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

SATURN, 2006

Danny Weyns

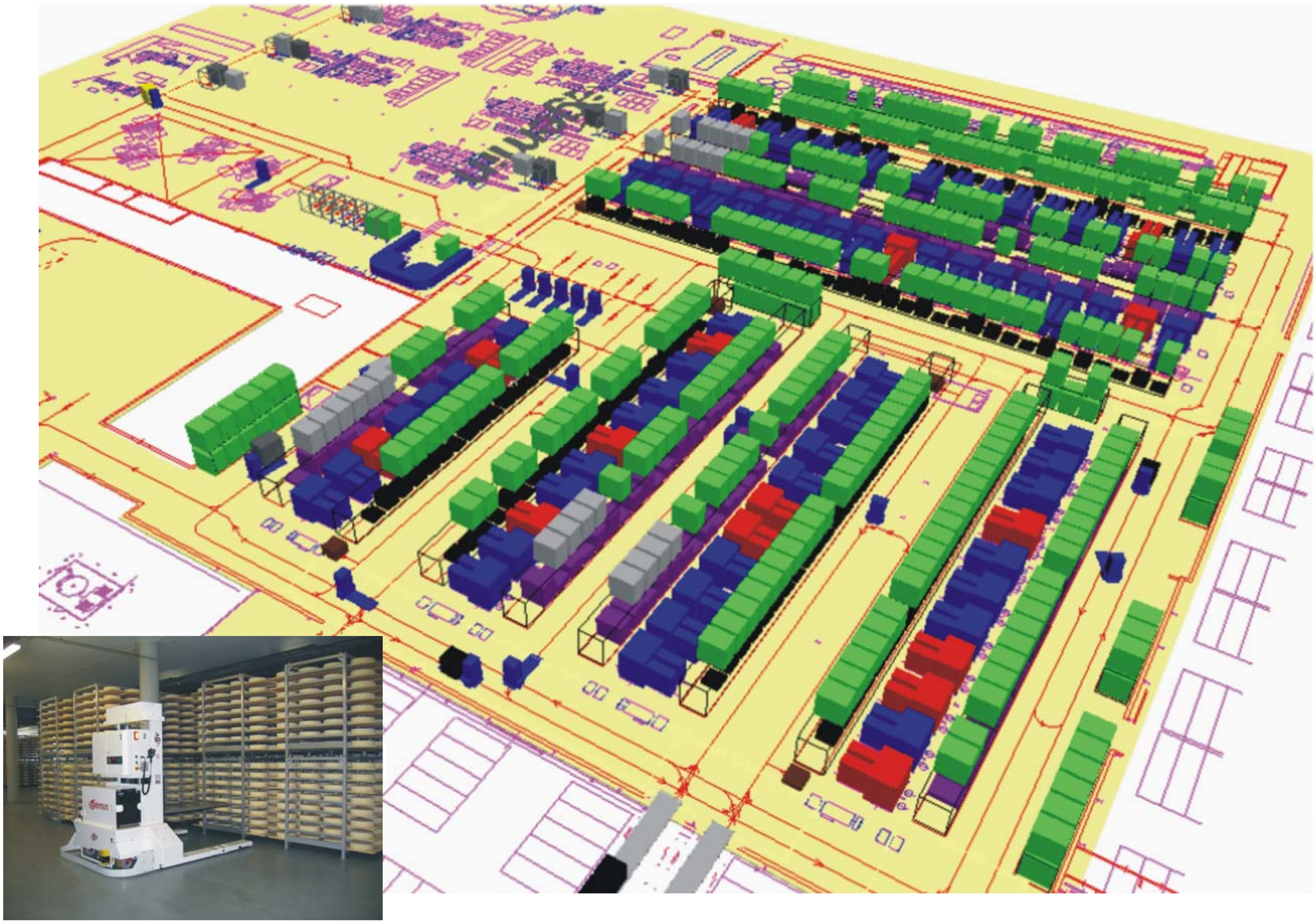
DistriNet, Dept. Computer Science,

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



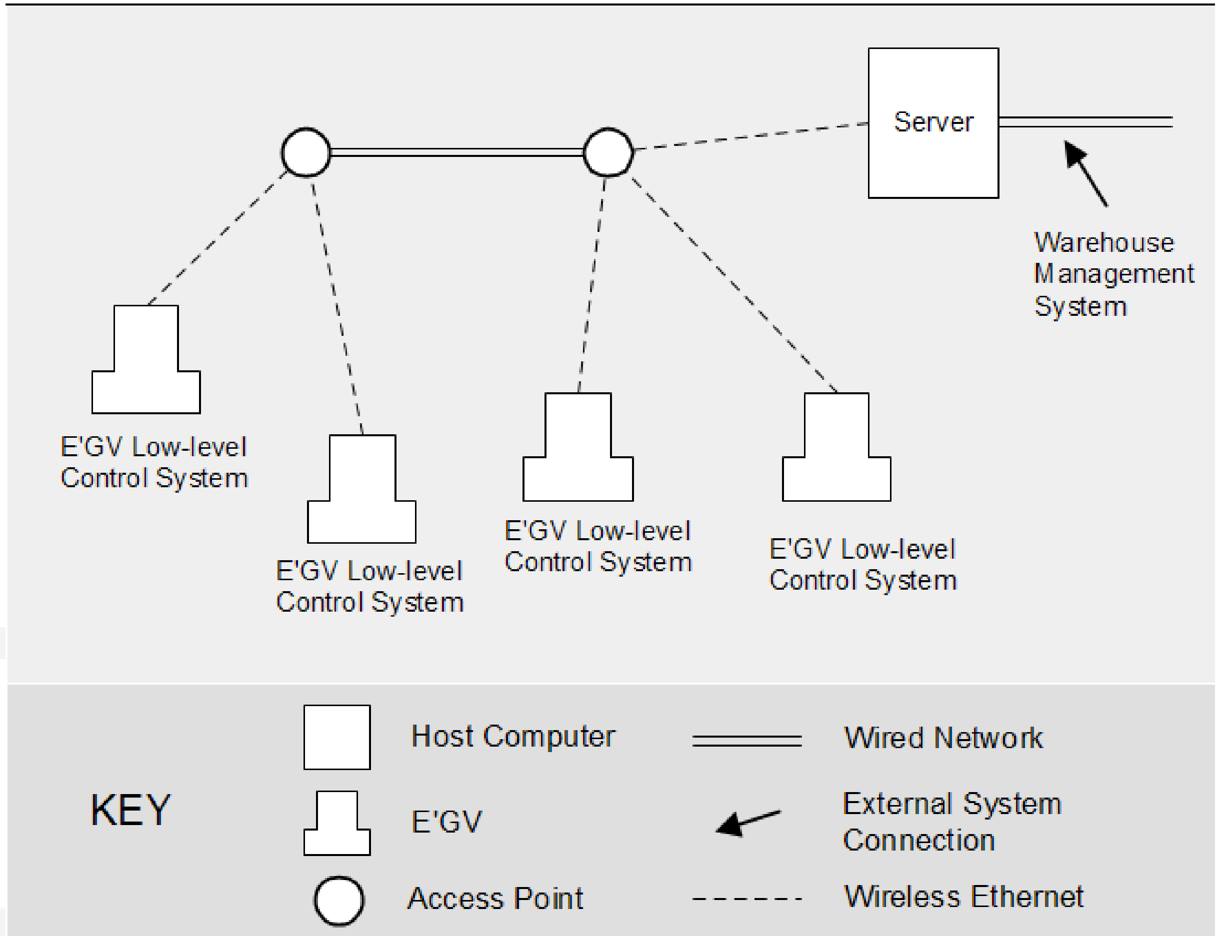
Main Functionalities

- **Transport assignment**
- **Execution transports**
- **IO with machines**
- **Collision avoidance**
- **Deadlock prevention**
- **Battery charging**

Main Quality Goals

- **Performance**
  - o Transports/hour – bandwidth
- **Flexibility**
  - o Deal with change autonomously, exploit opportunities
- **Openness**
  - o 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)

EMC^2 Project

- **Collaboration Egemin – DistriNet**
- **Project: 2004 – 2006 (4 FT)**
- **Main Goal**
  - o Cope with quality requirements: flexibility and openness
  - o 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

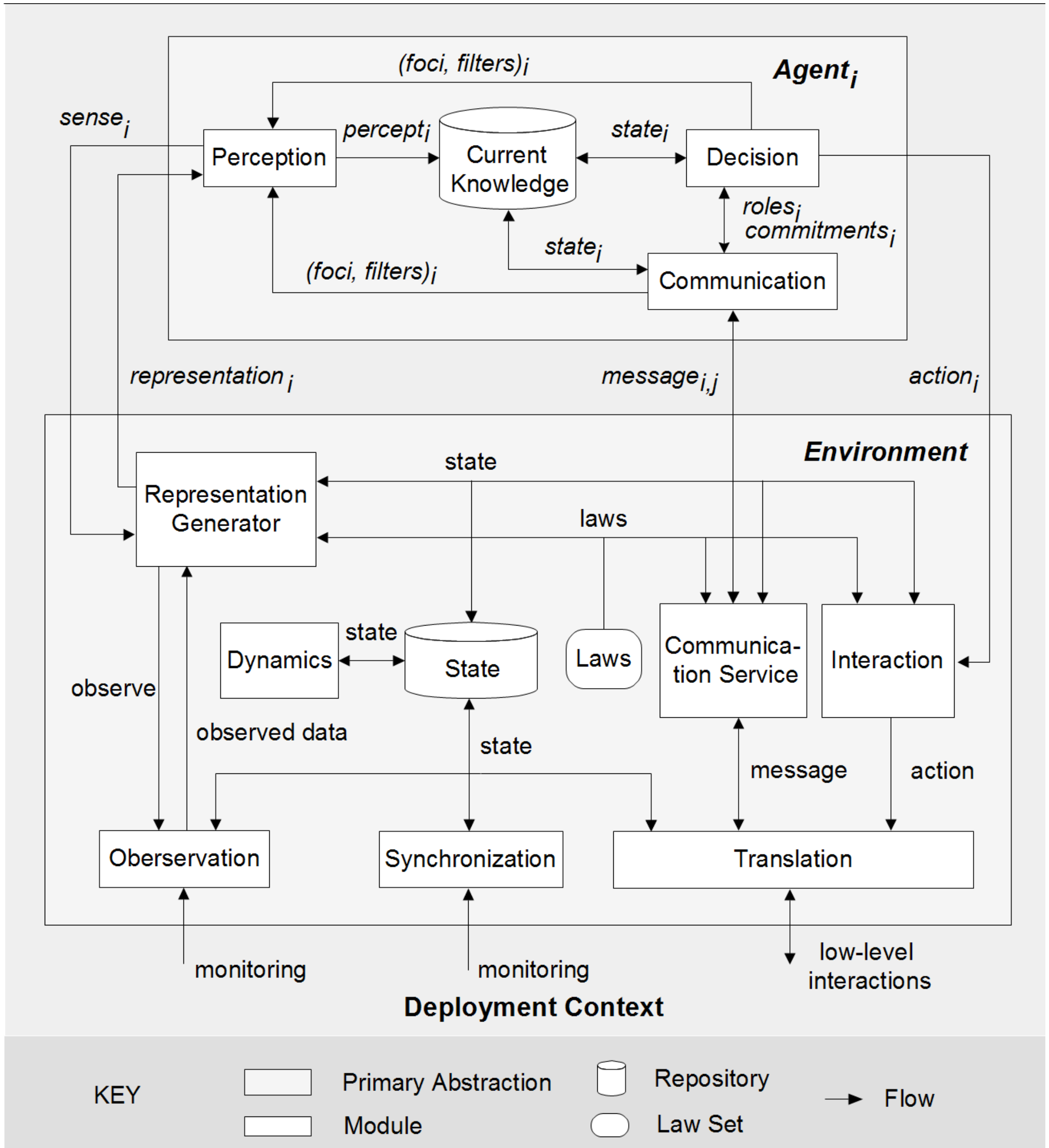
Overview

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

Software Architecture

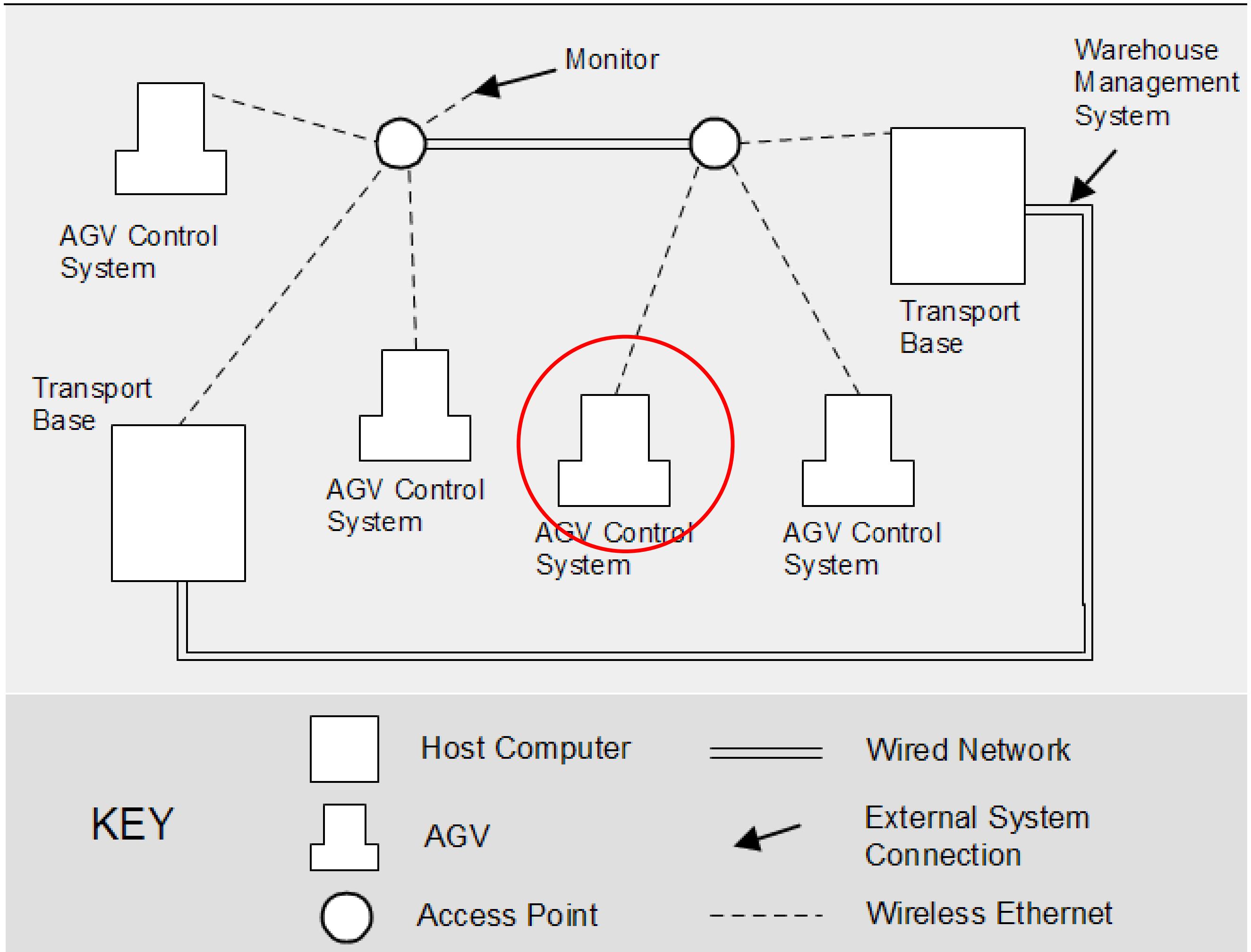
- Architectural design process
  - **Principles from Attribute Driven Design (ADD)**
    - Recursive decomposition: select drivers, apply architectural approaches
  - **Guided by:**
    - Reference architecture for situated MAS
    - ObjectPlaces middleware
- Documentation
  - **Architectural views / view packets**
    - Deployment -- Module -- Component and Connector

Overview of the reference architecture

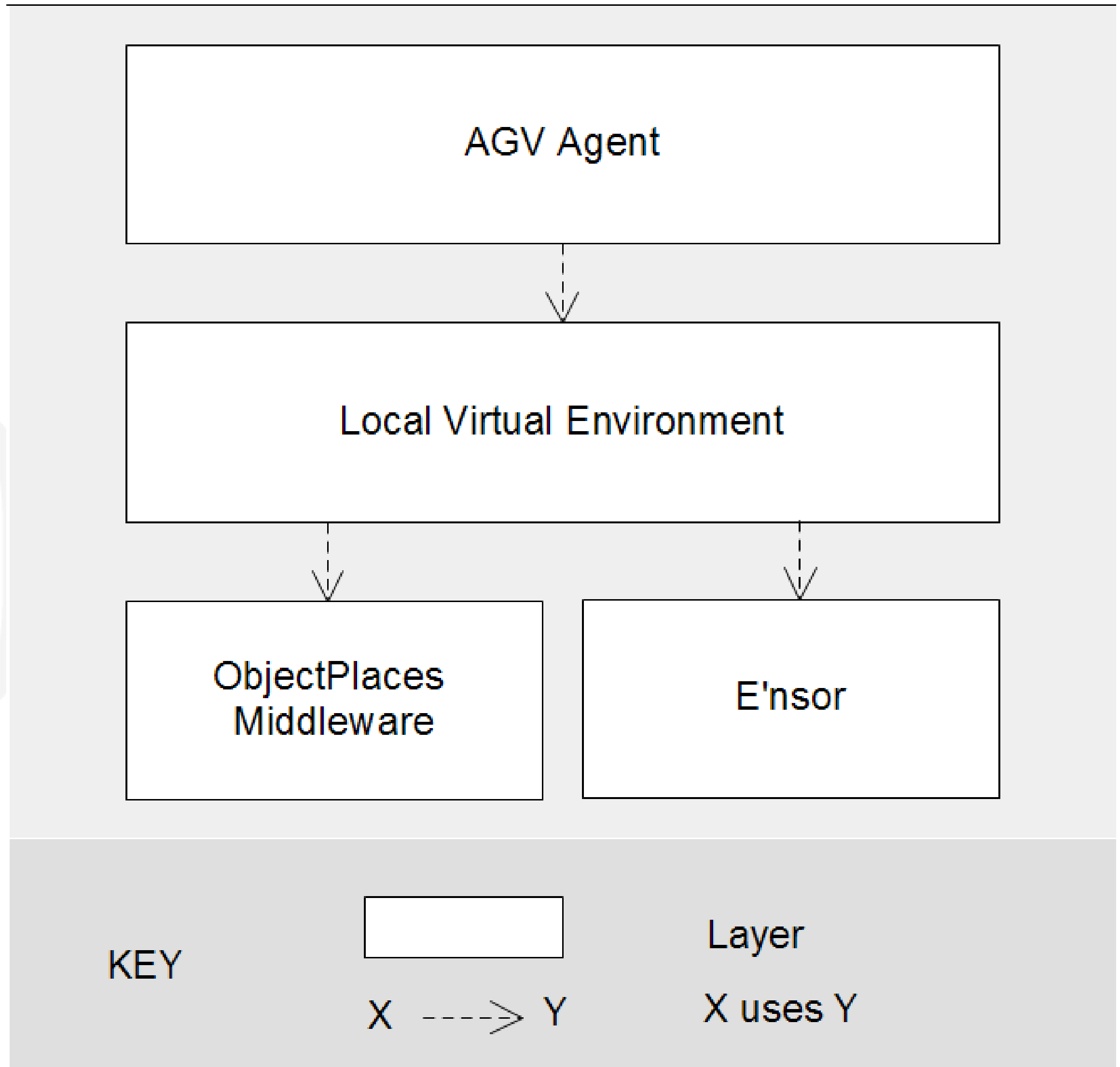


Katholieke Universiteit Leuven
departement computerwetenschappen

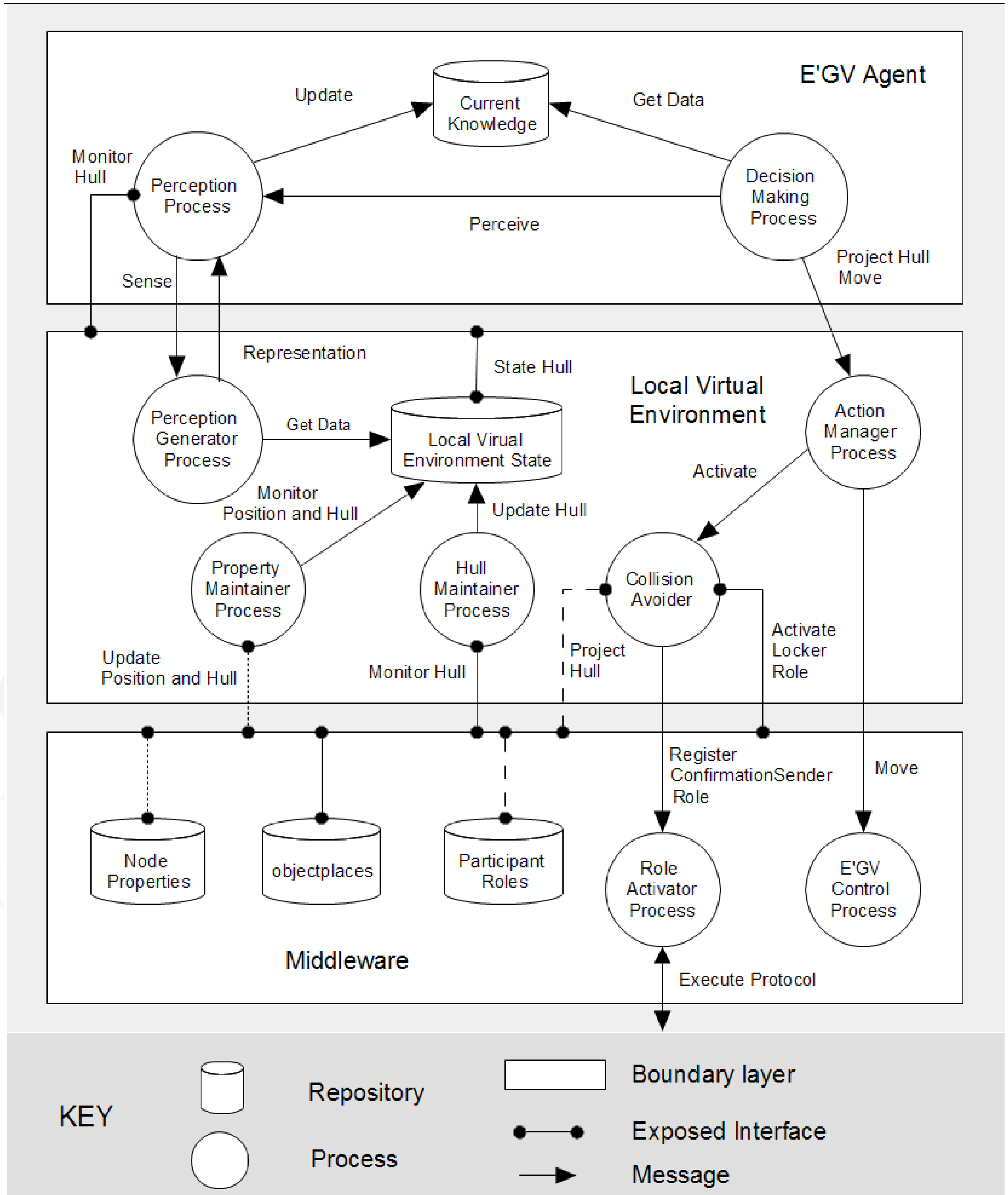
Deployment View: System



Module Uses View: AGV Control System



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**
  - o Utility Tree
  - o Analysis of architectural approach
- **Some lessons learned**

Katholieke Universiteit Leuven
departement computerwetenschappen

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
  - o Utility Tree
  - o Analysis of architectural approach
- **Some lessons learned**

Katholieke Universiteit Leuven
departement computerwetenschappen

Utility tree
(fragment)

Utility	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
	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 bloks a path, AGVs choose an alternative route (if it exists)

relative importance

complexity to realize

Katholieke Universiteit Leuven

departement computerwetenschappen

Overview

- AGV Transportation System
- Software Architecture
- ATAM
  - o Utility Tree
  - o 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.

Architectural decisions	Sensitivity	Tradeoff	Risks	Nonrisks
AD 1 Choice for .NET remoting	S2			NR3
AD 2 Agent located on machine controls E'GV		T2	R2	
AD 3 Dynamic Contract-Net protocol for transport assignment		T3		
AD 4 Two steps deadlock prevention mechanism			R3	
AD 5 Unicast communication in Middleware	S3			

Overview

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

Some Lessons Learned

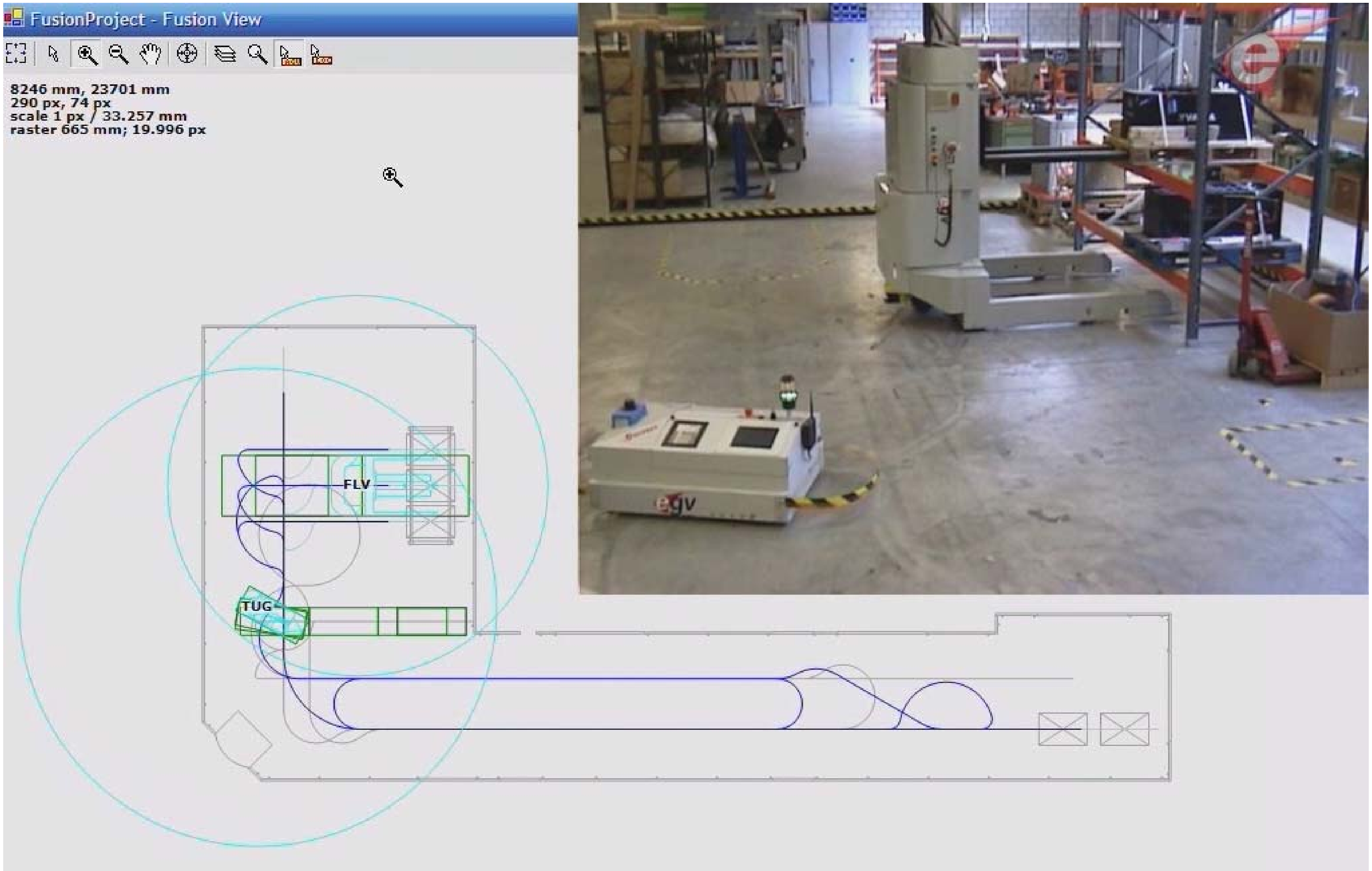
- **Software architecture**
  - o We gained a better insight in
    - Role of SA in building complex systems
    - Relationship between MAS and SA
  - o Qualities trade off (flexibility versus performance)
  - o SA constraints the system implementation
  - o Lack of tool support to document SA

>

Some Lessons Learned

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

Katholieke Universiteit Leuven
departement computerwetenschappen

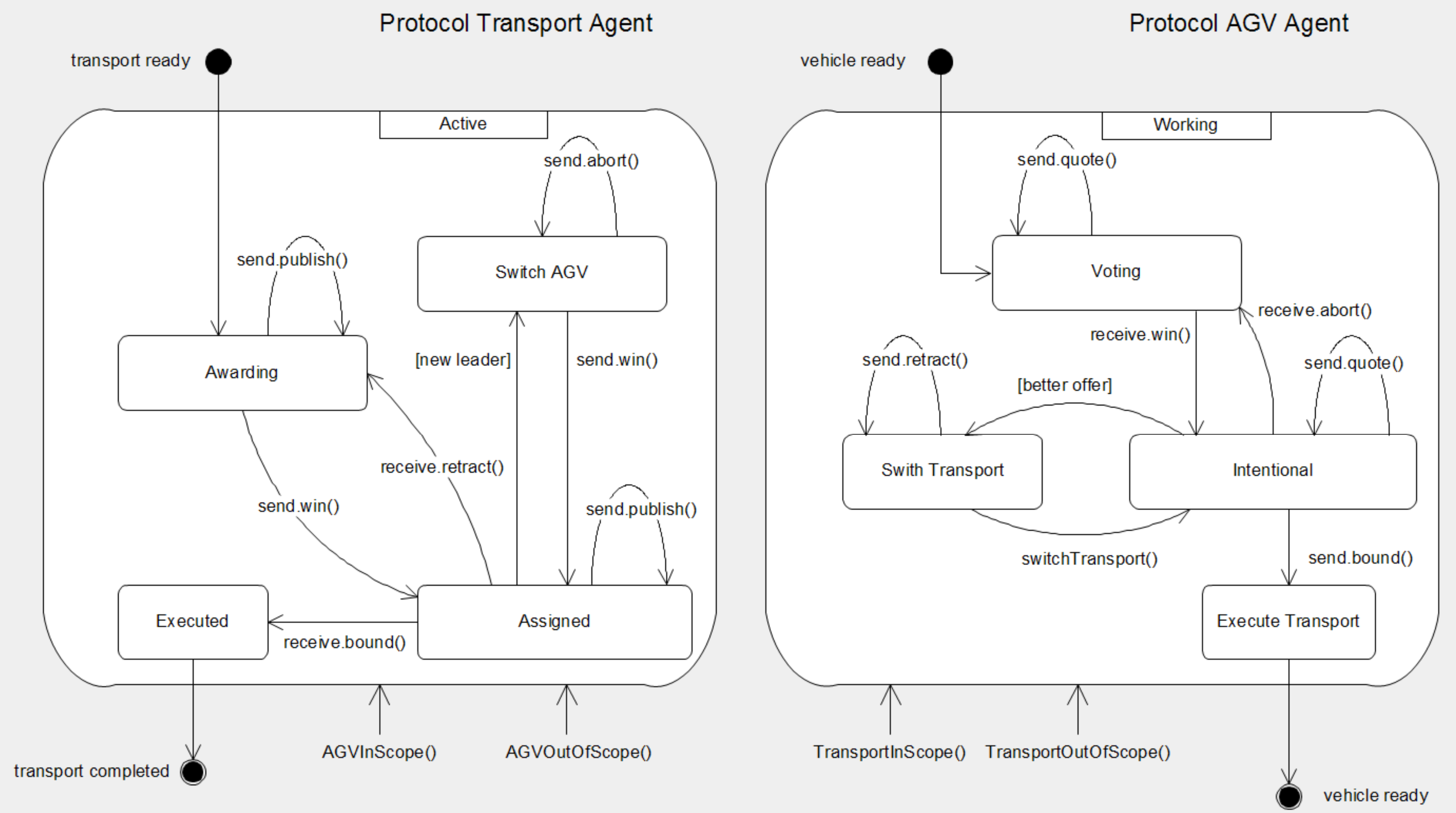


Thanks!

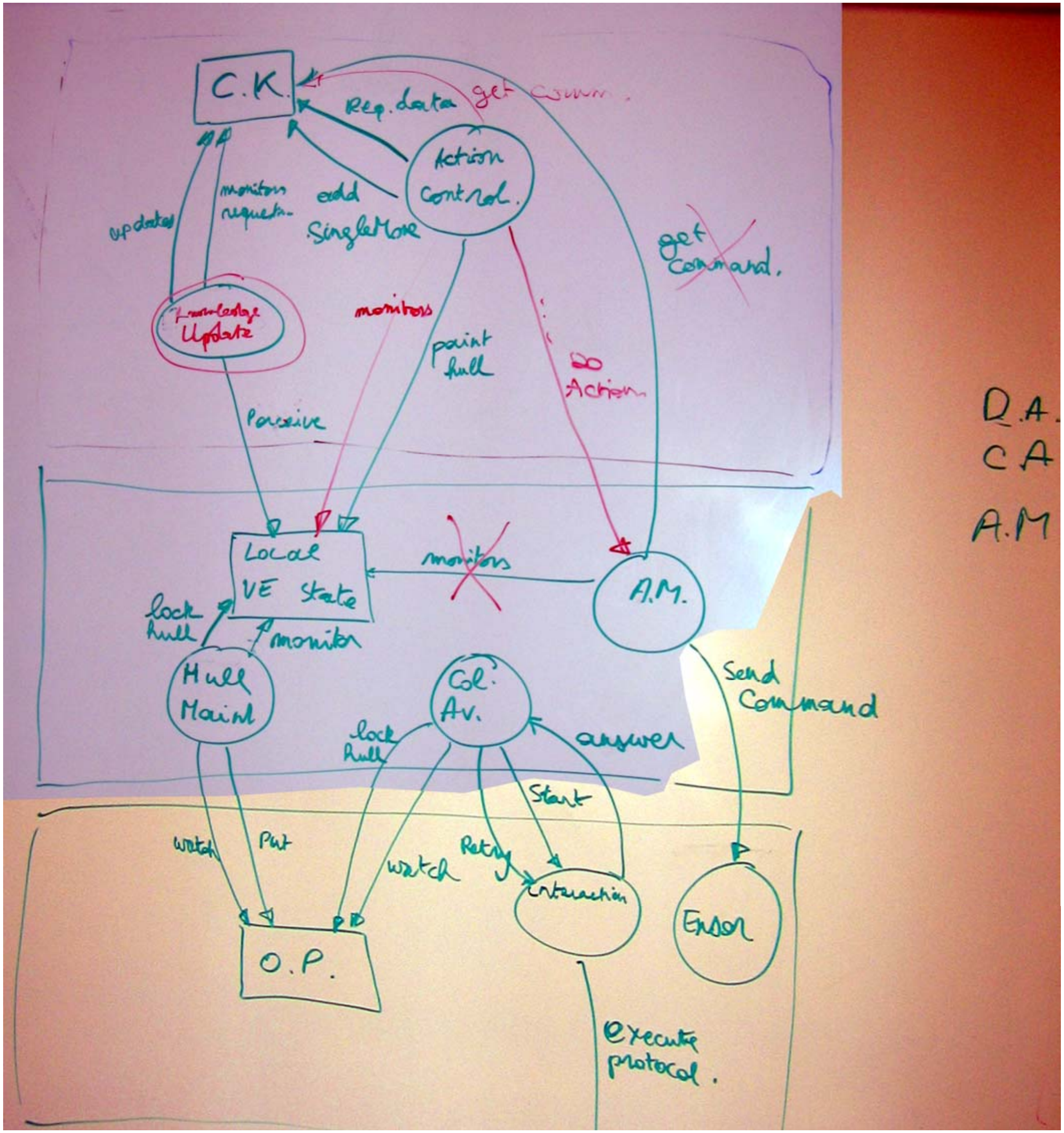
Katholieke Universiteit Leuven
departement computerwetenschappen

17 May 2006

Analysis Architectural approach



KEY UML Statechart



Katholieke Universiteit Leuven

17 May 2006

B-usage experiments