Proceedings of the 17th World Congress
The International Federation of Automatic Control
Seoul, Korea, July 6-11, 2008

Agent oriented software-development for networked embedded systems with real time and dependability requirements in the domain of automation

A. Wannagat*, B. Vogel-Heuser*

** Chair of Embedded Systems, University of Kassel,*
Kassel, Germany, (e-mail: wannagat@uni-kassel.de, vogel-heuser@uni-kassel.de).

Abstract: A method for integrated development of multi agent PLC based control systems using IEC 61131-3 will be introduced. Dependability of technical plants will increase if control behaviour can be adapted during runtime. This is achieved by dynamic reconfiguration of faulty devices, e.g. sensors, at run-time. The replacement is based on analytical redundancy that is represented by a network of sensors and their interdependencies. This method supports the developer to sketch a suitable modular software-architecture in regards to the hardware-architecture and the specific requirements of real-time and dependability. The integration of classical methods like fault-tree-analysis and graph-theory enables the developer to build a knowledge-based system so that the agents would have the ability to recognize faults, seize the needed actions within their scope and fulfil their goal. Recent investigations using a model-plant showed the feasibilities of this approach.

1. INTRODUCTION

The requirement of highly flexible and reconfigurable systems is growing due to rising cost pressure, shorter product life cycles and mass customization (Harrison and Colombo, 2005). Correspondingly the industry has a growing interest in flexible, scaleable, reusable, adaptable and highly reliable systems.

Due to the power of modern controller hardware it is possible to make even very complex software based decisions at runtime. This flexibility can be used to react on failures, in order to increase the reliability or to improve the efficiency by adapting to changing requirements. These strategies are usually specified during the engineering process to ensure the predictability of the system. The consequence is a static software design, where additional flexibility leads to a complex software design implementing all possible combinatorial operation sequences.

A well suited approach for developing decentralized, complex software systems with high flexibility is the paradigm of agent oriented software engineering. In agent oriented software, development of an agent is defined as an encapsulated software unit with a defined goal. An agent autonomously fulfils its goal and continuously interacts with its environment and other agents (Wooldridge and Jennings, 1995). To achieve its goals and to develop controlling strategies at run-time, agents need knowledge about the process and the automation system. If a component fails, the strategy has to be adapted. The agent must be able to detect the failure and be aware of the impact on the system, as well as to reconfigure the system dynamically.

The difficulty in developing an agent system is to define the action space of an agent precisely enough to ensure both, the reliability of the system and the fulfilment of the required performance and product quality. However neither methods nor tools that are adapted to design agents for industrial real-time applications exist (Mubarak et al., 2007).

The lack of suitable methods for industrial applications is actually recognized by several working groups. The national project AgentAut (Lüder et al., 2006) as well as the European projects Pabadis (Klemm and Lüder, 2003) and 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 (Socrades) and RI-MACS (RI-MACS) use agents to organize the coordination of communication networks between distributed devices. Agents are not applied for open or closed loop control purposes in the field control level. In all these projects the flexibility of agents is primarily used to realize an optimized planning of production program at runtime and not to increase the dependability of the system regarding real time requirements.

Due to these reasons, the development of flexible agent oriented real-time systems is highly desirable but still a challenge. The fulfilment of requirements like real-time, dependability and flexibility needs a systematic development method that considers all aspects of the system to be designed.

2. DESIGN ASPECTS FOR AN AGENT BASED SYSTEM

The approach is subdivided into four major aspects, i.e. architecture, requirements and boundaries, diagnosis and fault management, and knowledge base. These four aspects are substantial for the whole engineering process, which is separated into to main phases. The first phase is intended to analyse the given system, subdivide it into modules,

collecting the relevant requirements and capturing the analytical dependencies between both, the requirements and the modules.

The second phase is intended to build a suitable structure of the software, designing the behaviour of modules and agents as well as creating a knowledge base for the agents.

To reduce the complexity and enhance the reuse of the engineering process, we examine the system under three major views (according to Lauber and Göhner, 1999) in all steps of the proceeding. The technical system, which describes the mechanical parts of a plant, the automation control system, which includes controllers, networks, sensors and actuators, and the technical process itself, which describes the manufacturing of the product. The advantages of these different views are reduction of complexity for the design, and the opportunity to modify each view separately from each other without affecting the other views. Nevertheless, these are all views on the same system and most modules will have more than one view.

This separation into three views fits nearly perfectly to an agent based approach. Agents uses and offers functionalities by negotiation. One agent type represents the parts which have to be produced and uses the functions (skills) the other agent types offer. The first type of agent needs information about the technical process, to know what are the required functions as well as the necessary sequence to produce the desired product. The second type of agents offers functions that belong to modules of the underlying technical system. The challenge for the engineer is to subdivide the functions of the technical process in such detail until they can be fulfilled with a combination of the technical system functions. As well as the recipe of the technical process the technical system can have one or more different possible sequences to manufacture the product.

However, this coordination is bound to many requirements regarding the real time behaviour, logistical aspects and the reliability of a production system. According to this, it is important to specify exactly these aspects during the design phase and to create the responding behaviour boundaries (activity space) for the agents.

The collected requirements during the analysis phase are the base for the development of the agent system. The engineer design a suitable modular architecture based on the technical system with all its' relevant requirements. The flexibility of an agent based system depends on the boundaries of the activity space. These aspects are substantial for the knowledge base used to enable the agents in fulfilling their goals and reacting flexibly to disturbances. The agent must be able to estimate the influence of its action on the system at any given time. This is a prerequisite for reliability and the performance of the system, because an agent must know when he is going to violate requirements or when there is no solution for further operation within the given boundaries. In the latter case the technical process has to be stopped.

All chosen types of diagrams are similar to existing domain specific notations and easy to devolve them into PLC Programming suites for example. The diagram for the automation system can easily be used to configure a PLC, the specific description of behaviour based on state-machines can automatically be transformed into sequential function chart of the IEC 61131-3. (Vogel-Heuser, 2005)

2.1. Architecture

The basis of the software architecture for the agent is the model of the whole system, which is subdivided into three views. The goal is to reach a modular structure, where every module is functional and structural united. The structure is mainly oriented at the technical system and consists of sensors and actuators of the automation control system, residing at the most bottom level. The functionality deduced from the requirements of the technical process has to be fulfilled by one or a combination of modules of the technical system. At runtime the agents will use the modules' functions to observe and act on the technical process. The chosen diagrams base upon Internal-Block and Activity Diagrams of the SysML (Hause, 2006). SysML (Systems Modelling Language) base on UML and is a domain-specific modelling language for systems engineering applications.

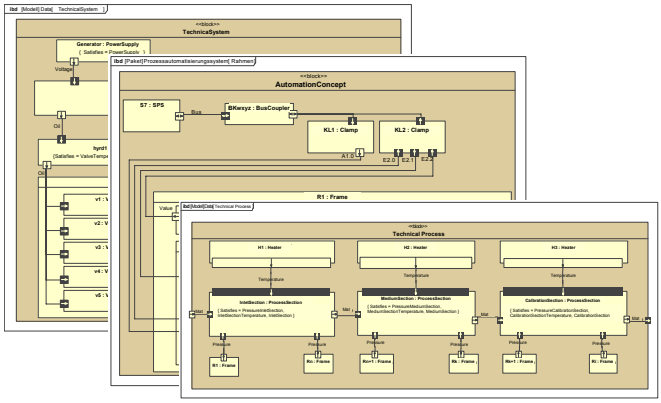

Fig. 1. Different views on a system (see fig. 7, 9, and 10 for details)

Because of real-time-requirements, agents are not used on the lowest level, i.e. the actuator and sensors level, in this approach. An agent-typical negotiation to compensate failures of sensors or actuators is not possible within a single PLC-cycle of some milliseconds. Consequently there are no agent-typical mechanisms on this level. Instead a uniform module-structure, which allows the diagnosis and messaging of failures and as a consequence the definition of measures to compensate failures, has been developed.

In some cases both the diagnosis of a failure and the measure are not constricted to the same module. Fault-Tree-Analysis (FTA) is used as a method to define the interfaces that are necessary to communicate occurring failures.

2.2. Requirements and Boundaries

The requirements are the basis for specifying the action space for the agents and to define their goals as well as the parameter to achieve them. The survey of these requirements

is separated into three steps, based on the same three views mentioned above (fig. 3).

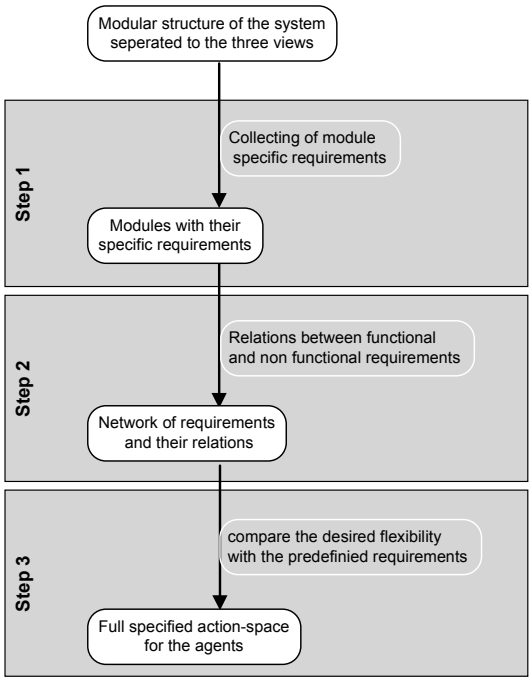

Fig. 2. Requirements: From Analysis to Action Space

In the first step, the requirements related to contained modules, interfaces and connections are collected separately for each view. Unlike the modules, which can be related to more than one view, these requirements are strictly related to their view. The same module can be viewed under different aspects. For example a valve is an actuator for the automation control system; it has a mechanical representation in the technical system and it controls the flow rate from the technical process view. This leads to two advantages: firstly, it reduces the complexity because the requirements survey is separated to three views for each element and secondly, it is the first connection between the requirements of different views, when they are related to the same element.

In the second step, relations between functional and non functional requirements (first step) need to be analysed and linked to each other using the requirement and the parametric constraint diagram of SysML. The predefined functions of the modules are linked to the appropriate requirements and boundaries regardless of the three views. The result is a network of requirements and their relations, which makes it easy for the developer to get an overview of functionalities, requirements and boundaries.

In the third step, the developer evaluates if the desired flexibility may be reached with the predefined boundaries and specifies how much flexibility the agents should have. According to the action space, which allows the agents to navigate and to achieve their goal, its structure will be defined within the defined boundaries. Every parameter has to be checked regarding its influence on the time delay according to real time requirements and the failure probability of its related functionality regarding dependability. Using these relations the agents are able to achieve their goal by changing the relevant parameters dependently in relevance to the possible side effects.

2.3. Diagnosis and Fault management

To fulfil the requirements regarding the reliability of the plant, the agents have to know the effect of their actions as well as to estimate the significance of a changing environment regarding their requirements within the whole system.

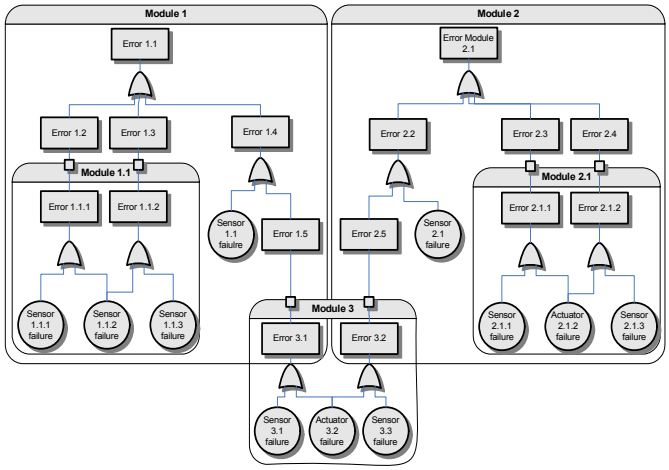

Fig. 3. Failure tree combined with the module structure

On the basis of a Fault-Tree-Analysis (FTA, Vesely, 1981) and the same modular structure as used before for the software architecture, the developer is able to specify the relationship between the functionality of a module and its sub modules. The goal is to specify the probability for correct execution of each function based on functions of the related subsystem until it reaches the basis elements of the automation control system at the bottom level. In this way it should be possible to calculate the probability of a malfunction when the quality of a sensor measurement changes. Finally, the knowledge about the relation between the quality of an automation control system element, e.g. the quality of the sensors' measurement, and the precision of the actuators' action, and the functionality of a module is implemented in the agents. This relation is used to calculate both the risk of a failure regarding the observed changes and the effect of possible counteractions, i.e. replacement of a sensor by a virtual (calculated) sensor value. Each agent is able to observe all elements of the automation control system and to relate the real values to calculated values of its internal system model. It is able to detect faulty elements and also to calculate virtual sensor values in case of failure. The second mechanism is based on a distributed detection of failures. It is a matter of fact that an occurring failure and the primary cause are not always in the same module. Therefore a failure-interface to send messages to corresponding modules is implemented in each module. The failures that needs to be communicated are defined by the cross cuts between the branches of the failure tree and the capsules of the modules.

Knowledge base

The knowledge base is an essential part of agents and includes a model of the environment as well as requirements and boundaries. While the latter aspects describes and structures the action space, the system model (fig. 5) is used to realize dynamical redundancy. Static redundancy needs a redundant component, which can be used in case of failure. In our approach a so-called dynamic redundancy is realized, which is based on a software approach using still existing sensors with compensation exemplarily explained in the following. In case of a sensor failure, analytical dependencies of the model are used to calculate estimated value at runtime. Katzke and Vogel-Heuser (2005) introduced auto reconfiguration service, which is a concept of UML-PA (UML for process automation, a UML profile) and based on a reconfiguration class using the mechanisms of inheritance and overriding. Using knowledge based approach, this virtual sensor can be used like a physical one even if its characteristics, e.g. precision, would have changed. Dynamic redundancy using agents is a substantial instrument to increase the availability by using such a mathematical model as part of their knowledge base. It can also be used to detect failures when real measurements are compared continuously with calculated ones. It is possible to detect the defect sensor by comparing it with other values. This virtual sensor values can be used to substitute real sensors at runtime to increase the availability of the whole system.

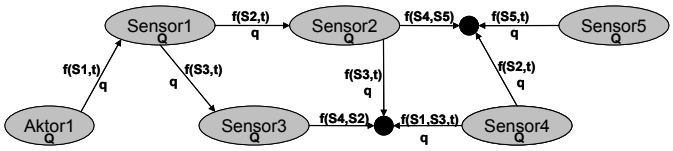

Fig. 4. Analytical dependencies of sensor and actuator values

These mathematical dependencies are part of the knowledge base and can be modelled using a directed graph (Chartrand). Each node represents a component, e.g. a sensor or an actuator, and each line has functional correlation between them and can be used to calculate a virtual sensor value in case if there is no valid measurement available. Because of the substitution of real nodes with virtual ones, the structure of the graph will not change and it is possible to calculate values even on base of virtual sensors as long as no algebraic loops occur. Expecting that these estimated values are less precise than the original ones, the loss of quality is represented by a factor q at each connection between two nodes. Additionally every sensor has a quality value that represents the precision of measurement. The quality value of a virtual sensor is calculated by multiplying the quality value of the source sensor and the quality factor of the connection. Depending on the new quality-value the agents can plan its actions according to their requirements.

3. EVALUATION

The introduced concept was evaluated by two applications, a sorting machine (lab, Wannagat, et al., 2007) and a continuous thermo-hydraulic press, which is a real industrial application. Fig. 6 shows the example of a continuous fibre board press application. It is composed of up to 80 separately controlled frames. Each frame consists of 5 separately controlled cylinders with sensors for pressure and distance. The sensors and actuators are linked via a field bus to four PLCs, which are the runtime environment of the control software. One PLC may control between 10 and 20 frames. The failure of one of the components results in failure of the entire control chain.

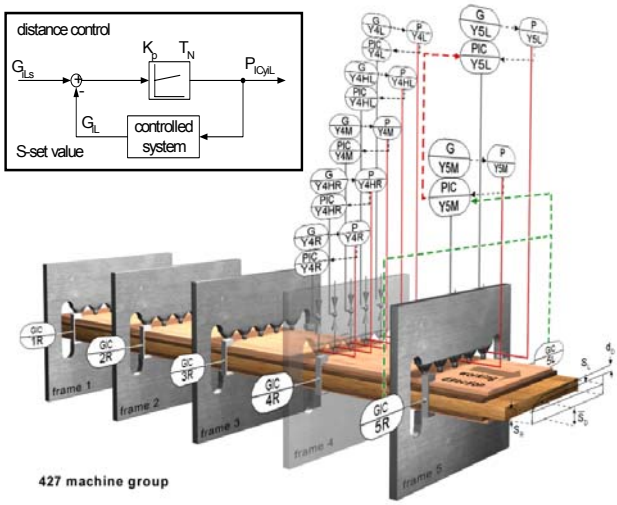

Fig. 5. First identified entities of the sample application

The requirements of the technical process (fig. 7) are modelled using internal block diagram which is part of the SysML. It shows the three sections of a continuous thermo-hydraulic press from a material point of view. The mat is fed into the press on the left side and will be heated and pressurized by the inlet section with high temperature and pressure. The medium section is characterized by lower pressure and lower temperature, while the glue in the mat starts to harden. The final or calibrating section is in charge of producing a proper surface with low pressure and low temperature.

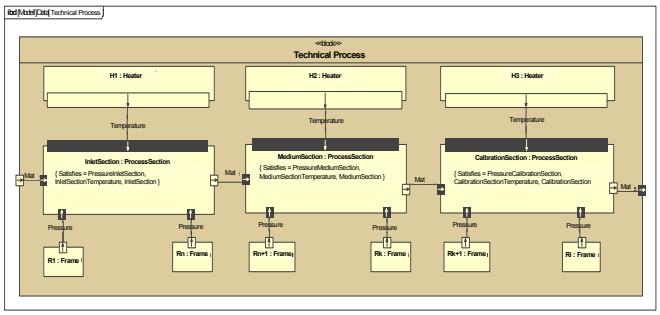

Fig. 6. Internal block diagram technical process

From the initial description an activity diagram (fig. 8) of this process is designed showing the three sections and the process steps and its boarders represented by swim lanes.

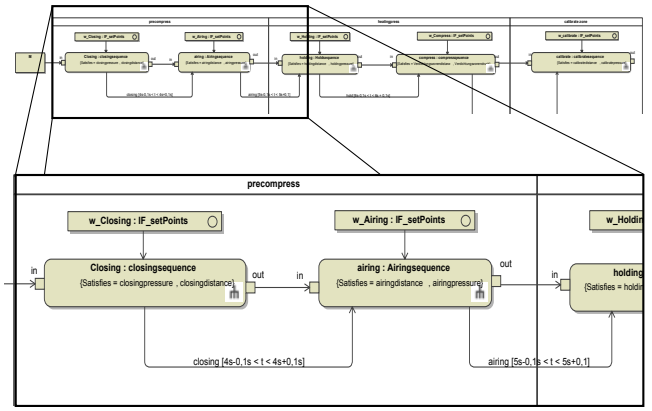

Fig. 7. Activity diagram technical process

The technical system consists of a generator and the hydraulic system and its interface to the mat (technical process view) the hydraulic main valve and the five valves and pressure cylinders of each frame. The particle board mat is pressurized by these cylinders.

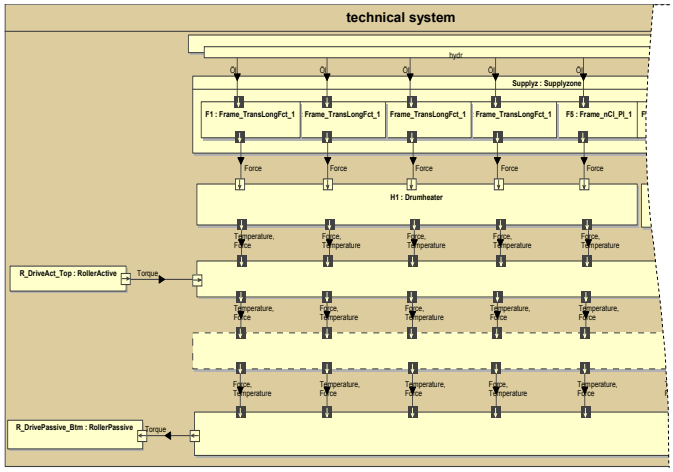

Fig. 8. Internal Block diagram technical system

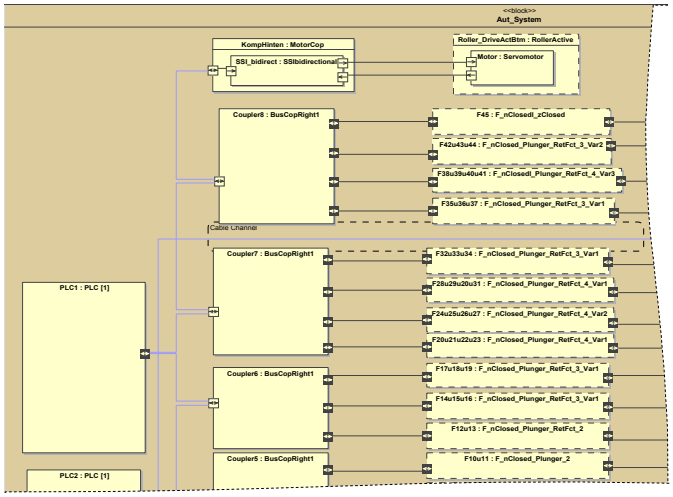

Fig. 9. Internal Block diagram automation concept

The automation control system (fig. 10) represents the chosen automation concept in this case with a PLC (S7) connected via a bus coupler to the input and output connectors of the single frame. The distance sensor and the pressure sensor as input value and the proportional valve with its set value as out value and its position value as input as well. The generator (bottom right) shows the connection to the technical system in dashed line (because the generator is not part of the automation control system, it is coming from another view). By representing the most important links between different views using dashed lines the usability and understanding is increased. This presentation supports the awareness of the interfaces and links, and interfaces to the other engineering disciplines. At first an example of requirement being influenced by different disciplines would be discussed regarding an actuator failure in the press application example. Secondly a failure of a sensor will be introduced with its impact on the dependability.

The material in the middle zone of a continuous press has to be pressed for 20±2 seconds (given a constant speed of the material through the press) by 220°C±1°C and 150±2 bar/cm². The agents have to meet these process related requirements (PTP) with the given precision when they control the transport speed, the temperature or the pressure. If the engineer is able to find and describe dependencies between these boundaries and the product quality then the agent can use these dependencies to find an appropriate operating point. In case of an actuator failure, e.g. hydraulic pressure sub system frame 2 (F2), the pressure would decrease (dot-dash-dot line) and leave the allowed tolerance band between PTPmax and PTPmin at frame 2 (white band). This would lead to the stoppage of the whole line (fig. 11).

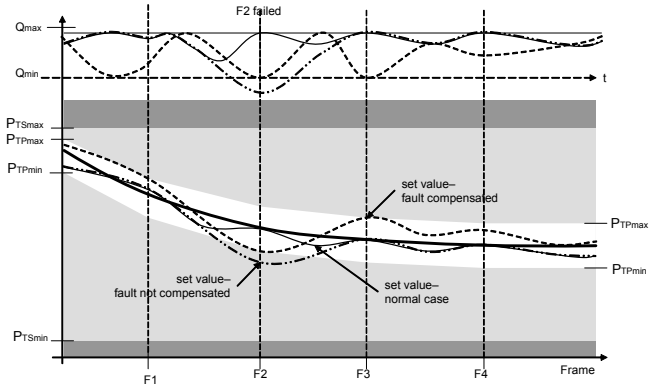

Fig. 10. Compensation of an actuator failure by the agent (Quality (Q), Pressure technical system (PTS), Pressure technical process (PTP))

The agent approach tries to compensate the actuator failure of frame 2 by increasing the pressure in frame 1 and 3. Due to the heated plate along the press there will be a resulting pressure in frame 2, which is above the necessary minimum (PTPmin at frame 2) and inside the allowed tolerance (white area) to achieve the panel quality. The strictest requirement of the different views regarding this aspect has to be covered by the according agent and is the boundary of the agent activity space. The white area in figure 11 is the allowed tolerance band for product (panel) quality, i.e. process view. The light grey area is the tolerance for machine security, i.e. technical system point of view (PTSmin and PTSmax). The pressure restrictions of the automation control systems view (PACmin and PACmax) are not included because they are less

restrictive, e.g. the necessary current for the valves to reach the pressure is available.

The network of requirements enables the agent to acquire if the plant is still operating in an acceptable state (automation control system and technical system) and whether the product quality is in the required tolerance band or not. The discussed application example showed that in case of an actuator failure the plant can still operate with the required panel quality. By that the dependability of the plant is increased.

The benefit of an agent oriented compared with a classical approach is its behaviour adaptation during run time on basis of the knowledge base (fig. 5) and under the constraints of dependability requirements (fig. 3). The benefit is a result of the additional knowledge included during run-time. If it is necessary to predefine all measures and their parameters (min., max.) before run-time, worst case scenarios will be used and according parameters chosen. The agents' knowledge base and mechanism allows reaction based on changes in environmental conditions and by that changed precision of sensors and/or actuators during run-time, by adapting on the actual values and not on the worst case values with the appropriate changes. The process operation time will be longer under the prerequisite that the process operation is still beneficial with reduced precision, speed etc. This leads to higher dependability of the production line.

4. CONCLUSION

The developed method is conceptual universal and includes domain-specific modelling languages from the analysis phase to the implementation. The combination of the modular and agent-oriented approach supports standard automation devices, e.g. PLCs. Focussing on embedded systems in automation there are strict constraints concerning operating systems and programming languages, i.e. for factory automation PLC based hardware structure (Hard or Soft PLC) with IEC 61131-3 as programming language and proprietary RTOS. An agent oriented approach needs to be implemented in this environment. The agents have to be executable on standard industrial controllers that are often distributed and connected by field busses. This structure neither represents a uniform platform nor allows it an unrestricted communication between agents. The same applies to the use of resources of controllers and field bus by the agents. The process control tasks use a large part of the limited resources and shall not be affected by the additional activities of agents.

REFERENCES

Chartrand, G. (1985). Directed Graphs as Mathematical Models. In: *Introductory Graph Theory*, pp. 16-19. Daver, New York.

Harrison, R. and A.W. Colombo (2005). Service-oriented architectures for collaborative automation. In: *Industrial Electronics Society*, IECON.

Hause, M. (2006). The SysML Modelling Language. *Fifth European Systems Engineering Conference*, Edinburgh, 2006.

Katzke, U. and B. Vogel-Heuser (2005). Design and Application of an Engineering Model for Distributed Process Automation. In: *American Control Conference*, pp. 2960-2965, Portland.

Klemm, E., A. Lüder (2003). Agentenbasierte Flexibilisierung der Produktion bei Verwendung von vorhandenen Steuerungssystemen. In: *atp – Automatisierungstechnische Praxis*, **Vol. 45**.

Lüder A., J. Peschke, R. Sanz (2006). Design Patterns for Distributed Control Applications. *atp international*, **Vol. 3**, pp. 32-40.

Lauber, R. and P. Göhner (1999). *Prozessautomatisierung 1*, Springer Verlag, Berlin.

Mubarak, H., P. Göhner, A. Wannagat, B. Vogel-Heuser (2007). Evaluation of agent oriented methodologies. In: *atp international*.

Peschke, J., A. Lüder, H. Kühnle (2005). The PABADIS'PROMISE architecture - a new approach for flexible manufacturing systems. In: *Industrial Electronics Society (EFTA 2005)*, pp. 491-496. Catania.

RI-MACS (Radically Innovative Mechatronics and Advanced Control Systems). *European Project: FP6 NMP-IST Joint Call 2*.

Socrades (Service-oriented cross-layer infrastructure for distributed smart embedded devices), *Integrated Project, European Commission, Information Society Technologies*, Frame Programme 6, 2006–2009.

Vesely, W. (1981). *Fault Tree Handbook*. NUREG-0492, Nuclear Regulatory Commission, Washington DC.

Wannagat, A., B. Vogel-Heuser, H. Mubarak, and P. Göhner (2007). Bestimmung automatisierungstechnischer Anforderungen bei der agentenorientierten Entwicklung flexibler eingebetteter Echtzeitsysteme. In: *Automation im gesamten Lebenszyklus,* GMA-Kongress.

Vogel-Heuser, B., U. Katzke, D. Witsch, D. (2005) Automatic Code Generation from a UML model to IEC 61131-3 and system configuration tools. *5th International Conference on Control & Automation (ICCA)*, Budapest

Wooldridge, M.J. and N.R. Jennings (1995). Intelligent agents: Theory and practice. *The knowledge Engineering Review*, **10(2)**, pp. 115-152.