Applying the ATAM to an Architecture for Decentralized Control of a Transportation System

Nelis Boucké, Danny Weyns, Kurt Schelfthout, and Tom Holvoet

Distrinet, KULeuven, Celestijnenlaan 200A, Leuven, Belgium
{nelis.boucke, danny.weyns, kurt.schelfthout,
tom.holvoet}@cs.kuleuven.be

Abstract. For two years, we have been involved in a challenging project to develop a new architecture for an industrial transportation system. The motivating quality attributes to develop this innovative architecture were flexibility and openness. Taking these quality attributes into account, we proposed a decentralized architecture using multiagent systems (MASs). A MAS consists of multiple autonomous entities that coordinate with each other to achieve decentralized control. The typical advantages attributed to such decentralized architecture are flexibility and openness, the motivating quality attributes to apply MAS in this case.

The Architecture Tradeoff Analysis Method (ATAM) was used to provide insights wether our architecture meets the expected flexibility and openness, and to identify tradeoffs with other quality attributes. Applying the ATAM proved to be a valuable experience. One of the main outcome of applying the ATAM was the identification of a tradeoff between flexibility and communication load that results from the use of a decentralized architecture.

This paper describes our experiences in applying the ATAM to a MAS architecture, containing both the main outcomes of the evaluation and a critical reflection on the ATAM itself.

1 Introduction

For two years, Distrinet [1] has been involved in a challenging R&D project (EMC^2 [2]) to develop a decentralized architecture for an industrial transportation system. Our industrial partner, Egemin N.V. [3], is a Belgian manufacturer of Automatic Guided Vehicles (AGVs) and control software for automating logistics services in warehouses and manufactories using AGVs. Traditionally, one computer system (central server) is in charge of numerous complex and time-consuming tasks such as routing, collision avoidance, or deadlock avoidance; the AGVs themselves have little autonomy. This traditional architecture has successfully been deployed in numerous practical installations, but the evolution of the market has put forward new requirements for AGV Transportation systems [4]. Especially in highly dynamic systems, where the situation changes frequently, problems are experienced. A new and innovative architecture is needed that offers additional qualities, like flexibility and openness, to cope with the highly dynamic environments.

Taking these quality attributes into account we proposed a decentralized architecture using multiagent systems (MASs). Typical advantages attributed to a MAS architecture are flexibility and openness, being the motivating quality attributes to apply MAS for the

C. Hofmeister et al. (Eds.): QoSA 2006, LNCS 4214, pp. 180–198, 2006.

AGV transportation system. A second motivation, from the research perspective, was the opportunity to evaluate MASs and our reference architecture [5] in a real industrial application and asses if it really fulfilled the attributed quality attributes. The Architecture Tradeoff Analysis Method (ATAM) [6,7] was used to provide insights wether our architecture meets the expected flexibility and openness and to identify tradeoffs with other qualities. This paper describes our experiences in applying the ATAM to the MAS-based architecture, containing both the main outcomes in terms of tradeoffs and what we have learned and a critical reflection on the ATAM itself.

Overview. The remainder of this paper is structured as follows. Section 2 describes the requirements, the motivation and a short overview of the MAS architecture. Section 3 describes the outcomes of the ATAM workshop. Section 4 reflects on the ATAM workshop. Section 5 describes related work and we conclude in section 6.

2 Decentralized Architecture for Automatic Guided Vehicles

An AGV transportation system uses unmanned vehicles that are custom made to be able to transport various kinds of loads, from basic or raw materials to completed products. Typical applications are repackaging and distributing incoming goods to various branches, or distributing manufactured products to storage locations. An AGV uses a battery as its energy source. AGVs can move through a warehouse guided by a laser navigation system, or following a physical path on the factory floor that is marked by magnets or cables that are fixed in the floor. Egemin N.V., our industrial partner for the EMC^2 project, develops and delivers such AGV transportation systems tailored to the needs of the specific production-plant or warehouse. Thus AGV transportation systems is a product-line system that is used in several concrete products with different functional and (possible contradicting) quality requirements.This section describes the main functionalities, the important quality attributes, the motivation to apply a MAS architecture and a short overview of the MAS architecture for the AGV Transportation system.

2.1 Main Functionalities

The main functionality the system should perform is handling transports, i.e. moving loads from one place to another. Transports are generated by client systems. Client systems are typically warehouse management systems, but can also be particular machines, employees or service operators. In order to execute transports, the main functionalities the system has to perform are:

- Transport assignment: transports are generated by client systems and have to be assigned to AGVs that can execute them.
- Routing: AGVs must route efficiently through the layout of the warehouse when executing their transports.
- Gathering traffic information: although the layout of the system is static, the best route for the AGVs in general is dynamic, and depends on the current conditions in the system. Gathering traffic information allows the system to adapt the routing of the AGVs to these dynamic conditions.

– Collision avoidance: obviously, AGVs may not collide. AGVs can not cross the same intersection at the same moment. Similar safety measures are also necessary when AGVs pass each other on closely located paths.
– Deadlock avoidance: since AGVs are relatively constrained in their movement (they cannot divert from their path), the system must ensure that AGVs do not find themselves in a deadlock situation.

When an AGV is idle it can park at a free park location; however, when the AGV runs out of energy, it has to charge its battery at one of the charging stations.

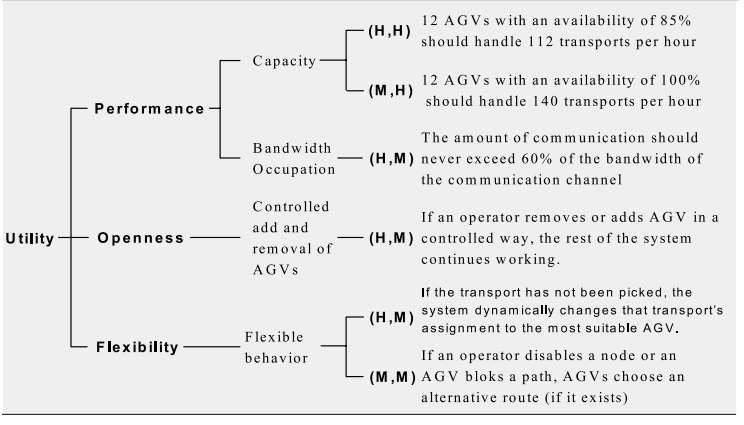

Fig. 1. Excerpt of utility tree for AGV transportation system

2.2 Quality Requirements

Fig. 1 shows an excerpt from the utility tree to illustrate important quality attributes: performance, openness and flexibility. The utility tree is an instrument to make quality attributes explicit in the form of scenarios, to structure these scenarios in a hierarchical fashion and to make the importance of each scenario explicit by putting priorities on them. Concretely, each scenario is assigned a ranking that expresses its priority relatively to the other scenarios. Prioritizing takes place in two dimensions. The first mark (High, Medium or Low) of each tuple refers to the importance of the scenario to the success of the system, the second to the difficulty to achieve the scenario. The tree serves as a guidance for design and evaluation of this architecture.

In the dictionary [8] flexibility is defined as *the quality of being adaptable, i.e. being able to change—or be changed—to fit changed circumstances.* Flexibility enables a software system to be adaptable with respect to variable circumstances during execution. For example, an AGV transportation system exposes flexibility if it is able to deal with disturbances in load supplies and exploiting opportunities. Openness is defined as *the quality of being open, i.e. characterized by an attitude of ready accessibility, affording unobstructed entrance and exit.* Openness enables a software system to cope with components that come and go during execution. For example, an AGV transport system exposes openness if this system is able to continue its work when AGVs leave or enter

the system. Flexibility and openness are of importance because they were the motivating quality attributes to come up with a new architecture [4]. Finally, *performance* is of high importance because the AGV transportation system is expected to process transports as efficiently as possible: a client wants a minimal number of vehicles to handle the transportation task load.

2.3 Motivation for MAS Architecture

Taking into account the quality attributes of the previous section and our experience with MAS [5,9], we proposed to use a decentralized architecture realized by a multi-agent system (MASs).A MAS consists of a set of agents, situated in an environment, that cooperate to solve a complex problem in a decentralized way [10,11,12]. An agent is an autonomous entity that has local access to the environment. Agents can flexibly adapt their behavior according to the changing circumstances in their vicinity. The general idea of using a MAS architecture for the transportation system is to put more autonomy in the AGV vehicles itself allowing for both flexibility and openness. In the decentralized solution, the AGV vehicles and the transports become autonomous agents that make decisions based on their current situation, and that coordinate with the other agents to ensure the system as a whole processes the transports [4].

The motivations to start the project and apply a MAS architecture for the transportation system are twofold. Firstly, the evolution of the market put forward new requirements. Although the traditional centralized architecture was deployed in numerous companies, it was less suited for other companies. This was a major motivating factor for Egemin to apply a MAS architecture for a transportation system. Secondly, it provided the opportunity to evaluate MASs in a real industrial application. In this way we could obtain insights obtain insights wether our architecture meets the expected flexibility and openness quality attributes, and we could identify tradeoffs with other quality attributes.

2.4 Short Overview of the Architecture

This section provides a short overview of the architecture. The description does not cover the complete architecture but is meant for illustration purposes. For more details we refer to the architectural documentation of [13, pag 69-138].

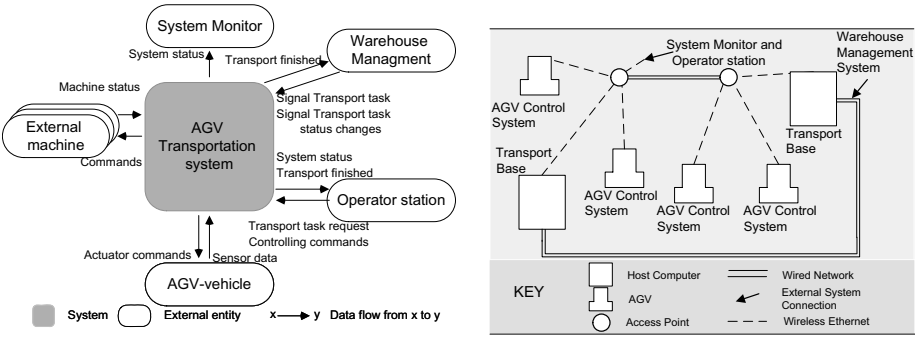

Fig. 2. Context diagram (left) and deployment view (right) of the AGV transportation system

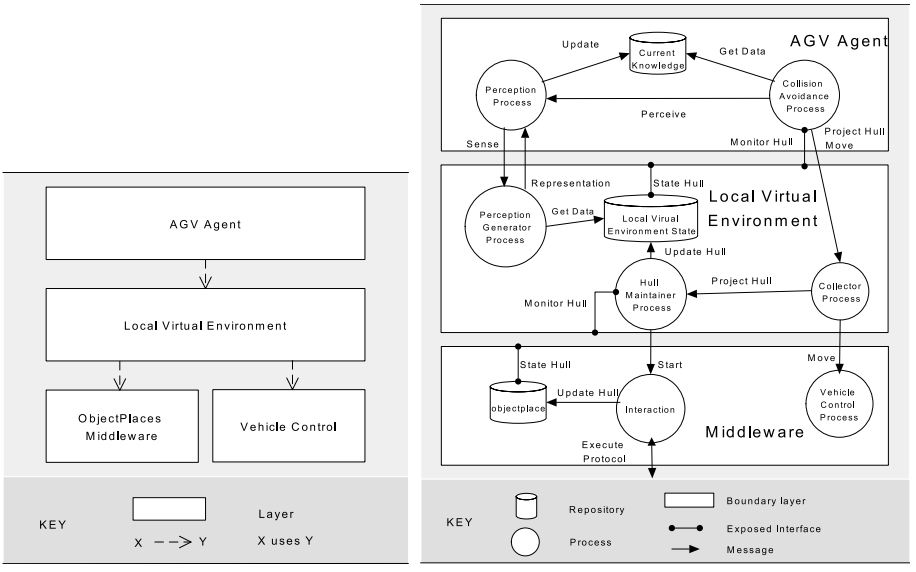

Fig. 3. Layered uses view (left) and process view (right) of AGV control system

Context Diagram and Deployment. Fig. 2 depicts the context diagram and deployment view of the *AGV Transportation System*. The AGV transportation system consists of two subsystems, *Transport Base* and *AGV Control System*. Transport bases receive transport requests from the *Warehouse Management System*, and are responsible for assigning the transports to AGVs. For each transport, a new transport agent is created. The transport agent is responsible for assigning the transport to an AGV and for reporting the status of the transport to the warehouse management system. The example contains two transport bases, each one responsible for one zone of the layout. AGV control system is responsible for ensuring that the AGV completes the assigned transport. Each AGV machine is equipped with low-level control software, called *Vehicle Control*, that takes high level commands from the AGV control system like move, pick, drop, ... The vehicle control system handles the physical interaction with the environment such as staying on track, turning, or determining the current position. The transport bases are deployed on stationary hosts, while the AGV control systems are deployed on the mobile AGV machines. The communication infrastructure provides a wired network that connects the warehouse management system with the transport bases and a wireless network that enables communication between mobile AGVs and with the transport bases. Debugging and monitoring the system is possible trough the *System Monitor* and the *Operator Station*. The *External Machines* represent possible machines the AGV Transportation has to interact with.

The AGV Control System. Now we zoom in on the internals of the AGV control system. The left side of Fig. 3 depicts the module view of AGV control system. The layers architectural pattern is used to manage complexity. The *AGV Agent* is responsible for executing transports and controlling the AGV. The *Local Virtual Environment* offers

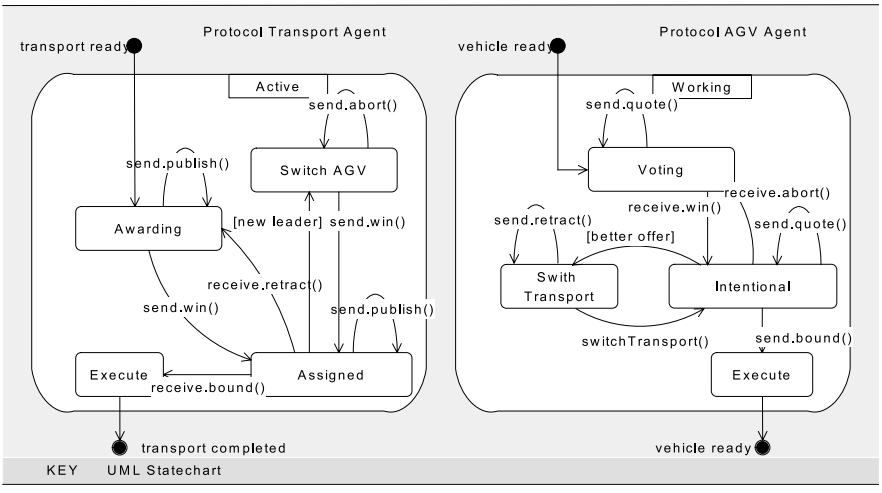

Fig. 4. Behavioral description of AGV agent (left) and transport agent (right) in the transport assignment protocol

a medium that agents can use to exchange information and coordinate their behavior. The local virtual environment handles distribution and locality, using the support offered by the ObjectPlaces middleware [9,14]. Besides a medium for coordination, the virtual environment also serves as a suitable abstraction that shields the AGV agents from low level issues, such as translating the commands for the *Vehicle Control*. The *ObjectPlaces Middleware* supports abstractions for protocol based coordination in a distributed system. The right side of Fig. 3 illustrates the processes, data repositories and interaction needed for moving around. First, the AGV agent projects a hull in the virtual environment. Such a hull demarcates the physical area the AGV will occupy during the movement. In case of conflicts, the local virtual environment executes a mutual exclusion protocol to decide which AGV can move on. The AGV agent perceives the virtual environment and only invokes the move command after it gets the permission to move on. For more information we refer to [4].

Transport Assignment. Now we zoom in on how transports are assigned to AGVs and explain the basics of the protocol. Transport assignment is done by a dynamic Contract Net (CNET) protocol [15], with multiple transport agents negotiating with multiple AGV-agents (many-to-many). A CNET protocol uses a market mechanism to assign tasks to agents. As soon as a new task enters the system, the transport agent announces that a new task is available. An available AGV-agent can bid on the task. The bid is dependent on the distance between the AGV and the task. After a fixed period of time, the transport agent chooses the best AGV-agent and the task is assigned to this agent.

The dynamic CNET protocol extends this protocol by delaying definitive assignment of the task until the AGV effectively picks up the load. Delaying definitive assignment is needed because many things can happen while the AGV moves towards the pick-up location of a transport. New tasks that are better suited for this AGV can show up, e.g. being much closer or more urgent; AGVs can become unavailable, because

of a failure or because they have to go in maintenance; an AGV better suited for the task, e.g. an AGV closer to the pick-up location becomes available [16]. The dynamic CNET protocol allows agents to regularly reconsider the situation in the environment and adapt the assignment if opportunities arise. Figure 4 depicts a description of the protocol behavior. As soon as a transport enters the system the transport agent will go to state `Awarding` and send publish messages. Free AGVs are in the `Voting` state and will answer with a bid. After some time, the transport agent will select a winner, send a win message and go into the `Assigned` state. The winning AGV will go the `Intentional` state and start driving to the pick location. Both the transport and AGV agent can still decide to switch the transport (the `Switch` state). It is only when the `Execute` state is reached, when the AGV picks up the load, that the protocol is ended. For more details we refer to [13, pag. 108-112].

3 Applying the ATAM

The Architectural Tradeoff Analysis Method (ATAM [6,7]) was used to evaluate if the system meets the expected performance, flexibility and openness quality attributes and to identify possible tradeoffs. The ATAM is a workshop that involves all important stakeholders. The workshop has two cornerstones: (1) make explicit and prioritize quality requirements for the system and identify tradeoffs between the quality attributes; (2) identify and make architecture approaches explicit and identify possible alternatives. Evaluating the architecture for large projects early in the development process is needed because the architecture represents an enormous risk in a development project. Making bad architectural decisions may substantially slow down the project and even lead to failure. Additionally, if some problems are identified, it is easier and cheaper to change the architecture in early stages of the development process.

In this section we describe our motivation to apply the ATAM, the context and the approach followed for the ATAM and the main outcomes from the workshop.

3.1 Motivations to Apply the ATAM

The decisions to apply the ATAM was taken in a the stage of the project where the software architecture started to take shape. There was an agreement amongst the stakeholders that a reflection about the software architecture was needed before investing much effort in the implementation. We selected the ATAM to reflect on the architecture because the method is well documented and has already proven its value in other studies [6].

The motivation for stakeholders to suggest a reflection about the architecture were threefold. Firstly, there are several quality attributes attributed to a MAS architecture, but MAS technology is relatively new and has not yet entered mainstream commercial organizations. The EMC^2 project, in which this evaluation took place, was started to test the feasibility of MAS architectures for the AGV transportation system and to evaluate if MAS effectively offers the attributed quality attributes. Reflecting early about the architecture was essential and the ATAM provided a first step of this evaluation. Secondly, the ATAM provides a way to explicitly tradeoff the new quality attributes of flexibility

and openness against other (possibly better known) qualities of the AGV transportation system. In general, quality attributes can never be achieved in isolation, improving one quality attribute affects others (positively or negatively). Although tradeoffs were somewhat considered during the design of the architecture, the ATAM provides an explicit and structured approach with input from all stakeholders to investigate these tradeoffs. Finally, the ATAM provided an opportunity to finalize the architectural documentation and to wrap-up a phase of discussing the quality attributes and essential architectural choices. Because architectural approaches and quality attributes are made explicit, presented and discussed together with the important stakeholders, an agreement is reached between all stakeholders. The bundled results of the ATAM paves the way to start building the software system.

3.2 Context and Method

This was our first experience with the ATAM itself, none of us had previous hand-on experience with the method. Additionally, our industrial partner has limited experience with software architecture in general. Due to a lack of experience with ATAM we decided to operate in two phases. At first, we did an intensive preparation phase with one evaluator and four important stakeholders. Secondly, we organized a single day ATAM workshop with the complete group of stakeholders. The initiative to perform an ATAM came in a stage of the project where the basic structure of architecture became clear and an early prototype with the basic functionality was available.

We decided to evaluate the architecture for one concrete product, namely a tobacco warehouse. In this application, bins with tobacco are stored into a warehouse and 12 AGVs bring the full and empty bins from the warehouse to different tobacco-processing machines. The warehouse measuring 75 by 55 meters and has an average of 112 transports per hour. An 11 Mbits wireless Ethernet is available for communication with the mobile vehicles, but the network is also used by the warehouse management system. The warehouse itself is split into different zones according to the type oftobacco-processing machines. The machines can be put in normal-capacity mode or high-capacity mode. When the machines are in normal-capacity mode the AGVs should be spread evenly over the different zones and they should stay in that particular zone. When the machines in a zone are put in high-capacity mode, the supply of tobacco to these machines gets absolute priority and AGVs can leave a normal capacity-zone to maximize the speed of provisioning in the high-capacity zone. Additionally, there arise lots of opportunities. Examples are better suited AGVs becoming available or new task who are on the way of available AGVs (as explained in the section on the dynamic CNET protocol 2.4). Due to the flexibility in behavior needed with these mode switches, the tobacco warehouse lends itself perfectly for the ATAM workshop.

Specification of Requirements Before the ATAM. The functional requirements were discussed and made explicit using scenario's during several kick-off meetings of the project. Obviously, the exact functional requirements and the scenario's evolved over time as our understanding of the system increased.

On the contrary, the quality attributes were handled in a less structured fashion. It was clear from the start that flexibility and openness were the motivating factors to

Step	Start	Activity
1	9:00	Introduction on ATAM and program
2	10:00	Present business driver
	10:45	Break
3	11:00	Present architecture
4	12:00	Identify architectural approaches
	12:30	Lunch
5	13:45	Generate Utility tree
6	14:45	Generate Utility tree
	15:45	Break
7	16:00	Analysis of architectural approaches
8	17:00	Closing

Fig. 5. Program of ATAM workshop

come up with a new and innovative architecture, but these qualities were only specified at a high level. Additionally, various architectural decisions that involved a tradeoff between quality attributes were made on an ad-hoc basis.

Preparation. According to the template in [6, pag 71], the first steps during the preparation are establishing the partnership and preparing the necessary material for holding the ATAM workshop itself. Next to the usual preparation we performed two main activities. Firstly, as our understanding of software architecture and quality attributes increased, we realized that structuring and concretizing the quality attributes is essential. It was only then that we produced a first version of the concrete quality attribute scenario's and the utility tree (as described in 2.2). Apart from learning and reading about the utility tree, it took at least four full days of discussion with four stakeholders and one evaluator to build up a decent utility tree. This tree was used as input for the discussion about quality attributes on the ATAM workshop itself. The end result can be found in [13, pag 140-143]. Secondly, the architectural documentation of the system was not well structured and some views were only documented partially or were lacking. This was the case for the process view where several documents were missing and the deployment view that was lacking. During the preparation of the ATAM the architectural team worked hard to complete the architectural documentation following the guidelines and examples of Clements et al. [17].

The ATAM Workshop. The ATAM itself was conducted by a team of three evaluators and nine stakeholders. Fig. 5 describes the program of our ATAM workshop. Originally, we planned to start the discussions on analysis of scenarios at 14:45 (based on the schedule of [6, pag 79]), but the discussion on the utility tree brought up important issues that required additional discussion. The main artifacts produced on the workshop are the utility tree, the architectural approaches and an analysis of the architectural approaches with respect to the quality attributes. All presentations and artifacts produced in the context of the ATAM are bundled in a technical report [13].

Afterwards. Next to the typical activities of finishing up documents and updating documentation we performed two main activities. Firstly, we performed a second round of analysis of the architectural approaches with a team of four main stakeholders and one

evaluator to structure the results and to ensure that all risk and sensitivity points were covered. Secondly, we performed a detailed analysis of several risks by consulting experts, running tests and we developed an additional prototype to perform a number of tests. The tests are described in the next section.

3.3 Main Outcomes

Applying the ATAM proved to be a valuable experience since it exposed several tradeoffs and points of improvement. In this section we provide several examples of problems that have been identified and how we have dealt with them.

Tradeoff Flexibility Against Communication Load. One of the main outcomes of applying the ATAM was the identification of the tradeoff between flexibility and the communication load, inherent in using decentralized control. In the AGV transportation system, there are several functional requirements that require more communication in a decentralized architecture (compared with the centralized architecture). Examples are transport assignment, deadlock avoidance, flexible routing and collision avoidance. In this section we focus on the tradeoff between flexible transport assignment (quality attribute scenario 3) and communication load (quality attribute scenario 5) as an example.

Fig. 6 contains an overview of our analysis of architectural decisions, both from the perspective of flexible transport assignment (left table) and the perspective of communication load (right table). We only included the architectural decisions relevant for the tradeoff between flexibility and communication load. As can be seen, the tradeoff is identified in several design decisions. The tradeoffs T1, T2, T3 represent manifestations of flexibility versus required bandwidth in different circumstances. The nonrisk NR2 is about limiting the communication load and the risk R2 is about the risk involved in having to much communication. The sensitivity point S3 is about using .NET remoting with unicast communication or using multicast as underlying implementation framework.

The tradeoff between flexibility and communication was not completely new for the architectural team, they had always realized that the decentralized approach needed more communication. However, on the ATAM workshop the assembled stakeholders (including several experienced developers of previous AGV transportation systems) put this tradeoff forward as a crucial factor for the applicability –and the success– of the decentralized approach. The ATAM provided the perfect opportunity to unravel the tradeoff in a structured way and to uncover the essential architectural decisions. The analysis of architectural approaches (step 7) revealed several architectural decisions that never really had been made. For example, the discussions revealed the choice for unicast in the middleware (AD8) producing some overhead. The influence of unicast, multicast and broadcast on the communication bandwidth is currently being investigated. As a second example, the discussion about AD7 revealed that the choice for a dynamic Contract Net (based on [15,10], described in section 2.4) protocol has a significant impact on the amount of communication in the system. Especially, how fast the protocol needs to reconsider the situation in the environment was an important choice that had not been made. The result of the ATAM regarding the tradeoff was that we gained better insight

Scenario: As long as a transport has not been picked, the system dynamically changes that transports assignment to the most suitable AGV			
Decisions	Sensitivity	Tradeoff Risk	Nonrisk
AD1 Negotiating agents		T1	NR1
AD2 Locality	S1		NR2
AD3 Separation of coordination and decisions making			NR3
AD4 Dynamic Transport Assignment		R1	

Scenario: The amount of communication, with maximal 12 AGVs and a maximal load of 140 transports per hour, does not exceed 60% of the bandwidth of the 11Mbps communication channel.			
Decisions	Sensitivity	Tradeoff Risk	Nonrisk
AD5 Use .Net remoting	S2		NR4
AD6 Agents located on AGVs		T2 R2	
AD7 Dynamic transport assignment		T3	
AD8 Unicast communication in Middleware	S3		

AD1	An agent is associated with each AGV and each transport in the system. To assign transports, multiple AGV agents negotiate with multiple transport agents (many-to-many protocol). Agent continuously reconsider the (possible changing) situation to improve transport assignment, until the load is picked. The negotiating agents approach for transport assignment was noted as non-risk NR1 from the perspective of flexibility. The continuous reconsideration of transport assignments implies a significant cost of communication. Flexibility versus required bandwidth was registered as tradeoff T1.
AD2	For their decision making, agents only take local information (from themselves and other agents in the neighborhood) into account. Using only local information and only exchange information with other local agents limits the amount of communication needed and was registered as non-risk NR2. The most suitable range varies per type and information and may even vary over time e.g. the number of candidate transports, vehicles to avoid collisions. The determination of this range for various functionalities is a sensitivity point, denoted as S1.
AD6	Each AGV machine is controlled by an agent that is physically deployed on the machine. This decentralized approach induces a risk with respect to the required bandwidth for inter-agent communication, recorded as risk R2. Decentralization of control implies a tradeoff between communication cost on the one hand, and flexibility and openness on the other hand, noted as T2.
AD7	The dynamic CNET protocol for transport assignment (as described in 2.4) enables flexible assignment of transports among AGVs. Yet, the continuous reconsideration of the transport assignment implies an additional communication cost. This tradeoff was denoted as T3 (see also T1).
AD8	The middleware uses unicast communication because this is supported by .NET remoting. However, some messages have to be transmitted to several agents, causing overhead. This potential problem was registered as sensitivity point S3.

Fig. 6. Analysis of architectural decisions from perspective of flexibility (left) and communication load (right)

in the main difficulties, agreed to further investigate the communication load and put forward a concrete testing plan.

Post ATAM Tests on Communication Load. The test on the communication load were performed in two ways. Firstly, we investigated to vary the period between reconsiderations for the transport assignment protocol. Obviously, this period strongly affects

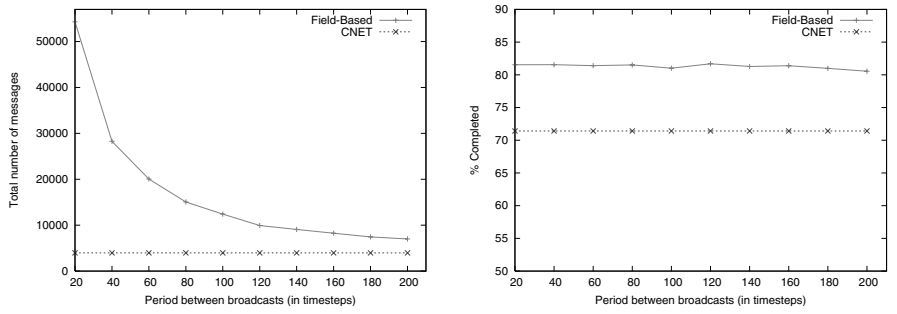

Fig. 7. Test results of varying the reconsideration period (Field-based is with reconsideration, CNET is without reconsideration)

the amount of messages being sent. Making the period too short produces a huge number of useless messages, but each agent in the system has up-to-date information. On the other hand, if the period is too long, AGVs may have outdated information and probably miss some opportunities. The tests are performed in an in-house developed simulator and are published in [18].

Figure 7 summarizes the most important results. The left figure shows the expected decrease in number of messages sent if the period between two broadcasts increases. BR20 corresponds to reconsidering the situation each 20 time steps. With a reconsideration rate of each 200 time steps (BP200, i.e. every 10 s) the number of messages sent with the dynamic protocol is 1.7 times higher then a protocol without reconsideration. The right-hand side of fig. 7 depicts the percentage of transports handled as a function of the reconsideration rate. It can be seen that the percentage of completed tasks fluctuates around 81,5% and slowly decreases, the difference between BR20 and BR200 is only 1% but still significantly better then the protocol without reconsideration. The results show that the communication load of a dynamic protocol can drastically be reduced by lengthening the period between reconsiderations without losing the advantages of a flexible protocol.

Secondly, we performed a stress test to measure the bandwidth usage of a the prototype implementation of the AGV transportation system on a real factory layout using a 11Mbps IEEE 802.11 wireless network. Fig. 6 shows the bandwidth usage relative to the bandwidth, divided into four tests. The first test has three AGVs, of which two were artificially put in deadlock (a situation which is avoided under normal operation, this situation is created by manually positioning the AGVs), because then the collision avoidance protocol is continually restarted, and never succeeds. The second test has 3 AGVs driving around freely. The third test has five AGVs driving around freely. The fourth tests has five AGVs, all artificially in deadlock. During the time in between test runs, AGVs were repositioned manually (thus the system is suspended in the time in between the tests). All the tests are worst case tests (with deadlock) that we artificially created but that are prevented during normal operation.

The general conclusions after the tests was that the decentralized approach seemed feasible for 12 AGV with a 11Mbps IEEE 802.11 wireless network. The tests showed that the communication load for five AGVs under normal operation (e.g. the time between

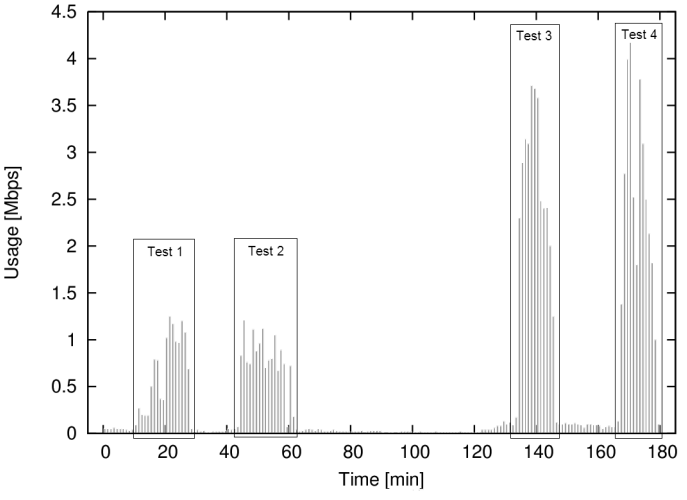

Fig. 8. Results of stress tests on bandwidth usage (in %)

154 and 168 min) is very low and obviously within the bounds of the available bandwidth. Looking to the stress tests, the communication load is a factor higher but still within the bounds. But there is a significant increase in communication load between the tests with three and five AGVs.

Architecture Constraints the Implementation. A second interesting issue that came up during the ATAM was that the implementation was not always conform with the architecture. Non conformance was identified in two phases of the evaluation. Firstly, while building up the architectural documentation in preparation of the ATAM and while discussing this with the developers of the prototype system. Secondly, during the discussion after the architect presented the architectural approaches on the ATAM workshop itself.

For example, consider Fig. 3 containing a view describing the layers and the usage relations between layers of the internal AGV control system (left) and a view describing the processes (mapped on the layers) and the interaction between these processes. During the discussion the issue came up that the developers accessed a repository in AGV agent layer from a process that was in the Local Virtual Environment layer, instead of using the appropriate interface. This breaks the encapsulation of layers and the implementation had to be changed. The reverse also happened. During an early prototype, the developers proposed a simplification of the architecture, making some processes and repositories obsolete. In this case the architectural documentation was adapted to reflect this change. From this and other discussions we learned that the *architecture constraints the implementation* and that it is very important to closely work together with the development team.

Improving the Architecture. Finally, the ATAM exposed several opportunities to improve the architecture. Two examples are provided.

A first example is an issue raised by a developer of the original centralized system. The AGV transportation system is built on top of a library that supports logging, persistence, security, etc. This library is also used in other projects and evolved since the start of the project (which still used an old version of the library). The architecture had to be adapted to better align with the library, so that the library can evolve without changing the AGV transportation system.

A second example is the use of a Free-flow tree [19,20,21] as decisions infrastructure inside the AGV agent. Free-flow trees are a proven technique to control mobile robots in highly dynamic circumstances where the robot must perform several activities in parallel. But the activities in the AGV transportation system are rather sequential (look for a task → drive to the task → perform the task → ...). Additionally, a free-flow tree is not so easy to extend or change which is essential in the AGV transportation system with often changing requirements.

4 Reflection on the ATAM Workshop

Although the ATAM itself was not worked out in full detail, the ATAM workshop was a valuable experience. It was the first time that there was such an in-depth discussion with a complete group of stakeholders. Everybody agreed that their insight had improved on: (1) the importance of software architecture in software engineering; (2) the importance of business drivers for architectural design; (3) the importance of making explicit and prioritize quality attributes with the stakeholders; (4) the strengths and weaknesses of the architecture and architectural approaches. Especially, we improved our understanding of the quality attributes and the other stakeholders improved their understanding of the fundamental architecture of the system and the important design decisions. Finally, we gained better insight into the tradeoffs associated with the flexibility and openness quality attributes, which is the most valuable outcome of this ATAM-workshop.

Some critical notes are:

- A thorough and complete evaluation with the ATAM of a realistic industrial application is not manageable in a single day. It would indeed be better, as suggested in the ATAM documentation, to organize a three day workshop.
- The utility tree proved to be the most important instrument in the ATAM. Several in-depth discussions have contributed to a better understanding of the required qualities and the importance of tradeoff between different quality attributes. But coming up with a utility tree proved to be difficult, time consuming, and at times tedious. A lack of experience and clear guidelines of how to build up such a tree hindered and slowed down the discussion. Good preparation of the tree and a good chairman to manage the discussion are essential. A suggestion, brought to us on the latest SATURN workshop[1], was to prevent brainstorming to come up with appropriate scenario's. Instead, every stakeholder should prepare two or three scenarios in advance. These scenarios serve as a good starting point for building up the utility tree, and speed up and improve the discussions.
- During discussions on the utility tree, there was a tendency by the architects to introduce quality attributes to motivate certain architectural decisions. This is of

[1] http://www.sei.cmu.edu/architecture/saturn/

course the reverse of what should happen: quality attributes influence the architecture and not the other way around. Similar problems were identified for the ALMA [22] evaluation method. The ATAM partially supports resolving such issues by obligating the stakeholders to prioritize the scenarios, thus filtering out less important scenarios. Still, the evaluation team must be watchful for artificially introduced scenarios and encourage everyone to think what the client really needs.

- During the discussions there was a tension between the AGV transport system architecture developed for several automation projects (the product-line architecture) and the fact we evaluated it within a single project (the product architecture). We decided to evaluate the architecture for a concrete product for three reasons. Firstly, Egemin installed dozens of systems, each of them tailored to the specific needs of the client. The installed products both differ in functionality, required quality attributes and can largely differ in size and complexity. Early in the preparation phase it became clear that the differences in requirements largely hindered the discussions. Often, stakeholders came up with contradicting requirements of different clients. Especially the variation in relative importance of qualities with respect to each other posed a problem. Secondly, the evolution of the market has put forward new requirements for AGV Transportation systems. Especially in highly dynamic systems, where the situation changes frequently, problems are experienced. This are the target systems for our architecture and we only wanted to focus on these type of requirements. Finally, the fact that the AGV Transportation system is a product-line system introduces specific qualities by itself. For example, the product-line architecture must be tailorable to specific needs of clients in different products. But the ATAM is made to evaluate a single architecture and not a product-line architecture.

  To improve the discussions, the main stakeholders selected a single product (a tobacco warehouse). The tobacco warehouse was the best fit to the criteria that the product needed to be a highly dynamic system (the main reason to introduce the decentralized architecture) and represents the typical size and complexity of systems installed by Egemin. The choice for the single product and the reason to work with this single product where clearly communicated to all stakeholders of the ATAM. After this choice the discussions improved and became more focussed. Still, the tensions between product-line architecture and the concrete product architecture sometimes hindered the discussions. At the time of finishing this paper an extension of the ATAM to handle such a product line architecture has been published [23].
- A final remark is that there is a lack of good tool support to document software architectures. Currently, drawing architectural diagrams and building up the architectural documentation incurs much overhead. Especially changing the documentation and keeping everything up-to-date (e.g. cross references and relations between different parts of the documentation) turned out to be hard and time consuming. Good tool support would be helpful.

5 Related Work

Architecture in MAS. Initially, architecture in MAS was associated with the internal agent architecture. There exist large number of internal architectures, some examples

are Belief Desire Intention (BDI), the subsumption architecture [24] and the free-flow tree [20]. Later on, the focus of architectural development shifted to social structures in MAS. Several methodologies specifically tailored to develop software offer support for such social structures, e.g. the organizational metaphor of Gaia methodology [25] and the society and environment model of the SODA methodology [26]. Finally, there are some approaches that take an architectural-centric approach to software development, using the classical notion of software architecture. PROSA [27] offers a reference architecture for coordination in manufacturing control. [28] proposes an aspect-oriented approach to develop MAS architectures, stressing the importance of separation of concerns. Our work differs from previous work on software architecture because it is quality attribute driven. The existing architectural approaches mainly focus on the system functionality but do not reason explicitly about qualities.

Shehory [29] considers MAS from the perspective of architectural styles and reasons about the qualities that are typically attributed to the MAS styles. The author explicitly recognizes flexibility and openness as important qualities for MAS architectural styles. Our work in this paper extends this with architectural evaluation to asses wether the architecture effectively fulfills the expected qualities.

Decentralization. Decentralized control of automated warehouse transportation systems is an active area of research [4]. [30] also discusses a behavior based architecture for decentralized control of AGV vehicles and [31] discusses a decentralized cognitive planning approach for collision-free movement of vehicles. Ong [32] gives an extensive overview of decentralized agent based manufacturing control and compares the pros and cons of centralized versus decentralized control. According to Ong, the advantages of decentralized control are that processing power is used more economically and that it is more reliable. The disadvantages are that performance may be affected by the speed of communication links, that there is a tradeoff between reactivity of the system to disturbances and performance and that suboptimal decisions may be made due to the lack of global information.

Experiences with Architectural Evaluation for MAS. Woods et al. [33] specifically describes experiences with an ATAM for collaborative agent-based systems. The quality attributes identified in the report are performance predictability, security against data corruption and spoofing, adaptability to changes in the environment (protocols, message formats, new types of agents) and availability and fault tolerance and are specifically aimed at applications of MAS technology for the internet or network infrastructures.

Architectural Evaluation. A good overview of other architectural evaluation methods can be found in [34,35]. Several of these architectural methods are aimed at a specific quality attributes (e.g. SAAM, ALMA, ...) and do not support evaluation with new type of quality attributes. Other architectural evaluation methods are primarily performed by the design team only, e.g. ARID, SBAR, ... For us, the ATAM workshop provided a good opportunity because of the explicit treatment of tradeoffs between different qualities and architectural approaches and the involvement of all stakeholders. Additionally, it aligned perfectly with our architectural documentation who was based on [36,17].

6 Conclusions

In this paper, we reported our experiences with applying the ATAM to a MAS architecture for an AGV transportation system. The main quality requirements in this application are flexibility, openness and performance. The report contains both the main outcomes of the architecture evaluation and a critical reflection on our experiences with the ATAM itself. One important insight we derived from evaluating the architecture is the tradeoff between flexibility and communication load in the decentralized architecture.

The ATAM workshop was a very valuable experience. We summarize three important lessons learned. First, we learned that assessing the qualities of the system can be done directly on the software architecture, early in the development process without the need for a complete implementation. A second important lesson we learned is that structuring and concretizing the quality attributes is essential when building complex large scale systems. Third, we learned that there is a strong connection between MAS and software architecture. Applying the ATAM deepened our understanding of this relationship.

Egemin plans to apply the first results of the project in one of the systems that is currently in the startup phase. In particular, Egemin experiences the need for improved flexibility in this application and plans to apply the order assignment approach applied in the MAS architecture. The highly dynamic nature of this application urges for improved flexibility and that is one of the qualities where the MAS architecture can make the difference.

Acknowledgement

The EMC^2 project and Nelis Boucké are supported by the Institute for the Promotion of Innovation through Science and Technology in Flanders (IWT-Vlaanderen).

References

1. Distrinet: Distrinet research group website. (`www.cs.kuleuven.ac.be/cwis/research/distrinet/`)
2. Egemin, DistriNet: Emc^2: Egemin modular controls concept. (IWT-funded project with Distrinet and Egemin. http://emc2.egemin.com)
3. Egemin: Egemin website. (`www.egemin.com`)
4. Weyns, D., Schelfthout, K., Holvoet, T., Lefever, T.: Decentralized control of E'GV transportation systems. In: International Conference on Autonomous Agents and Multi-Agent Systems, Industry Track. (2005) 25–29
5. Weyns, D., Holvoet, T.: A reference architecture for situated multiagent systems. In: 3rd International Workshop on Environments for Multiagent Systems, E4MAS06. (2006)
6. Clements, P., Kazman, R., Klein, M.: Evaluating Software Architectures: Methods and Case Studies. Addison Wesley Publishing Comp. (2002)
7. Kazman, R., Klein, M., Clements, P.: Atam: Method for architecture evaluation. Technical Report CMU/SEI-2000-TR-004, SEI, Carnegie Mellon University (2000)
8. WordNet 2.1., Princeton University Cognitive Science Library. (http://wordnet.princeton.edu/)

9. Schelfthout, K., Weyns, D., Holvoet, T.: Middleware for protocol-based coordination in dynamic networks. In: MPAC '05: Proceedings of the 3rd international workshop on Middleware for pervasive and ad-hoc computing, New York, NY, USA, ACM Press (2005) 1–8
10. Wooldridge, M.: An introduction to Multiagent Systems. John Wiley & Sons, LTD (2002)
11. Ferber, J.: Multi-agent Systems, An Introduction to Distributed AI. Addison-Wesley (1999)
12. Weyns, D., Parunak, H.V.D., Michel, F., Holvoet, T., Ferber, J.: Environments for multiagent systems: State-of-the-art and research challenges. In: Revised papers of the E4MAS workshop at AAMAS'04. Volume LNCS 3374. (2005)
13. Boucké, N., Holvoet, T., Lefever, T., Sempels, R., Schelfthout, K., Weyns, D., Wielemans, J.: Applying the Architecture Tradeoff Analysis Method (ATAM) to an industrial multi-agent system application. Technical Report CW431, Departement of Computer Sience, KULeuven (2005)
14. Schelfthout, K., Holvoet, T.: Coordination middleware for decentralized applications in dynamic networks. In: DSM '05: Proceedings of the 2nd international doctoral symposium on Middleware, New York, NY, USA, ACM Press (2005) 1–5
15. Smith, R.G.: The contract net protocol: high-level communication and control in a distributed problem solver. Distributed Artificial Intelligence (1988) 357–366
16. Boucké, N., Weyns, D., Holvoet, T., Mertens, K.: Decentralized allocation of tasks with delayed commencement. In Chiara, G., Ciorgini, P., van der Hoek, W., eds.: EUMAS04 Proceedings. (2004) 57–68
17. Clements, P., Bachman, F., Bass, L., Garlan, D., Ivers, J., Little, R., Nord, R., Stafford, J.: Documenting Software Architectures, Views and Beyond. Addison Wesley (2003)
18. Weyns, D., Boucké, N., Holvoet, T.: Gradient field based task assignment in an agv transportation system. In: International Conference on Autonomous Agents and Multiagent Systems (AAMAS). (2006)
19. Rosenblatt, J.K., Payton, D.W.: A fine-grained alternative to the subsumption architecture for mobile robot control. In: In Proceedings of the IEEE International Conference on Neural Networks. Volume 2. (1989) 317–324
20. Tyrrell, T.: Computational Mechanisms for Action Selection. PhD thesis, University of Edinburgh. Centre for Cognitive Science (1993)
21. Weyns, D., Steegmans, E., Holvoet, T.: Protocol based communication for situated multiagent systems. In: In Proceeding of the Third International Joint Conference on Autonomous Agents and Multi-Agent Systems, AAMAS'04, ACM Press, New York (2004) 118–126
22. Lassing, N., Bengtsson, P., van Vliet, H., Bosch, J.: Experiences with ALMA: Architecture-level modifiability analysis. Journal of Systems and Software **61**(1) (2002) 47–57
23. Olumofin, F.G., Misic, V.B.: Extending the ATAM architecture evaluation to product line architectures. In: IEEE/IFIP Working Conference on Software Architecture, WICSA. (2005)
24. Brooks, R.: Intelligence without representation. Artificial Intelligence **47** (1991) 139–159
25. Zambonelli, F., Jennings, N., Wooldridge, M.: Developing multiagent systems: the gaia methodology. ACM Transactions on Software Engineering and Methodology **12**(3) (2003)
26. Omicini, A.: Soda: Societies and infrastructures in the analysis and design of agent-based systems. Lecture Notes in Computer Science **1957** (2001)
27. Brussel, H.V., Wyns, J., Valckenaers, P., Bongaerts, L., Peeters, P.: Reference architecture for holonic manufacturing systems: Prosa. Computers in Industry **37** (1998)
28. Garcia, A., Kulesza, U., Lucena, C.: Aspectizing multi-agent systems: From architecture to implementation. Software Engineering for Multi-Agent Systems III **LNCS 3390** (2004) 121–143
29. Shehory, O.: Architectural properties of multiagent systems. Technical Report CMU-RI-TR-98-28, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA (1998)
30. Berman, S., Edan, Y., Jamshidi, M.: Decentralized autonomous agvs in material handling. Transactions on Robotics and Automation **19**(4) (2003)

31. Pallottino, L., Scordio, V.G., Frazzoli, E., Bicchi, A.: Decentralized cooperative con ict resolution for multiple nonholonomic vehicles. In: AIAA Conference on Guidance, Navigation and Control. (2005)
32. Ong, L.: An investigation of an agent-based scheduling in decentralised manufacturing control. PhD thesis, University of Cambridge (2003)
33. Woods, S.G., Barbacci, M.: Architectural evaluation of collaborative agent-based systems. Technical Report CMU/SEI-99-TR-025, CMU/SEI (1999)
34. Dobrica, L., Niemela, E.: A survey on software architecture analysis methods. IEEE Transactions on Software Engineering **28**(7) (2002)
35. Babar, M.A., Zhu, L., Jeffery, R.: A framework for classifying and comparing software architecture evaluation methods. In: Proceedings Australian Software Engineering Conference (ASWEC). (2004)
36. Bass, L., Clements, P., Kazman, R.: Software Architectures in Practice (Second Edition). Addison-Wesley (2003)