Verteilte Simulation auf Basis der High Level Architecture (HLA) in zivilen Anwendungsgebieten

Steffen Straßburger
Institut für Simulation und Grafik
Fakultät für Informatik
Otto-von-Guericke-Universität Magdeburg

Email: strassbu@web.de
WWW: http://isgsim1.cs.uni-magdeburg.de/~strassbu

Überblick

- Motivation und Zielstellung
- HLA als state-of-the-art für Simulationsinteroperabilität
- Migration von HLA in zivile Simulationssysteme
  - Anforderungen an Simulationssysteme
  - Alternativen für den Entwurf von HLA-Schnittstellen
- Praktische Evaluierung von HLA in zivilen Applikationen
- Zusammenfassung und Ausblick

Überblick

- Motivation und Zielstellung
- HLA als state-of-the-art für Simulationsinteroperabilität
- Migration von HLA in zivile Simulationssysteme
  - Anforderungen an Simulationssysteme
  - Alternativen für den Entwurf von HLA-Schnittstellen
- Praktische Evaluierung von HLA in zivilen Applikationen
- Zusammenfassung und Ausblick

Einleitung – Rahmenbedingungen der Dissertationsarbeit

- 1997/98: HLA war eine vielversprechende Technologie, die noch in ihren Kinderschuhen steckte und langsam Form annahm
  - Angewendet für militärische Trainingssimulationen, aber mit dem Anspruch Interoperabilität zwischen einem breiten Spektrum von Simulationsapplikationen zu unterstützen
  - Skeptik von Seiten alter DIS Nutzer
- Ziviler Simulationsmarkt in einem sehr konsolidierten Zustand
  - Überschaubare Anzahl von hochspezialisierten Simulationssystemen mit komfortablen Modellierungsparadigmen
  - Interoperabilität zwischen Systemen nicht existent

Motivation: Interoperabilität zwischen Simulationssystemen in zivilen Simulations-anwendungen häufig erforderlich

- Ein komponentenbasierter Ansatz zum Erstellen komplexer Simulationsmodelle mit *heterogenen* Simulationssystemen fehlt
  - Komposition komplexer Modelle durch Kombination verschiedener Teilmodelle (entwickelt im dafür bestgeeigneten Simulator) ist wünschenswert
  - Wiederverwendung existierender Modelle durch deren Re-Kombination
  - Verteilte Simulation: Kombination von physisch verteilten Modellen sollte möglich sein

  ➪ Ein Plug-and-Play Standard zur Schaffung (verteilter) Simulationsmodelle fehlt.

Motivation: Interoperabilität mit Nicht-Simulationskomponenten

- Simulationsprojekte benötigen in der Praxis häufig eine Verbindung zu Systemen wie
  - Geografische Information Systeme (GIS)
  - Leitstellen (Command and Control) Systeme
  - Externe Visualisierungen
  - On-line Datenquellen
- Verschiedene proprietäre Lösungen zur Schaffung von Interoperabilität
  - Unkomfortable Nutzung (z.B. Socket-Schnittstellen)
  - „Vereinfachte“ Synchronisationsmechanismen
  - Keine Standardisierung von Schnittstellen und Daten

Existierenden Standard-Architekturen für Interoperabilität fehlte Funktionalität

- CORBA, DCOM:
  - Interoperabilität für allgemeine Anwendungen
  - Keine Simulationsunterstützung
- PVM, MPI:
  - Schnelle Kommunikationsprotokolle für paralleles Rechnen
  - Mängel bzgl. Interoperabilitätsaspekten
- DIS, ALSP
  - Vernetzung bestimmter Typen militärischer Trainingssimulatoren
- Nur HLA hatte den Anspruch, Interoperabilität für ein breites Spektrum von Simulationsanwendungen zu unterstützen.

Zielstellung: Kann HLA zum benötigten Interoperabilitätsstandard für zivile Simulationsanwendungen werden?

- Aufstellung eines umfassenden Anforderungskatalogs für Simulationssysteme, die „HLA-kompatibel“ werden sollen
- Konzepte zum Design von HLA-Schnittstellen in einer nutzerfreundlichen Art
  - Verstecken von HLA-Funktionalität wann immer möglich
  - Absicherung eine hohen Grades von Flexibilität bzgl. HLA-Funktionalität
- Einschätzung des Anwendungspotentials
- Identifikation potentieller Schwächen von HLA und notwendiger Verbesserungen

Überblick

- Motivation und Zielstellung
- HLA als state-of-the-art für Simulationsinteroperabilität
- Migration von HLA in zivile Simulationssysteme
  - Anforderungen an Simulationssysteme
  - Alternativen für den Entwurf von HLA-Schnittstellen
- Praktische Evaluierung von HLA in zivilen Applikationen
- Zusammenfassung und Ausblick

HLA – der State-of-the-Art für Simulationsinteroperabilität (1)

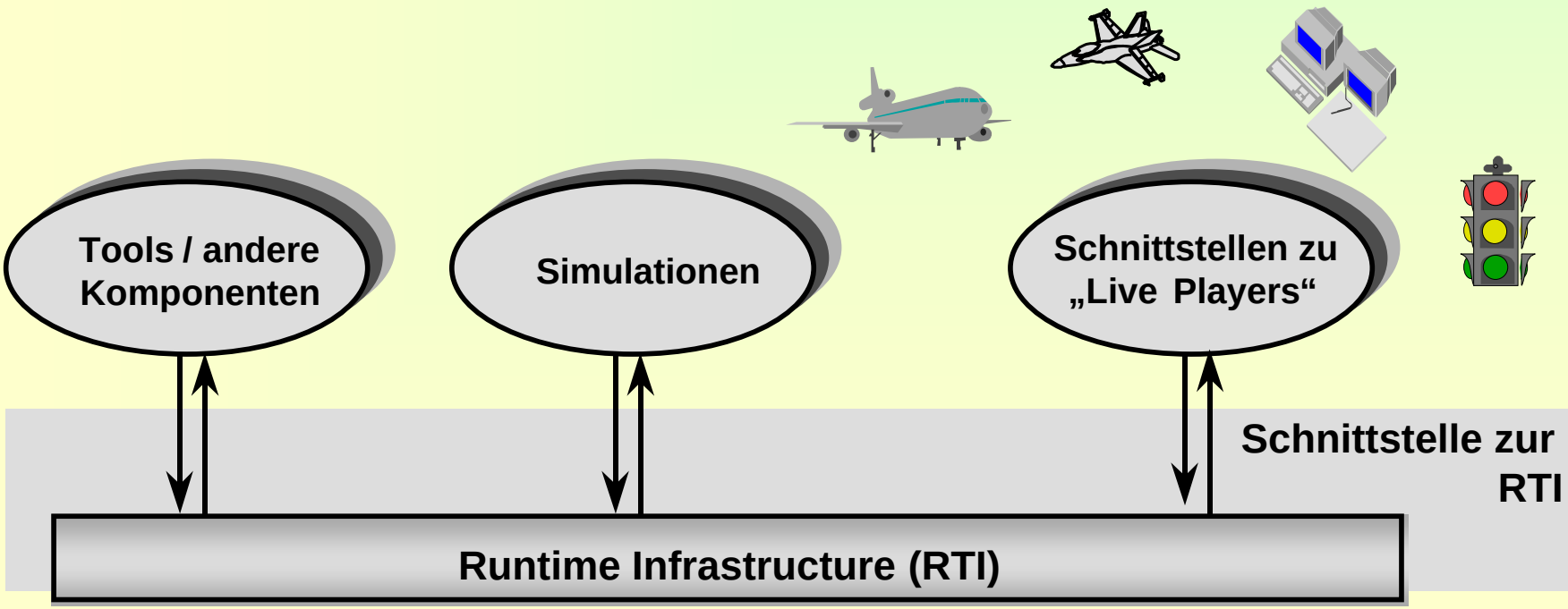
- Architektur zur Kombination individueller Simulationen (Federates) zu einem koordinierten Ensemble (Federation)
- DMSO stellt bereit
  - HLA Standard
  - Infrastructure Software (Runtime Infrastructure, RTI)
  - Support Tools
- Architektur zur Unterstützung von *Interoperabilität* und *Wiederverwendbarkeit* von verschiedenen Arten von Programmen

HLA – der State-of-the-Art für Simulationsinteroperabilität (2)

- HLA wird definiert durch
  - HLA Regeln
  - HLA Objektmodell-Templates (OMT)
  - HLA Interface Spezifikation
- HLA‘s Fähigkeiten übersteigen die von
  - Vorgängern wie DIS und ALSP wegen des „all-umfassenden“ Ansatzes
  - verwandten Technologien wie CORBA und DCOM wegen der simulationsspezifischen Dienste

HLA – Funktionale Übersicht

Datenkollektoren/ Umweltinformationssysteme
Passive Viewer
Command & Control Systeme
...



Überblick

- Motivation und Zielstellung
- HLA als state-of-the-art für Simulationsinteroperabilität
- Migration von HLA in zivile Simulationssysteme
  - Anforderungen an Simulationssysteme
  - Alternativen für den Entwurf von HLA-Schnittstellen
- Praktische Evaluierung von HLA in zivilen Applikationen
- Zusammenfassung und Ausblick

Unterschiede zwischen dem zivilen & militärischen Simulationsbereich

- Militärischer Bereich:
  - Simulationen werden oft „programmiert“ in Sprachen wie C++, Java oder Ada
  - Konstruktive/Virtuelle und Live Simulation
- Ziviler Bereich:
  - Nutzung von C++ zur Simulation eher selten & unerwünscht
  - Verlässliche und komfortable Werkzeuge existieren
    - Simulationssprachen, Integrierte Simulationsumgebungen, Graphische Nutzerschnittstellen, Integrierte Visualisierung, Bausteinbibliotheken, usw.
    - ACSL, Arena, Automod, eMPlant, GPSS/H, Matlab/Simulink, Modsim, Pro Model, Simplex, SLX ...

Zur Entwicklung von HLA-Federates muss eine Simulation zwei Arten von Anforderungen erfüllen

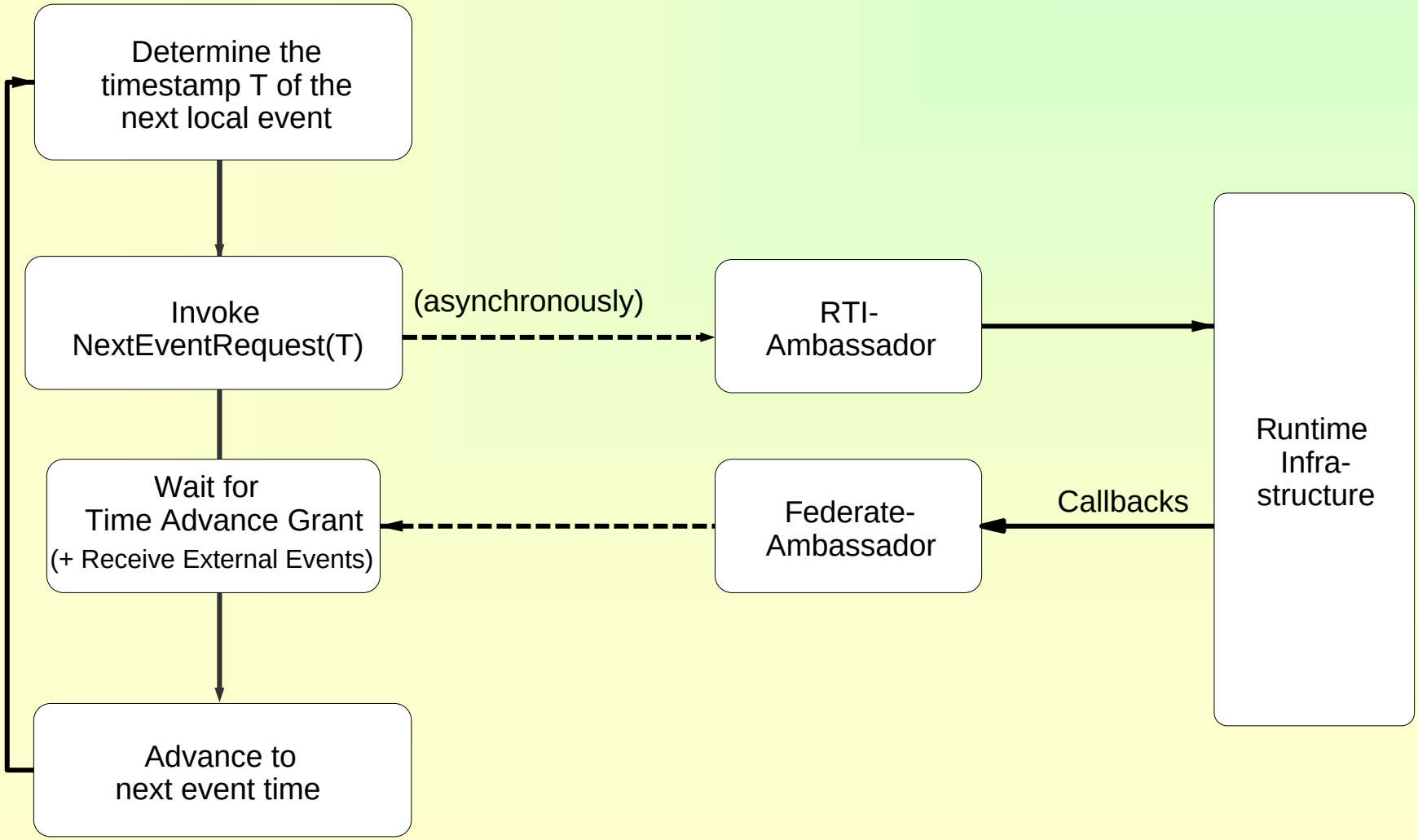
(1) Anforderungen, die aus den Bedingungen einer verteilten Simulation entstehen

(2) Anforderungen, die aus der erforderlichen Konformität zum HLA Programmier-paradigma resultieren

Anforderungen zur Teilnahme an einer verteilten Simulation

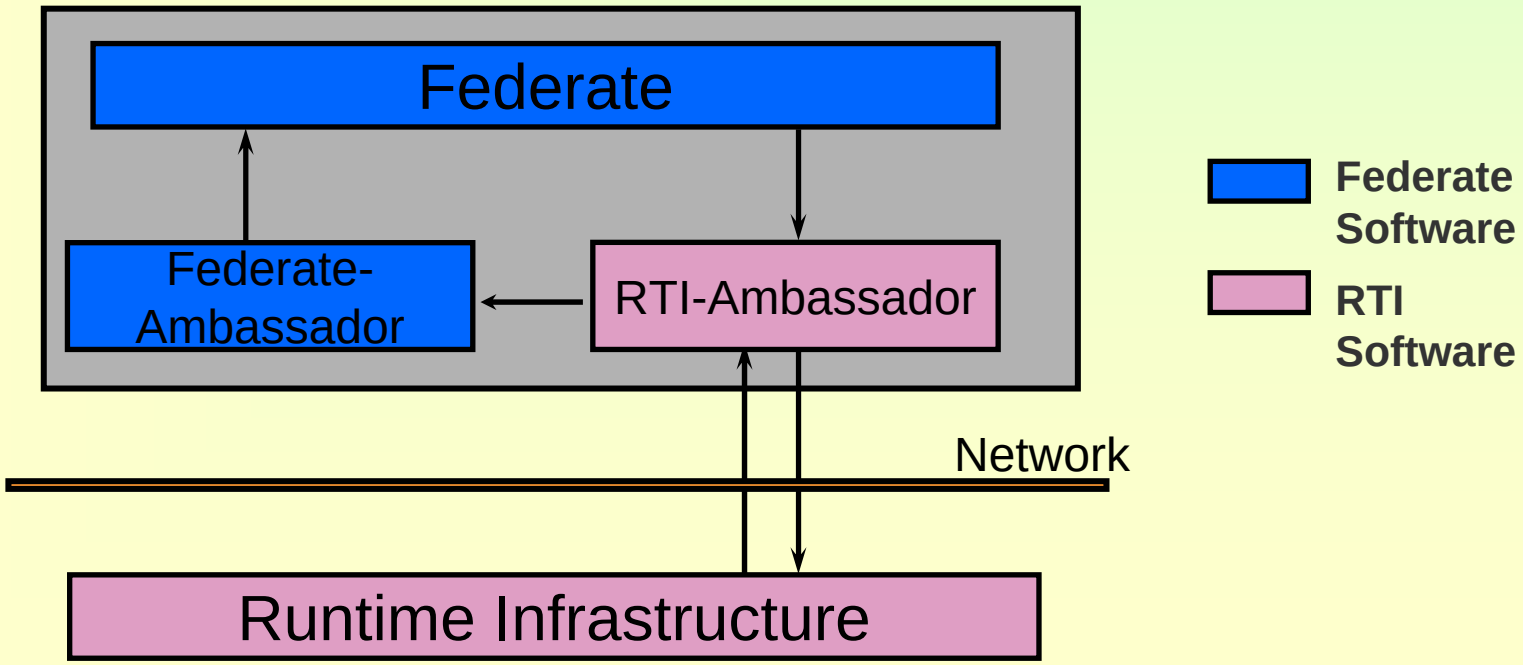
- Datenaustausch und Darstellung
  - Kompatibilität der Datentypen
  - Lokale vs. globale Datenobjekte
  - Zugriff auf globale Datenobjekte
- Synchronisation
  - Koordinierung der Zeitfortschritte notwendig
  - Zeitschritt- und ereignisgesteuerte Verfahren
  - Konservative vs. optimistische Protokolle

Lokale Zeitfortschritte müssen mit anderen Teilnehmern koordiniert werden



Anforderungen bzgl. des HLA-Programmierparadigmas

- “Botschafter Paradigma”: HLA Interface Spezifikation fordert Kommunikation über zwei Objekte (C++,Java,ADA)



Integration von Simulationssystemen in HLA durch Entwicklung von HLA-Schnittstellen

- Identifikation von angepassten HLA-Schnittstellen als Mittel HLA-Funktionalität in Simulationssystemen verfügbar zu machen
- HLA-Interfaces helfen bei der Erfüllung der Anforderungen
  - Lösung technischer Probleme
    - Implementierung der Botschafter (Modell-unabhängig!)
    - Verfügbarmachung eines an den Simulator angepassten HLA API‘s
  - HLA-Interfaces können eigene Intelligenz besitzen
    - Automatisierung bestimmter Aufgaben (z.B. Synchronisation, Updates)
    - Konvertierung von Datentypen

Ansätze zur Entwicklung von HLA-Schnittstellen

- Programmierungstechnische Sicht
  - Erweiterung des Simulatorkerns
  - Nutzung externer Programmierschnittstellen
  - Erweiterung eines Zwischencodes
  - Entwicklung eines Gatewayprogramms
- Nutzersicht
  - Explizite Ansteuerung der HLA-Schnittstelle
    - Mapping von HLA-API in ein Simulator-spezifisches API
  - Implizite Ansteuerung der HLA-Schnittstelle
    - Komplette Automatisierung der HLA-Funktionalität

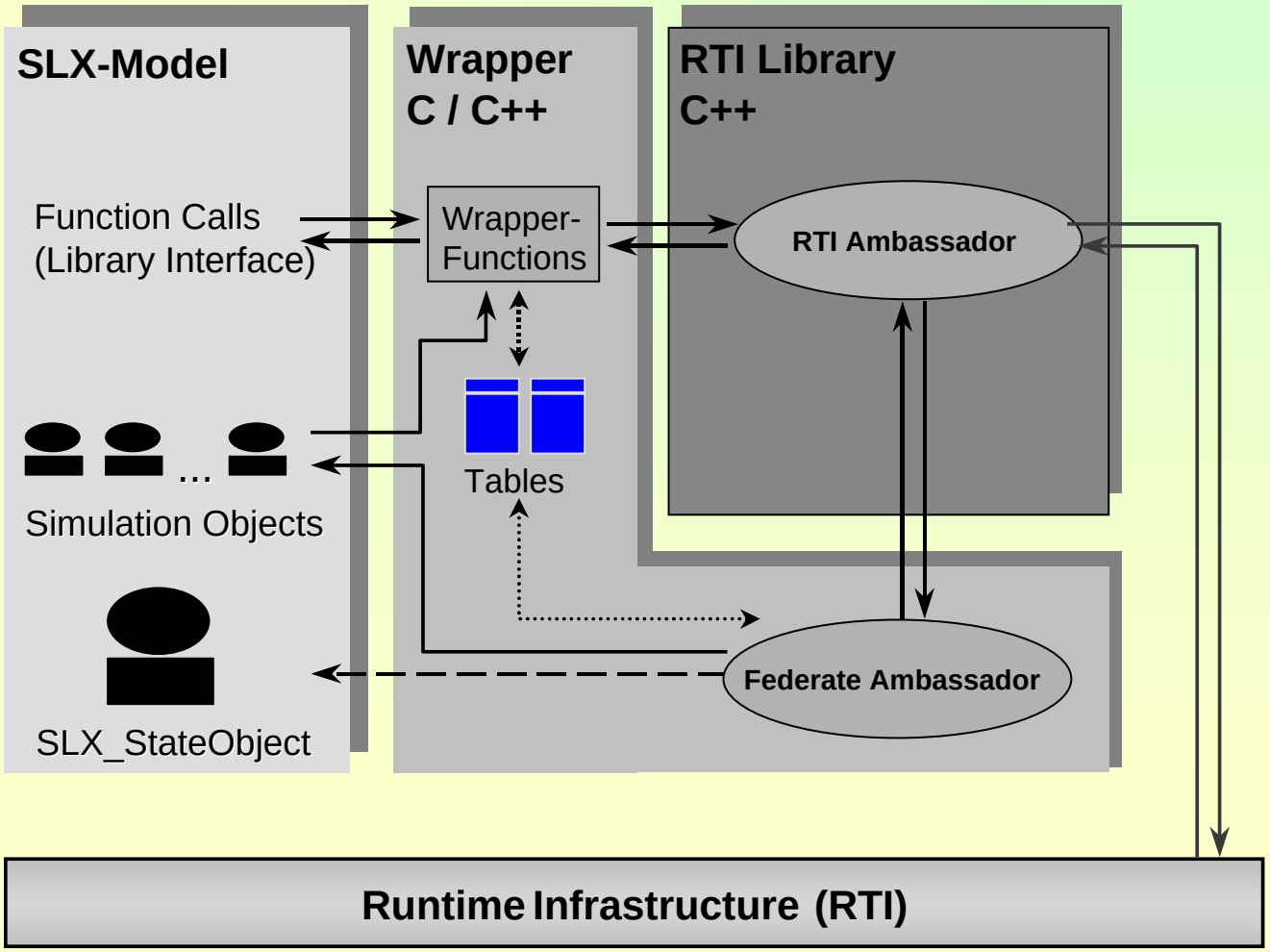
Ansätze zur Entwicklung von HLA-Schnittstellen

- Programmierungstechnische Sicht
  - Erweiterung des Simulatorkerns
  - Nutzung externer Programmierschnittstellen
  - Erweiterung eines Zwischencodes
  - Entwicklung eines Gatewayprogramms
- Nutzersicht
  - Explizite Ansteuerung der HLA-Schnittstelle
    - Mapping von HLA-API in ein Simulator-spezifisches API
  - Implizite Ansteuerung der HLA-Schnittstelle
    - Komplette Automatisierung der HLA-Funktionalität

Untersuchung von zwei Systemen: SLX und Simplex3

- SLX
  - Nutzung der Bibliotheksschnittstelle (Windows DLL) um auf eine spezielle Wrapper-Bibliothek zuzugreifen
  - Wrapper-Bibliothek kapselt RTI-Funktionalität in eine für den Simulator verständliche Form
  - Expliziter Zugriff auf HLA-Funktionalität im Modell
- Simplex
  - Erweiterung des Quelltextes des Laufzeitsystems
  - Auf HLA-Funktionalität wird implizit zugegriffen, d.h. HLA Funktionsaufrufe werden automatisch generiert wann immer nötig

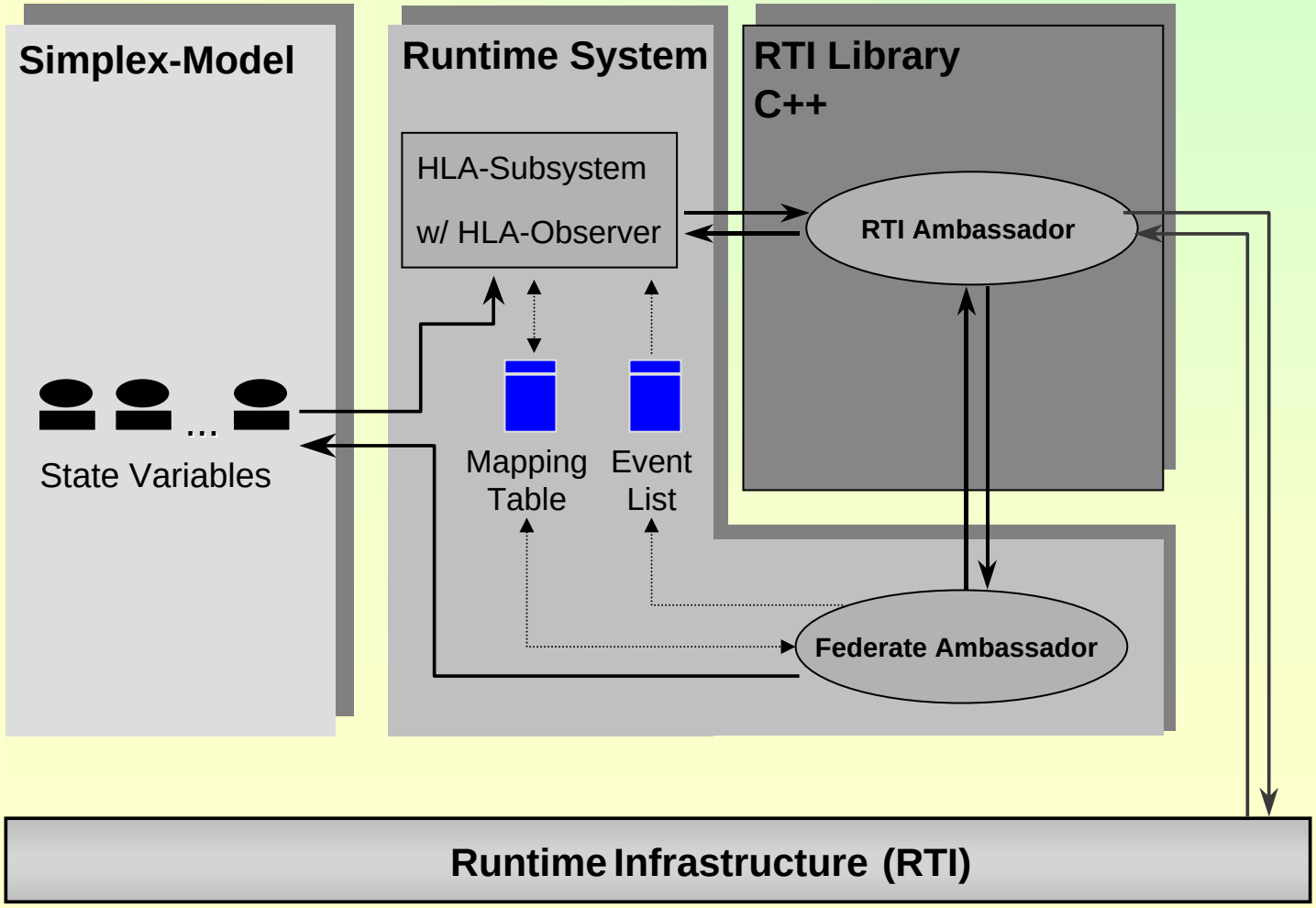
Das SLX-HLA-Interface nutzt den expliziten Ansatz



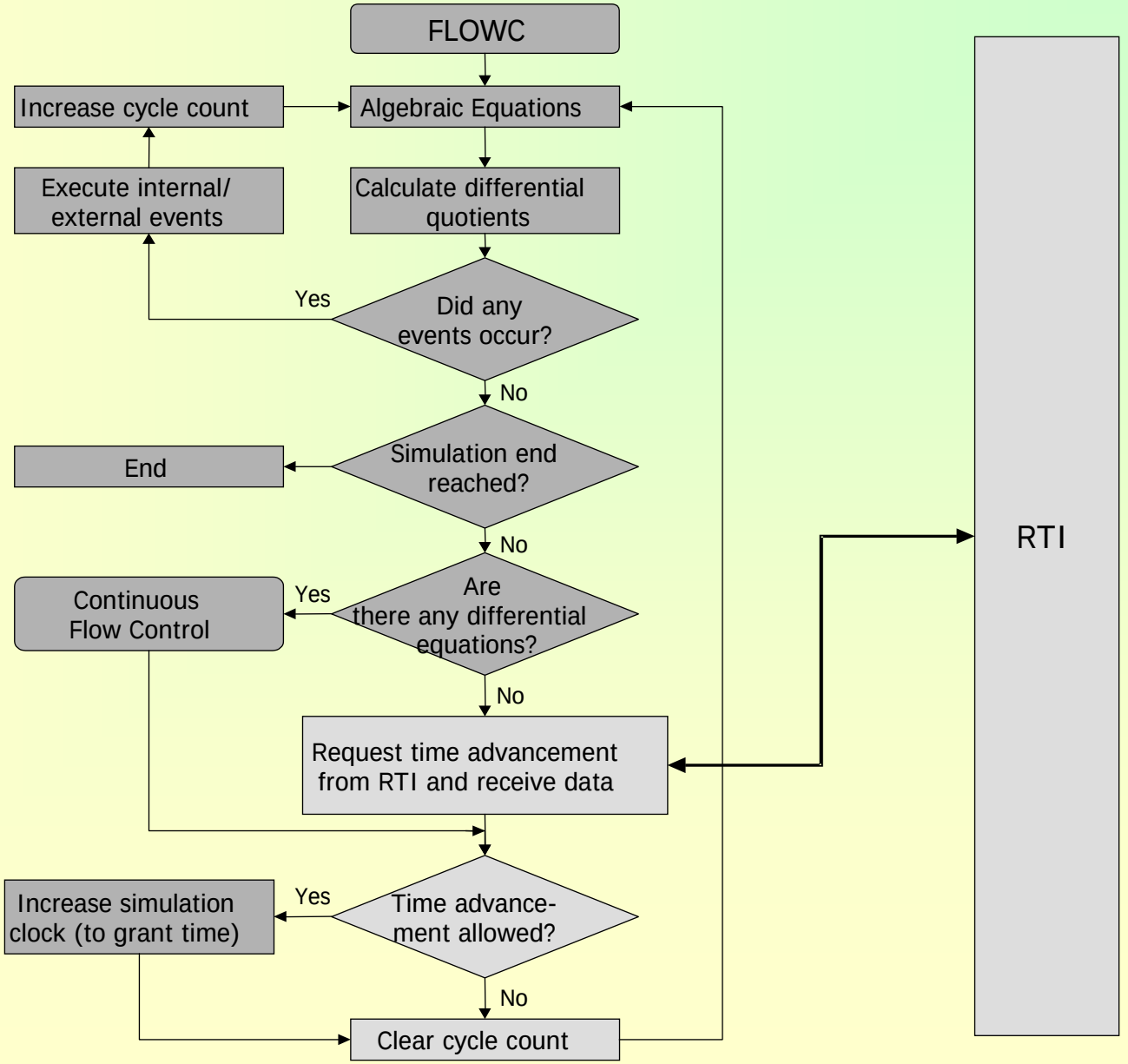
SLX Synchronisationsroutine

```
//Fork the synchronization thread with lowest priority in the entire model
fork priority -2 {
        forever {
            //determine time stamp of next local event
            NextEventTime= next_imminent_time();

             //Request advancement
            grantTime = RTI_NextEventRequestAvailable(NextEventTime);


            //Advance to grantTime
            advance grantTime-time;

            //Let external events (updates, interactions, etc.) take effect
            RTI_ReflectControlVariableChanges();

            //all outside changes have been taken care of;
            //yield control to simulation tasks
            yield;
        }
}
```

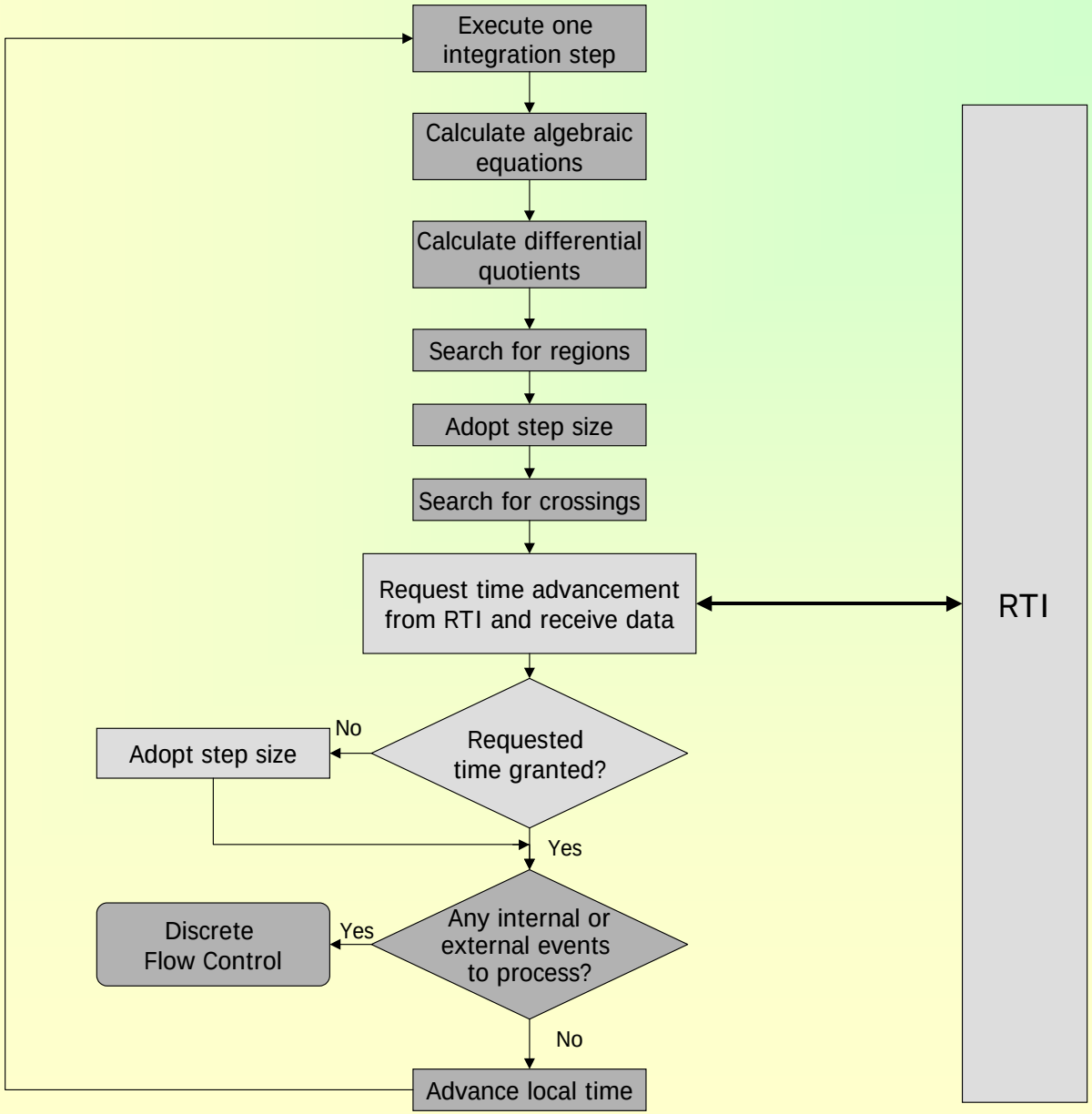
Das Simplex3-HLA-Interface basiert auf dem impliziten Ansatz



Integration des HLA Time Management in die Simplex Ablaufkontrolle (Diskreter Teil)



Integration des HLA Time Management in die Simplex Ablaufkontrolle (Kontinuierlicher Teil)



Vergleich der HLA-Schnittstellen von Simplex3 und SLX

- Simplex3: Impliziter Ansatz
  - Minimiert den Modellierungsoverhead
  - Erfordert Zugriff auf den Quelltexte des Simulators
  - Kommt mit Kompromissen bzgl. Performance und Flexibilität
- SLX: Expliziter Ansatz
  - Stellt hohe Flexibilität der Schnittstelle zur Verfügung
  - Erfordert Modellerweiterungen
  - Übertragbar auf alle Systeme mit externen Programmierschnittstellen

Überblick

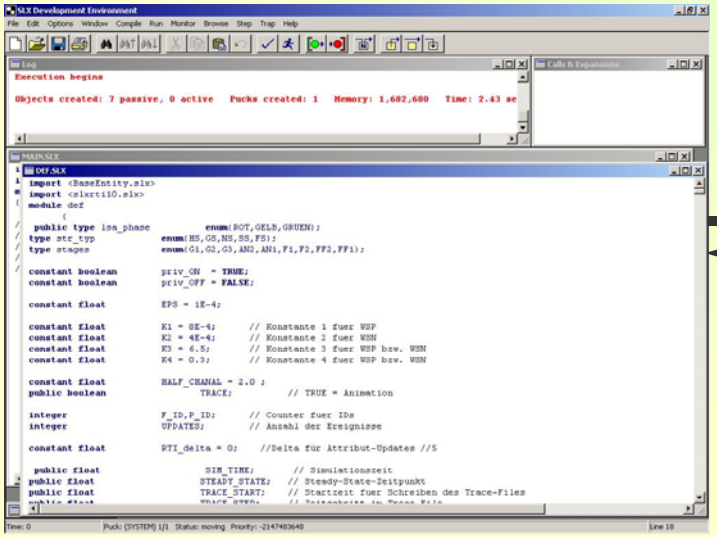
- Motivation und Zielstellung
- HLA als state-of-the-art für Simulationsinteroperabilität
- Migration von HLA in zivile Simulationssysteme
  - Anforderungen an Simulationssysteme
  - Alternativen für den Entwurf von HLA-Schnittstellen
- Praktische Evaluierung von HLA in zivilen Applikationen
- Zusammenfassung und Ausblick

Praktische Evaluierung von HLA in zivilen Applikationen

- Evaluierung von verschiedenen Interoperabilitätsaspekten
  - Zeitmanagement-Interoperabilität
  - Software Interoperabilität (C++/Java)
  - Hardware Plattform Interoperabilität (Intel/Sun/SGI)
- Identifizierung möglicher Schwächen von HLA
  - Haben zivile Anwendungen spezielle Bedürfnisse, die von HLA nicht adäquat adressiert werden?
  - Ist die Komplexität von HLA handhabbar?

Zivile Anwendungen: Verteilte Fahrsimulation

- Kombination eines Echtzeitfahrsimulators mit einer ereignisgesteuerten Verkehrssimulation

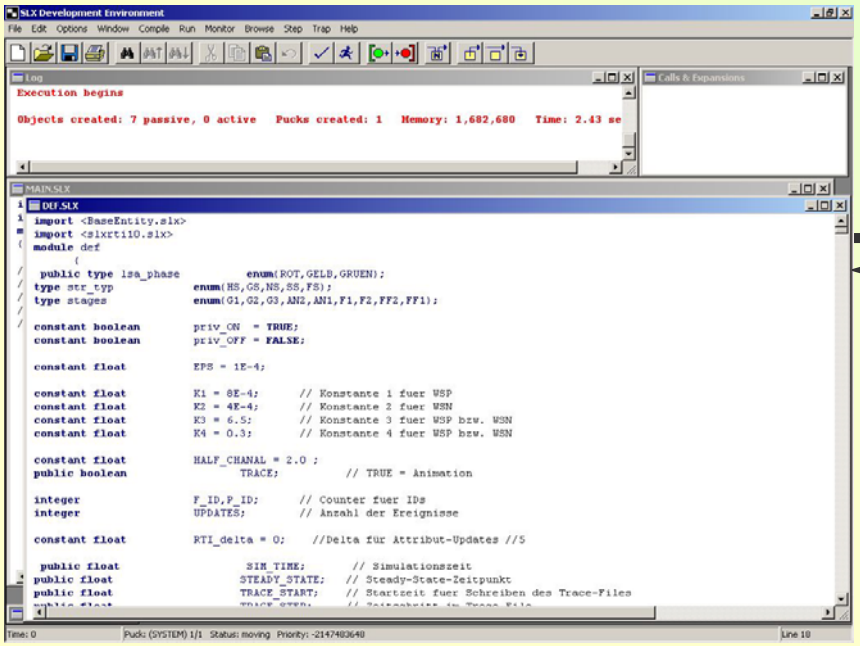


SLX/Windows NT/PC

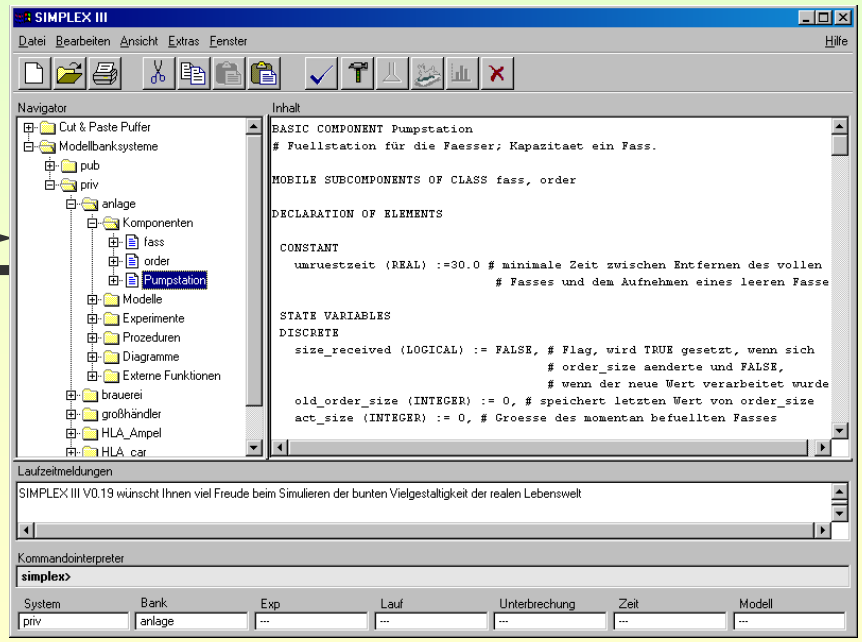
C++/SGI/Irix

Zivile Anwendungen: Logistische Probleme/Supply Chain Simulation

- Evaluierung von Interoperabilität zwischen diskreten und kombinierten Simulationsmodellen
- Evaluierung von Ownership Management Diensten



SLX/Windows NT/PC



Simplex3/Windows NT/PC

P4 Delta 1A File: E:\HLA-Prototypen\Abfuellanlage\Abfüllanlage_en.lay

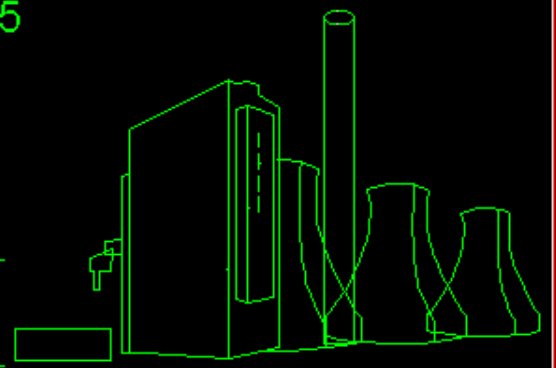
File Mode View Run Class Path Setup Help

9318.73 2%

Federate Logistics

(SLX)

Number of Trucks	Current	Total
Deliveries	6	11
to Magdeburg	2	4
to Dresden	4	7
Returns	4	5



Federate Chemical Plant

(Simplex 3)

Statistics

Waiting Orders: 0

Barrels Filled: 11

Empties Received: 1

State

busy / available

Current Barrel

50 l

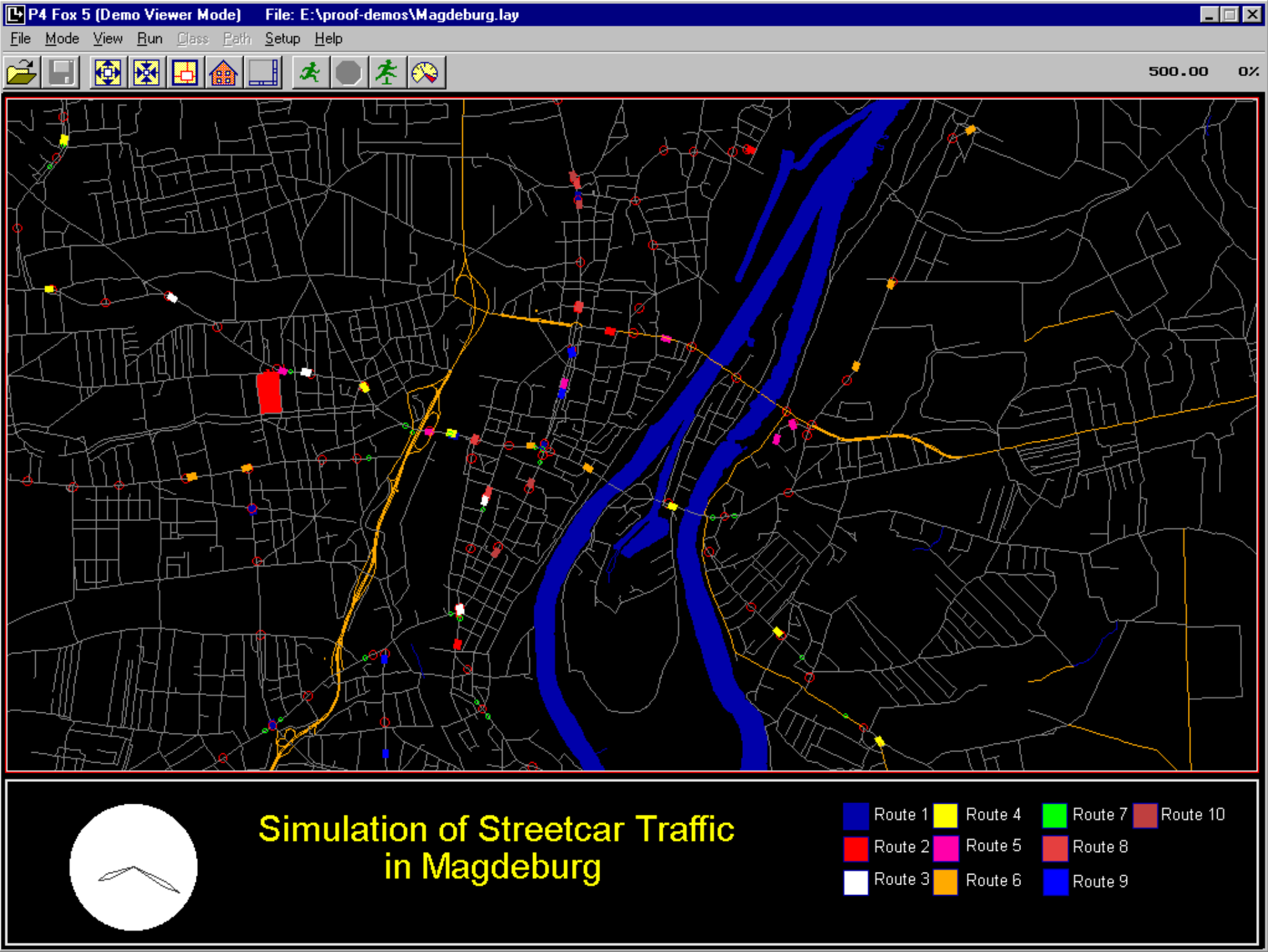
Filling Progress

100
75
50
25
0

0 75 150

Zivile Anwendungen: Integration von on-line Daten

- Nutzung der HLA-Mechanismen zur transparenten Integration verschiedener Datenquellen in Simulationen
- Straßenbahn-Federation
  - Online-Daten von Straßenbahnpositionen werden empfangen
  - Integration in ein fahrplanbasiertes Simulationsmodell
  - Möglichkeit des Umschaltens zwischen online/offline Daten und reiner Simulation
- Nutzbarkeit der hier angewendeten Mechanismen im Bereich der Fabrikplanung, Integration von Planungsdaten, etc.


P4 Fox 5 (Demo Viewer Mode) File: E:\proof-demos\Magdeburg.lay
File Mode View Run Class Path Setup Help
500.00 0%
Simulation of Streetcar Traffic in Magdeburg
Route 1
Route 2
Route 3
Route 4
Route 5
Route 6
Route 7
Route 8
Route 9
Route 10

Überblick

- Motivation und Zielstellung
- HLA als state-of-the-art für Simulationsinteroperabilität
- Migration von HLA in zivile Simulationssysteme
  - Anforderungen an Simulationssysteme
  - Alternativen für den Entwurf von HLA-Schnittstellen
- Praktische Evaluierung von HLA in zivilen Applikationen
- Zusammenfassung und Ausblick

Zusammenfassung

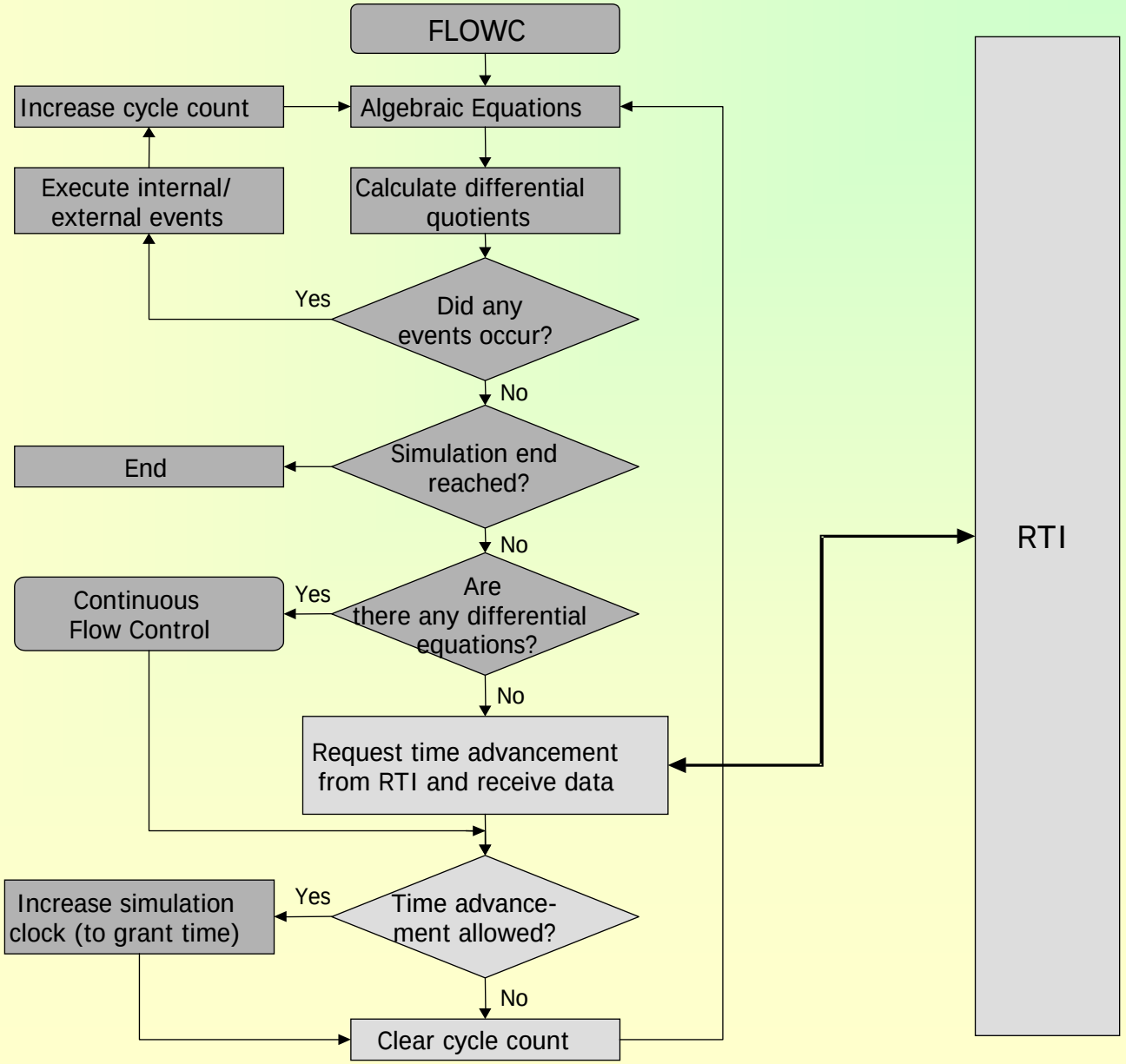
- Interoperabilität für Simulationsanwendungen im zivilen Bereich ist notwendig, aber bisher mangelhaft
- HLA kann die bestehenden Mängel überwinden, wenn die aufgestellten Anforderungen erfüllt werden
- HLA-Schnittstellen für Simulationssysteme können unter Nutzung verschiedener Ansätze entwickelt werden
  - Methodische Aspekte (explizit, implizit)
  - Technische Aspekte
- Proof-of-Concept Implementierungen für SLX, Simplex und Pro Model
- Demonstration sinnvoller Anwendungen

Ausblick

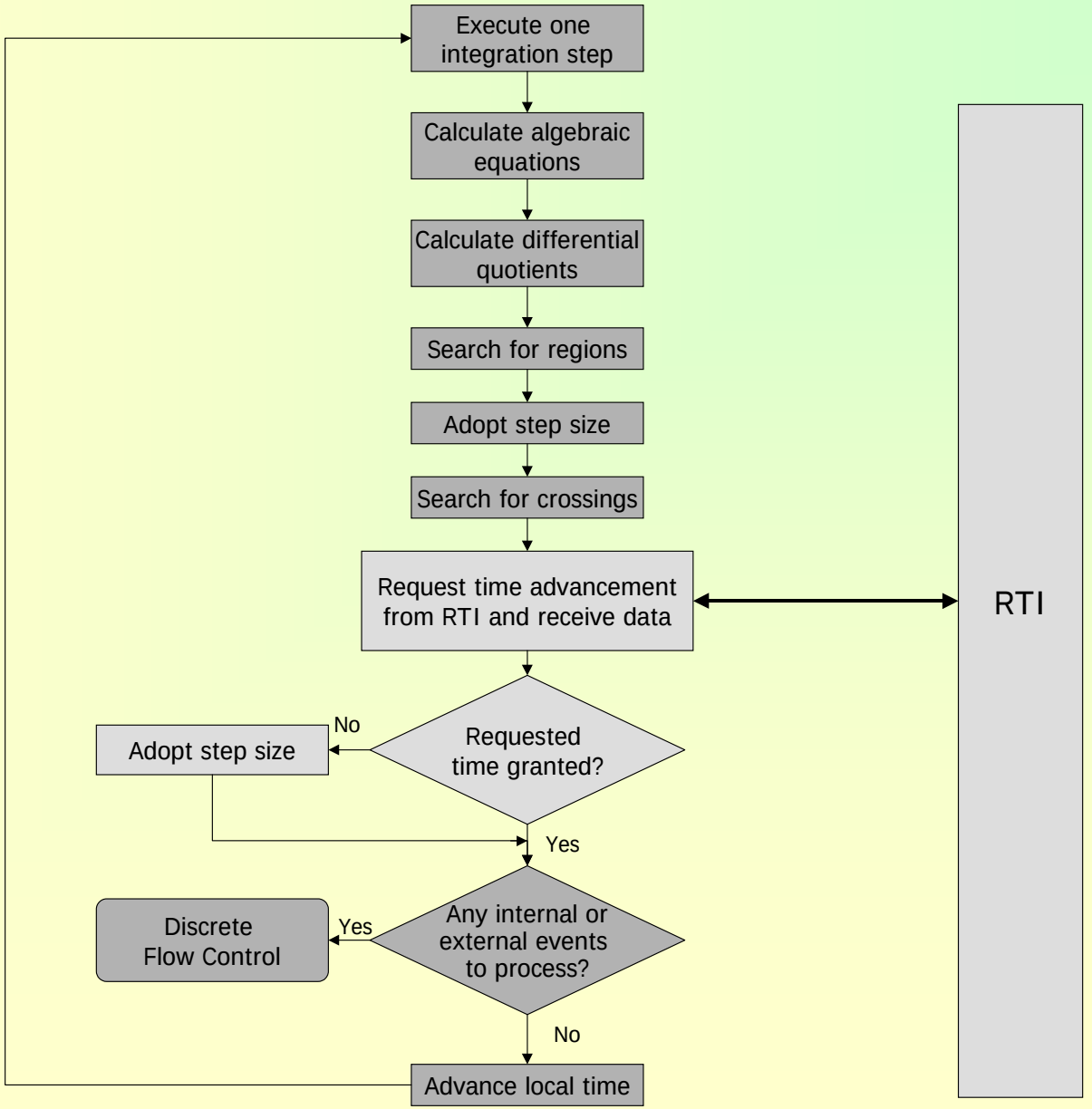
- Plug-and-Play Fähigkeiten noch in der Zukunft
  - Integration von Schnittstellen in die Systeme durch ihre Hersteller notwendig (impliziter Ansatz)
  - Verbesserungen von HLA bzgl. Handhabbarkeit und Nutzerfreundlichkeit notwendig
- Standardisierung im zivilen Bereich noch mangelhaft
  - Semantische Interoperabilität (z.B. Referenz-FOMs für Produktion, Fertigung und Logistik)
  - Minimal-Menge von HLA-Diensten, die alle Simulatoren unterstützen sollten
- Nachweis des Mehrwertes für industrielle Anwendungen

Backup-Slides

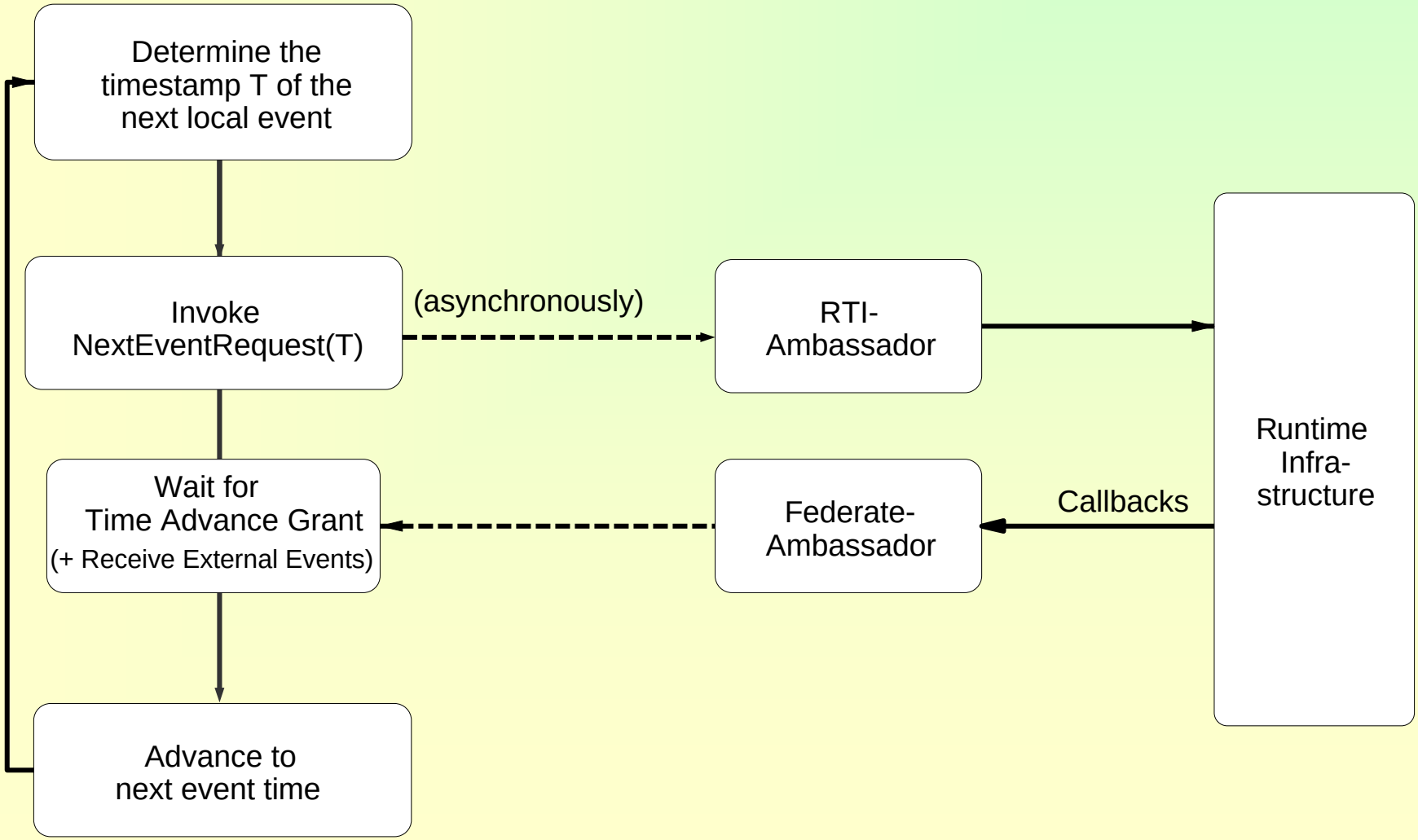
Incorporation of HLA Time Management into Simplex Flow Control (Discrete Part)



Incorporation of HLA Time Management into Simplex Flow Control (Continuous Part)



Local time advances must be coordinated with other participants



SLX Synchronization Loop

```
//Fork the synchronization thread with lowest priority in the entire model
fork priority -2 {
        forever {
            //determine time stamp of next local event
            NextEventTime= next_imminent_time();

             //Request advancement
            grantTime = RTI_NextEventRequestAvailable(NextEventTime);


            //Advance to grantTime
            advance grantTime-time;

            //Let external events (updates, interactions, etc.) take effect
            RTI_ReflectControlVariableChanges();

            //all outside changes have been taken care of;
            //yield control to simulation tasks
            yield;
        }
}
```