4. Agentenorientierte Simulation & HLA

Bisher beschäftigte sich die Vorlesung mit Techniken zur Effizienzsteigerung von Simulatoren
Im folgenden Kapitel steht mehr der Modellierungs- und Modellbildungsaspekt im Vordergrund
Alternative Konzepte zur Modellbildung:
Nicht mehr ein (monolithisches) Simulationsprogramm, sondern Simulatoren, die aus der Interaktion unterschiedlicher und autonomer Teilsysteme resultieren.
Zwei unterschiedliche Ansätze:

4.1 Agentenorientierte Simulation

(neues Paradigma zur Beschreibung von Simulationsmodellen)

4.2 HLA (High Level Architecture)

(Infrastruktur zur Kopplung von Simulatoren)

4.1 Agentenorientierte Simulation

Begriff des Agenten ist schillernd und schwer fassbar
Ein Versuch der Definition durch Abgrenzung von anderen Konzepten und der Beschreibung von Verhalten:

- Agenten besitzen (wie auch Objekte) eine klare Abgrenzung zu ihrer Umgebung
- Agenten interagieren mit der Umgebung reaktiv oder proaktiv (d.h. reagieren auf Grund zeitlicher Bedingungen)
- im Gegensatz zum Objekt wird dem Agenten eine eigene „Intelligenz“, Autonomie“, „Sozialfähigkeit“ etc. zugestanden
- Agenten haben ein individuelle und lokale „Weltsicht“
- Agenten reagieren flexibel auf sich wandelnde Umgebungen
- Agenten sind in der Regel mobil (im Netz)

Agenten sind ursprünglich in der KI beheimatet, werden heute aber auch in vielen anderen Systemen verwendet

Vorsicht: In vielen heutigen Systemen sind sogenannte Agenten nichts anderes als nebenläufige Objekte!

Struktur eines Agentensystems:

- Aktive Einheiten = Agenten
- Umgebung eines Agenten
  besteht aus anderen Agenten, Objekten der Modellwelt z.B. Ressourcen, der realen (Rechner-)Umgebung
- Interaktions- und Kommunikationsmechanismen zwischen Agenten

Agentensystem: Agenten + Umgebung +Interaktion

Konzeptuelle Ebene ⇒ Implementierung!?

Einsatz von Agenten in der Simulation:

1. Agenten als Designkonzept für Simulationssysteme
2. Agenten als Gegenstand der Simulation
3. Agenten als Konzept zur Modellierung dynamischer Systeme

Agenten als Designkonzept für Simulationssysteme

- Simulationssystem besteht aus einer Reihe autonomer Agenten
- Kopplung der Agenten über gemeinsame Infrastrukturkomponenten (z.B. HLA)
- Agenten führen spezifische Aufgaben im Rahmen der Simulation aus (z.B. Modellentwurf, Experimentdesign, Parallelisierung, Auswertung, Optimierung, ...)
- Ansatz ist insbesondere interessant für domänspezifische Simulationssysteme (z.B. Flugsimulatoren: Agenten realisieren Komponenten, einfacher Austausch der Agenten ändert das simulierte Szenario (anderer Flugzeugtyp, anderer Flughafen, ...))

Agenten als Gegenstand der Simulation

Autonome Softwareagenten spielen eine immer größere Rolle
(web-basierte Tools, Agenten als dynamische Suchmaschinen,...)

- Testen solcher Softwaresysteme ist schwierig
  (Verhalten des Agenten hängt stark von der Umgebung ab)
- Agenten benötigen in praktischen Situationen Informationen um Entscheidungen zu treffen und lernen oftmals aus getroffenen Entscheidungen

Simulation kann als Hilfsmittel eingesetzt werden, um

- die reale Umwelt des Agenten nachzubilden und den Agenten per Simulation zu testen
  (Fehler sind reproduzierbar, Testfälle einfach durchzuspielen)
- den Agenten zu trainieren, bevor er in die reale Umwelt entlassen wird
  (Trainingszyklen per Simulation deren Ergebnisse bekannt sind oder die zufällig erzeugt werden)

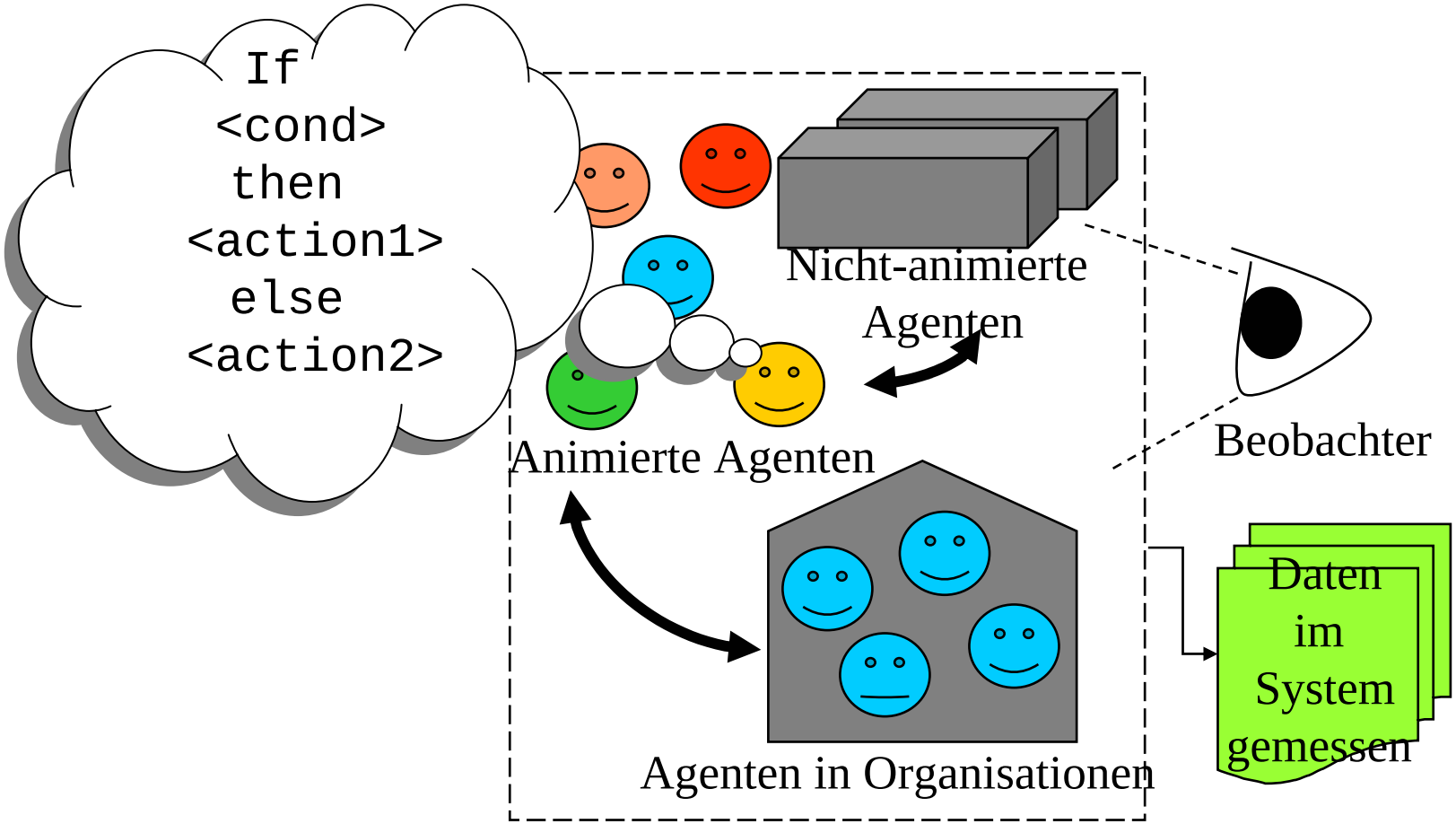
Agenten zur Modellbildung dynamischer Systeme

Konzept der Multi-Agenten-Systeme wird zur Modellierung verwendet

Typische Eigenschaften

- System besteht aus mehreren >> 1 Agenten
- Agenten reagieren auf Stimuli aus der Umgebung und proaktiv von sich aus (u.U. getrieben durch probabilistische Entscheidungen)
- Struktur des Systems ist dynamisch
  (neue Agenten können hinzukommen, alte das System verlassen)
- Agentensystem läuft verteilt auf mehreren Rechnern, ohne dass explizit eine gemeinsame virtuelle Zeit existiert
  (also kein Synchronisationsprotokoll notwendig, implizite Synchronisation lokaler Zeiten durch Interaktion der Agenten)
- neben Softwareagenten können auch Hardware- oder menschliche Agenten an der Simulation teilnehmen

Modellentwurf oft bottom-up: Erst Agenten, dann deren Interaktion



Erzeugung einer künstlichen Welt (aus Stef99)

Anwendungsgebiete: Reale Systeme, die aus interagierenden, intelligenten Individuen mit geringer oder ohne globale Kontrolle bestehen

Beispiele:

- Autos im Straßenverkehr
- Fußgängerströme in der Innenstadt
- Menschen in sozialen Gruppen
- Preisbildung im Markt
- Verhalten von Fischschwärmen
- Ausbreitung von Computervieren im Internet

Oft soziale oder biologische Systeme mit autonomen Individuen

In Zukunft aber auch Systeme autonomer Maschinen, Roboter etc.

Voraussetzung der Modellierung:
Kaum globale Kontrolle, ansonsten Probleme ähnlich zu paralleler Simulation!

Verhalten von Agentensystemen auf mindestens zwei Ebene:

- Lokale Sicht jedes einzelnen Agenten
  - beschränkte Information/Weltsicht als Basis für Entscheidungen
- Globale Sicht auf das System durch Beobachtung der Agentenpopulation
  - Auswertung der Beobachtungen schwierig, in Echtzeit oft unmöglich

Realisierung von Agentensystemen:

- Modellierung, Realisierung von Agenten mit heutigen Systemen ohne große Probleme möglich (z.B. Java)
- Kommunikation erfordert entsprechende Infrastruktur(z.B. Corba)
- Experimentelle Plattformen existieren (z.B. Swarm)

Probleme agentenorientierter Simulation:

- Verhalten des Simulators oft nicht reproduzierbar
- Zustand des Systems nur lokal und nicht global konsistent beobachtbar
- Bisher kaum Möglichkeiten statistischer Auswertungen und statistisch gesicherter Resultate

Chancen agentenorientierter Simulation:

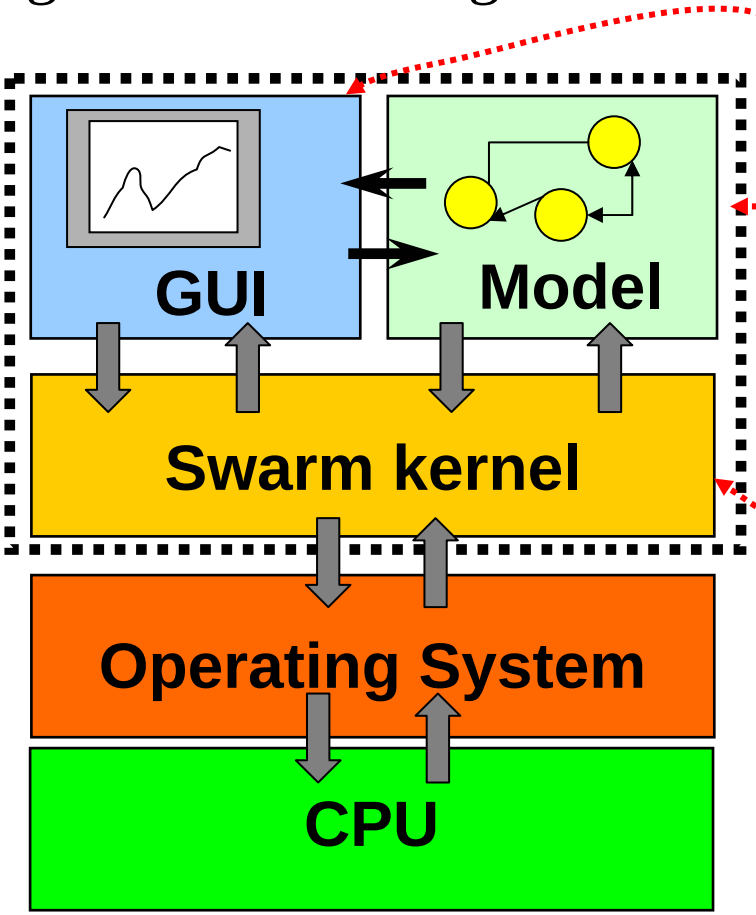
- Klasse modellierbarer Verhalten wird erweitert
- Mathematische Modellierung erreicht bei manchen realen Phänomenen ihre Grenzen
- Viele reale Systeme bestehen aus autonomen Einheiten
- Simulation sozialer Systeme bisher nur mäßig erfolgreich (gilt aber bisher auch für agentenorientierte Ansätze)
- Einbindung von Menschen erlaubt neue Simulations- und Untersuchungsszenarien

SWARM: Ein Beispiel für ein agentenorientiertes Simulationssystem
(Folien zum Teil aus Stef99 entnommen)

Historie und Aufbau:

- Entwicklung am Santa Fe Institute der Universität New Mexiko seit 1994
- erste Versionen 1995, bis heute Weiterentwicklungen
- Swarm ist eine Sammlung von C libraries zur Erstellung von agentenorientierten Simulatoren
- Lauffähig unter Unix und diversen Windows Varianten
- Tcl/Tk basierte GUIs
- Objektorientierte Realisierung
- Zahlreiche Tools zur Beobachtung und Auswertung von Modellen
- Modelle laufen auf einem Rechner (es existiert eine (virtuelle) Ereignisliste über alle geplanten Ereignisse!)

Zentrale Strukturelemente:
Swarms beschreiben eine Menge von Agenten, die über eine gemeinsame Ereignisliste koordiniert werden

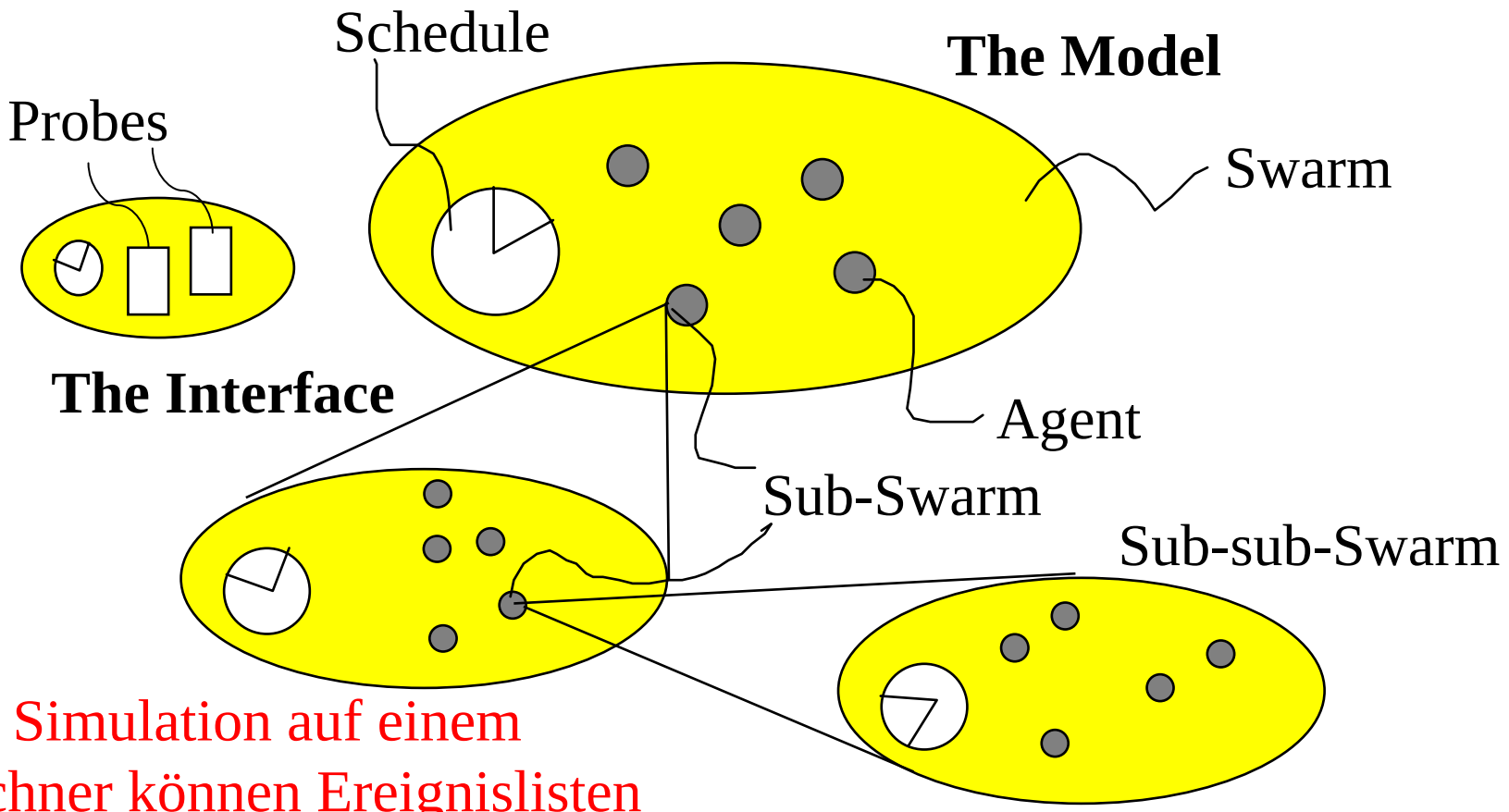


Modell mit lokaler Visualisierungs- und Auswertungsmöglichkeit

Bereitstellung der virtuellen Maschine zur Ausführung der Simulation

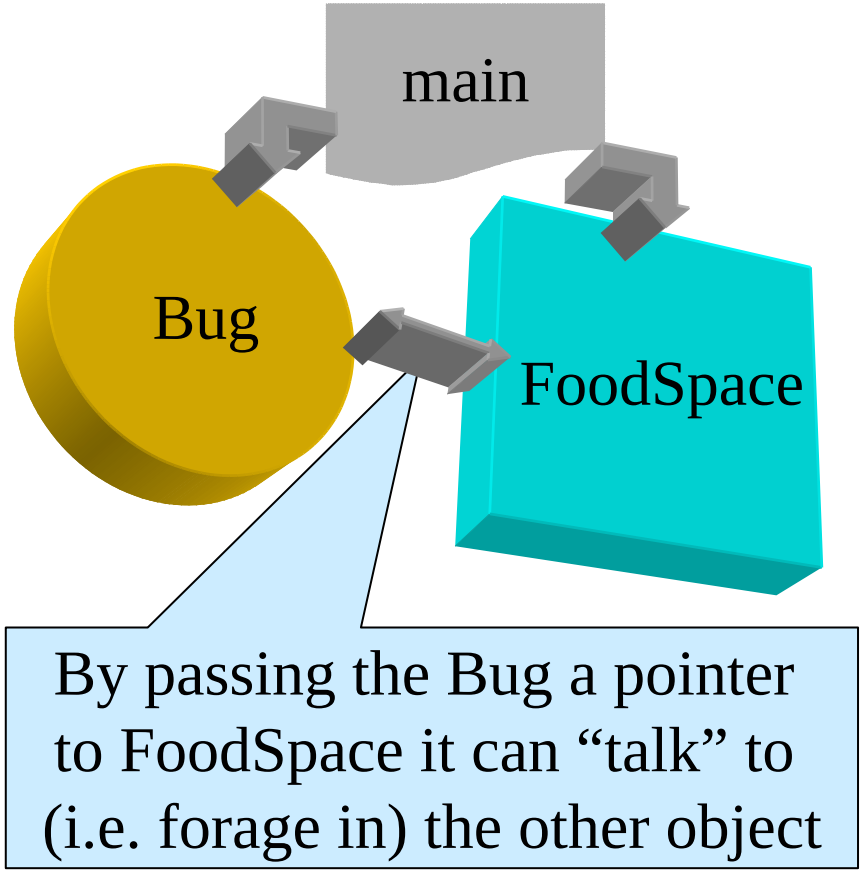
- globale Ereignisliste für den Schwarm
- Simulationsfunktionalität

Swarms können hierarchisch generiert werden



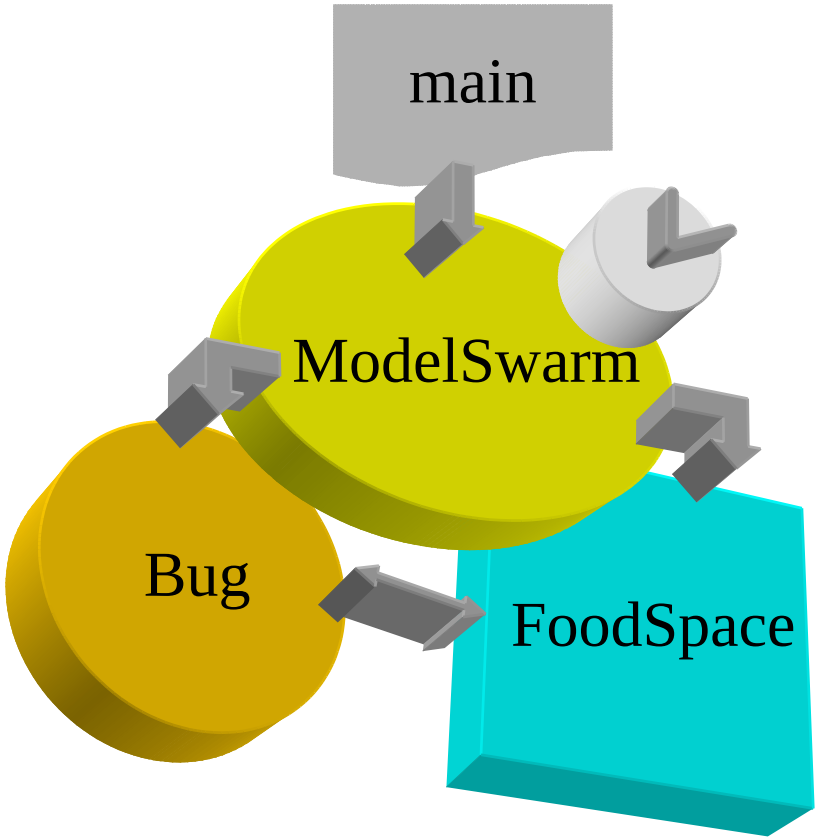
Bei Simulation auf einem Rechner können Ereignislisten zusammengefügt werden.

- Agenten sind Objekte
  (abgeleitet aus Agentenklasse der swarm library)
- Umgebung eines Agenten besteht aus anderen Agenten

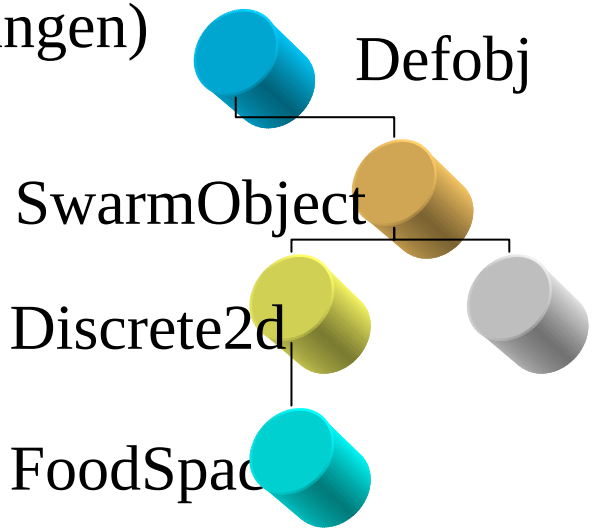


- Käfer werden als Agenten modelliert und leben in ihrer Umwelt
- Umwelt wird durch einen weiteren Agenten modelliert
  - zwei- oder dreidimensionale geometrische Struktur
  - Verteilung von Futter im Raum
- Interaktion durch Bewegung der Käfer im Raum

Sichtweise des Modells zur Simulation

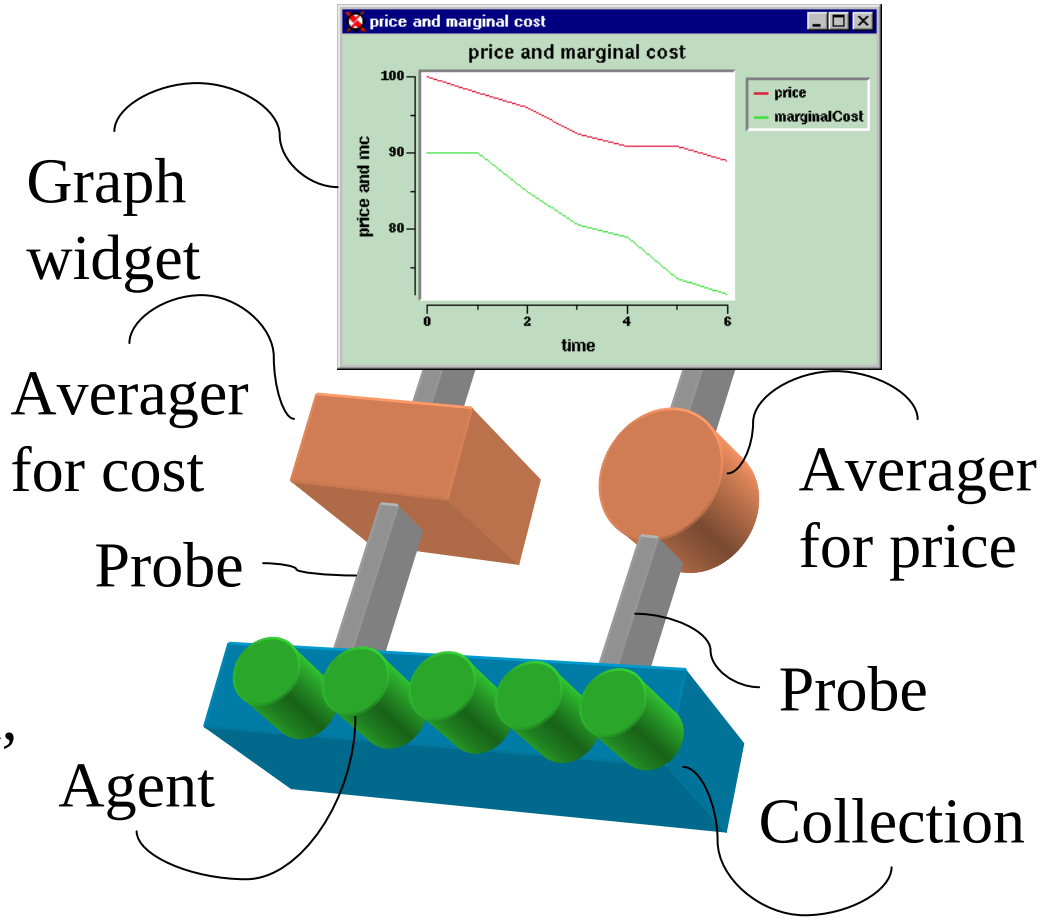


- Käfer und Umgebung führen Ereignisse aus
- Scheduling der Ereignisse im Swarm Objekt
- Basisklassen für Standardagenten vorhanden
  (z.B. zwei oder dreidimensionale Umgebungen)

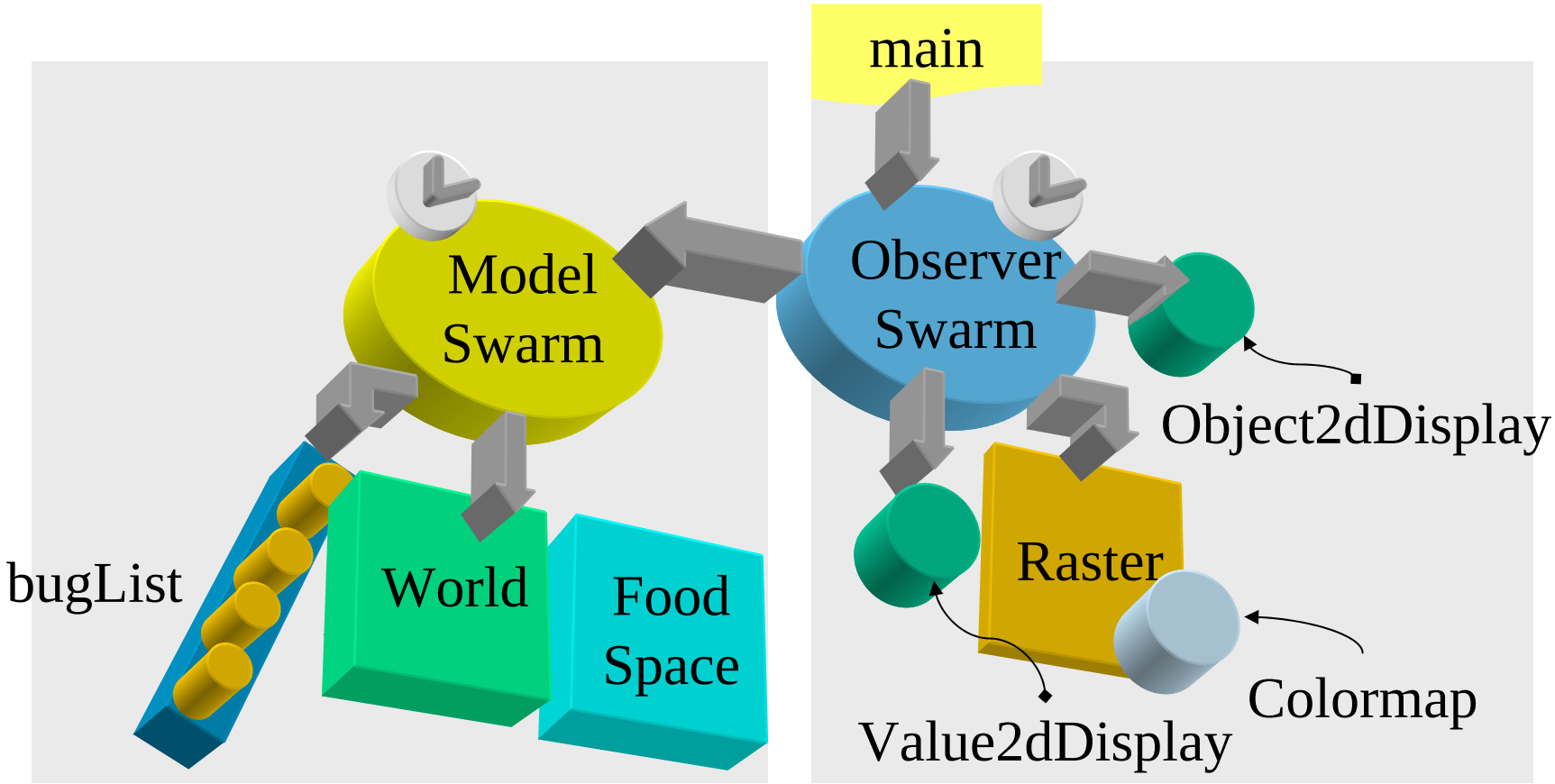


Beobachtung des Modells durch Beobachtung der Agenten

- Beobachtungspunkte sind im Agenten vorzumerken
- Variablenwerte werden übermittelt
- Beobachtung auf Anfrage oder proaktiv bei Eintreten von Ereignissen
- Aufbereitung der beobachteten Werte zur Laufzeit (auch graphisch)
- im wesentlichen Animation, keine Unterstützung zur statistischen Auswertung



Modellaufbau inkl. Beobachtungsagenten



4.2 HLA (High Level Architecture)

Entwicklung des US Department of Defense (DoD) wird heute von vielen Organisationen und Firmen eingesetzt

Motivation:

- Kein einzelnes Simulationssystem oder –paradigma erfüllt allein alle Anforderunge, die an Simulationssysteme gestellt werden
- Für Teile komplexer Systeme existieren oft schon Simulatoren, die bei der Modellierung des Gesamtsystems zusammengeführt und ergänzt werden müssen
- Zur Durchführung von Simulationsexperimenten wird ein umfangreiche Infrastruktur benötigt, die von einzelnen Simulatoren nicht geboten wird
- Verteilte Rechnerarchitekturen sollen in Simulationsexperimenten genutzt werden

HLA ist eine Infrastruktur zur Kopplung von Simulationssystemen, wobei Simulatoren eingebunden werden können, ohne dass größere Änderungen durchzuführen sind.

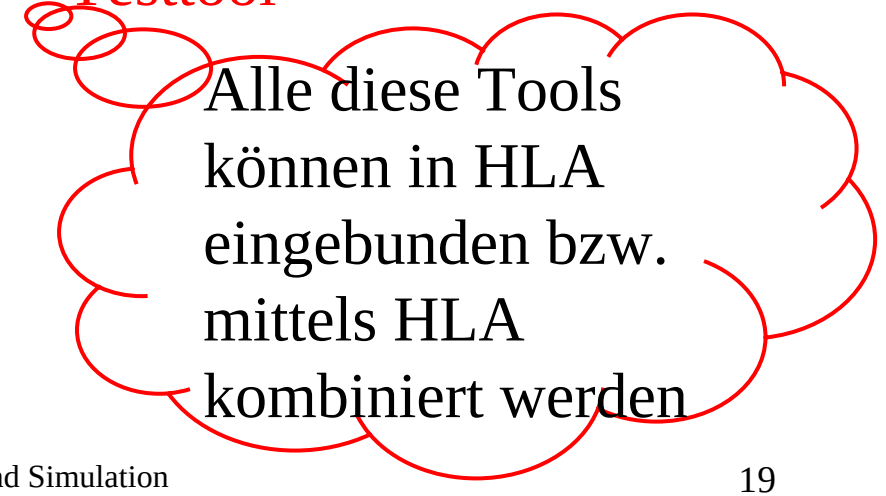
- Erste Veröffentlichungen des HLA-Standards durch DoD 1996
- Heute als DoD und IEEE-1516 Standard etabliert

Was ist HLA?

- Objektorientierte Softwarearchitektur
- Komponentenbasiert Softwarearchitektur
- System zur Prozesskommunikation
- System zur verteilten Datenhaltung
- Verteilte Umgebung
- System zur Synchronisation und zum Zeitmanagement

Was ist HLA nicht?

- Simulator
- Modellierungstool
- Analysetool
- Benutzerschnittstelle
- Testtool



Eine HLA Simulation besteht aus einer Menge von Partnern (federates) und wird Föderation (federation) genannt

Partner können
- Simulatoren
- Hardware
- Software
- Personen
- ...

sein und können die Föderation verlassen oder neu eintreten

Eine Föderation beinhaltet mindestens zwei Partner, die interagieren
Durch HLA wird die Interaktion von Partner unterstützt, auch wenn sie sich an unterschiedlichen Stellen befinden

- HLA beschreibt keine Implementierung oder legt keinen Programmierstil fest, sondern besteht im wesentlich aus den folgenden Teilen
  - Menge von HLA-Regeln
  - Festlegung der Zuständigkeiten
  - Gewährleistung von Interaktion zwischen Partnern
  - Anforderungen an einzelne Partner
- Definition der HLA-Schnittstelle
  - Definition Laufzeitschnittstelle (runtime interface (RTI))
  - Identifikation von notwendigen „call back“ Funktionen
- Definition des HLA-Objektmodells
  - Standard zur Struktur und Repräsentation von Daten in HLA

Es existieren inzwischen einige, auch frei verfügbare Implementierungen von HLA!

HLA-Regeln

Für federations:

- müssen HLA-Objektmodell gemäß object model template haben
- alle Objektdarstellungen in den federates und nicht im RTI
- Austausch von Daten und Instanzen nur über RTI und nicht direkt
- federates müssen mit der RTI gemäß vorgegebener Interface-Spezifikation kommunizieren
- ein Attribut einer Objektinstanz kann zu einem Zeitpunkt nur einem federate gehören

Für federates:

- müssen HLA—Simulation-Objektmodell (SOM) gemäß object model template haben
- müssen in der Lage sein, sämtliche Objektattribute im SOM zu publizieren
- müssen in der Lage sein, Besitzrechte von Attributen dynamisch zu übertragen/zu akzeptieren
- müssen Attributänderungen gemäß Anforderung mitteilen
- müssen ihre lokale Zeit verwalten und mit RTI koordinieren

HLA-Objektmodell

Primäres Ziel: Interoperabilität zwischen Teilnehmern (federates)

$\Rightarrow$ keine Einschränkung bzgl. des Inhalts der Objekte

aber Festlegung der Struktur und Schnittstellen

HLA Objektmodell Hauptvehikel der Interoperabilität, da Regeln und Schnittstellen unabhängig von Plattformen formuliert werden

Objektmodell beschreibt:

- ausgewählte Objektmenge zur Darstellung der realen Welt für eine Föderation
- Attribute, Assoziationen, Interaktionen der Objekte
- Detaillierungsgrad dieser Objekte, einschließlich räumlicher und zeitlicher Auflösung

HLA bietet Schablonen (Templates), um die Objektmodelle von Simulationen zu beschreiben

HLA spezifiziert drei Typen von Objektmodellen:

- HLA Federate Object Model (FOM)
  - Festlegung der Objekte, Attribute und Interaktionen in der gesamten Föderation
- HLA Simulation Object Model (SOM)
  - Definition der Informationen, die der Simulator als Teil einer Föderation bereitstellen kann
- HLA Mangaement Object Model (MOM)
  - Objekte zum Management von Föderationen

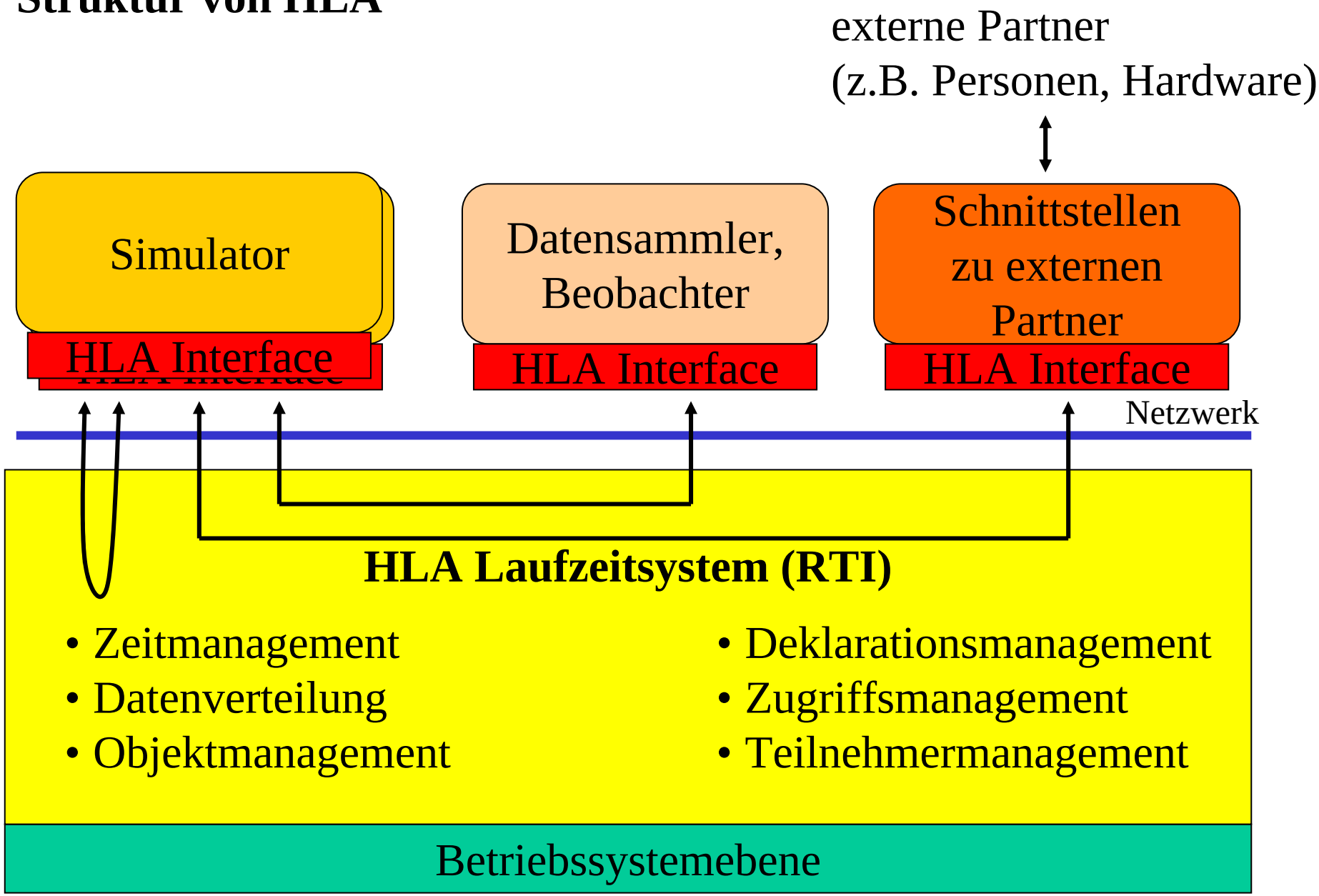
Alle Objekte werden gemäß Object Management Template (OMT) beschrieben

Unterstützung bei Erstellung durch vorhandene Tools (z.B. OMDT)

Partner können Objekte

- publizieren (d.h. für andere sichtbar machen)
- zeichnen (d.h. die publizierten Objekte anderer Partner benutzen)

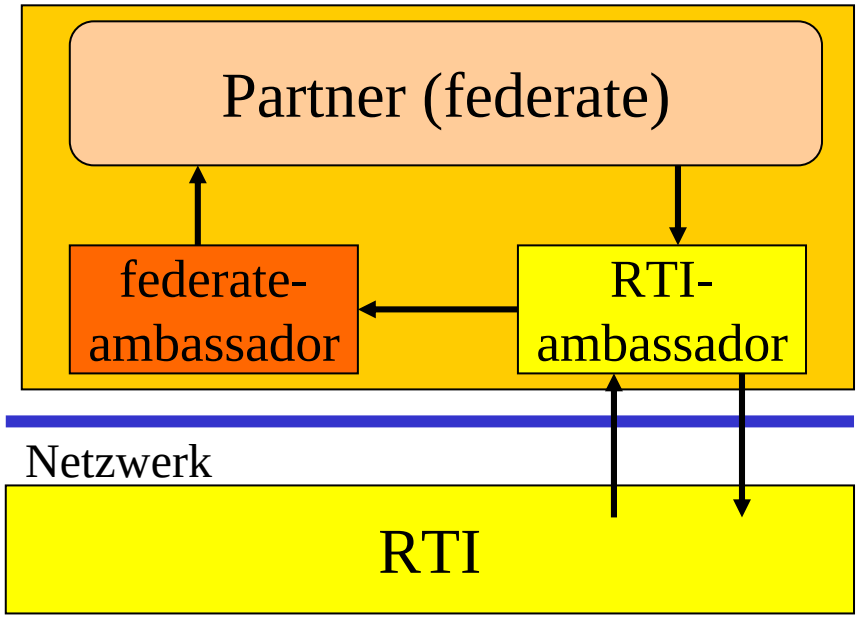
Struktur von HLA



Basiskomponenten

- Simulatoren oder allgemeine Partner der Föderation (federates)
  - funktionale und autonome Komponenten
  - keine Anforderungen an interne Struktur eines Teilnehmers
  - Festlegung der Schnittstellen
- Runtime Infratsructure (RTI)
  - Verteiltes Betriebssystem für Teilnehmer
  - Funktionalität zum Teilnehmermanagement
  - Kommunikation zwischen Teilnehmern (auch gruppenorientiert)
- Runtime Interface
  - Definition der Kommunikationsschnittstellen zwischen Teilnehmern und RTI
  - Trennung von Kommunikation und Simulation
  - Unterstützung des Zeitmanagement der Partner
  - Schnittstellendefinition ist unabhängig von
    - der Implementierung
    - dem verwendeten Objektmodell

Definition der HLA Schnittstellen



Kommunikation über Botschafter
d.h. Aufruf von Methoden
spezieller Klassen
Einbindung der
Kommunikationsklassen
verfügbar in C++, Java, Corba,
Ada

Kommunikation nicht nur unidirektional vom Partner ausgehend, sondern oftmals auch von anderen Partnern (vom RTI) initiierte Kommunikation

$\Rightarrow$ call back Methoden/Funktionen müssen realisiert werden
(Grundgerüste dafür sind vorhanden,
z.B. in C++ abstrakte Objektklasse)

Teilnehmermanagement

- Erzeugung und Zerstörung von Föderationen
- Beitreten und Austreten von Partnern
- Föderationsausführung anhalten, wiederherstellen, speichern, fortsetzen

Deklarationsmanagement

- Publikation von Objektklassen/Interaktionsklassen
- Abonnement von Objektklassenattributen/Interaktionsklassen
- Kontrolle von Udaptes und Interaktionen

Zugriffsmanagement

- jedes Attribut hat genau einen Besitzer (keine shared objects)
- Anfrage nach Besitzrechten an einem Attribut
- Anfrage nach Abgabe der Besitzrechte an einem Attribut
- Mitteilung der Besitzübergabe an Attributen

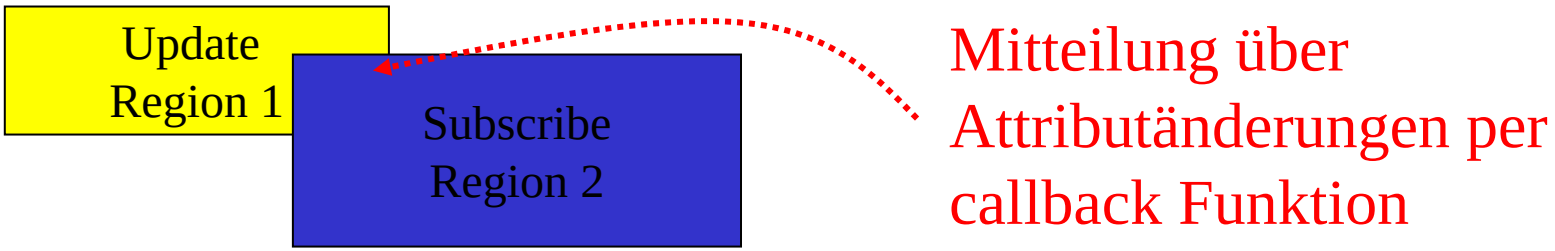
Objektmanagement

- Registrieren und entdecken von Objekten
- Aktualisieren von Attributwerten
- Senden und Empfangen von Interaktionen
- Entfernen von Objekten
- Ändern von Transport- und Sortiermechanismen
  - Transportmechanismus: zuverlässig, bestmöglich etc.
  - Sortiermechanismus: Sortierung nach Zeitstempel, Empfang, ...

Datenverteilungsmanagement

- Definition des routing-spaces
- subscribe region für Attribute, an denen ein Prozess interessiert ist
- update region für Attributänderungen, die ein Prozesse ausführt

  Attributeänderungen werden nur in der Schnittmenge mitgeteilt



Zeitmanagement

Kontrolle des Zeitfortschritts der Teilnehmer einer Simulation

Möglichkeiten des Zeitfortschritts

- schrittweise und unabhängig
  Zeit schreitet proportional zur Realzeit fort, ohne Synchronisation durch RTI
  (z.B. men-in-the-loop, hardware-in-the-loop Simulationen)
- schrittweise und koordiniert
  Zeit schreitet proportional zur Realzeit fort,
  Koordination mit anderen Teilnehmern erfolgt über RTI
  (z.B. Simulationskomponenten in Realzeitsimulationen)
- asynchron und koordiniert
  Teilnehmer schreiten so schnell wie möglich voran,
  Synchronisation zur Sicherung der Zeitkausalität

Komponenten des Zeitmanagements

- Teilnehmer beantragen Fortschritt
  - Time Advance Request (nächster Zeitschritt)
  - Next Event Request (nächstes Ereignis)
  - Flush Queue Request (alle Ereignisse)
- Mechanismen zur Steuerung des Zeitfortschritts
  (konservative und optimistische Variante siehe parallele Simulation)
  - bei konservativer Synchronisation müssen Teilnehmer Zeitfenster (lookahead) garantieren, in denen sie keine Nachricht senden
  - bei optimistischer Synchronisation müssen Teilnehmer Zustandssicherung und Rücksetzen realisieren
  - Mischung von konservativer und optimistischer Synchronisation ist möglich
- Nachrichtenordnung und –synchronisation
  - RTI garantiert zeitkonsistente und zuverlässige Auslieferung von Daten

Einsatzgebiete von HLA

- Wiederverwendung von Simulatoren
  - Simulatoren müssen um HLA Schnittstelle erweitert werden und können in größeren Umgebungen benutzt werden
- große heterogene Simulatoren
  - HLA erlaubt Kopplung diskreter und kontinuierlicher Simulatoren
- parallele Simulation
  - Protokoll ist realisiert, Details in den Simulatoren sind zu implementieren
- web-basierte Simulation
  - Realisierung von HLA auf verteilter Infrastruktur

Aber Problem mangelnder Effizienz paralleler Simulation wird natürlich durch HLA nicht beseitigt, sondern eher verstärkt!!