9 AN INTELLIGENT AGENT VALIDATION ARCHITECTURE FOR DISTRIBUTED MANUFACTURING ORGANIZATIONS

Francisco P. Maturana
Raymond Staron
Kenwood Hall
Rockwell Automation, Mayfield Heights, OH, USA
{fpmaturana, rjstaron, khhall}@ra.rockwell.com

Pavel Tichý
Petr Šlechta
Vladimír Mařík
Rockwell Automation Research Center, Prague, CZECH REPUBLIC
{ptichy, pslechta, vmarik}@ra.rockwell.com

In this paper, we focus on validation of Multi-Agent System (MAS) behavior. We describe the simulation architecture and the system design methodology to accomplish the appropriate agent behavior for controlling a real-life automation system. The architecture is explained in the context of an industrial-sized water cooling system. Nevertheless, it is intended to operate in a wide spectrum of control domains. In general, after the design of the control system is accomplished, a set of validation procedures takes place. The current needs are to validate both the control and the agent levels as integrated parts. Hence there is a need to establish a general architecture and methodology for easing the commissioning process of the control solution.

1. INTRODUCTION

Distributed systems such as manufacturing, supply chains, service industry, and information infrastructures require a flexible structure for the integration of their components to fulfill the market requirements of this century. Solutions to such requirements can be found in Intelligent Agent technology which provides an appropriate framework to integrate knowledge with efficient production actions (Brooks, 1986) (Wooldridge and Jennings, 1995) (Shen, *et. al.*, 2001).

The validation of agent behavior is not only a local to-the-agent issue but a global issue, where the interaction of the agents and associated latencies should also be modeled as part of the system. This requirement introduces an interesting complexity into the design of the architecture.

Distributed organizations emerges as a result of the dynamic interactions of its intelligent components, which can be human or artificial (intelligent agents or holons), or a hybrid (Christensen, 1994) (Mařík, *et. al.*, 2001). To validate the task sequences and interactions of the agents, it is required to understand the agent context from multiple views. We describe the gluing technology to integrate the pieces of the organization from the device level into upper enterprise levels. We focus on the validation infrastructure which is based on a Simulation Development Environment (SDE). The SDE is merged with the agent and control systems. The validation system allows the designer of the agents to verify the feasibility of the agents' actions prior to final commissioning of the system. This is a contribution to the system architecture to improve the design and performance of the future agent-based organization. The agent behavior can be refined exhaustively prior to its final deployment, without ad-hoc investments or complicated equipment in-the-loop. This infrastructure is synchronized with controllers to mimic the real-time operations in order to obtain good representations of the events occurring in the real world. We demonstrate the new infrastructure on an industrial-sized cooling system.

2. AGENT ARCHITECTURE FOR CONTROL

In the past, development of Agent architectures was focused on experimental systems of reduced scale (Maturana, *et. al.*, 2002) (Chiu, *et. al.*, 2001) (Tichý, *et. al.*, 2002). In those experiments, the foundation architecture for highly distributed control agents was established. Step by step, the new requirements were introduced into the extensions of the automation controllers to enable the creation of distributed intelligence in control.

We anticipate that agents will be distributed among multiple automation devices or Programmable Logic Controllers (PLCs) and therefore an agent infrastructure is needed to fit well the manufacturing environment, information networks, and enterprises in general. Each agent represents a physical process or machine or device and coordinates its operations with other agents. The MAS architecture is organized according to the following characteristics:

- Autonomy: Each agent makes its own decisions and is responsible for carrying out its decisions toward successful completion.
- Cooperation: Agents combine their capabilities into collaboration groups (clusters) to adapt and respond to diverse events and goals.
- Communication: Agents share a common language to enable interoperation.
- Fault tolerance: Agents possess the capability to detect equipment failures and to isolate failures.

2.1 Automation Architecture

In agent-based control, the controllers have an agent infrastructure for enabling the component-level intelligence. With this, it is possible to distribute the intelligence among multiple controllers using different agent sizes and populations. In this architecture, controllers of various sizes and capacities can be deployed. Different network connectivity can be used to exploit the distributed intelligence dimension that is added by the agents.

Regardless of the network topology (e.g., backbone or ring), the relationship among the agents is kept loosely coupled. There are dynamic interactions among the agents occurring during the decision-making process. These dynamic interactions establish logical relationships among the agents temporarily. The agents are designed based on FIPA specifications (http:\\www.fipa.org) and ContractNet protocol (Smith, 1980) to create and coordinate their activities throughout logical links. To enable the agent-based automation architecture, it was required to modify the controller's firmware. A common software infrastructure is shared among the different controllers.

The application software represents the physical components and processes of the facility under control by the agents. Each agent represents a physical device such as a valve, water service, heat load, etc. After the agent is created, it is ready to begin operations by carrying out initialization procedures (capability registration) and waiting for external messages or events from the control systems.

The agents contact each other within and outside the controllers via Job Description Language (FIPA/JDL) messages. FIPA/JDL is used by the agents to represent planning, commitment, and execution phases during the task negotiation. Information is encoded as a sequence of hierarchical actions with precedence constraints. JDL is also used to encode plan templates. A plan template is a representation of the agent behavior as parametric scripts. A parametric script has entry variables whose values are set during the planning process. Moreover, the script has associations with internal-to-agent functions which are executed to fulfill local decisions.

When an agent accepts a request, an instance of a plan template is created to record values emerging during the planning process. Requests are propagated throughout the organization using the Contract Net protocol. The requests visit multiple agents and negotiation clusters are formed.

For inter-organization conversations, the agents emit messages outside their organization via wrapping JDL messages inside FIPA envelops. This implementation includes Directory Facilitators (DFs) functionality to be FIPA compliant. A DF performs capability registration and matchmaking. For each capability request, a DF provides a list of agents that coincide with the requested capability. For instance, an overheating component requests cold water from its water service. This is a cooling process capability.

2.2 Intelligent Agent Architecture

The agents are goal-oriented entities. They organize the system capabilities around system missions. There are agents exclusively programmed to emit missions. Other agents are programmed to handle the mission requirements and the execution control. This type of distributed responsibility is easily handled using the agent programming methodology. Information is fractioned into small pieces and each of these is associated with separate agents. Importantly, agents are divided according to class types. Thus, information is encapsulated under a class type as a template to be used by the derived instances of that class type.

Goals emerge dynamically and these are agreed upon by the agents throughout negotiation. For instance, an agent that detects a water leakage in a pipe of the cooling system establishes a goal to isolate the problem. The agent then informs

adjacent agents to evaluate the problem according to their views and borrowed data. This is the origin of a group based goal, which is to isolate the leaking pipes, in spite of the cost of operation. This action exceeds the pre-assigned priorities. Isolation is the highest possible priority.

The architecture of an agent has four components (Tichý, *et. al.*, 2002); planner, execution control, diagnostics, and equipment model. Important part is the execution control component that acts as control proxy, which translates committed plans into execution control actions. These actions are synchronized with the control logic programs. It also monitors events from the control logic and translate them into response-context events to be processed by the planner component.

Both the controller infrastructure and agent architecture facilitate the creation of agents for controlling the physical system. Thus, the control engineer and system engineer can experiment with distributed intelligence and control in a flexible manner. However, the puzzle is incomplete from the solution validation point of view. In general, after the automation system has been modeled in software, it is tested on physical pilot systems until a fine tuning of the control system is achieved. But with the introduction of the agent software, this operation becomes more difficult because the system has a larger number of control variables to be tested and stabilized. Therefore, physical testing of such a system becomes impractical. Hence, there is a need to incorporate a validation system to help the solution modeling process, with a minimum of manual operation and pre configuration cycles.

3. SIMULATION ARCHITECTURE

The general tendency in validation systems for automation control is to build physical prototypes or scaled down models of the real system. This practice is ideal from the accuracy of the observations that are extracted from the operations of the system during validation. Nevertheless, it is also practical to develop simulated models to enable extensive validation process.

Information is organized under the agent scope. This information relates to the transactions that occur during the planning process and during the execution of the plans. Agents enable the construction of more advanced strategies for controlling the system (according to the emergent behavior perspective). Advanced control strategies imply physical changes into the pilot facility, which adds cost and process uncertainty. From the predictive side of the spectrum, more advanced strategies allow for proactive diagnostics. But this requires the equipment to produce specific signatures that are generally obtained after a certain number of service hours. In simulation this can be done efficiently. Therefore, the obvious conclusion is to pursue an integrated architecture that includes all three elements: (1) control, (2) agents, and (3) simulation.

The main components of the validation system are: (1) agent/control software, (2) SDE, (3) soft controller, and (4) simulation. Figure 1 shows these components.

- *Agent/control software*: This component represents the agent and control software creation. Other publications describe more details about this component. This component produces three files types: (a) Agent object code (Agents.o): executable agent code to be placed in RAM of the controller; (b) Ladder logic code (.L5k): control programs written in ladder

logic; and (c) Tag symbol topology (.xml): This represents the inputs and output variables of the field devices.

- *SDE*: This component takes the tag symbol topology and the simulation library to help the user match the control and simulation variables. The variable matching is a critical task that is generally performed manually in very separate contexts and by different people. Commonly, the tag symbols from control do not match the symbols from the simulation models. Another component of this architecture is a tool for importing the symbols from either source (control or simulation) into the other. In this manner, symbols from one source can be made available into the other for ensuring 100% correspondence among the symbols. This component produces an association file which is used to create a proxy. The proxy synchronizes the controllers and simulator clocks and also does data exchange.
- *Soft controller*: This component is an exact emulation of a hardware based controller. It allows for the creation of multiple controllers inside a single chassis as well as communication cards such as Ethernet/IP and ControlNet. This component is intended to contain agents and control programs, i.e., the behavior of the real multi-agent system is emulated in this environment (Mařík, *et. al.*, 2004).
- *Simulation*: This component represents the simulation environment. In this, the application-domain process is modeled using user-preferred techniques and languages. The fundamental idea is to deploy Commercial-Off-The-Shelf (COTS) simulation packages (e.g., Matlab, SolidWorks, Arena, etc.). COTS simulations are more practical from the industrial world point of view. Based on the majority of the cases observed, the usage of commercial simulators is more constructive than writing ad-hoc simulations.

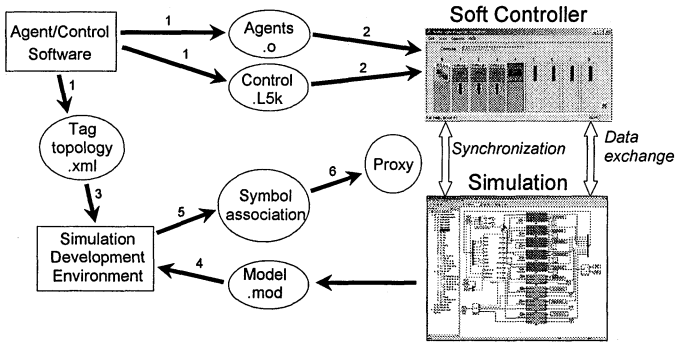

Figure 1 – Simulation system architecture

The simulation architecture adds many degrees of freedom to the design and validation of the agent-based automation system. With this, multiple strategies can be treated as equal and tested using a single computer without incurring into additional investment. Reusability of the infrastructure is a very relevant attribute. The following sections focus on the system design methodology.

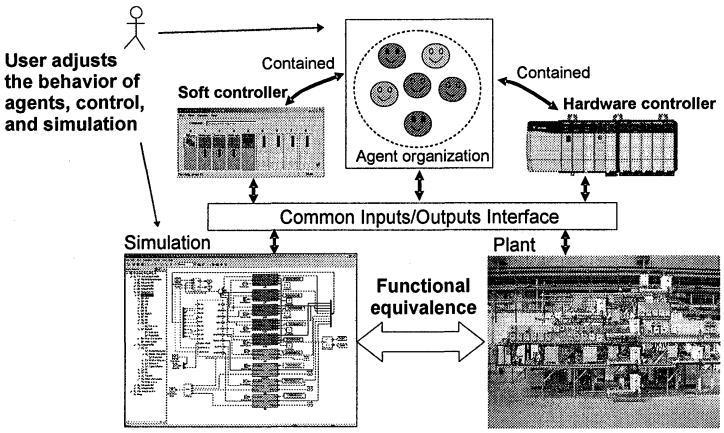

Figure 2 – Software equivalence

The design and validation process is as follows:

1) Users create a first prototype of the agent and control code to suit the characteristics of the physical plant;
2) A simulation engineer creates a simulation model of the physical plant;
3) Users create some desired agent strategies for fulfilling advanced control;
4) Agents and control software are downloaded into the soft controller;
5) Users match the tag topologies and create the synchronization proxy;
6) The integrated system is executed and observed to verify if the desired behavior is fulfilled correctly by the agents and control programs;
7) Users modify the behaviors by changing simulation, control, or agents;
8) Repeat Step (6). If users add more components or if users change the input/output configuration of the initial components, then repeat Step (5);
9) All the desired behaviors have been fulfilled to complete satisfaction; and
10) Software is transported into the physical plant and industrial controllers for final commissioning.

Another important aspect of this methodology is the common input/output (I/O) interface. Both the simulation and the physical plant expose the same set of I/O signals. Therefore, the agent software that interacted with the simulation will see no difference when connected to the physical equipment, because the interconnection has been done through a common I/O set. Nevertheless, at the hardware level, it is expected that some changes will occur regarding the characteristics of the equipment. It is understood that simulation can be very accurate in some cases, but it is still an idealization of the real situation. Nonetheless, these proposed changes are considerably lower than those occurring in a conventional commissioning process, yet from the lab into the pilot facilities.

4. SYSTEM MODELING

Figure 3 shows the cooling system under study. This cooling system is water based and it is currently used at a Navy site to mimic the cooling system of the DDG-51

destroyer class ship of the US Navy. This system is used for evaluation of advanced auxiliary machinery concepts. The cooling system is a reconfigurable fluid system platform with component-level intelligence. It includes the plumbing, controls and communications, and electrical components that mimic the real-life operations.

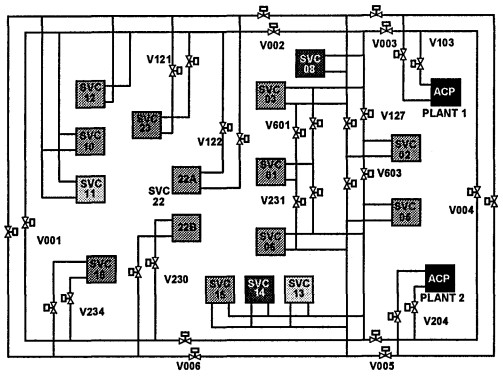

Figure 3 – Cooling system

Immersion heaters provide stimuli for each service (SVC boxes in Figure 3) so as to model actual heat transfer. Essentially, there are 3 subsystems, plants, mains and services. There is one plant per zone (i.e., currently 2 plants: ACP boxes). There are two types of services, vital (14) and non-vital (2). While in operation, under normal conditions, the cooling system is segregated in two zones to maintain the cold water from each source separate. These two zones increase the survivability of the system in case of damage occurring on one side.

The water from the cooling plants (named ACP plants) should never be mixed. Cooling flow is controlled by each service using a local flow circuit. As more services demand cooling the relative demand on the plants is increased. Under low load conditions, it is possible for one ACP to handle all the loads. However, high loading conditions will require that non-vital or low priority loads be shed from the cooling loop until a future time at which time the heat load and water distribution could be balanced again.

The ACP plants were modeled as a single agent each, which included pipes, valves, pumps, an expansion tank, and water-level, pressure, flow and temperature sensors. The main circulation piping is partitioned among 'T' pipe sections, i.e., passive agents. Load agents include a heat generator and a temperature sensor. Water Services agents include valves and flow sensors. There are standalone valves in the main circulation loop for the supply and return lines. This partitioning gives us a total of 68 agents.

5. RESULTS

The results will be presented in terms of the specific models (control, tags, and simulation) of the cooling system. There is no specific target result that can be easily

pinpointed from this work but the capability to integrate agent control and simulation for validation purposes.

The following describes the system's topology. In Figure 4, the mapping of the physical I/O tags as a fragment of the tag symbol topology. The symbol information was automatically extracted from the agent/control models in XML format (refer to Section 3).

```
<Component name="SVC03">
<Tags>
<TAG name="SVC03allReqClose" host="slx1"
    access="r" type="boolean" value="0" />
<TAG name="SVC03allReqOpen" host="slx1"
    access="r" type="boolean" value="0" />
<TAG name="SVC03anyFailedToOpen" host="slx1"
    access="r" type="boolean" value="0" />
<TAG name="SVC03anyIsIsolatingLeak" host="slx1"
    access="r" type="boolean" value="0" />
<TAG name="SVC03reqClose" host="slx1" access="r"
    type="boolean" value="0" />
<TAG name="SVC03valveOK" host="slx1" access="r"
    type="boolean" value="0" />
<TAG name="SVC03waitingForRepair" host="slx1"
    access="r" type="boolean" value="0" />
</Tags>
<Component/>
```

Figure 4 – Cooling service I/O topology

An agent is a component (e.g., SVC03) with a set of tag elements. Each element identifies the name of the tag (e.g., 'SVC03allReqClose'), the name of the controller that contains the tag (e.g., 'slx1'), an access attribute which is 'r' for reading or 'w' for writing, depending on whether the simulation reads or writes into the variable, and a value type and an initial value. Figure 4 only shows a fragment of the tag topology for one service, we have approximately 2000 tags for the whole system.

Next, we explain a use case that was based on a Matlab/Simulink simulation of the cooling system. The simulation is a qualitative model, which includes water flow dynamics and heat transfer simulation for each of the components. Figure 5 shows a partial model of one of the cooling regions. It has five loads (SVC05, SVC06, SVC13, SVC14, and SVC15). Each simulation sub-model has I/O symbols that are imported to the SDE for subsequent matching.

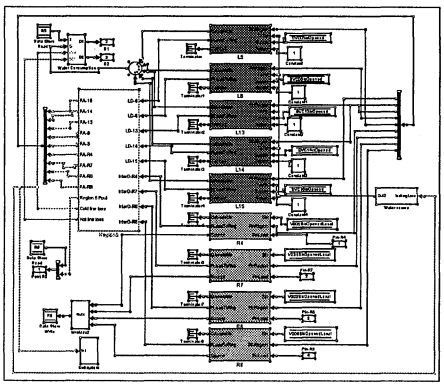

Figure 5 – Simulation sub-model (Simulink view)

After completing the matching of the symbols and proxy configuration, the system was executed to observe its behavior. The experiment shown in Figure 6 consists of emitting a system mission request into the cooling system. The request is to provide cooling under cruise conditions. The cooling system agents tried a configuration by emitting a series of sub-requests to different sections of the cooling system. The initial attempt (see the left part of Figure 6) failed because there was a problem in the water route discovery process. This experiment also failed for other missions such as cooling in 'battle' and 'in-port' modes.

In this experiment, we demonstrated the capability to observe and debug the system's behaviors using a simulation system, real agents and formal control algorithms. After deducing the probable causes of the error, the agent and control code was modified and next experiment was executed. The right part of Figure 6 shows the results. Now, the mission request went through some additional layers marking a successful completion.

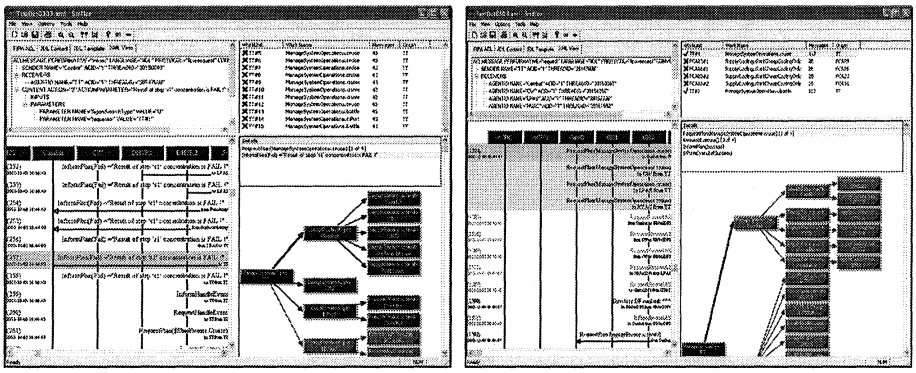

Figure 6 – Results of the first execution (left) and improved performance (right)

This experiment showed that the modification of the code eliminated the problem partially, since there were some failing conditions for the other missions. Without the tool to experiment with partial changes, this troubleshooting process would have been extremely hard and tedious using the real equipment. Progressively, as we continued debugging the system more errors appeared until the system was completely cleaned out to operate as expected.

We think that it is important to remark that the troubleshooting procedure described above does not replace the commissioning phase. On the contrary, it complements the final delivery of the solution by accelerating the process of eliminating errors from the system ahead of time and in arbitrary locations chosen by the designers of the system.

7. CONCLUSION

The above results give explanation of what could be done with the agent/control validation system. In this work, we presented a set of prototype tools and procedures

well aligned with industrial automation. Other more complex interactions have been experimented with excellent results. One immediate observation is the reduction of the design time from the beginning of the modeling until obtaining a good working model. It has been also observed that the number of modification cycles increased but these were processed faster. This technique prevented our team from experimenting with real expensive equipment. The debugging and validation tasks were partitioned among multiple users. Each user pinpointed specific advantages and deficiencies of the system. Current research efforts are on the improvement of the new tools and on the establishment of a general methodology to create agents and agent validation environments.

7. REFERENCES

1. Brooks A.: "A Robust Layered Control System for a Mobile Robot", IEEE Journal of Robotics and Automation, 2(1), 14-23, 1986.
2. Chiu S., Provan G., Yi-Liang C., Maturana F., Balasubramanian S., Staron R., and Vasko D.: "Shipboard System Diagnostics and Reconfiguration using Model-based Autonomous Cooperative Agents", ASNE/NAVSEA Intelligent Ship Symposium IV, Philadelphia, PA, April 2001.
3. Christensen J.H.: "Holonic Manufacturing Systems: Initial architecture and standards direction", First European Conference on Holonic Manufacturing Systems, Hanover, Germany, 20pp, 1994.
4. IEC (International Electrotechnical Commission), TC65/WG6, 61131-3, 2nd Ed., Programmable Controllers - Programming Languages, April 16, 2001.
5. Mařík V., Pěchouček M., and Štěpánková O.: "Social Knowledge in Multi-Agent Systems". In: Multi-Agent Systems and Applications (Luck M., Mařík V., Štěpánková O., Trappl R. eds.) LNAI 2086, Springer-Verlag, Heidelberg, pp. 211-245, 2001.
6. Mařík V., Vrba P., and Fletcher M.: "Agent-based Simulation: MAST Case Study". Accepted by the 6th IFIP International Conference on Information Technology for Balanced Automation Systems in Manufacturing and Services (BASYS'04), Vienna, Austria, 2004.
7. Maturana F., Staron R., Tichý P., and Šlechta P.: "Autonomous Agent Architecture for Industrial Distributed Control". 56th Meeting of the Society for Machinery Failure Prevention Technology, Section 1A, Virginia Beach, April 15-19, 2002.
8. Maturana F.P., Tichý P., Šlechta P., and Staron R.: "Using Dynamically Created Decision-Making Organizations (Holarchies) to Plan, Commit, and Execute Control Tasks in a Chilled Water System". In Proceedings of the 13th International Workshop on Database and Expert Systems Applications DEXA 2002, HoloMAS 2002, Aix-en-Provence, France, pp. 613-622, 2002.
9. Shen W., Norrie D., and Barthès J.P.: "Multi-Agent Systems for Concurrent Intelligent Design and Manufacturing". Taylor & Francis, London, 2001.
10. Smith R. G.: "The Contract Net Protocol", High-level Communication and Control in a Distributed Problem Solver. In IEEE Transactions on Computers, C-29(12), pp. 1104-1113, 1980.
11. Tichý P., Šlechta P., Maturana F.P., and Balasubramanian S.: "Industrial MAS for Planning and Control". In (Mařík V., Štěpánková O., Krautwurmová H., Luck M., eds.) Proceedings of Multi-Agent Systems and Applications II: 9th ECCAI-ACAI/EASSS 2001, AEMAS 2001, HoloMAS 2001, LNAI 2322, Springer-Verlag, Berlin, pp. 280-295, 2002.
12. Wooldridge M. and Jennings N.: "Intelligent agents: theory and practice", Knowledge Engineering Review, 10(2), pp. 115-152, 1995.