Designing Agents for Manufacturing Control

Sven Brückner[#], Jo Wyns, Patrick Peeters[†], Martin Kollingbaum[‡]

[#]Daimler-Benz AG
Research and Technology 3
Alt-Moabit 96A
D-10559 Berlin
Germany
Sven.Brueckner@dbag.bln.daimlerbenz.com

[†]Katholieke Universiteit Leuven
Department of Mechanical Engineering
Celestijnenlaan 300B
B-3001 Leuven
Belgium
{Jo.Wyns, Patrick.Peeters}@mech.kuleuven.ac.be

[‡]University of Cambridge
Manufacturing Automation and Control Systems Group
Mill Lane
Cambridge
UK, CB2 1RX
mjk27@eng.cam.ac.uk

Abstract

The hype in agent research will not last forever. If the agent research community does not succeed in bringing its results into real-world application within the next two or three years it will share the fate of some other AI sections which are today thought to be just academic. Multi-agent systems applied to manufacturing are candidates for bringing the first breakthrough. To achieve that, methodologies for the design of agent systems for manufacturing by software developers are needed. This paper reports on the design process chosen in the MASCADA project, whose goal is not only to develop an algorithmic solution to manufacturing problems but also to show a transfer strategy for these results.

Introduction

This paper reports on the design process of a multi-agent system for manufacturing control. This work is embedded in the MASCADA project, which aims at the development of an agent-based manufacturing control system and a design methodology to enable software developers to bring this algorithm finally into real-world applications. To illustrate the design process, the car body painting at the Daimler-Benz AG plant in Sindelfingen (Germany) is taken as an example. It is also the main test case of the MASCADA project.

The focus of this paper is not a description of the agent-based control system for Sindelfingen, but the design process chosen to come to such an application. The design process itself is still being evaluated and revised in the MASCADA project.

It is important for the agent research community to transfer their results to software companies in order to get this new and exciting technology finally implemented in real-world applications. Therefore not only the results themselves are important, but also the methodology to apply these results in real-world applications without being an agent researcher. If the agent community does not get the scientific results into the real world, the agent hype of the last years will vanish as fast as it has started.

The design process itself is going to be introduced in the first section. And the reminder of this paper will elaborate on each of the steps of this process. To give the examples a solid foundation and to show that the application covers a real-world problem, the first design step, the application analysis, is going to be rather detailed.

The Design Process

The design process chosen is an evolutionary method, based on several iteration steps to optimize and refine the

reached solution.

It starts with a very intensive analysis of the domain. This is necessary to enable the agent designer to understand the real-world processes and the optimization goals. It is assumed that there is already an existing production system which has to be provided with a new control system. That assumption reflects not only the situation in Sindelfingen but it also applies to most of today's agent projects. This assumption is offering a migration strategy.

The outcome of this first step is a process model and an ontology. The process model will be used to identify the single agents. The ontology helps to specify the distribution of knowledge in the agent system and gets useful when a larger or wider distributed group of project partners is supposed to cooperate.

After the application domain analysis, the design of the multi-agent system begins. The design is an iteration process where each loop results in a more and more refined and optimal agent system. This process can be stopped after any number of loops depending on the criteria the control system has to fulfill. The single steps in this design loop are:

1. Domain analysis / Determination of the goals of that iteration.
2. Specification of the agent types and their typical interactions.
3. Specification of each agent type's knowledge, abilities and behavior.
4. Identification of the agents in the application.
5. Evaluation of the system.

This design process is a specialization of design methodologies suggested in (Parunak 1997) and (Burmeister 1996). These methodologies do not focus on the manufacturing domain and therefore they cannot suggest, as this paper does, an abstract concept from where to start the design. The Parunak paper presents techniques for an informal evaluation of the design which should be applied here as well. The distinction in three models (agent model, organizational model, cooperation model) as Burmeister suggests is not that explicit in the manufacturing control domain. But its aspects are still valid issues to apply.

The examples in the following sections show, how the design process is applied to the Sindelfingen domain.

The Application Domain[1]

The domain considered is the last section of the painting center of the Daimler-Benz AG passenger car plant in Sindelfingen, Germany. This section covers two painting steps and their corresponding recovery procedures, all used to produce a perfectly painted car body.

[1] Different domain analyses can be found in the deliverable of work package one in the MASCADA project. It will be available on the MASCADA WWW-site: www.mech.kuleuven.ac.be/project/mascada/welcome.html

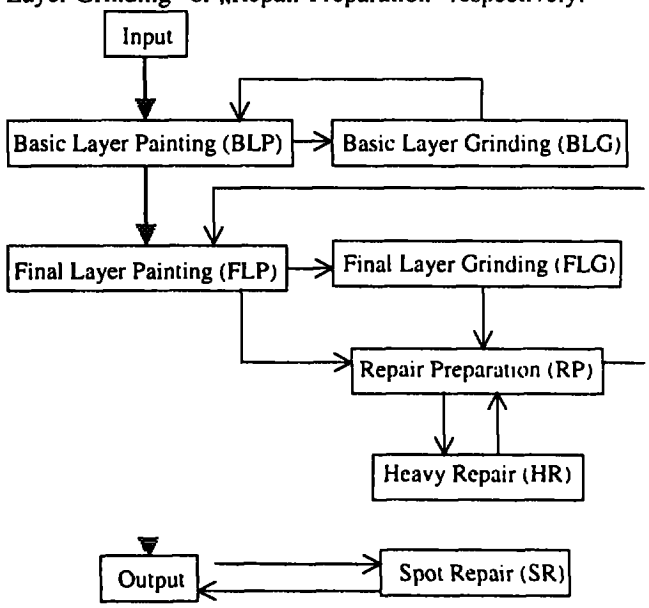
At first glance it seems to be a simple, straight forward task for a control system to transport the car bodies to the different processing stations and eventually out of the building.

However, a deeper analysis reveals two problems: feedbacks and buffering.

A feedback loop inserts car bodies after one or more recovery steps back into the production system. Because it is not possible to throttle back the input stream into the system, this results in an increased and mixed input stream of car bodies, every time a car body does not pass a quality check. The increased input stream increases the load of the processing units, which on their turn results in more reparations and as a consequence in an even more increased feedback.

The following picture shows an abstract model of the production process. The nodes in the depicted graph are the processing steps, while the arcs are virtual streams of car bodies.

The picture shows three main feedback loops. One into the first main processing step „Basic Layer Painting" via the recovery processing „Basic Layer Grinding" and the other two leading into the „Final Layer Painting" step via „Final Layer Grinding" or „Repair Preparation" respectively.



Picture 1 - Feedback Loops in the Process Model

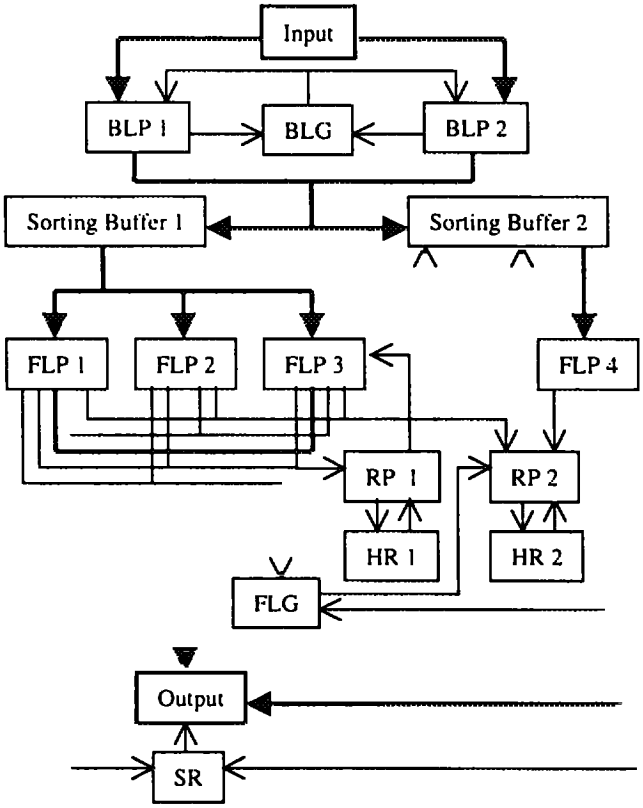
A quality check is performed after each main processing step determining if a car body can pass on to the next main step (or to the exit) or if it has to come back via one of the recovery steps. Which of the recovery actions is to be taken is decided there as well.

Buffering between the processing steps is another control problem. Two types of buffering are possible in the system. One is used to store car bodies during shift changes and breaks or to buffer breakdowns and other delays in the succeeding resources. The buffers of this kind are simple, roughly first-in-first-out storage buffers.

The other, more complicated buffer type is a sorting

before. Those buffers are able to access virtually any car body within its storage for the next output. The function of these sorting buffers is to rearrange car streams. Because the performance of the "Final Layer Painting"-yield mainly depends on the batch size of cars that have to be painted in the same color, each "Final Layer Painting" unit has a preceding sorting buffer to rearrange the input stream in a stream of cars that have to be painted in the same color. As a consequence, the performances of these buffers have a great impact on the overall performance of the manufacturing

In order to get a complete model of the manufacturing process, it is necessary to specify the number of processing units of each type and the transport and buffering conditions between them.



Picture 2 – Complete Process Model

The transport system imposes strong constraints on the ability of the control system to shift the streams between the processing units during an efficiency breakdown of the processing units. An efficiency breakdown is a situation where the overall number of cars going into a feedback loop increases drastically and therefore the load of the system reaches dangerous levels.

As long as the load of the system is low or the efficiency of the painting lines is high enough to ensure that at least the same number of car bodies leaves the system as new car bodies are coming in, the control job is easy. But if the inflow of cars increases drastically or the efficiency is breaking down the decisions to be taken are getting to complex for a centralized system to handle.

Take an efficiency breakdown in FLP1 as an example. The amount of car bodies going through RP1 is increasing drastically while less car bodies can leave the system. But all car bodies coming out of RP1 are going into FLP3 without being sorted in SB1. That results in a higher load for FLP3 and an almost certain decrease in FLP3 efficiency (partially unsorted stream). A lower efficiency means a higher load for RP1 and so on. A good control system would have to throttle back the input into FLP1 because the real source for the trouble in FLP3 and RP1 lies there. So it is very complicated to find the right balance and to keep it. In a distributed control system this balance could be the emerging feature of the agents behavior.

First Iteration

In a first step of each iteration, the goals that have to be reached at the end of that iteration must be defined. In case of the manufacturing control system of the Sindelfingen plant the goals of the first iteration are:

1. Eventually, every car body must have been processed.
2. The performance of the new control system should be at least as good as the performance of the existing one. The performance measures used are: throughput, yield, idle resources, congestion levels and operability.
3. The new control system must be able to cope with drastic disturbances like complete breakdowns of processing capabilities.
4. The new control system must try to balance the system even during quality breakdowns.

So the aim of the first iteration is the development of an agent-based manufacturing control system, that is able to safely control the manufacturing process. Other requirements such as reaching optimal results in terms of the production output are maintained for future iterations. This is sensible because the manufacturing process itself is already complex enough. The control system should be simple in the beginning and get more complex after the basics have been understood.

Agent Types and Interactions (First Level of Abstraction)

The PROSA manufacturing control reference architecture (Van Brussel et al. 1998, Bongaerts et al. 1996) is used as the basis for the design of the agent-based manufacturing control system. PROSA, standing for Product-Resource-Order-Staff Architecture, originates from a research project on Holonic Manufacturing Systems (Valckenaers et al. 1998). It defines the entities that make up a manufacturing control system, their responsibilities, and their relationships.

The PROSA concept has been developed with a background in Holonic Manufacturing Systems but for the purpose of this paper the difference between holons and

agents is irrelevant as (Bussmann 1998) suggests.

PROSA defines three types of essential (basic) agents: Product-Agents, Resource-Agents, and Order-Agents. Product-Agents take care of product and process related technological aspects such as which operations need to be performed to achieve the product. Resource-Agents take care of resource aspects such as driving a machine at optimal speed and maximizing its capacity. And finally, the Order-Agents take care of logistical concerns about customer demands and due dates. A fourth type of agent is the Staff-Agent which is optional. The Staff-Agents may assist the basic agents in performing their task more optimally.

The following sections show, on the basis of the Sindelfingen example, how the agent types of a manufacturing control system can be derived from the PROSA concept. Because the application design in the Sindelfingen domain is still in the first iteration, the agents are still very simple in terms of knowledge and interactions. Further iterations will surely introduce some Staff-Agents and make the basic agents more complex.

The Order- and Product-Agents

Starting from the PROSA concept, the agent types for the Sindelfingen application are derived. The most obvious agent is the Car-Agent. A Car-Agent is a specialization of the Order-Agent concept. This agent is responsible for an order to be fulfilled. An order represents the task to process a car body in a certain way with a set of parameters specifying all necessary processing steps in full detail.

One special feature is added to the Car-Agent: the Car-Agent is not only responsible for an order, but also for the car body that corresponds to that order. If this car body gets so badly damaged during processing that the order cannot be fulfilled the Car-Agent is responsible for extracting the car body safely from the production system instead of trying to find another car body to fulfill the order. After the release of the car body, the Car-Agent can be killed as well. As a consequence, there will never be a Car-Agent without a car body and vice versa.

The Sindelfingen application is rather static in terms of product- and process changes. Therefore the functionality of the Product-Agent to provide the Order-Agent with processing information is integrated into the Car-Agent itself. No non-abstract agent type was derived from the Product-Agent during the first design loop.

The Resource- and Product-Agents

For the Sindelfingen application, the abstract concept of a Location-Agent is derived. The Location-Agent is a specialization of the Resource-Agent. It represents a clearly defined area of some transport units and possibly one or more processing units. Hence a Location-Agent's area has some entries and some exits where car bodies can enter or leave respectively. In this area, the agent has complete control over the transport or processing of the car body. It is able to change the direction of a car body based on knowledge about possible transportation goals in the overall layout and the (partial) mapping onto exits of its own area. Therefore, the location of a car body is not a mere parameter of the Car-Agent but it represents the fact that the car body is under the physical control of the Location-Agent.

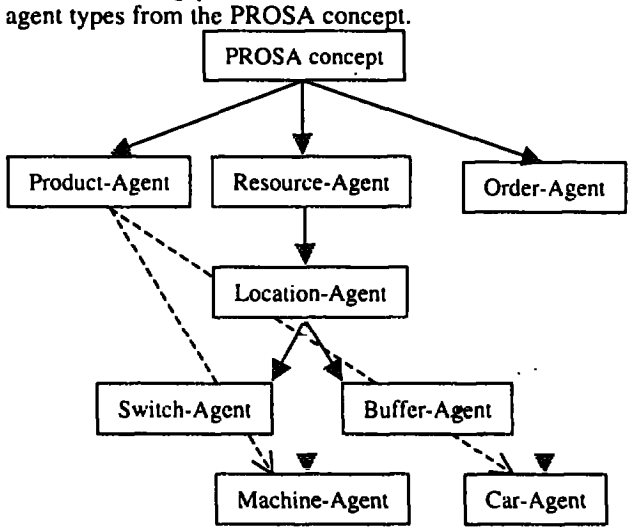
From the Location-Agent, three different agents are derived: the Switch-, the Machine- and the Buffer-Agent.

Switch-Agents are Location-Agents with only transport units in their area. Therefore they represent a pure transport resource taking car bodies in through its entries and sending them out through their exits. Inside its area of influence the Switch-Agent can direct the car bodies according to its policy.

A Machine-Agent has at least one processing unit within its area and therefore it provides some (aggregated) processing to the incoming car bodies. It could also switch car bodies along its processing units but its main focus is the processing. The functionality of the Product-Agent to provide the Machine-Agent with data on how to process an order correctly has been integrated into the Machine-Agent itself. With its aggregated processing abilities, the Machine-Agent already knows all the actions necessary to fulfill a proposed processing step without asking any other agent.

A Buffer-Agent is very similar to a Switch-Agent. It also contains no processing units within its area. But because of its internal layout and its special transport strategy a Buffer-Agent is able to store car bodies within its area. The Buffer-Agents area can be made up of simple transport units used to store car bodies or it could contain a special buffering device. In the second case the Buffer-Agent just wraps it.

The following picture shows the derivation of the used agent types from the PROSA concept.



Picture 3 - Agents Derived from PROSA

Types of Interaction

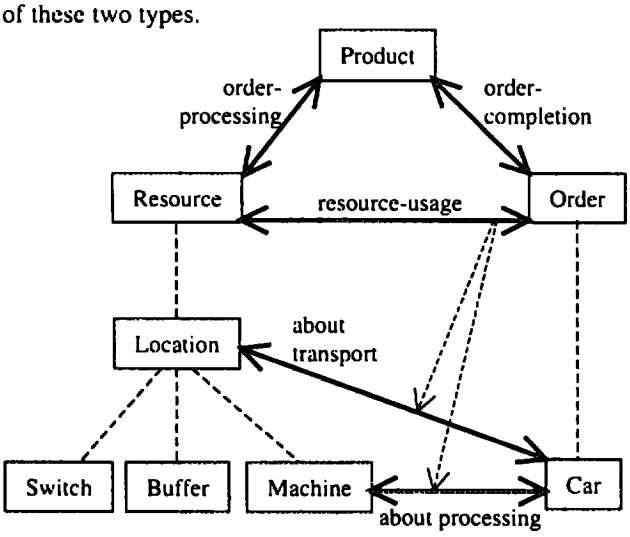
The abstract types of interaction can be derived from the PROSA concept as well. An Order-Agent interacts with a

Resource-Agent about the usage and resource for the order. And the Product-Agent is providing the other two agent types with information on the further production steps of the order. So there are interactions on resource-usage (Order-Machine), order-processing (Product-Machine) and order-completion (Product-Order).

Applying these abstract classes of interaction to the intended application and the derived agent types is the next step in the design process. Starting with the responsibilities of the non-abstract agent types, the types of interaction can be specified in more detail.

In the Sindelfingen case only one of the three abstract interaction types are still visible in external agent behavior, because there is no non-abstract agent type derived from the Product-Agent. The order-processing interaction is hidden in the internal knowledge of the Machine-Agent on how to actually fulfill a processing step. And the order-completion interaction takes place internally when the Car-Agent determines the next processing step based on the car body's current processing state.

The resource-usage interaction type is split up according to the different resources the Resource-Agents are responsible for. In general there are two different resources handled by agents. Transport resources are handled by all Location-Agents and therefore each agent is able to participate in interactions about transport. Only Machine-Agents are responsible for processing resources. So the interactions about processing only happen between the Machine-Agents and Car-Agents. The agent specification, second level of abstraction, will specify all the interactions of these two types.



Picture 4 - Interaction Types

Agent Specification (Second Level of Abstraction)

During this design step , the knowledge, abilities and behavior specific to each agent type is going to be specified. Furthermore the interactions between the agents are fleshed out. The result of this step is going to be the first specification of the implementation of the agent types. Questions arising about specific agents in the application are tackled when the general agent type is in an applicable form. This specification is concerned with the distributed algorithm in general.

Specification of a Multi-Agent System

In object-oriented design the Unified Modeling Language (UML) has emerged as a quasi standard for the specification. It does not only cover the structure of the object model, but it also contains constructs to describe the use of the system in typical situations (use-cases) and the message passing process between objects (sequence diagrams).

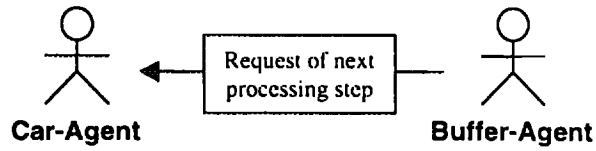
Agents derived from the PROSA concept seem to be quite simple compared to more complex agents used in domains like information gathering or human-machine interfacing. They are still close to the object-oriented world and it seems worthwhile to look at object-oriented methods like UML for their specification.

Another reason making the UML approach interesting as a specification format is the interactivity of the agents. Besides specifying what the single agent is responsible for and what it knows, the interactions between the agents in certain situations are very important. These interactions provide triggers for many actions of the agents and if they are not specified correctly the implementation of the agent system would not be possible.

So, which UML constructs can be used for the agent specification? Use-cases for the specification of important situations and the global behavior of the agent system; sequence diagrams for the concrete definition of the agents interactions and an object model to describe the knowledge of an agent.

For the first version of the MASCADA multi-agent system for the painting center in Sindelfingen important situations in the agent interactions were specified using a use-cases-like format. These situations are (i) the determination of the fulfillment of the next processing step beginning with the completion of the previous one and (ii) the routing of a car body through the transport system.

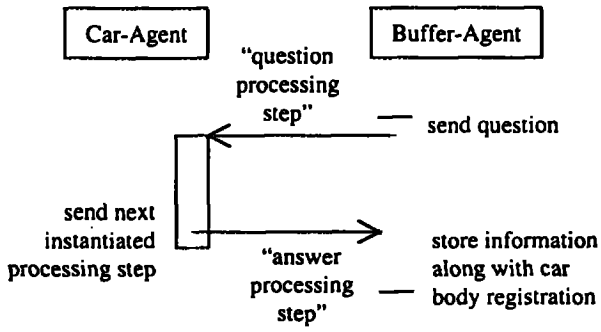
As an example, the representation of the request of the next processing step sent by a Buffer-Agent to a Car-Agent. The representation of this situation looks as follows:



Picture 5 - Use-Cases for Situations

After the identification of the situations where the agents have to interact, the interactions themselves have to be defined. The appropriate tool for specifying the interactions are sequence diagrams. In a sequence diagram the exchanged messages, the internal actions and even the usage of different threads for the processing of events can be defined.

The simple interaction on the next processing step can be represented in this short sequence diagram:



Picture 6 - Sequence Diagrams for Interactions

The names of the used messages are written above the arrows depicting the exchange of messages between agents. Internal actions are named next to the "lifeline" of each agent and the threads are the boxes on top of the "lifelines".

The next step of the interaction specification is the formal definition of each message in the interaction. Each message is defined by its message name (or ID), the sending and receiving agent type and the data sent by the messages parameters. A message specification could look like this:

```
(<Buffer-Agent> <Car-Agent> <Protocol-Number> question_processing_step)
```

With the specification of the sent messages the external behavior of an agent is defined. To complete the specification of the agents, the internal actions and the knowledge of each agent has to be added. Both are emerging from the previous steps. The internal actions have already been named in the sequence diagrams and some knowledge of an agent has shown up in the declaration of the messages. But the agent designer should always check if there are some internal interactions with the underlying systems which could lead to more internal actions than the interactions between the agents suggest.

The specification of the internal actions will not use any UML derived features. The actions could be specified in graphical form like a program graph. But a verbal description should be sufficient. The structure of the agents knowledge could be represented in an object diagram. But often it is not so structured and a listing if the data items is more appropriate.

Identifying Agents for the Application

However, the agent types have been completely specified, the design process still hasn't produced a working control system for the manufacturing process yet. The specified multi-agent system is still based on abstract agent types without any application to the actual layout and special features of the actual application. So the next step will ground the multi-agent system into reality.

To reach that goal, the agent designer has to identify all agents in the application. That means the identification of all instantiations of each agent type. This is done by looking at the domain analysis. First of all Machine-Agents and some Buffer-Agents can be found in the process model of the painting center. Then the agent designer has to look at the actual layout of the transport system to find all the Switch-Agents and probably some more Buffer-Agents.

A rule of thumb during this design phase is that there should be no areas in the production system without an agent which include more than linear transport. How fine grained the multi-agent system will be in the end and in whose agent's area the single transport units will end up is a design decision which can be revised in later iterations of the design process when the need arises.

Further Iterations

After having implemented and tested the agent-based control system in a simulation system, the results have to be analyzed. To measure the quality of the new control system, a simulation of the current control system using the same test scenarios has to be compared with the results from the new control system. In the Sindelfingen application it will be possible to emulate the behavior of the current control system with the agent-based control system. That decreases the resources needed for the implementation of the simulation.

The evaluation of the agent-based control system is done in simulation. To simulate the underlying processes correctly, data from the in-depth analysis are needed. The digital mock-up includes the production equipment, the actual layout of the resources and their correct behavior. It is important for a correct evaluation to get the simulation as close to reality as possible.

To create the right test scenarios for the simulation, domain knowledge from the analysis of the application is needed. The evaluation should take place in real-world scenarios. This is important for the later acceptance of the new control system because it would be much to expensive to set up real-world experiments in the plant.

The analysis of the tests of the agent-based control system should focus on the following issues:

- Is the implementation correct?
- Is the algorithm working correctly?
  - ✓ no car body that has been processed wrongly
  - ✓ no car body lost by its agent
- How good is the algorithm qualitatively?
  - ✓ deadlocks, fairness, stability
- How good is the algorithm quantitatively?
  - ✓ numbers of processed car bodies
  - ✓ load of the processing units
- Is the chosen algorithmic approach feasible?

In answering all these questions the evaluation will lead to a set of goals the next iterations of the design process will have to reach. Eventually the evaluation of the reached design will show that all objectives of the project are fulfilled and the design of the multi-agent system is completed. The next step will then hopefully be an implementation of the multi-agent control system in a real

product. This could uncover new problems with the real-world which could lead to further design loops.

For the Sindelfingen application the following iterations in the design process will focus on more advanced issues like optimal operational results, maintenance, human control in assigning strategies, emerging strategies and adaptation. Furthermore a feedback from the control systems design into the actual layout of the plant is going to be addressed. One outcome of the MASCADA project should be suggestions on how to change the layout to get even better results with an agent-based control system.

In general this evaluation step bridges the gap between the design of a multi-agent system for a certain global behavior and the specification of the single agents. An example is the agent-based control system for Sindelfingen. The specification of the agents never mentions scheduling explicitly but the multi-agent system shows an emergent scheduling behavior during runtime. Therefore the design problem is to reach a certain global behavior of a complex system only through the specification of the single entities it is made of. And the design process itself can be seen as an evolutionary approach.

Conclusion

This paper reported on a process to design a multi-agent system for the control of manufacturing systems. This methodology was chosen in the MASCADA project to realize an application in the painting center of the Daimler-Benz AG car plant in Sindelfingen, Germany.

The design process starts with an analysis of the application domain and then iterating the specification of agent and interaction types, and agent identification. It can be interpreted as an evolutionary design approach to achieve an emergent global system behavior through the (local) design of the systems components.

The first iteration of the design loop is aimed at a control system delivering the basic functionality to safely run the production system. Further iterations add more functionality in terms of the performance measures and usability of the system.

During the design process tools adapted from the object-oriented design methodology UML were used to standardize the representation of the specification for all projects. It is hoped that this will help to create a universally applicable format for agent specification to ease the transfer of agent research results to software developers.

At the moment the design process itself is undergoing evaluation and refinements not only in the Sindelfingen application, but also in other test cases like electronic assembly or steel casting.

Acknowledgements

This paper presents research results obtained through work sponsored by the European Community (ESPRIT LTR 22728). All MASCADA partners – Katholieke Universiteit Leuven, A.I.Systems, Brussels, Daimler-Benz AG Research & Technology 3, Berlin, University of Cambridge and VTT Automation – contributed to this work and assume the scientific responsibility.
www.mech.kuleuven.ac.be/project/mascada/welcome.html

References

Van Brussel, H., Wyns, J., Valckenaers, P., Bongaerts, L., and Peeters, P. 1998. Reference Architecture for Holonic Manufacturing Systems: PROSA. *Computers in Industry*, special issue on intelligent manufacturing systems.

Bongaerts, L., Wyns, J., Detand, J., Van Brussel, H., and Valckenaers, P. 1996. Identification of Manufacturing Holons. In *Proceedings of the European Workshop for Agent-Oriented Systems in Manufacturing, Berlin, Germany, 26-27/9/96*, Eds. S.Albayrak and S.Bussmann.

Valckenaers, P., Van Brussel, H., Bongaerts, L., Wyns, J., and Peeters, P. 1998. Holonic Manufacturing Control at K.U.Leuven. INCOM98, IFAC Symposium on Information Control Problems in Manufacturing, June 24-26, 1998, Nancy, France.

Bussmann, S. 1998. An Agent-Oriented Architecture for Holonic Manufacturing Control. *In Proceedings of the First International Workshop on Intelligent Manufacturing Systems*, 1-12.

Burmeister, B. 1996. Models and Methodology for Agent-Oriented Analysis and Design. In *Proceedings of the Workshop on Agent-Oriented Programming and Distributed Systems (KI'96).*

Van Parunak, H., Sauter, J., Clark, S. 1997. Toward the Specification and Design of Industrial Synthetic Ecosystems. In *Proceedings of the Fourth International Workshop on Agent Theories, Architectures , and Languages (ATAL'97).*