Multi-Agent-based Controller Architecture for AGV Systems

Lluís Ribas-Xirgo, Ismael F. Chaile
Microelectronics and Electronic Systems Department
School of Engineering, Univ. Autònoma de Barcelona
Campus UAB, 08193 Bellaterra, Spain
{Lluis.Ribas, IsmaelFabricio.Chaile}@uab.cat

Abstract

Automated guided vehicles (AGVs) are increasingly being used in industrial facilities to improve efficiency of internal transportation processes. In this work, we have analyzed what are the requirements for AGV systems and made a correspondence to our controller architecture, which relies on multi-agent systems.

We show that, by using our agent-based model, it is possible to rapidly deploy AGV system controllers and attain higher degrees of robustness and flexibility than by using conventional approaches.

1. Introduction

Supervisory control and data acquisition (SCADA) software must be tailored for the plants where they are used. The cost of this customization grows with the complexity of the plants, and adding more devices to control like the AGVs makes them yet more complex.

Complexity must be tackled by abstraction and division. Note that these systems can be split into two aspects: that of the system application (transformation and storage of materials) and that of the material transportation [1].

A similar approach is done in this work, which focuses on the transportation aspect of applications from the industrial (e.g. manufacturing and warehousing) and public infrastructure (e.g. traffic and logistics) domains.

Even with this division, the complexity of the transportation SCADA is high and so it is the cost of deploying the corresponding systems. Typically, it is further minimized by implementing very constrained policies adapted to a particular plant. As a consequence, any change to the plant, which includes layout of the traffic network and addition or suppression of AGVs, implies changing the controller.

In this paper, we present a controller's model for AGV systems that uses an agent-based model (ABM) to plan the operations of the AGVs for the next control cycle. Because of using ABM, AGV addition or removal does not require any change to the controller. Besides, as agents are autonomous entities, it automatically adapts to changes to the plant.

The paper is organized as follows. The next section is devoted to describe the common requirements of the supervising and controlling software for AGVs. Then, the agent-based controller's model for internal transportation systems is proposed. In section 4, a suitable architecture is matched to the former model so that it conforms to the previously cited requirements. The result design framework and process is detailed in section 5. Main results and impact of the work are discussed at the concluding section.

2. AGV system SCADA requirements

Providers of AGV-based solutions try to offer robust and flexible systems to their clients, but they have also to take into account backward compatibility, safety and security. Apart from the fact that these systems must comply with the former requirements, they must also include the features (some of them referred to in [2]) that shall be explained in the rest of the section.

2.1. Management of work orders

The order management software (OMS) of a plant generates specific work orders for the transportation system. These work orders include collection and delivering spot identifications as well as data about the material to be transported and priority of the load.

Note that there must be some manual override to create work orders in the presence of any error whichever its cause, and that work orders have priorities that are used to solve traffic conflicts.

Work orders are dynamically allocated to AGVs, either automatically or manually. These task allocations can change even after AGVs have started their jobs, if other AGVs can finish them at lower costs. This can happen when AGVs can be turned off and on at any moment, either because they are used in manual mode for a while, because they undergo maintenance operations or for some other reason.

2.2. Human-machine interface

The HMI must offer a real-time view of the status of the transportation system, must be able to communicate with it in accordance to the user access rights, and must record data about its working so to be able to present them in an intelligible form to the corresponding users and to report statistical information about system performance and lists of specific operation events.

2.3. Power management

AGVs are battery operated hence their work capacity strongly depends on maximizing available energy at any moment and battery life.

Work orders can only be assigned to AGVs with enough energy to fulfill them but, amongst all candidates for a given assignment, there is an ordering that depends on time to job completion and on efficient energy usage [3]. Additionally, optimization of energy consumption and battery lifespan is a complex problem itself. In fact, they may compromise each other.

2.4. Infrastructure

Supervisory control must be run on a fault-tolerant system and communications need be safe, secure and, again, robust.

Tolerance to failures is achieved through redundancy. In the case of communications, it is implemented not only by duplicating network elements but by using appropriate protocols in data transmission.

2.5. Communication between system manager and vehicles

The control loop begins with data acquisition (from AGVs to system manager) and ends with actuators' actions (from system manager to AGVs), and the control cycle should be short enough to avoid any wait on the vehicles side.

Additionally, the system manager has to communicate with other elements in the plant, which are not part of the transportation system, but that are related to the application. For instance, it must control peripheral devices' PLCs involved in load/unload operations.

Data traffic over the communication network must be minimized because minimal traffic means lower costs, lesser dependence on spurious communication interrupts and greater capability of introducing redundancy and security than with usual data transference rates in conventional control schemas.

Minimizing data traffic is somehow contradictory with having a solid real-time control. However, it is possible to exploit the characteristics of the transportation systems to do so without compromising the control quality. For instance, accurate localization of AGVs is not required (can be estimated) and data that have not changed (that is essentially the same, e.g. the difference does not go over a given threshold) do not need to be sent.

3. Agent-based SCADA for AGVs

Agent-based models for transport logistics are used mainly to support decision taking but not to automate processes [4]. However, current computing systems are able to run ABMs fast enough to close control loops in real-time provided that they operate at a high enough abstraction level [5, 6].

Our approach uses a single ABM tool to simplify the development framework and minimize the development costs.

3.1. ABM of transportation systems

The proposed ABM has a relatively simple architecture (see Fig. 1), that organizes agents into two classes: the one for the external elements (application-related agents, A_j) to the transportation system and the one for the vehicles or *taxis* (agents T_i with links to physical, R_i, and virtual, V_i, lower-level layers).

The model is run under inputs that come from external agents and physical elements and generates outputs for the latter ones. This control loop might be too slow for many applications unless physical elements have embedded some controllers and relation with the ABM is done at a higher level of abstraction.

As for the latter aspect, note that high-level parts of taxis, $\{T_i\}$, send high-level orders (e.g. "go to the next landmark", "take the next turning to the right" or "dock at the machine pier") to their low-level parts ($\{R_i\}$), which at their turn, reply with high-level answers.

However, even with this solution, ABM has to be executed fast enough to interact at real time with the physical elements. This requires agents to be efficient in taking decisions and to have simple communication protocols that enable negotiations to occur within a few messages.

3.2. ABM-based controllers

The idea is that transport orders from the applications are handled by taxis in an autonomous manner, with minimal information from other agents, including those who may act as planners and routers.

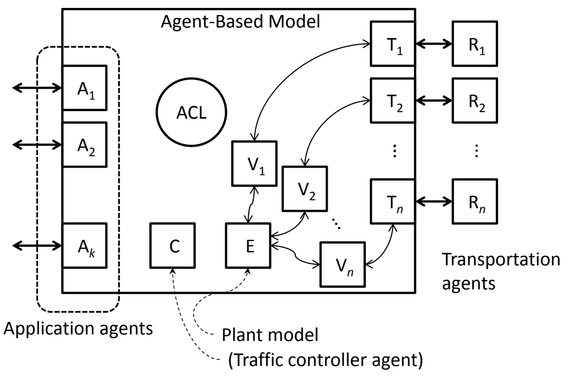

Figure 1. ABM of transportation systems.

Fig. 2 illustrates how this control scheme is organized. Topology of the plant and the number of taxis are among the variables that configure the model that is used for controlling the real plant.

The higher level modules of the taxis ($\{T_i\}$) get orders from agents that represent other modules of the application ($\{A_j\}$) and try to fulfill them.

To do so, taxis must negotiate with application agents $\{A_j\}$ and other workmates which jobs they take and, when in transit, how can they be done in the more efficient way. In taking the decisions, taxis have knowledge of their own state and the state of their lower-level counterparts ($\{V_i\}$). Results of deliberations are transformed into requests to the $\{V_i\}$ and also to the real robots $\{R_i\}$. The last set of requests is, in fact, the output of the ABM controller. And the inputs include the replies to these requests from robots, hence closing the loop between the controller and the controlled system.

4. Implementing agent-based SCADA

This section is devoted to explain how to construct such a system from the model with the requirements that have been listed in section 2.

4.1. Platform

A typical binding between architecture components and model elements would have the ABM running on a computer, the low-level parts replicated on real AGVs and a communication infrastructure.

This solution is more compatible with a quick development process because only low-level parts have to be ported to real AGVs. Besides, it makes inter-agent communication faster, as it runs in a single computer.

In any case, central servers must have backup machines and communications' infrastructure must support encryption and must have redundancy.

4.2. Software architecture

The software follows the client-server architecture. On the client side, the users can interact with the system in accordance with their role (supervisors, operators, observers, *et cetera*). On the server side, the software gets inputs from the clients, provides data about its status (an "image" of the system), and implements the functional process (i.e. ABM) and database logic.

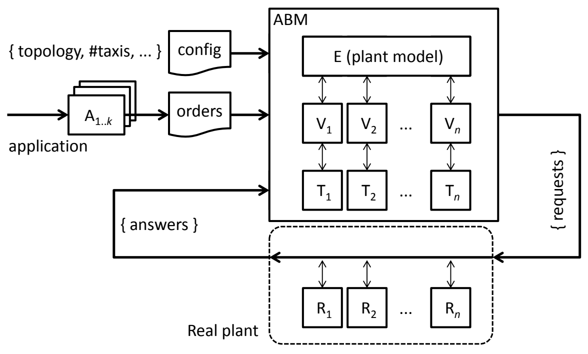

Figure 2. Software architecture of the proposed agent-based controller.

4.2.1. Presentation tier

It is build on top of a web-server to provide users with a broad accessibility through "situated" web-clients so interaction with the system not only depends on the user-permits but also on the access point. For example, direct commands to vehicles can only be accepted from nearby access points, whereas performance statistics may be observed also from access points at managerial departments.

In the ABM, this layer is an application-specific agent which has access to the internal representation of the controlled system and which must be compliant to HMI requirements.

4.2.2. Logic tier

The middle layer contains the set of agents that control the operation of the transportation system, including those that interface with the other tiers and the rest of the software of the plant/company.

4.2.3. Data tier

This layer contains all services related to the controllers' database access. It contains a copy of the data on plant and agents' states for a system recovery from emergency stop or failure, but also as historical information for further analysis. Also, the database includes all static information about the plant and application-specific agents, like the layout of the traffic network or the localization of fixed agents.

4.3. From ABM to prototype

The design process starts with the creation of the multi-agent model and ends with the embedding of the low-level controller of taxis into the real AGVs.

The transportation system for the study case consisted of several AGVs that move on a traffic network with a layout style that is commonly found on many plants, though it is inspired on an automated laboratory with a single collection point and several delivery spots [7].

The system's prototype (Fig. 3) had three real AGVs that moved around the plant by following lines on the ground. The plant layout has dead-end alleys that stand for loading/unloading spots and small segments crossing the main lines that account for landmarks to support AGVs self-location.

The ABM for that system has been built on Netlogo, with taxi controllers following the organization of the proposed architecture, thus divided into high-level ($\{T_i\}$) and low-level parts.

High-level parts $\{T_i\}$ include a basic conflict-solving strategy based on work-order priorities.

The low-level parts of taxis have been implemented in Netlogo ($\{V_i\}$) and embedded into real AGVs ($\{R_i\}$). In our case, we have used simple Parallax Boebots to emulate the AGVs of a plant.

There is an application-specific agent to provide taxis with work orders, but no other agents have been programmed, as the focus of this work was on the transportation part.

Besides, a series of methods have been implemented to assist programming taxis, namely to characterize taxis, to synchronize simulation with reality and to keep record of control-cycle time periods.

The characteristics of a taxi with respect to a plant are represented with data related to the cost (currently, only the time) it takes to itself to get to a node from another or to perform some action at a node. These data are taken from messages between the ABM and the physical part of taxis. Note that taxis may end up by having very different "views" of the traffic network and behaving in a different manner.

The model also includes a mechanism to verify that all measured delays go above twice the model worst-case execution time (WCET) for a single step. In case of failure, inputs from the plant are taken into account out of time and, eventually, lost. There are some alternatives to operate with delays closer to the WCET or to minimize it [8].

The prototype has been working fine (see video at www.youtube.com/atch?v=Af90YT5AVCk) for aspects regarding the controller's architecture. However, it has been implemented without the other two tiers in the full software architecture that has been proposed in this paper, which is left for future work.

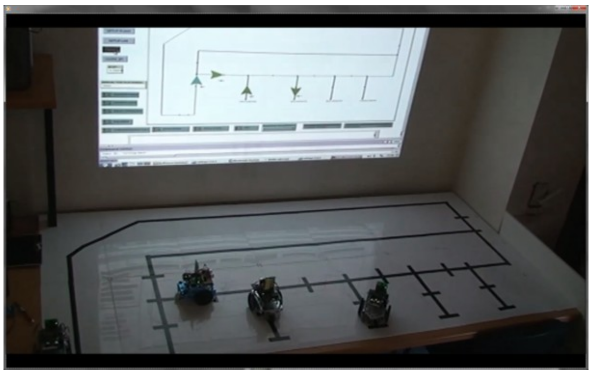

Figure 3. A prototype of an agent-based SCADA for AGVs, with HMI at the background.

5. Conclusions

Manufacturing, logistics and public transportation are increasingly incorporating AGVs to transport material or people inside the traffic network. In this paper we have proposed a model to build the controllers of their transportation systems.

The proposed architecture uses an agent-based model to keep controllers' design complexity at manageable levels. Agents are classified into two classes so that application-specific ones act as representatives of other elements of the full system and transportation ones are in charge of controlling the real AGVs.

We have also looked at the typical requirements of multi-AGV systems in the industry and taken into account to devise an appropriate architecture for their implementation.

Finally, we have developed a simple framework with Netlogo to rapidly design, prototype and deploy AGV systems. The framework has been used to fully develop the agent-based SCADA for AGVs for a case study, which worked fine.

Noteworthy, once the low-level parts are ready and verified, the system requires little configuration and re-programming to adapt to other plants. Besides, adding or removing agents (taxis, HMIs, et cetera) may cause overall performance changes but can be done without modifying the software.

In the near future, we shall include some soft real time synchronization services for non critical events such as temporary obstacles so to improve the quality of the system representation.

References

[1] S. Schreiber, A. Fay. "Requirements for the benchmarking of decentralized manufacturing control systems." In *Proc. of Emerging Techs. & Factory Aut.* (ETFA). 2011.

[2] Tuan Le-Anh, M.B.M. De Koster. "A review of design and control of automated guided vehicle systems." In *Research & Management*. ERIM Report Series. May'04.

[3] T. Kawakami, S. Takata. "Battery Life Cycle Management for Automatic Guided Vehicle Systems." In M. Matsumoto *et al*. (eds.) *Design for Innovative Value Towards a Sustainable Society*. pp. 403–408. 2012.

[4] P. Davidsson, *et al*. "An analysis of agent-based approaches to transport logistics." In *Transportation Research Part C*. pp. 255–271. 2005.

[5] A. Fernández-Caballero, J.M. Gascueña. "Developing Multi-Agent Systems through Integrating Prometheus, INGENIAS and ICARO-T." In *Proc. of Int'l. Conf. on Agents and Artificial Intelligence* (ICAART). 2009.

[6] T. De Wolf, T. Holvoet. "Towards Autonomic Computing: Agent-Based Modelling, Dynamical Systems Analysis, and Decentralised Control." In *Wkshp. on Autonomic Computing Principles & Architectures*. 2003.

[7] Ll. Ribas-Xirgo, *et al*. "Multi-Agent Model of a Sample Transport System for Modular In-Vitro Diagnostics Laboratories." In *Proc. of Int'l. Conf. on Emerging Technologies and Factory Automation* (EFTA). 2012.

[8] P. Mathieu, Y. Secq. "Environment Updating and Agent Scheduling Policies in Agent-based Simulators." In *Proc. of Int'l. Conf. on Agents and Artificial Intelligence* (ICAART), 170–175. 2012.