Web-basierte Animation verteilter Simulationen auf Basis der High Level Architecture (HLA)

K.-C. Ritter / U. Klein / S.Straßburger / M. Diessner
Universität Magdeburg, Institut für Simulation und Graphik
Postfach 41 20
D-39016 Magdeburg
Email: {ritter@isg, uklein@isg, strassbu@sunpool, diessner@sunpool}.cs.uni-magdeburg.de

Zusammenfassung

Der Beitrag befaßt sich mit der High Level Architecture, die die Erstellung interoperabler und wiederverwendbarer Simulationsmodelle unterstützt. Zur Animation verteilter HLA-Simulationen zur Laufzeit wird das Web-basierte Animationssystem Skopeo eingesetzt, welches als eigenständiger Teilnehmer konzipiert ist. Der Prozeß der Integration von Skopeo in die HLA, die Anforderungen einer Visualisierungskomponente an HLA-basierte verteilte Systeme sowie die sich dadurch ergebenden neuen Möglichkeiten werden beschrieben.

1 Motivation

Das Gebiet der verteilten Simulation auf der Basis kooperierender, interoperabler Teilsysteme hat durch die High Level Architecture (HLA) neue starke Impulse bekommen. Aus dem militärischen Anwendungsfeld kommend und als Nachfolger solcher Protokolle wie DIS und ALSP gedacht, wird es vorwiegend zur Kopplung größerer, mit eigener Visualisierung ausgestatteter Komponenten eingesetzt.

Im Rahmen verschiedener Arbeiten, die Möglichkeiten untersuchen HLA auch in zivilen Anwendungsfeldern zu etablieren, war und ist es notwendig, die im zivilen Bereich üblichen Simulationstools auf HLA-Kompatibilität zu untersuchen bzw. geeignet zu erweitern. So wie dies derzeit in den Simulationssprachen GPSS/H und insbesondere SLX möglich ist, Simulationsmodelle als vollwertige Mitglieder einer HLA-basierten verteilten Simulation zu entwickeln und einzusetzen, beschreibt dieser Beitrag die analogen Arbeiten auf dem Gebiet der Animation: Ziel ist die Integration eines Animationstools in die HLA-Welt. Hierzu wurde das Java-basierte Visualisierungssystem Skopeo[6] auf die Integrationsmöglichkeiten hinsichtlich HLA untersucht und später um Module für die Kommunikation mit HLA erweitert.

Über die rein passive Beobachterrolle, die HLA-Skopeo zunächst einnimmt und auf die sich die nachfolgenden Ausführungen beziehen, hinaus lassen sich bereits vielfältige weitere Einsatzmöglichkeiten erkennen, wie sie sich z.B. über interaktive Visualisierungskomponenten erreichen lassen; über diese aktive Rolle wird gegen Ende des Beitrages berichtet (siehe Abschnitt 3.6.).

2 Die High Level Architecture

Die HLA bietet Unterstützung bei der Umsetzung der Forderungen nach Wiederverwendbarkeit und Interoperabilität von Simulationsmodellen. Außerdem erlaubt sie, die Schwierigkeiten mit monolithischen Modellen bei Änderungen der Funktionalität oder Konnektivität zu umgehen.

Als Kommunikationsgrundlage wird eine einheitliche Schnittstelle (*HLA Interface*) definiert, welche die Teilnehmer eines gemeinsamen Simulationslaufs (*Federates*) bereitzustellen haben, um in Kontakt mit der *Runtime Infrastructure* (RTI), die zur Laufzeit Basis-, Koordinations- und Kommunikationsdienste bereitstellt, zu treten. 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 definierten Form zu dokumentieren (*Object Model Template*). Weiterhin legt HLA zwingende Verhaltensregeln für Federates und Federations fest. Ein Vorteil der Architektur ist die Offenheit für verschiedene Arten von Federates; sowohl Softwarebausteine wie z.B. Datenbanken und Beobachter als auch reale Elemente wie Sensoren und andere Hardware können bei Einhaltung der HLA-Spielregeln bei Federation Executions mitwirken (Bild 1).

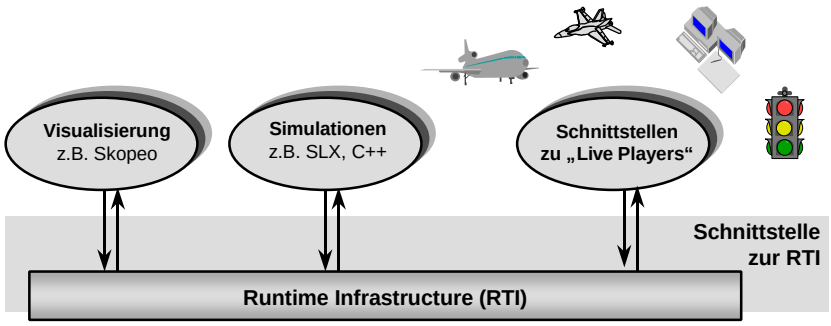

Bild 1: *Aufbau der High Level Architecture*

Wesentliche Elemente des Erfolges von HLA sind das transparente Zeitmanagement, die "intelligente" Art des Datenaustauschs sowie die bereits angesprochenen Objektmodelle.

Das HLA Zeitmanagement erlaubt es, die Art und Weise des lokalen Zeitfortschrittes gegenüber anderen Federates transparent zu halten. Zwischen zeitschrittgesteuerten, ereignisorientierten, kontinuierlichen und echtzeitgetriebenen Federates kann eine Interoperabilität hergestellt werden, die konservative und optimistische Synchronisationsverfahren nebeneinander erlaubt.

Primär wird das zeitliche Verhalten eines Federates dabei durch zwei Eigenschaften charakterisiert:

- *time constrained*: dieser Schalter gibt an, ob das Federate durch den Zeitfortschritt der anderen Federates beeinflußt wird.
- *Time regulating*: hiermit kann eingestellt werden, daß die RTI die Zeit des Federates bei der Berechnung des Zeitfortschritts anderer Federates berücksichtigt.

Der Datenaustausch zwischen den Federates wird durch ein sogenanntes Interessenmanagement etabliert, durch das die einzelnen Federates bei Eintritt in die Federation Execution mitteilen, welche Informationen (Objekt- und Interaktionsklassen) sie publizieren (modellieren und bekanntgeben) können bzw. welche Informationen sie abonnieren wollen. Dadurch ist es der RTI zur Laufzeit möglich festzustellen, ob und zu welchen Federates Updates zu liefern sind.

Das Data Distribution Management (DDM) erlaubt eine noch genauere dynamische Eingrenzung der zu übertragenden Daten über den Wertebereich der Objektattribute (z.B. Einschränkung auf alle sichtbaren Objekte eines abonnierten Typs) und erlaubt somit eine drastische Reduzierung des Kommunikationsaufwandes.

Die RTI-Software setzt intern auf TCP/IP auf und ermöglicht damit eine umfassende Flexibilität von der Verteilung der Federates bis hin zur Wahl des Kommunikationsmediums.

Die Schnittstelle zwischen Federate und RTI bedient sich des Ambassador-Paradigmas, welches von der Entsendung von Botschaftern zur jeweils anderen Softwareseite ausgeht. Das RTI (vertreten durch eine einzubindende Softwarebibliothek) entsendet den RTIAmbassador, welcher Funktionsaufrufe des Federate entgegennimmt; umgekehrt hat das Federate den bereits abstrakt vorhandenen FederateAmbassador auszuimplementieren, um Funktionsaufrufe der RTI ausführen zu können. Über die konkrete Umsetzung der Skopeo-Anbindung durch den SkopeoAmbassador wird später berichtet.

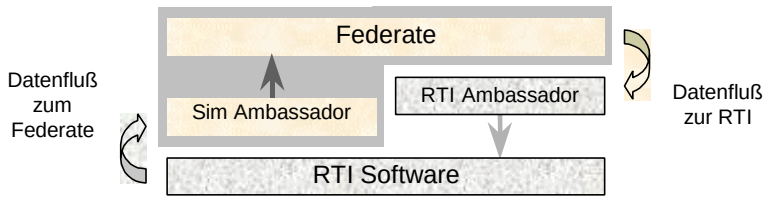

***Bild 2**: Das Botschafterparadigma der Federate-RTI-Kommunikation*

2 Verknüpfung von Skopeo und HLA

Mit der Größe und Komplexität der Systeme nimmt auch die Bedeutung der Animationskomponente zu. Dieses gilt insbesondere für verteilte Systeme, wie sie durch HLA aufgebaut werden.

In diesem Abschnitt wird eine Lösung vorgestellt, wie durch den Einsatz des Web-basierten Animationssystems Skopeo eine plattform- und damit ortsunabhängige Lösung

für die Visualisierung in HLA-Federations bereitgestellt werden kann. Es wird auf Anforderungen, Schwierigkeiten und Lösungswege für die Implementierung verwiesen.

2.1. Konzeptionelle Anforderungen

In der verteilten Simulation können voneinander abhängige Simulationen ihren lokalen Zeitfortschritt nicht mehr autonom durchführen; vielmehr ist dieser (bei HLA mittels der RTI) zu koordinieren [2, 3]. So ist es z.B. bei ereignisgesteuerten Simulatoren nötig, die Ereignisketten von sich am Zeitfortschritt beteiligenden Federates miteinander zu synchronisieren bzw. verknüpfen. Im Falle eines rein passiven Animationsfederates ist sicherzustellen, daß die (ggf. von einer Vielzahl von Federates stammenden) Visualisierungsinformationen in kausaler Reihenfolge empfangen werden und der Zeitfortschritt der Animation sich dem jeweiligen Zeitfortschritt der Federation, in der das Animationsfederate Mitglied ist, anpaßt, ohne selbst Einfluß auf das Zeitmanagement anderer Federates zu nehmen.

Hiervon unbeeinflußt kann die Visualisierungszeit bleiben, die - bei einer Kopplung von Federation - und Visualisierungszeit- dem aktuellen Federationstatus entspricht, die bei einer Entkopplung jedoch trotz weiterer Beobachtung der Vorgänge in der Federation eine eigenständige Visualisierung nach anderen Kriterien (Echtzeit, Visualisierung vergangener Zeitabschnitte etc.) erlaubt.

Eine weitere konzeptionelle Anforderung betrifft den Datenaustausch zwischen Federates sowie die Notwendigkeit der Darstellung externer Daten (Objekte, Attributwerte) im Federate (*ghosten*). Das HLA Konzept sieht vor, daß ein Federate für sich interessante Daten, die durch andere Federates modelliert werden, „abonniert“ und in der Folgezeit über Änderungen bezüglich dieser Daten informiert wird und diese ggf. intern abbildet. Im Falle eines Animationsfederates sind alle zu animierenden Objektklassen zu abonnieren, so daß die notwendigen Visualierungsinformationen vollständig über den Update-Mechanismus des RTI bezogen werden können.

Damit in engem Zusammenhang steht die Notwendigkeit, die Informationen in das Objektmodell zu integrieren, die für die Animation benötigt werden und über die meist rein logischen Zusammenhänge, die eine Simulation benötigt, hinausgehen: z.B. Geometrieinformation, Orte, Pfade etc. Auf diesem Gebiet besteht eine breite Variationsvielfalt, und die beiden folgenden Varianten spiegeln dies wieder:

- Die Visualisierungsinformationen wie z.B. Layout, Hintergrund, Pfade etc. werden während der Vorbereitung außerhalb des gemeinsamen Ereignisraumes der Federation, also allein im Bereich des Animationsfederate, erstellt.
- Die Visualisierungsinformation wird vollständig in das Objektmodell integriert, so daß die notwendigen Informationen zur Laufzeit übertragen werden; dies erlaubt eine dynamische Gestaltung der Visualisierung bis hin zur Integration von Darstellungsobjektattributen (z.B. in VRML2).

2.2 Programmierungstechnische Anforderungen

Bei der Beschreibung der Federate-RTI-Kommunikation wurde bereits auf das Botschafterparadigma und die Notwendigkeit eingegangen, eine Softwarebibliothek einzubinden und einen FederateAmbassador (hier: SkopeoAmbassador) zu implementieren, dessen Funktionen von der RTI-Softwarebibliothek aufrufbar sein müssen (Callback Functions).

Die RTI-Software liegt zur Zeit in C++ für zahlreiche Unixvarianten sowie für Windows95/NT vor sowie in einer plattformunabhängigen Java-Version (zur Zeit noch Beta-Stadium).

Da Skopeo am Institut für Simulation und Graphik entwickelt wurde und der Programmkode vorliegt, war die Ausgangssituation zur Erfüllung der programmiertechnischen Anforderungen besser als bei zahlreichen anderen untersuchten Werkzeugen. Weiterhin unterstützt die hohe Modularität von Skopeo die problemlose Anbindung weiterer Kommunikationsmodule.

Unsere Untersuchungen zeigten, daß es weniger die konzeptionellen, sondern vielmehr die programmierungstechnischen Forderungen sind, die potentielle Federates vor schwierige Aufgaben stellen.

3 Skopeo – Systemanimation im Web

In diesem Abschnitt wird die Architektur von Skopeo kurz vorgestellt. Es werden Visualisierungs-, Kommunikations- und Interaktionsmöglichkeiten aufgezeigt. Einige Punkte der Implementation werden kurz angerissen.

3.1. Skopeo – Das System

Skopeo basiert auf Java Technologie und ist geeignet, in jedem Java-fähigen Standard Web Browser, wie Netscape oder dem Microsoft Internet Explorer, zu arbeiten. Es läuft sowohl als Stand-alone Applikation als auch als Applet in einem Browser. Der Kernel ist vom jeweils ausgeführten Modell und vom aktuellen Betriebssystem unabhängig. Ist eine Animation einmal geladen, läuft sie komplett auf der Client-Maschine [12].

Eine offene und modulare Systemarchitektur des Skopeo-Kernels, die in Bild 3 gezeigt wird, ist die Voraussetzung für eine Arbeit im Web, ermöglicht doch erst diese Architektur die Anbindung unterschiedlichster Datenquellen an einen Animationskernel, der in einem „Sandbox“-Environment beim Client läuft.

Für den Skopeo-Kernel gibt es keine Unterschiede zwischen Echtzeitdaten (im Folgenden als online-Daten (onD) bezeichnet) und in Trace-Dateien konservierten Daten (offline-Daten, ofD). Beide Datenformen werden durch die Systemuhr beobachtende Threads in die Datenbasis des Kernels eingebracht.

Diese Architekturbesonderheit, die aus den Notwendigkeiten von Multithread- und Multiuserbetriebssystemen heraus geboren wurde, legte die Idee nahe, Skopeo in HLA-Szenarien zu integrieren.

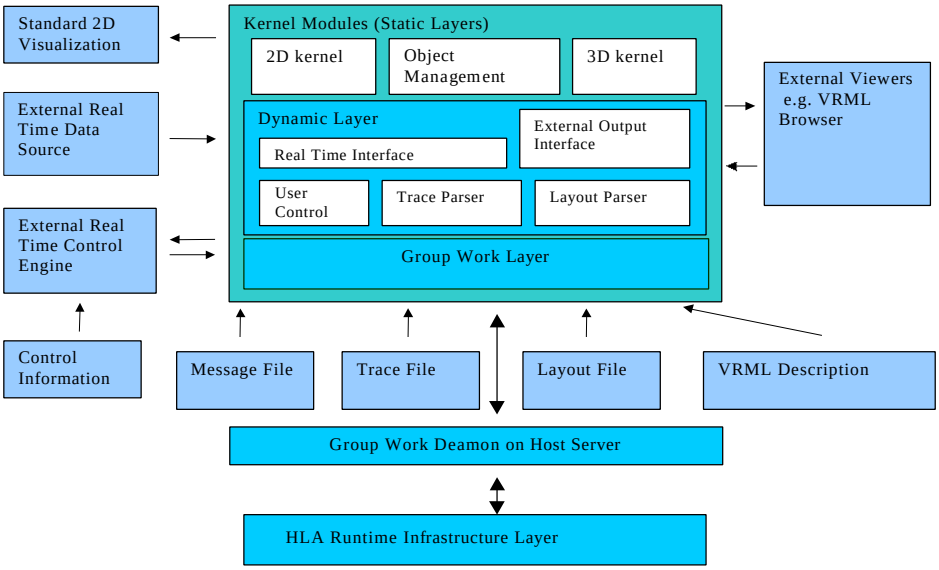

***Bild 3**: Architektur von Skopeo*

An dieser Stelle stellt sich die Frage, wie lassen sich Daten in die geschützte Umgebung beliebiger Web-Browser, „Sandbox“ genannt, liefern.

3.2. Kommunikationsgrundlagen zwischen Animation und Federation

Für die ofD wurde ein einfacher Deamon auf dem Skopeo-Host-Server installiert, der die vom Client angeforderten Tracedaten von beliebigen URLs holen und an den Client weiterverschicken kann. Ein wenig komplizierter verhält es sich im Bereich der onD. Für den gesamten Bereich der onD wurde ein Arbeitsgruppenkonzept (Workgroup WG) für verteilte Animationen im Web entwickelt und bereits implementiert.

Dieses WG ermöglicht eine umfangreiche Kommunikation aller angeschlossenen Web-Browser untereinander und von beliebigen externen Datenquellen mit allen Klienten. Wobei das Konzept streng nach den von Java auferlegten Sicherheitsrestriktionen arbeitet. Diese WG, die ursprünglich für die Kommunikation von Skopeo-Sitzungen untereinander konzipiert war, wurde so erweitert, daß ein HLA-Ambassador als Teil der WG agieren kann.

Für den Aufbau einer WG hätten sich grundsätzlich folgende Verfahren angeboten: Sockets, HTTP/CGI, RMI oder DCOM. Vergleicht man diese Verfahren hinsichtlich Plattformunabhängigkeit, Abstraktionsebene, Leistung, Sicherheit, Standardisierung,

Sprachunabhängigkeit und Bindungsfähigkeit an andere Applikationen, so bildet CORBA die z.Zt. beste Alternative. Basis für das WG bietet die folgende Architektur:

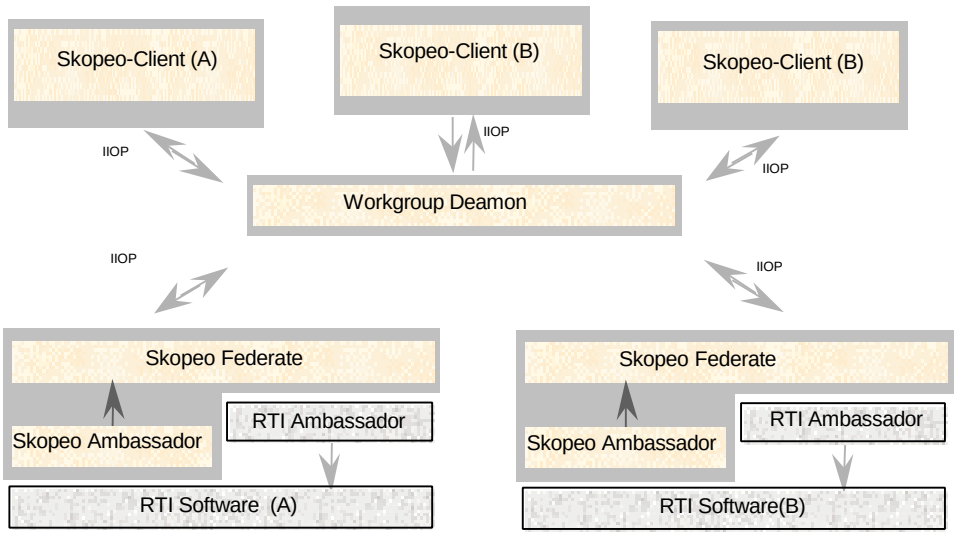

***Bild 4**: Genereller Aufbau der WG*

Diese im obigen Bild dargestellte Architektur ist nahezu die einzig mögliche Architektur, um den Zugriff beliebiger Web-Klienten auf HLA-Federationen zu gewährleisten.

Eine weitere interessante Variante, die der Skopeo-Kernel bietet, auf die aber an dieser Stelle nicht weiter eingegangen werden soll, ist die Verknüpfung von onD und ofD, da mehrere Datenströme zur selben Zeit in die Datenbasis des Kernels eingespeist werden können. Hieraus ergeben sich Möglichkeiten des Vergleiches zwischen verschiedenen Systemen.

3.3. Visualisierung

2D-Standardausgabe

Die Standardausgabe des Skopeo-Kernels bildet ein außerhalb der HTML-Seite erzeugtes skalierbares Fenster (vgl. Bild 5, rechts), das durch seine Größe an die Rechenleistung der Client-Rechner angepaßt werden kann.. In diesem Fenster wird die Szene mittels zweidimensionaler Vektorgraphik dargestellt. Diese Visualisierung ist unabhängig vom verwendeten Betriebssystem und steht auf jedem Client-System zur Verfügung.

Die ersten Versionen von Skopeo benutzen einen Visualisierungsbereich fester Größe und Position direkt in der HTML-Seite. Diese Art der Visualisierung stellte sich jedoch

durch die unterschiedlichen Leistungen der Client-Maschinen als nicht praktikabel heraus und wurde daher durch das oben beschriebene externes Fenster ersetzt.

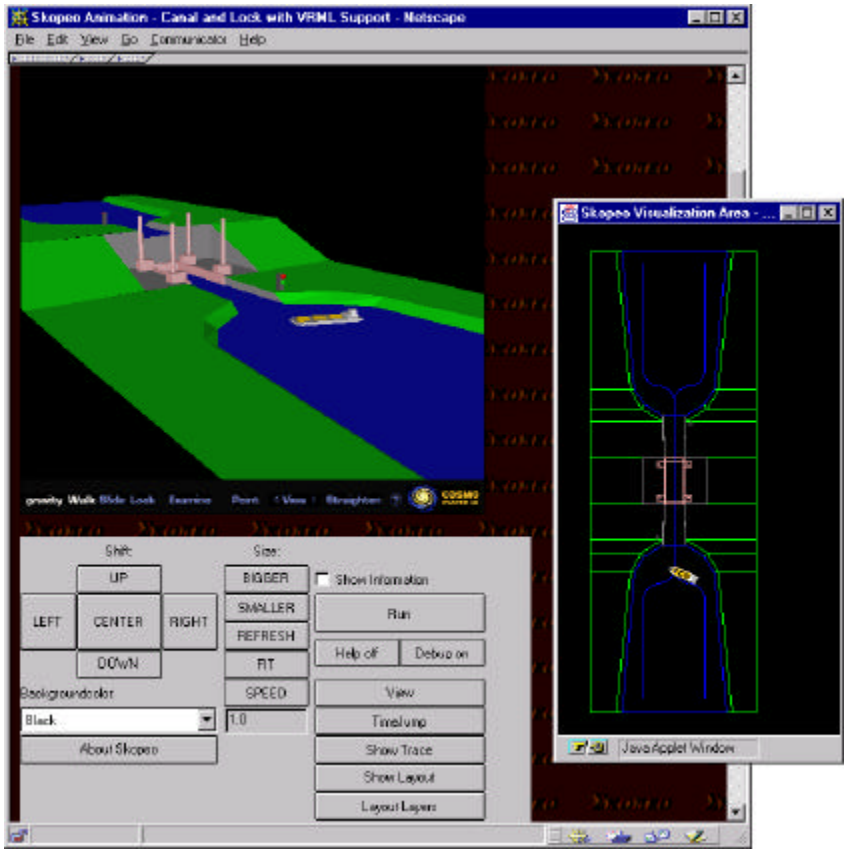

***Bild 5**: 3D und 2D-Animation eines Kanal-Schleuse-Systems mit Skopeo*

VRML 2.0

Neben der 2D-Standardausgabe ist es Skopeo auch möglich, mit anderen Applikationen zu kommunizieren, die sich auf der selben HTML-Seite befinden. So kann eine Visualisierung und Animation über VRML-Browser wie z.B. CosmoPlayer von SGI erfolgen. Diese zusätzlichen Visualisierungskomponenten, die meist in Form von PlugIns zur Verfügung stehen, sind nicht plattformunabhängig. Sie stehen daher nur auf einigen Betriebssystemen zur Verfügung und führen daher zu einer Einschränkung der Portabilität des Gesamtsystems.

VRML ist ein Dateiformat zur Beschreibung von Objekten im Raum. Diese Objekte können außer dreidimensionalen geometrischen Strukturen auch eine Reihe anderer

Elemente enthalten. Dies schießt Töne und Bilder mit ein. Weiterhin lassen sich Wechselwirkungen und Veränderungen der Objekte festlegen. Da zu VRML selber keine Möglichkeiten zur Kommunikation mit anderen Rechnern im Internet, sieht man vom Einbinden anderer Dateien auf Servern im Internet ab, gehören, muß der Datenaustausch, zwischen Szene und Umwelt, über höhere Programmiersprachen realisiert werden. Hierbei wird Java favorisiert [14]. Der Skopeo-Kernel nutzt diese Schnittstellen.

Ein Beispiel für eine Nutzung von VRML ist die Anbindung des CosmoPlayers von SGI für die Bereitstellung von dreidimensionalen Animationen (vgl. Bild5, links oben).

3.4. Zeitmanagement in Skopeo und der Federation

Eine der wesentlichsten Punkte für die Anbindung von Komponenten in eine HLA-Federation ist das Zeitmanagement der einzelnen Komponenten (vgl. Abschnitt 2.1). Skopeo verfügt über eine interne Zeitsteuerung, deren Basis die Systemuhr des Client-Rechners bildet. Aus der Systemzeit werden dann alle Zeitproportionalitäten berechnet.

Der Ablauf der Federation-Zeit ist (mit Ausnahme einer reinen Realtime-Federation bzw. bei Anwesenheit eines schrittmachenden *Pacing*-Federate) im Gegensatz zur Systemzeit unregelmäßig und ist für Zwecke einer zeitkontinuierlichen Animation ungeeignet.

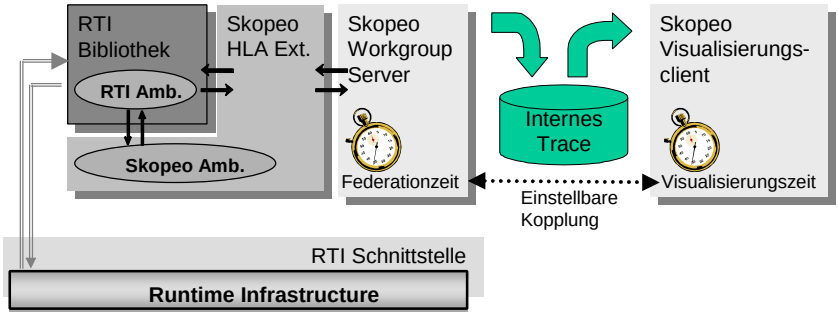

***Bild 6**: Zeitmanagement in der Federation und in Skopeo*

Eine flexible Lösung bietet eine Entkopplung: Federation-seitig hereinkommende Informationen werden umgesetzt und in ein internes Trace geschrieben, welches (sich ggf. stetig füllend) für eine „spätere“ Animation zur Verfügung steht. Dieses Verfahren hat folgende Vorteile:

- es erlaubt eine zeitkontinuierliche Wiedergabe des Trace,
- der Trace ist speicherbar, so daß Skopeo somit eine Rekorderfunktion bietet
- die Animationsgeschwindigkeit läßt sich während des Laufes ändern.

Diese prinzipielle Entkopplung ist jedoch in der Praxis nur dann einsetzbar, wenn es nicht auf die exakte Darstellung des animierten Systems zur aktuellen Federation-Zeit ankommt (vergleichbar einer Post-run Animation). Eine Kopplung von Federation und Skopeo wird in den meisten Fällen zu einer ruckenden Animation führen, deren

Darstellungen jedoch immer die aktuellsten Systemzustände zeigt und damit als Basis für Interaktivität dienen kann.

Somit lassen sich zusammenfassend drei Verfahren für die Synchronisation definieren:

- stets auf aktuellstem Stand (Nachteil: ruckend)
- aktuell-gemittelt: je nach Umfang der Mittelwertbildung steht eher die Aktualität oder die fließende Animation im Vordergrund; kein konstanter Animationszeitfortschritt
- entkoppelt: Nutzer gibt Zeitfaktor für Animation an. Ist die Federation schneller, hinkt die Animation hinterher, ohne jedoch Informationen zu verlieren; droht die Animation die Federation zeitlich zu überholen, ist ggf. mit kurzzeitig unterbrochener Animation zu rechnen.

3.5. Visualisierungsinformationen für Skopeo: Objektmodelle und Schnittstellenfunktionalität

Nach Beitritt in eine Federation meldet HLA-Skopeo über die publish/subscribe-Mechanismen der HLA-Schnittstelle das Interesse an, über alle Objekte und Interaktionen der zu animierenden Objekt- bzw. Interaktionsklassen informiert zu werden (Subscribe-Funktionalität). In der derzeitigen Ausbaustufe, in der lediglich eine rein passive Rolle eingenommen wird, werden keine Objekte publiziert. Sämtliche Visualisierungsinformation gelangt in der Folge über Update-Nachrichten der RTI.

Im Objektmodell sind zur Visualisierung zumindest logische Orte zu definieren, so daß es Skopeo möglich ist, im Zusammenhang mit außerhalb der Federation verfügbaren (Layout-) Information über Objektidentifikation, Objekttyp und Ort eine Animation abzuleiten.

In weiteren Ausbaustufen wird eine umfangreichere Integration der visualisierungsrelevanten Informationen in die Objektmodelle untersucht werden, so insbesondere die attraktive Möglichkeit, neben Geometriedaten auch Darstellungsinformation in Form von VRML2-Knoten als Objektattribut dynamisch zu variieren.

3.6. Interaktivität

Wie bereits angedeutet gibt es grundsätzlich zwei Modi, in denen eine Animation in Beziehung zu HLA stehen kann: passiver oder aktiver Modus. Im passiven Modus zeichnet Skopeo nur für bestimmte Attribute der Modellobjekte. Aus den zeitlichen Änderungen dieser Attribute wird dann die Animation berechnet. Es erfolgt keine Rückkopplung vom Nutzer.

Weit komplizierter ist das Verhalten im aktiven Modus. Hierbei kann der Nutzer direkt in der Animation in die Simulation eingreifen. Diese Interaktivität wird im nächsten Abschnitt beschrieben.

An dieser Stelle sollen unterschiedliche Arten der Interaktion auf Web- und Java-Basis diskutiert werden. Die einfachste Art der Interaktion läßt sich durch in Java implementierte Dialogboxen erreichen. Diesen Dialogboxen stehen sämtliche Standardelemente des API zur Verfügung. Hierzu zählen Buttons, Scrollleisten, Radio- und Checkbuttons ...

Solche Interaktion sind gut geeignet um modell- und geometrieunabhängige Werte an die Federation zu übergeben. Zu solchen Übergaben können z.B. generelle Start- oder Stopanforderungen gehören. Die Implementation von modellspezifischen Dialogboxen treibt den Modellierungsaufwand jedoch unnötig in die Höhe. Aus diesem Grund wurde nach anderen Mitteln der Interaktion gesucht.

Gefunden wurden diese Grundlagen direkt in den Möglichkeiten von VRML2. VRML2 bietet neben der 3D-Visualisierung auch Möglichkeiten, Interaktionen zwischen Nutzer und Federation zu unterstützen. Zu diesem Zweck können sogenannte Touch-Sensoren an den VRML-Objekten angebracht werden. Diese Sensoren können über Skriptknoten im Inneren der VRML-Objekthierarchie „Berührungen“ von VRML-Objekten durch den Benutzers erkennen. Dann werden Callback-Routinen im Skopeo-Kernel die notwendigen Ereignisse generiert, um die WG über diese Interaktionen zu informieren. Die WG kann dann die Federation informieren.

Skopeo bietet in seiner heutigen Implementationsstufe Möglichkeiten an beliebige Objekte in der Animation Touch-Sensoren anzubringen. Bei Berührung des Objektes durch den Nutzer wird die Objektnummer und ein Attribut für die Berührung an die Federation zurückgegeben.

Kritisch anzumerken sei, daß selbst bei heutigen high-end PCs die Verwendung von Touch-Sensoren im VRML2 recht sparsam erfolgen sollte, da jeder Touch-Sensor einen internen Skriptknoten in der Objekthierarchie erfordert und somit das Laufzeitverhalten des Gesamtsystems negativ beeinflußt.

4 Zusammenfassung und Ausblick

Es wurde gezeigt, daß es bei Erfüllung der oben vorgestellten Anforderungen und Konzepte möglich ist, Web-basierte Applikationen als HLA-konforme Federates zu nutzen, wobei die Effizienz der vorgestellten Methoden beträchtlich differiert. Auch sind gewisse „Kosten“ wie z.B. für Koordination und Kommunikation, die für die Vorzüge wie Modularität und Wiederverwendbarkeit anfallen, zu berücksichtigen. Wurde für ein Simulations- bzw. Animationstool der Initialaufwand zur Erstellung einer RTI-Schnittstelle erbracht, so reduziert sich der eigentliche, HLA-bedingte Aufwand auf die Implementation der Modellkomponenten in den einzelnen Federates und die Aufstellung der Objektmodelle.

An der Erstellung einer solchen allgemeinen Lösung für das Animationstool Skopeo und den Simulator SLX wird derzeit gearbeitet. Generell ist zu erwarten, daß die HLA aufgrund ihrer Flexibilität neue Anwendungsfelder, -techniken und -qualitäten erschließen kann; für klassische Simulationstools erlaubt dies einen vielversprechenden Schritt in Richtung wiederverwendbarer und interoperabler Simulationsmodelle.

5 Literatur

[1] Defense Modeling and Simulation Office (DMSO). 1997. *The High Level Architecture Homepage*. URL http://www.dmso.mil/projects/hla/.

[2] Defense Modeling and Simulation Office (DMSO). 1997a. *HLA Time Management Design Document, Version 1.0, dated 15 August 1996*. Verfügbar auf der HLA Homepage (DMSO 1997).

[3] Department of Defense (US). 1996. *High Level Architecture Rules, Version 1.2, dated 13 August 1997*. Verfügbar auf der HLA Homepage.

[4] Department of Defense (US). 1997. *High Level Architecture Interface Specification, Version 1.2 Draft 6, dated 1 August 1997*. Verfügbar auf der HLA Homepage.

[5] Department of Defense (US). 1997a. *High Level Architecture Object Model Template, Version 1.2, dated 13 August 1997*. Verfügbar auf der HLA Homepage.

[6] Ritter, K.-C. 1996. *The Skopeo Animation System*. Verfügbar unter der URL http://simos2.cs.uni-magdeburg.de/Skopeo/Ani.html.

[7] Fujimoto, R. *Parallel Discrete Event Simulation.* In Communications of the ACM, 1990, no. 10, pp. 30-53.

[8] Fujimoto, R. M. 1997. *Zero Lookahead and Repeatability in the High Level Architecture*. Proceedings of the Spring 1997 Simulation Interoperability Workshop, March 3-7, Orlando. Paper No. SIW97S-046, available online at http://www.dmso.mil/projects/hla/papers/.

[9] Klein, U. *Zivile Anwendungspotentiale der High Level Architecture (HLA).* Tagungsband Symposium Neue Technologien in der wehrtechnischen Simulation, 30.09.-02.10.1997. Bundesakademie für Wehrtechnik und –verwaltung, Mannheim.

[10] Klein, U., S. Strassburger, J. Beikirch. *Distributed Simulation with JavaGPSS based on the High Level Architecture*. International Conference on Web-based Modeling and Simulation, Jan. 11.-14. 1998, San Diego, in Vorbereitung.

[11] Klein, U. and S. Straßburger. 1997. *Die High Level Architecture (HLA): Anforderungen an interoperable und wiederverwendbare Simulationen am Beispiel von Verkehrs- und Infrastruktursimulationen*. Proceedings of the 11th Simulation Symposium ASIM 97 11.-14. Nov. 1997, Dortmund, Germany. In Vorbereitung.

[12] Lorenz, P. and K.-C. Ritter. 1997. *Skopeo: Platform-Independent System Animation for the W3*. In Deussen, O. and P. Lorenz (Ed.), Proceedings of the Simulation und Animation Conference Magdeburg, March 6-7, 1997. SCS European Publishing House San Diego / Erlangen / Ghent / Budapest 1997, Seiten 12-23.

[13] Ritter, K.-C.. 1996. *Systemanimation im Internet.* Diplomarbeit. Institut für Simulation und Graphik. Universität Magdeburg.

[14] Henriksen, J. O. 1997. *SLX and Proof Animation* In Deussen, O. and P. Lorenz (Ed.), Proceedings of the Simulation und Animation Conference Magdeburg, March 6.-7., 1997. SCS European Publishing House San Diego / Erlangen / Ghent / Budapest 1997, Seiten 287-294.