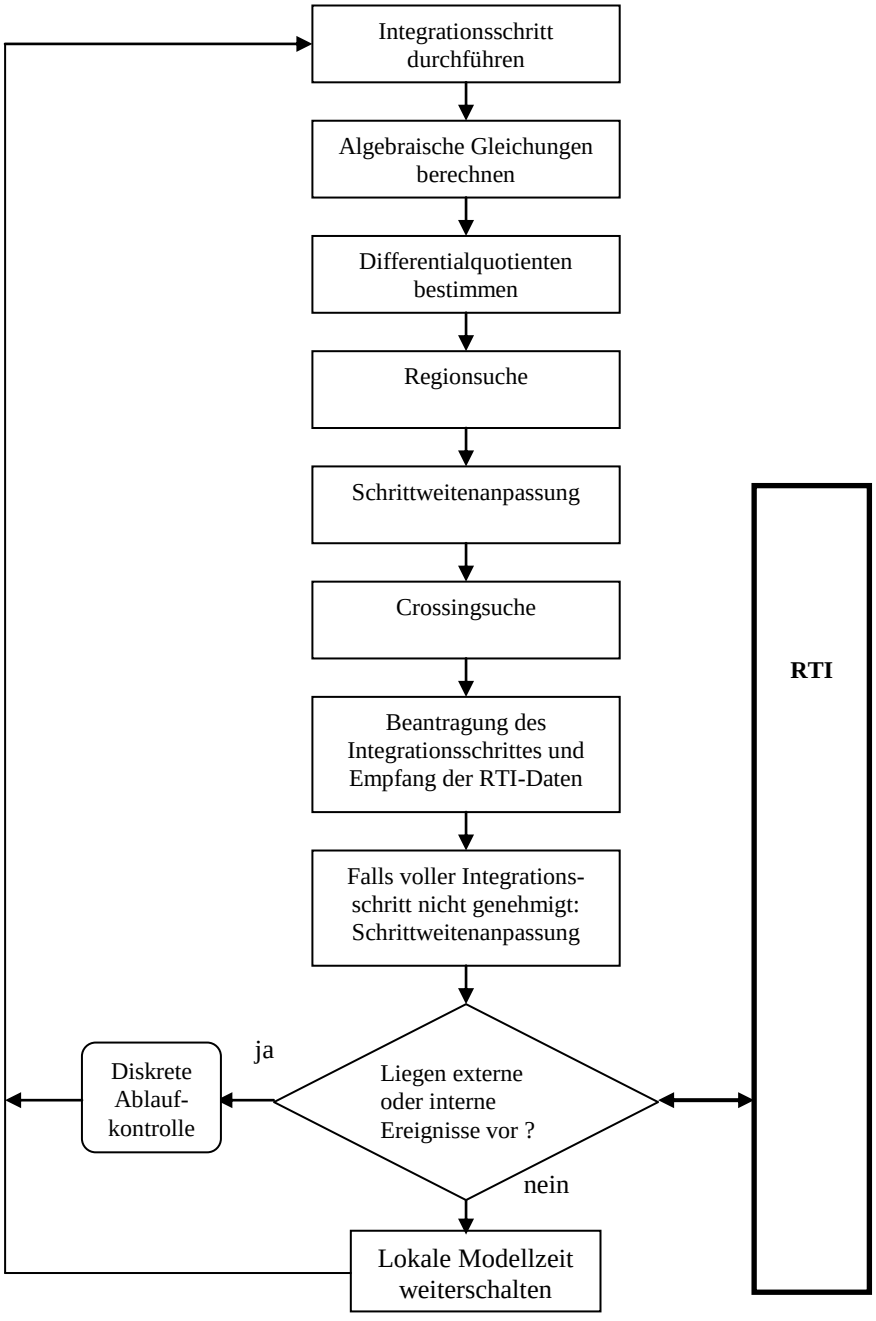

Abbildung 5: Zeitliche Synchronisation mit der HLA bei kontinuierlichen Modellen von SIMPLEX III

Ein Ansatz für das Synchronisationsproblem bei lose gekoppelten kombinierten Modellen besteht darin, die HLA-Regeln quasi zu umgehen: Es könnten zuerst die Differentialgleichungen gelöst und somit der nächste Ereigniszeitpunkt bestimmt werden. Im zweiten Schritt könnte dann (quasi nachträglich) der Zeitfortschritt zu dem bestimmten Ereigniszeitpunkt beantragt werden. Daraufhin können zwei Fälle eintreten: Der Zeitfortschritt wird wie gewünscht gewährt. In diesem Fall kann normal fortgefahren werden. Der zweite Fall ist der weitaus schwierigere: Es wird mitgeteilt, daß vor dem beantragten Zeitpunkt noch ein anderes (externes) Ereignis mit einem kleineren Zeitstempel bearbeitet werden muß. Das Problem in diesem Falle ist, daß nur bis zu diesem Zeitpunkt hätte integriert werden dürfen und somit zu weit integriert wurde. Der Lösungsansatz hierfür ist, vor jeder Integration den aktuellen Zustand des Modells zu speichern und in diesem Falle wiederherzustellen (Roll-Back) und bis zum ‚richtigen' Zeitpunkt zu integrieren.

Dies erfordert jedoch relativ tiefgreifende Eingriffe in das Simulationssystem, gestattet es aber, den Aufwand der zeitlichen Synchronisationen auf das absolut notwendige Maß zu minimieren. Dieses Minimum wird erreicht, wenn eine zeitliche Synchronisation in einem Federate nur noch im Falle einer Datenkommunikation mit anderen Federates erfolgt. Der Vorteil dieser Lösung ist, das sie gleichermaßen für kontinuierliche wie auch für diskrete Modelle anwendbar ist. Zur Zeit wurde jedoch nur eine Lösung implementiert, die jeden internen Zeitfortschritt im SIMPLEX-Modell mit HLA koppelt (Abbildung 5).

4 Referenzbeispiele

4.1 Modell einer Fußgängerampel

Als erstes Referenzbeispiel zur Kopplung von SIMPLEX III und SLX wurde das wohlbekannte Modell einer Fußgängerampel herangezogen [7]. Dieses rein diskrete Modell hat den Vorteil, daß es sehr überschaubar ist und trotzdem alle zur Erprobung eines HLA-Interfaces wichtigen Komponenten enthält. Die Federation besteht aus drei Federates. Diese simulieren

a) eine Fußgängerampel
b) einen Fußgängerstrom und
c) einen Fahrzeugstrom.

Die Ampel wird als Objekt im Sinne von HLA modelliert. Der Vorgang der Ankunft eines Fußgängers mit Grünanforderung stellt eine Interaktion im Sinne von HLA dar. In dem realisierten Prototyp sind jeweils alle Federates sowohl in SIMPLEX als auch in SLX implementiert und können beliebig kombiniert werden. Die HLA-Objektmodelle der drei SLX-Federates sind identisch mit denen der SIMPLEX-Federates. Nur das Modellverhalten der Teilmodelle (z.B.: Umschaltzeiten der Ampel) unterscheidet sich leicht, läßt sich jedoch durch die Wahl anderer Parameter für die Teilmodelle problemlos egalisieren.

Neben der HLA-basierten Kopplung der Modelle können die drei Basiskomponenten des Modells in SIMPLEX mittels einer zusätzlichen High Level Component auch alternativ als monolithisches SIMPLEX-Modell ausgeführt werden. Hierfür ist keine Änderung der Modellspezifikation der SIMPLEX-Modelle notwendig.

4.2 Modell einer Abfüllanlage aus der Verfahrenstechnik

Das zweite Referenzbeispiel, welches auch eine kontinuierliche Komponente enthält, modelliert eine verfahrenstechnische Abfüllanlage, in der die einzelnen Fässer kontinuierlich gefüllt und dann mittels eines Logistikmodells transportiert werden. Die Federation besteht lediglich aus zwei Federates:

a) der Abfüllanlage selbst sowie
b) einem Großhändler

Der Füllprozeß wird durch eine kontinuierliche Funktion dargestellt und in SIMPLEX simuliert. In der Abfüllanlage werden Fässer gefüllt und auf Anforderung an einen Großhändler geliefert. Diese Anforderung bzw. Bestellung wird unter HLA als ein Objekt modelliert. Bestätigt bzw. abgewiesen wird die Bestellung ebenfalls durch ein HLA-Objekt. Zur Abfüllanlage gehören noch zwei Lager mit beschränkter Kapazität für die Fässer. Eines ist für das Leergut, das andere beherbergt die gefüllten Fässer. Sind die Lager voll, kann kein Leergut mehr abgenommen bzw. kein leeres Faß mehr gefüllt werden.

Der Großhändler hat neben der Bestellung der Fässer lediglich logistische Probleme zu bearbeiten. Dies wird unter SLX simuliert. Das Hauptlager des Großhändlers sitzt ebenfalls in Passau und transportiert die bestellten Fässer nach einer bestätigten Bestellung aus der Brauerei mittels LKW zu den einzelnen Zweigstellen. Bei der Rückfahrt wird eventuell vorhandenes Leergut zur Brauerei gefahren. Die Transporte werden unter HLA als Interaktionen modelliert. Zweigstellen befinden sich außerdem noch in Dresden und Magdeburg. Die Auslieferung der Fässer vom Großhändler zum Händler wird vereinfacht durch eine exponentialverteilte zeitliche Verweilzeit jedes Fasses beim Großhändler nachgebildet, nach der das Faß vom Zustand gefüllt zum Zustand leer wechselt.

Dieses Referenzbeispiel demonstriert, daß man verschiedene Simulationssysteme mit ganz unterschiedlichen Fähigkeiten und Aufgabenfeldern (z.B. diskret und kontinuierlich) koppeln kann. Hierin liegt auch ein mögliches, zukünftiges Einsatzgebiet von HLA.

5 Zusammenfassung und Ausblick

Mittels des HLA-Standards ist es möglich, von Grund auf verschiedene Simulationssysteme für unterschiedliche Anwendungsgebiete und Zielgruppen miteinander zu koppeln. Daraus ergeben sich teilweise völlig neue Anwendungsmöglichkeiten, da durch HLA jedes Teilmodell in dem am besten geeigneten Simulationssystem modelliert und mit einem anderen Teilmodell in einem anderen Simulationssystem gekoppelt werden kann. Diese Kopplung kann auch kontinuierliche Modelle in einer verteilten Simulation aufnehmen.

Um eine Kopplung über HLA realisieren zu können, ist es für ein Simulationssystem notwendig, eine HLA-Schnittstelle zu implementieren. Hierfür sind verschiedene Konzepte möglich, die zum einen darauf basieren, dem Nutzer die Arbeit mit HLA praktisch gänzlich abzunehmen (SIMPLEX III) oder andererseits den Ansatz verfolgen, ihm vom Modell aus einen komfortablen Zugriff auf das HLA-Interface zu gewähren (SLX). Beide Lösungen haben ihre Vor- und Nachteile und somit ihre Berechtigung. Die Lösung für SIMPLEX III besticht durch ihre Nutzerfreundlichkeit, da kein „Hinzulernen“ oder Umdenken bei der Modellerstellung notwendig ist, hat aber gewisse Nachteile bezüglich der Nutzbarkeit aller von HLA gebotenen Möglichkeiten. Die Lösung für SLX bietet diese größere Flexibilität (z.B. dynamische Lookahead-Werte, nutzerdefinierte Update-Intervalle, etc.), hat jedoch den Nachteil der höheren „Learning Curve“, da am Modell Erweiterungen bezüglich HLA vorzunehmen sind (Synchronisation, Datenaustausch, Berücksichtigung externer Ereignisse). Durch Nutzung von Hybridformen aus diesen beiden Lösungen könnten die Vorteile beider Lösungen kombiniert werden. Zur Implementierung solcher Hybridformen bietet sich aufgrund bisher gesammelter Erfahrungen und der Verfügbarkeit des Quelltextes von SIMPLEX an.

Durch die Möglichkeit, ein Modell auf verschiedene Simulationssysteme zu verteilen, bieten sich auch optimistische Simulationsverfahren zur Modellausführung an. Die Integration derartiger Verfahren in klassische Simulationssysteme wäre für die Zukunft wünschenswert.

Referenzen

[1] Defense Modeling and Simulation Office (DMSO). *The High Level Architecture Homepage*. Online verfügbar unter http://hla.dmso.mil. 1998.

[2] 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.

[3] Straßburger, S., T. Schulze, U. Klein, J.O. Henriksen. 1998. *Internet-based Simulation using off-the-shelf Simulation Tools and HLA.* In Proceedings of the 1998 Winter Simulation Conference, eds. Medeiros, D.J. and E. Watson, Washington D.C.

[4] Klein, U., T. Schulze, S. Straßburger, H.-P. Menzler. *Distributed Traffic Simulation based on the High Level Architecture*. In: Proceedings of the Simulation Interoperability Workshop, Fall 1998.

[5] Straßburger, S., U. Klein. *Integration des Simulators SLX in die High Level Architecture*. Tagung Simulation und Visualisierung 1998 Magdeburg. Lorenz, P., Preim, B. (eds.), SCS Europe Publishing House, Delft, Erlangen, Ghent, San Diego.

[6] Schmidt, B. 1995. *Simplex II - Benutzerhandbuch*. SCS Publications, San Diego 1995.

[7] Schulze, T., U. Klein, S. Straßburger, H.-P. Menzler. *Die High Level Architecture (HLA) in der Straßenverkehrssimulation*. 12. Symposium Simulationstechnik ASIM 98, Zürich.

[8] Apsel, Th. *Konzeption und Aufbau eines universell einsetzbaren Simulationssystems*. in: Kerckhoffs, E., Lehmann, A., Pierreval, H., Zobel, R.: Advances in Simulation. SCS, Erlangen, Ghent, Istanbul, San Diego (1996).

[9] Reger, K. *Konzeption und Realisierung der Konfigurierbarkeit universeller Simulationssysteme*. in: Kerckhoffs, E., Lehmann, A., Pierreval, H., Zobel, R.: Advances in Simulation. SCS, Erlangen, Ghent, Istanbul, San Diego (1996).

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

[11] Lantzsch, G. *HLA-Interface für SIMPLEX III*. Diplomarbeit 1998. Technische Universität Dresden.