Architectural Design & Evaluation
Of An Industrial AGV Transportation System
With A Multiagent System Approach

SATURN, 2006
Danny Weyns
DistriNet, Dept. Computer Science,
Katholieke Universiteit Leuven Belgium
Overview

• AGV Transportation System
• Software Architecture, ADD
• ATAM
  o Utility tree
  o Analysis of architectural approach
• Some lessons learned
AGV Transportation System
Main Functionalities

- Transport assignment
- Execution transports
- IO with machines
- Collision avoidance
- Deadlock prevention
- Battery charging
Main Quality Goals

• **Performance**
  - Transports/hour – bandwidth

• **Flexibility**
  - Deal with change autonomously, exploit opportunities

• **Openness**
  - Deal with AGVs that dynamically leave and enter the system
Traditional approach

- **Centralized architecture**
  - Server assigns transports to AGVs, plans routes etc.
  - Low level control AGVs is handled by E’nsor software

- **Main quality attributes**
  - Configurability (server is central configuration point)
  - Predictability (server manages execution of functionality)
EMC² Project

- Collaboration Egemin – DistriNet
- Project: 2004 – 2006 (4 FT)
- Main Goal
  - Cope with quality requirements: flexibility and openness
  - Investigate feasibility of applying decentralized architecture for AGV transportation system
- Approach: Situated Multiagent System
Situated Multiagent System

- What is a situated multiagent system (MAS)?
  - Set of autonomous entities (agents) explicitly situated in a shared structure (an environment)
  - Agents select actions “here and now”, they do not use long term planning (locality in time and space)
  - Interaction is at the core of problem solving (rather than individual capabilities)

Decentralized control

Adaptive behavior

Collective behavior
Overview

- AGV Transportation System
- Software Architecture, ADD
- ATAM
  - Utility tree
  - Analysis of architectural approach
- Some lessons learned
Software Architecture

- Architectural design process
  - Principles from Attribute Driven Design (ADD)
    - Recursive decomposition: select drivers, apply architectural approaches
  - Guided by:
    - Reference architecture for situated MAS
    - ObjectPlaces middleware

- Documentation
  - Architectural views / view packets
    - Deployment -- Module -- Component and Connector
Overview of the reference architecture
Deployment View: System

![Diagram of System Deployment View]

**KEY**
- Host Computer
- AGV
- Access Point
- Wired Network
- External System Connection
- Wireless Ethernet
Module Uses View: AGV Control System

AGV Agent

Local Virtual Environment

ObjectPlaces Middleware

E'nsor

KEY

X ----> Y

Layer

X uses Y
Communicating Processes View: Move action AGV
Attribute-Driven Design

• ADD with reference architecture
  o Reference architecture
    blueprint for architectural design
    provides build-in mechanisms
  o ADD is helpful
    as a design approach
    for refinement
Overview

- AGV Transportation System
- Software Architecture
- ATAM
  - Utility Tree
  - Analysis of architectural approach
- Some lessons learned
Architecture Trade-Off Analysis Method

• Goals ATAM
  o Articulation of business goals
  o A concise presentation of the architecture
  o Utility tree
  o Mapping architectural decisions to quality requirements
  o Tradeoff points, risks, non-risks
ATAM for AGV Transportation System

- **AGV Software Architecture**
  - Developed independent of concrete system (Product Line like)
  - Evaluation in context of particular project (tobacco warehouse)

- **Preparation**
  - Preparation utility tree (+ 4 days / 3 stakeholders, 1 evaluator)

- **ATAM**
  - June 16th, 2005 -- 10 stakeholders, 2 evaluators
  - Presentations: ATAM, business goals, architecture, approaches
  - Generation utility tree - analysis architectural approaches
  - Round-up
Overview

- AGV Transportation System
- Software Architecture
- ATAM
  - Utility Tree
  - Analysis of architectural approach
- Some lessons learned
Utility tree
(fragment)

- **Performance**
  - Capacity
    - \( (H,H) \) 12 AGVs with an availability of 85% should handle 112 transports per hour
    - \( (M,H) \) 12 AGVs with an availability of 100% should handle 140 transports per hour
  - Bandwidth Occupation
    - \( (H,M) \) The amount of communication should never exceed 60% of the bandwidth of the communication channel

- **Utility**
  - Openness
    - Controlled add and removal
      - \( (H,M) \) If an operator removes or adds AGV in a controlled way, the rest of the system continues working.
  - Flexibility
    - Flexible behavior
      - \( (H,M) \) If the transport has not been picked, the system dynamically changes that transport's assignment to the most suitable AGV.
      - \( (M,M) \) If an operator disables a node or an AGV blocks a path, AGVs choose an alternative route (if it exists)

- relative importance
- complexity to realize
Overview

- AGV Transportation System
- Software Architecture
- ATAM
  - Utility Tree
  - Analysis of architectural approach
- Some lessons learned
## Analysis Architectural approach

**Scenario:** The amount of communication, with maximal 12 E’GVs and a maximal load of 140 transports per hour, does not exceed 60% of the bandwidth of the 11Mbps communication channel.

<table>
<thead>
<tr>
<th>Architectural decisions</th>
<th>Sensitivity</th>
<th>Tradeoff</th>
<th>Risks</th>
<th>Nonrisks</th>
</tr>
</thead>
<tbody>
<tr>
<td>AD 1 Choice for .NET remoting</td>
<td>S2</td>
<td></td>
<td></td>
<td>NR3</td>
</tr>
<tr>
<td>AD 2 Agent located on machine controls E’GV</td>
<td>T2</td>
<td>R2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AD 3 Dynamic Contract-Net protocol for transport assignment</td>
<td>T3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AD 4 Two steps deadlock prevention mechanism</td>
<td></td>
<td></td>
<td>R3</td>
<td></td>
</tr>
<tr>
<td>AD 5 Unicast communication in Middleware</td>
<td>S3</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Overview

- AGV Transportation System
- Software Architecture
- ATAM
  - Utility Tree
  - Analysis of architectural approach
- Some lessons learned
Some Lessons Learned

• **Software architecture**
  - We gained a better insight in
    - Role of SA in building complex systems
    - Relationship between MAS and SA
  - Qualities trade off (flexibility versus performance)
  - SA constraints the system implementation
  - Lack of tool support to document SA
Some Lessons Learned

- ATAM
  - Utility Tree = most important instrument, yet time consuming -> good preparation is necessary
  - A complete evaluation of a complex system such as the AGV system is not manageable in one day
  - Evaluation of specific case versus product line like basic architecture hindered the discussions
Thanks!
Analysis Architectural approach
B-usage experiments

% of 11 Mbits

time (min:sec)