Real-time capable software agents on IEC 61131 systems – Developing a tool supported method

Ursula Frank*, Josef Papenfort*,
Daniel Schütz**

*Beckhoff Automation GmbH, Verl, Germany (e-mail: j.papenfort@beckhoff.com, u.frank@beckhoff.com).
**Automation and Information Systems, Technische Universität München, München, Germany (e-mail: schuetz@ais.mw.tum.de)

Abstract: In this paper an approach that utilizes the paradigm of agent-orientation for the software development of automation systems, is presented. We discuss different methods for agent-oriented software development in general and motivate a novel approach that meets the special requirements for control software in production automation. We show how this novel approach can enhance the dependability of control software by integrating concepts for diagnosis and failure compensation in real-time. Furthermore, we present a modelling approach, that will be integrated into a market leading IEC 61131 software development tool and by that, allow application engineers to implement agent control software.

Keywords: Real-time algorithms; Programmable logic controllers; Knowledge-based control.

1. INTRODUCTION

Industrial production technology as well as the corresponding field in engineering are classified as manufacturing and processing technology. Process technology can be further divided into batch processing and continuous flow processing. Manufacturing technology is classified as a discrete process (Vogel-Heuser, 2009).

Due to the rising pressure of the costs in production engineering and shortening lifecycles of the produced products, the requirements regarding the flexibility of production systems against changes in the technical structure as well as changes in the production processes are increasing. These changes can be, for example the extension of production plants with additional components or the implementation of new production processes.

Furthermore, for production systems there are high requirements regarding their robustness against defects and failures in general, whereas the consequences of for example unplanned stops in production have different impacts on different production categories. In manufacturing systems such interrupts are unwelcome but mostly not very critical, because the treated material remains stable during the interrupts. On the contrary, the treating of a material in the process technology cannot be interrupted as easily, because the material continues its reaction during the interrupts. Continuous flow processes are extremely challenging to interrupt and a fast flowing material requires real-time reactions from the automation system. One of the most common sources of failures in the production plants as well as in the process plants are defects of sensors.

A control system may have different strategies to appropriately react on such failures, but it may not be the same for all classes of production processes. For example, shutting down a production process for maintenance may not be an appropriate strategy in process automation, whereas in manufacturing automation this might be the case. Instead of entirely aborting a process, forcing the controlled device to a stable state can be preferred to avoid further malfunctioning. This may allow the production to continue for a short time or until the process can be shut down safely. A third way to react on sensor failures is a dynamic reconfiguration of the system during runtime, using redundant devices or information. However, hardware backup for each sensor would double the cost of the sensor equipment of a plant. In some cases it may even not be possible to attach a redundant sensor technically to a plant due to spatial restrictions.

An approach to enhance the reliability of plants in production automation as well as in process automation using software agents was developed and presented in (Wannagat et al., 2008a). This approach integrates the concept of virtual sensor redundancy within software agents. The results from several demonstration scenarios show that the agents can autonomously compensate malfunctions of sensors without affecting the overall plant. By implementing the software agents controlling the production process directly on decentralized PLCs, the dynamic reconfiguration of the control software can be realized in real-time.

However, this promising approach is yet lacking an appropriate tool support as well as a supporting methodology for the development and implementation of agent systems. To be utilized in real industrial applications, a tool supported methodology has to be developed, that enables a successful development and implementation of agent oriented software by application engineers in industry. Therefore, in cooperation with the chair Automation and Information Technology at the Technische Universität München, a tool supported approach for the development of agent control software that meets the special requirements for control software in production automation is presented. We discuss different methods for agent-oriented software development in general and motivate a novel approach that meets the special requirements for control software in production automation.

We show how this novel approach can improve the dependability of control software by integrating concepts for diagnosis and failure compensation in real-time. Furthermore, we present an implementation of this approach into a market leading software development tool and by that, allow application engineers to implement agent control software.
Systems (AIS) of Technische Universität München, we are currently working on the realization of such a tool-support.

2. RELATED WORK

The main challenges regarding the realization of a tool support for the agent based systems’ development, are the seamless integration into standard engineering processes and an adequate support for application engineers. The approach for software agents we are pursuing, aims at an implementation of agents on PLC platforms. Therefore, seamless integration into the standard engineering process requires the embedding of the tool support in a standard software development environment for IEC 61131 systems as well as an integration of the procedure model into the current control software development procedures. This is also a basic requirement regarding an adequate support for application engineers. In a partly derived model, an application engineer has to integrate additional knowledge from operation experience and/or his personal experience. We need to offer an intuitive, easy to use modeling notation and procedure which is integrated into an already established PLC programming environment. Furthermore the developed methodology has to support application engineers in dealing with the restrictions, the increased flexibility and the modeling of an agents’ knowledge base. Finally, an automatic mapping from a modeled agent system to executable IEC 61131 control software has to be realized.

The agent-oriented software engineering (AOSE) is a relatively new approach in the system analysis and design. It allows the application of agent systems to software projects dealing with real-life distributed scenarios. In the following, brief overviews selected from the existing AOSE-methods are given separately and the methodology especially designed for the industrial field-level control is motivated.

For the agent-oriented system analysis and design, non-formal approaches have the most significant practical impact. The non-formal methods use structured natural language and graphical notations for the specification and design of the agent systems. They support the software development phases from the analysis and the design of the software up to the implementation of executable program code. The usability and the continuity of the support vary from method to method.

Some domain specific requirements restrict the utilization of most AOSE-methods for industrial applications. The lack of suitable methods for industrial applications has been noticed by several working groups. The national project AgentAut (Lüder et al., 2006) as well as the European project Pabadis promise (Peschke et al., 2005) work on an integrated method for distributed control systems and focus on the integration of PPC/MES and control level. The European projects SOCRADES (Socrates) and RI-MACS (RI-MACS) use agents to organize the coordination of communication networks between distributed devices. In these projects the main focus of agents is the adaption and the optimization of program planning of the production at runtime and not increasing the system reliability regarding the real-time requirements. Agents are not applied for neither open nor closed loop control purposes in the field control level.

As in the object-oriented approaches, modularity support and small interfaces are prerequisites for agent-oriented software development. Therefore, object-oriented approaches are favored as a basis for agent-oriented design but needs to be enriched with role modeling and, for deliberative agents, modeling of a knowledge base. In contrast to reactive agents, who decide on their actions by a set of rules, deliberative agents make their decisions based on an explicitly formulated model of the controlled system. Within these models, the basis for concepts like virtual sensors to enhance the reliability of a plant can also be integrated. Due to real-time requirements, additional concepts are required to model the time constraint for negotiation between agents and to restrict an agent’s solution space (Mubarak et al., 2007).

3. “META-MODEL” OF AN AGENT SYSTEM

In the following, we give a brief overview on the modeling approach which simplifies the handling of the complexity of an agent system. A more detailed application example will follow in section 4 of this paper.

The modeling approach we are using is based on the Systems Modeling Language (SysML). To reduce the complexity of the overall agent system, we separate the model system into four major aspects, i.e. the technical process, the technical system, the automation system and finally a service layer.

Fig. 1: Agent system in SysML

The first three views are derived from the definition of these terms in (Lauber and Göhner, 1999). The “service layer” connects the view “technical process” with the views “automation system” and “technical system” (Fig. 1). It consists of all services that are offered within the agent system. The modeling approach uses the SysML element “interface” for modeling a service and containing an “operation” to declare the function an agent offers by a particular service. Technical processes are encapsulated by “blocks” that describe process agents. Likewise the technical components of a plant are encapsulated in blocks for resource agents. These blocks have a reference, i.e. a pointer, to the block that describes the resource an agent controls.

3.1 Technical process view
As mentioned above the technical process view can be fully encapsulated by a process agent, i.e. an agent responsible for the technical process, modeled by blocks. The process agent’s functionality is described as the behavior of the particular block in an Activity Diagram (ACT) by sequencing and/or parallel “activities”. These activities are bound to the operations that are contents of the service interfaces. An Activity Diagram describing a process, can therefore be modeled by defining in which sequential or parallel order different services of the resource agents should be executed. Requirements regarding the process steps can be modeled using the language element “requirement”, bound to an activity as a “nested classifier”.

3.2 Service layer view

As mentioned, services of the agent system are modeled using the element interface of SysML in Block Definition Diagrams (BDD) describing the service layer view. As provided in the paradigm of object orientation in this approach too, an interface represents an abstract piece of structure and behavior that can be implemented by an entity of a modeled system. An operation contained in an interface is also declared in an abstract way by its name and its inputs and outputs. If an operation of a service interface is used by a process in an activity, to which requirements are bound, it is demanded that there will be a corresponding “value property” with a defined data type contained by the service interface. These value properties describe the boundary values the parameters of a block have to comply with. By modeling that, the block of a resource agent implements an interface service, it is implicated that this agent offers the service within the agent system. Additionally, by implementing a service interface, an agent also has to implement the value properties (boundary values) that are bound to this service.

3.3 Technical system view

The technical system view describes the technical structure and the behavior of a plant by hierarchically modeling its components and their interconnections. The top-level of the module hierarchy is given by the blocks that are controlled by the resource agents. These blocks are modular compositions starting from the sensors and actuators (field level) in Block Definition Diagrams (BDD). In addition to the boundary values, a resource agent may contain by implementing a service interface, further boundary values can be modeled for limitations of the technical system. Modular approaches to model automated process and manufacturing plants using the SysML were presented in (Schütz et al., 2011). These also contain the modeling of analytical dependencies in SysML Parametric Diagrams (PAR). In the approach presented in this paper, the Parametric Diagrams are used to model a part of the agents’ knowledge base.

3.4 Automation system view

The automation system view consists of the sensors and actuators of a component controlled by a resource agent. Being technical components, the sensors and actuators are modeled also within the technical system view. Therefore the information on the actuators and sensors a plant component contains can be derived from that view to be extended by virtual sensors within the automation system view. The semantics which we use to model the automation system view are based on the semantics of the Directed Graphs (DG) introduced by (Wannagat et al., 2008a). Each node of a graph represents a component, e.g. a sensor or an actuator, and each edge indicates an analytical correlation between them. Based on these correlations, virtual sensors can be modeled and implemented. These virtual sensors can be activated by a software agent in case there is no valid measurement available.

4. APPLICATION EXAMPLE

In this section we describe the procedure of a model-based implementation of an agent system using diagrams and notations based on SysML. As a demonstrator we have chosen the Hybrid Process Model (HPM), a hybrid laboratory plant which consists of processing technology parts as well as manufacturing technology and logistic parts. For the process engineering part, a production scenario can be implemented, consisting of handling a fluid with different process steps (heating, circulating, etc.) in a central processing workstation “MPS” before it can be filled into bottles together with different kinds of granulate at two filling stations.

In our application example we focus on the MPS of the processing engineering part and the process function circulating for a fluid. Figure 2 shows the reduced P&I-diagram of the MPS and the sensors and actuators that are used for the realization of this function. A certain volume of a fluid between $LS^- / I_{12}$ and $LS^+ / I_{13}$ will be circulated between the tank $T101$ and the tank $T102$. In the following we describe how an agent, that realizes this function within the software, can be designed with our method in SysML and how the model of the agent can be mapped to real-time control code for IEC 61131 runtime systems.

The subsections follow the numbering of the steps of the procedural model we take as a basis for the development of a real-time agent system in production automation (s. Figure 3). The procedural model proposes a successive development and covers all 4 major views of an agent system we introduced in the previous section. The steps No. 1 to No. 4 represent the modified classical procedure for the modular composition of real-time control software. Therefore only the steps No. 5 to No. 7 have to be considered as additional effort compared to the classical software development approach.

![Fig. 2: P&I-diagram of Hybrid Process Model (out-take)](image-url)
Due to the fact that we start our application example with an empty model of the agent system, the steps No. 1 to No. 3 as well as step No. 7 have to be accomplished. For adding a new production process into the control software of a production system, only the steps No. 4 to No. 6 (circles with solid line in Figure 3).

4.1 Modules of the Automation System

In the automation system view standard IEC 61131 function blocks (FB) represented by a SysML block (s. Figure 4) for the implementation of real sensors as well as for actuators have to be used from a library. The blocks for sensors will be featuring to modes: a hardware mode and a software mode.

In the hardware mode the sensor blocks will deliver the value measured by the real sensor to their output variables. In the software mode the sensor blocks will deliver a virtual value, read from a pointer address "sw_sens", to their outputs. The data type “T_RedValue” of the outputs is a defined data structure containing a parameter for the sensor value itself as well as a precision parameter that indicates the accuracy of a sensor. By calling operations “set_hard” and “set_soft” of this sensor blocks and setting the pointer address, an agent is able to switch a sensor between hardware and virtual mode.

4.2 Composition of the Technical System

Like the IEC 61131 function blocks for sensors and actuators, function blocks in general will be represented by SysML blocks. Thus, a modular composition of the control software will be enabled inside Block Definition Diagrams (BDD). For building an instance of a function block within another one, composite aggregations can be used inside a BDD. The Block Definition Diagram in which the composition of a block for the tank T102 is modelled as well as out-takes of its FB in IEC 61131-3 is shown in Figure 5.

With the model in this BDD, a function block “FB_Tank” with one analogue and one binary sensor for the filling level of a tank (compare T102 in Figure 2) is declared. Like the block for the tank, a block describing the complete processing workstation (MPS) can be composed modularly within Block Definition Diagrams. Afterwards, the resource agent controlling the MPS can be modelled as a SysML block “FB_Agent_MPS”, which references the SysML block declared for the MPS.

4.3 Modelling the process functions on the Service Layer

Due to the reference / pointer a resource agent contains to the block of a technical resource, inside the resource agent for the MPS of our application example it is possible to call the operations from the components of the MPS, e.g. pumping. In our application example the resource agent “FB_Agent_MPS” is able to switch the sensor blocks of the MPS between the hardware and the software mode. From a generic block for a resource agent, this agent adopts two operations: the operation “directed_graph” and the operation “precision_tree”. The implementation of these operations is described in the subsections 4.6 and 4.7 of this section.

Fig. 3: Procedure model for design of agent system

Fig. 4: Standard block (function block) for binary sensors

Fig. 5: BDD for the block of tank T102

Fig. 6: Modelling a resource agent

The SysML block for the resource agent also represents an FB. The reference in the SysML block can be translated to a pointer of the FB. By such a reference, a resource agent is able to access the operations, parameters and components in the software, i.e. the FB, of a technical resource (s. Figure 6).

Thus, the agent “FB_Agent_MPS” is able to switch the sensor blocks of the MPS between the hardware and the software mode. From a generic block for a resource agent, this agent adopts two operations: the operation “directed_graph” and the operation “precision_tree”. The implementation of these operations is described in the subsections 4.6 and 4.7 of this section.

4.3 Modelling the process functions on the Service Layer

Due to the reference / pointer a resource agent contains to the block of a technical resource, inside the resource agent for the MPS of our application example it is possible to call the operations from the components of the MPS, e.g. pumping. In our application example the resource agent “FB_Agent_MPS” implements a service interface that contains an operation for the function “circulating” (s. Figure 7).

Operations in the SysML model will be translated to actions the IEC 61131 code. Thus, the operation “circulating” of the block “FB_Agent_MPS” can be implemented as an action for example in the language Structured Text using the pointer to access the actions of the sub-components like the pump.
4.4 Describing the Technical Process

Like mentioned in chapter 3 of this paper, technical processes are modeled as the behavior of process agents that are described by a SysML block. These blocks, like resource agents, represent function blocks in the IEC 61131 code. It is planned that the activity diagram (ACT), which describes the behavior of a process agent will be available as an implementation language for SysML blocks / function blocks.

For our application example we assume that a technical process is implemented in an activity diagram of a process agent that uses the operation “circulating” allocated by the agent of the MPS. Operations can be used inside an activity diagram by describing an activity that calls an operation from a service interface. We further assume that there is a process requirement regarding the volume of fluid that has to be circulated. This requirement is described in natural language and bound to the activity that calls the operation from the service interface.

4.5 Implementing the process functions and constraints

Our approach demands, that if requirements are bound to an activity that calls an operation of a service interface, this service interface has to contain a corresponding parameter by a composite aggregation (s. Figure 9).

This parameter expresses the value a resource agent has to calculate for the evaluation if the requirement is satisfied. Thus, in our application example the service interface for the circulating of the fluid “S_Circulate” has to contain a parameter of a defined data type that expresses the minimum and maximum volume of fluid that is actually possible to be circulated by the workstation. The resource agent of the MPS that implements this interface does also implement this parameter as well. In the IEC 61131 code the function block will have a variable with a similar data type.

4.6 Evaluating the Technical System

The evaluation of a technical resource is done by the implementation of the operation “precision_tree” of a resource agent. Based on concept for a Fault-Tree-Analysis proposed by (Wannagat et al., 2008b), in this operation the ability of a resource agent to satisfy the process requirements can be evaluated. Due to the fact, that process requirements will also be represented by corresponding parameters of a resource agent, the evaluation consists of the calculation of these parameters inside the implementation of the operation “precision_tree”. Like any other operation modelled in the SysML model of an agent system, this operation will be translated to an activity that can be implemented in the languages specified in the IEC 61131-3.

Additionally it will be possible to implement this operation by modelling a parametric diagram following the semantics of this diagram proposed by (Schütz et al., 2011). Figure 10 shows the parametric diagram that implements the calculation of volume fluid that can be circulated by the processing workstation in our application example as well as the translation of this diagram to the IEC 61131 language Structured Text. Based on the precision values of the sensor blocks for the filling level of tank T102 the parameter “vol1” that corresponds to the process requirement is calculated. If for example the precision parameter of the upper filling level sensor has a low value, the probable volume of circulated fluid will change accordingly.

4.7 Extending the Automation System

In step No. 7 of the proposed procedural model virtual sensors are added to the resource agents. The modelling of virtual sensors as mathematical functions of the values of real sensors is based on the semantics of the directed graph suggested by (Wannagat et al., 2008a) for this purpose. For modelling this directed graph inside a SysML model we extended the notations of the parametric diagram as well as the meta-model of the SysML. Figure 11 shows the directed graph for the processing workstation MPS from the
application example. It represents the implementation of the operation “directed_graph” of the resource agent.

Fig. 11: Parametric diagram for Directed Graph

The constraint block “FB_VS_FL” represents an IEC 61131 function block that can be used as a virtual sensor for the three filling level sensors of the workstation (LA-, LS-, LS+).

5. CONCLUSIONS

Agent-orientation enables the changeability, scalability and reliability as integrated system features with simultaneous reduction of the system complexity. Systematical and tool-supported software development methods can advance the engineering process additionally. Unfortunately, from the wide variety of the existing AOSE-methods none has a significant impact for industrial applications on the field level. We introduced a novel AOSE-method based on SysML, developed for the agent based automation of process and manufacturing plants. This method follows an agent framework that was previously developed and further, allows the implementation of software agents directly on IEC 61131 systems.

To fulfill the main challenges mentioned in section 2: firstly, the seamless integration into standard engineering processes, secondly the adequate support of application engineers to model the agent based system and to support automatic code generation for PLCs, we are currently working on a tool support for the model based development of agent systems. The integrated tool will allow application engineers to develop agent systems for process and manufacturing automation systems by using the presented modeling approach. The control software will be generated by mapping the different parts of a modeled agent system onto generic software components, which are abstracted from the implementation framework and developed in preliminary work. Extensions of the generic components can either be programmed manually using the languages standardized in the IEC 61131-3, or generated automatically from behavior diagrams, like state charts or activity diagrams, that can be part of the agent model. The automatic generation of control code from UML behavior diagrams, integrated into a development tool is another preliminary work of our project partner, the chair Automation and Information Systems (AIS) from Technische Universität München (Witsch et al., 2010).

ACKNOWLEDGEMENT

The authors gratefully acknowledge the financial support of the German Research Foundation (Deutsche Forschungsgemeinschaft DFG) for the project KREAagentuse (Design, realization and evaluation of a tool-supported approach for the development of agent systems in automation, considering the usability, VO 937/8-1).

REFERENCES


