Architecture-Centric Development of an AGV Transportation System

Danny Weyns, Kurt Schelfthout, and Tom Holvoet

DistriNet, Department of Computer Science K.U.Leuven,
Celestijnenlaan 200 A, B-3001 Leuven, Belgium
{danny.weyns, kurt.schelfthout, tom.holvoet}@cs.kuleuven.be

Abstract. Architectural design plays a key role in software engineering. The software architecture is the backbone of the designed solution, it has the functional requirements of the system and satisfies the quality requirements. In our research, we put forward situated multiagent systems (situated MAS) as an approach to build distributed applications with demanding quality requirements such as flexibility and openness. In this paper we illustrate how we apply situated MAS to an Automatic Guided Vehicle (AGV) transportation system. We discuss the high-level structure of the software architecture and explain how the architecture aims to meet important quality requirements.

1 Introduction

Software architecture is generally acknowledged as a crucial part of the design of a software system [1]. The software architecture has the functional requirements of the system and aims to satisfy the quality requirements. A common practice to document a software architecture is by using a set of related *views* [2]. A view is a representation of a set of system elements and the relationships associated with them. A module view enumerates principal implementation units and relationships among these units such as "is-part-of" or "uses". A process view focuses on dynamic aspects of the system such as synchronization between process elements. Other views can be part of the documentation of an architecture such as a deployment view that describes the allocation of system elements to available processors.

In the last three years, we have studied the engineering of distributed applications with demanding quality requirements such as flexibility and openness. Example domains we focus on are network management and decentralized control of logistic machines in a warehouse. In our research, we put forward situated MASs as an approach to build such distributed applications. A situated MAS consists of a distributed environment populated with a set of agents that cooperate to solve a complex problem in a decentralized way. Intelligence in a situated MAS originates from the interactions between the agents, rather than from their individual capabilities. Situated agents exploit the environment to coordinate their behavior, e.g. via digital pheromones or gradient fields [3]. We have developed a reference architecture for situated MASs that offers a blueprint for developing the intended applications. This reference generalizes and extracts common functions and structures from various experimental applications we have studied. For a detailed discussion of the reference architecture we refer to [4,5,6].

M. Pěchouček, P. Petta, and L.Z. Varga (Eds.): CEEMAS 2005, LNAI 3690, pp. 640–644, 2005.

In this paper, we illustrate the architectural design of an AGV transportation system that is based on the reference architecture for situated MASs. The AGV transportation system is investigated in a R&D project in close cooperation with Egemin, a manufacturer of automated warehouse systems (http://www.egemin.com/). An AGV transportation system uses unmanned vehicles (AGVs) to handle *transports*, i.e. to move goods through a warehouse. Transports are generated by a *client system*, typically a business management program. An AGV uses a battery as energy source. AGVs can move through a warehouse guided by a laser navigation system, or by magnets or cables that are fixed in the floor. The low-level control of the AGVs such as staying on track on a segment, turning, picking a load or dropping it, determining the current position, etc., is handled by the AGV control software called E'nsor ® (Egemin Navigation System On Robot).

Besides traditional qualities such as performance and robustness, the market for AGV transportation systems requests for more flexibility. AGVs should be able to exploit opportunities, e.g., when an AGV is assigned a transport and moves toward the load, it should be possible for this AGV to switch tasks on its way if a more interesting transport pops up. AGVs should also be able to anticipate possible difficulties, e.g., when the battery level of an AGV decreases, the AGV should anticipate this and prefer a zone near a charge station. Customers also expect that the system is able to deal with AGVs leaving the system, or new AGVs entering the system. One example is maintenance. Currently, maintenance of AGVs is based on fixed worst-case rules. This leaves room for improvement by allowing AGVs to decide themselves when to leave the system for service.

In the next section we discuss the main high-level views of the software architecture and we illustrate how the quality requirements are realized. Finally, Sect. 3 concludes the paper.

2 Architectural Design of an AGV Transportation System

Contrary to the traditional approach applied by Egemin, where vehicles are controlled by one central server, in this project, we explore the feasibility of applying the paradigm of situated MASs to decentralize the control of the AGVs. In [7], Ong compares decentralized with centralized control. According to Ong, decentralized control: (1) is more economical w.r.t. required processing power, and (2) is more reliable. Limitations of decentralization are: (1) performance of the system may be affected by the communication links between nodes, (2) there is a trade-off between its performance and the reactivity of the system to disturbances, and (3) myopic decision making may occur due to the lack of global information.

Besides the advantages of decentralization listed by Ong, we believe that in principle, a MAS-based AGV transportation system also becomes more flexible. Since each AGV acts locally, it can better exploit opportunities and adapt its behavior under changing circumstances. On the other hand, bandwidth must be considered carefully to ensure that the communication network does not become a bottleneck. The challenge in the project is to support the current functionality, while aiming to improve flexibility and openness.

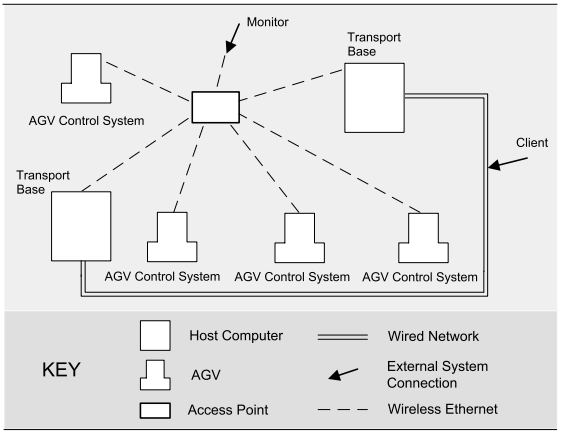

Fig. 1. Deployment view of the AGV transportation system

Deployment View of the System. The decentralized architecture consists of two subsystems, *transport bases* and *AGV control systems*. Transport bases receive transport requests from the client system, and are responsible to assign the transports to AGVs. The AGV control software is responsible to ensure that the AGV completes the assigned transport. Fig. 1 depicts the deployment view of the software architecture. Transport bases are deployed on stationary hosts. The AGV control systems are deployed on the mobile AGV machines. The communication infrastructure provides a wired network that connects the client system with the transport bases and a wireless network that enables communication between AGVs and transport bases.

Module Decomposition View of the Subsystems. Fig. 2 depicts the module decomposition view of the AGV control system and the transport base. For each requested transport, the *transport base manager* creates a new *transport agent* at the transport base. A transport agent is responsible for assigning its transport to an *AGV agent*. the client system. AGV agents, that are located in the AGVs, are responsible for executing the assigned transports.

Since the physical environment of the AGVs restricts how agents can use their environment, we introduced a virtual environment for agents to live in. This virtual environment offers a medium that agents can use to exchange information and coordinate their behavior. For example, to avoid collisions, AGV agents coordinate with other agents through the virtual environment. AGV agents mark the path they are going to drive in their environment using *hulls*. The hull of an AGV is the physical area the AGV occupies. A series of hulls then describes the physical area an AGV occupies along a certain path. If the area is not marked by other hulls, the AGV can move along and actually drive over the reserved path. If the AGV's hull intersects with others, only the AGV with the highest priority is allowed to move on. Afterwards, the AGV removes the markings in the virtual environment.

Since the only physical infrastructure available to the AGVs is a wireless network to communicate, the virtual environment is necessarily distributed over the AGVs. In

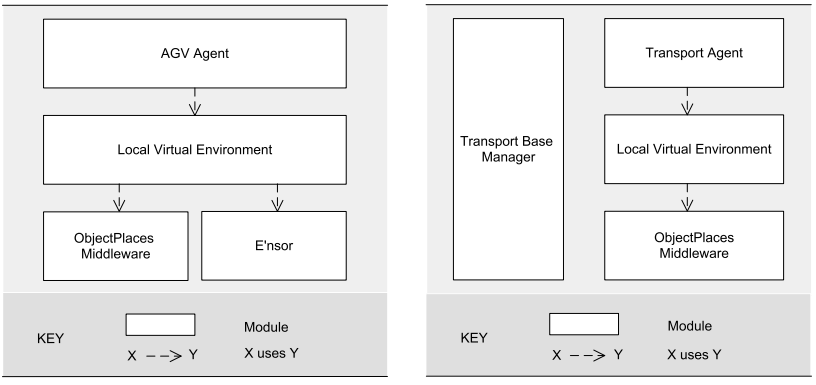

Fig. 2. Module view of the AGV control system on the left and the transport base on the right

effect, each AGV and each transport base maintains a *local virtual environment*, which is a local manifestation of the virtual environment. Synchronization of the state of the local virtual environment with neighboring machines is supported by the *ObjectPlaces middleware* [8].

Besides a medium for coordination, the virtual environment also serves as a suitable abstraction that shields the AGV agents form low-level issues, such as the physical control of the AGV. Therefore, we fully reused the E'nsor software.

Quality requirements. We have applied several architectural approaches to realize flexibility in the system. One example is transport assignment that is based on a flexible version of the Contract Net protocol. This protocol postpones final transport assignment until the load is picked. While the AGV is driving towards the load, the AGV agent and the transport agent are able to switch transport and AGV respectively. Openness in the system is basically realized by the virtual environment supported by the ObjectPlaces middleware. When an AGV leaves the system, or a new AGV enters, the ObjectPlaces middleware on neighboring machines will notice this and the local virtual environments will be updated accordingly.

3 Conclusion

In this paper, we illustrated how we have applied situated MASs as an approach to design an automated AGV transportation system. We discussed three high-level architectural views and illustrated how the architecture supports flexibility and openness. From the initial project phase we learned that the the reference architecture for situated MAS that underlies the software architecture of the AGV transportation system turned out to be an excellent guide for architectural design. On the other hand, the complexity of the application forced us to further decompose several modules of the reference architecture.

References

1. Bass, L., Clements, P., Kazman, R.: Software Architecture in Practice. Addison-Wesley (2003)
2. Clements, P., Bachman, F., Bass, L., Garlan, D., Ivers, J., Little, R., Nord, R., Stafford, J.: Documenting Software Architectures. Addison-Wesley (2003)
3. Weyns, D., Parunak, V., Michel, F., Holvoet, T., Ferber, J.: Environments for multiagent systems, state-of-the-art and research challenges. Lecture Notes in Computer Science, Vol. 3374 (2005)
4. Weyns, D., Holvoet, T.: Formal model for situated multi-agent systems. Fundamenta Informaticae, Vol. 63(1-2) (2004)
5. Weyns, D., Steegmans, E., Holvoet, T.: Protocol based communication for situated multiagent systems. 3th Joint Conference on Autonomous Agents and Multi-Agent Systems, New York (2004)
6. Weyns, D., Steegmans, E., Holvoet, T.: Towards active perception in situated multi-agent systems. Journal on Applied Artificial Intelligence, 18(8-9) (2004)
7. Ong, L.: An investigation of an agent-based scheduling in decentralised manufacturing control. Ph.D Disseration, University of Cambridge (2003)
8. Schelfthout, K., Holvoet, T., Berbers, Y.: Views: Customizable abstractions for context-aware applications in MANETs. Software Engineering for Large-Scale Multi-Agent Systems, St. Louis (2005)