HLA-based Logistic Simulation

Project experiences

27. Februar 2008

Dipl.-Inf. Michael Raab
Prof. Dr.-Ing. habil. Thomas Schulze
Prof. Dr.-Ing. Steffen Straßburger
Contents

1. Preconditions
2. Intention
3. Requirements
4. Implementation
5. Summary & Demonstration
6. Outlook
Preconditions

• Production of vehicles
• Preexisting SLX simulation models, level of detail very high
  - To simulate assembly and paint processes
  - Originally developed to support factory design
  - Also been used to support factory operations
• One model can be used to simulate exactly one sub area of the entire factory, e.g. transmission assembly, paint shop, ...
• Interrelationships are not modeled and can’t be investigated
• Preexisting models are generic → can be adapted to a special use case by editing a set of configuration data
Existing generic models

Configuration Files → Class S (Station) → Object S.1 → Executable Simulation Model

Class A (Assembler) → Object A.1 → Object A.x → Object S.1 → Object S.n → Object A.y

... → compile → ... → Simulation Run creates Result files
Original Layout of the factory
Production system

- Front axle Assembly
- Transmission Assembly (Type 1)
- Transmission Assembly (Type 2)
- Cabin Assembly
- Chassis Assembly
- Paint Shop
- Final Assembly
Project objectives

• Simulate the complex system
  - Investigate the dependencies between the different sub models
    • Dimensioning of buffers between the sub areas
    • Coordination of production schedules
    • Detection of bottlenecks in the complex system regarding the required throughput
• Reuse the existing simulation models
• Using the HLA as middleware for the distributed simulation
  - RTI 1.3NG
  - Expert knowledge from previous projects
  - Existing HLA-SLX Wrapper library
Requirements (1)

1. Simulation Management
   - Prepare the existing models to be connected using the HLA
   - Support the user in creating the Federation
     - Which model should be coupled? What’s their relation?
   - Encapsulation of RTI-Services against the user
   - Start & Manage all Federates (Simulations) automatically

2. Extension of the existing models
   - Creation of a “consistent” solution for all models
     - Utilization for both monolithic und distributed simulation runs
Requirements (2)

3. Modeling the flow of material and information
   - Exchange of produced assembly parts between the different sub models
   - Ensuring the access to relevant information which belong to other sub models (e.g. buffer fill level)

4. Carry out simulation experiments
   - Parameterize the complex model
   - Collect data about the complex model
   - Calculate results (e.g. progress of the buffer fill level, waiting times)
   - Visualization of the simulation processes
Federation Overview

- Federate FrontAxle Assembly
  - SLX-HLA-Interface

- Federate Final Assembly
  - SLX-HLA-Interface

RTI
Modeling the flow of material and information

- Problem to solve can be mapped on a standardized interoperability model (see WSC2007, Taylor et. al)
  - Type A.2 → Synchronous entity passing using a bounded buffer
Realizing the flow of material and information
Extending the existing generic models

Configuration Files

DS Config xml

Generic Model

Class S (Station)
Class A (Assembler)

Distributed Simulation Configuration module

DS related class definitions & extensions
Class E (Entity)
Class B (Buffer)

compile

Executable Simulation Model

Object S.1
Object A.1
Object A.x
Object S.n
Object A.y

DS Initialization
- Join Federation
- Create & Register Objects (Buffers)
- Publish, Subscribe Object Updates & Interactions

Synchronization Loop
- Manage advancement of Simulation Time
- Receive & Process Interactions & Attribute Updates
Run Experiments

- Federate FrontAxle Assembly
  - SLX-HLA-Interface
- ... (omitted)
- Federate Final Assembly
  - SLX-HLA-Interface

- RTI
  - Federate Observer
  - Distributed Simulation Control Centre
Distributed Simulation Control Centre

- written in C++ .Net

- Usage:
  - Start RTI
  - Load & Distribute Configuration
  - Start Simulation Models
  - Monitor Experiment at Runtime
  - Produce Simulation Results
    - Reports
    - Animations
Configuration & Start

Distributed Simulation Control Centre

- Overall model config xml
  - Read

- Create specific model config xml
  - Read

- FrontAxle
  - Read

- Final
  - Read

Start
Collecting data

- Observer Federate is used to monitor the progress of the distributed simulation
- Preexisting information like:
  - Transfer of produced vehicle parts (Interaction)
  - Update of Buffer fill level (Attribute update)
- Additional data using HLA-Interactions:
  - Waiting times
    - Blocking due to overfull buffers
    - Due to assembly part deficits
  - Daily production statistics (e.g. Demand vs. Production)
Calculate & display simulation results

- Sub model related results:
  - Demand vs. Production
- Buffer related results:
  - Fill level (average, max, progress), throughput
- Relation Model-Buffer:
  - e.g. model waiting times due to full buffers of successor models
- Relation Model-Model:
  - Waiting times due to assembly part deficits (e.g. caused by a predecessor model which produces to slow)
- Simulation results are automatically generated, in form of html reports und MS Excel Charts
Visualize Simulation Progress

- Proof Animation
  - Production progress (started parts vs. finished parts)
  - Buffer (throughput, content)
Outlook

• Improvement of user-interface
  - Experiment configuration (define model structure & relationships)
  - Optimization with distributed models
  - Model administration
• Validation of complex distributed models
• Improvement of run-time performance
  - Investigating dynamic Lookaheads
Modellierung des Material- & Informationsflusses (1) - backup