Increasing Flexibility and Availability of Manufacturing Systems - Dynamic Reconfiguration of Automation Software at Runtime on Sensor Faults

Andreas Wannagat, Birgit Vogel-Heuser

Abstract:

This proposal introduces a conceptual design of self-adapting system software to manage sensor failures in factory automation. The approach reconfigures the arrangement of software modules in real time to preserve the required stability of production processes without interrupts. Reconfiguration will be decided by rules from a knowledge base system. This paper discusses conventional, object oriented and agent based concepts, and focuses on modelling of these concepts. For discussion purposes, a real industrial application - a continuous thermo-hydraulic press will be presented as application example.

Keywords: *agents, programmable logic controllers, industrial production systems, availability.*

1. Introduction

Industrial production systems have very high requirements regarding their robustness against defects and failures. The production process shall not be interrupted or even hampered.

This paper describes an agent-based approach that reconfigures automation software at runtime to compensate failed sensors and actuators. It uses physical dependencies between process values in a production environment to install virtual sensors and it reconfigures the system input to an optimal presentation of the current process state. In plant automation, software systems are being used to control technical systems with sensors and actuators as input and output devices. Apart from office environments the surrounding of such field devices is very rough. Humidity and heat are influences that shorten the lifetime of sensors. Expensive and unwelcome downtimes of plants are the consequence of such failures. In case of physical defects of a sensors or an actuator, e.g. a drive, human intervention is not possible without expensive delays. In many cases, continuing the production process with some restrictions until the next scheduled service is preferable than interrupting it for unplanned repair work.

In case of sensor faults, defective devices can be bridged by reconfiguring the affected control loops at runtime. Sensors and actuators pose as interfaces between technical processes and the control equipment. Some additional sensors may be installed for monitoring purposes. Any faulty sensor, which is used in a control cycle, will results in uncontrolled behaviour, unless alternative solution is being used.

2. Required changes at runtime

Industrial automation can be classified as production technology and process technology, which is further subclassified as batch processing and continuous flow processing [2]. Consequences of unplanned production stops have different impact on different categories. In production systems such interrupts are unwelcome but they are not critical, as the treated material can remain

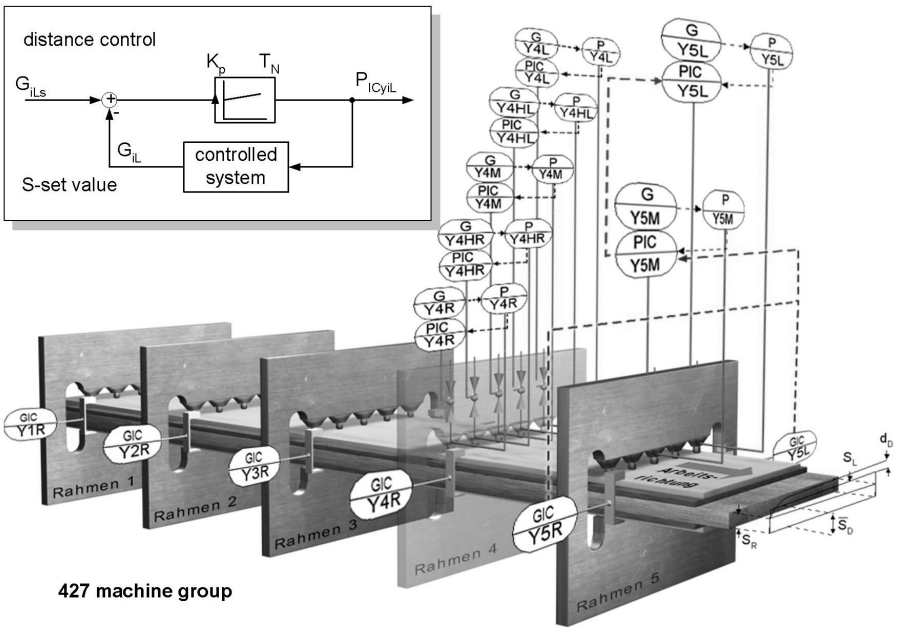


Fig. 1. Model of a continuous thermo-hydraulic press.

stable during the interrupts. The treating of material in process technology cannot be interrupt that easily, as the material will continue its reaction during the interrupts. Continuous flow processes are extremely challenging to interrupt. Fast flowing material requires real time reactions from the automation system for every change.

Therefore, a continuous flow technology example will be used for discussion purpose. For this purpose, alternative reactions for a continuous thermo-hydraulic press with failed sensors are being inspected. Fig. 1 gives an overview of this application. A press is composed of up to 80 frames. Each frame consists of 5 separately controlled hydraulic cylinders that are equipped with pressure and distance transducers. The sensors are physically linked to programming logic controllers (PLC), which execute the control software. The sensors are linked by a field bus system. One PLC may control between 10 and 20 frames. The failure of one component results in a failure of the entire control chain.

The press produces fibreboards from raw material like wood fibres and glue. [14] The raw material is sandwiched between steel belts that are pressurized by the hydraulic cylinders. Intense differences of pressure between two neighbouring frames are not allowed.

Basically, there are three different ways for a control system to react to a sensors failure. First, the system may shut down the production process and stop until the defective sensor is changed. However, interrupting a running system can be very expensive or even impossible. Continuous processes in particular cannot be aborted and may react with an unpredictable behaviour, if an actuator is driven by uncontrolled value. Fig. 2 illustrates this behaviour at the thermo-hydraulic press. If the distance sensor of a hydraulic cylinder has a malfunction, the control function may set the pressure too high. This jams the steel belt and the material in the press.

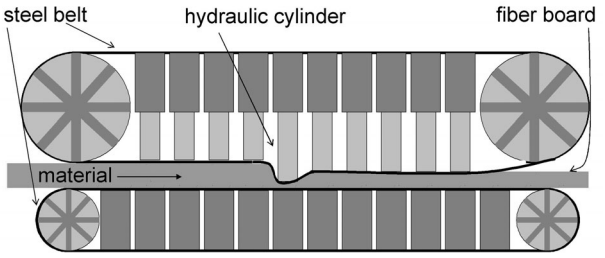


Fig. 2. Sketch of a continuous fiberboard press with an uncontrolled pressure value.

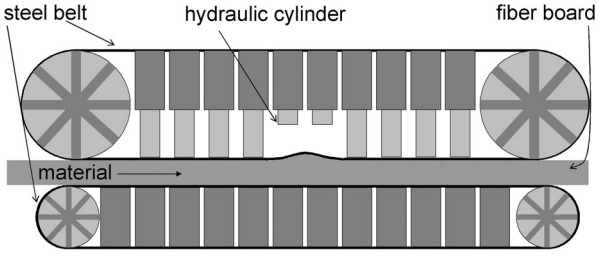


Fig. 3. Sketch of a continuous fibreboard press with two controlled units in a safe position.

Secondly, the jam can be avoided by forcing the controlled device to a stable state, e.g. a valve is opened completely or it is closed completely. Such stable state has to be predefined for each use case and it will allow continuing the production for a short time or shut down the process safety (Fig. 3).

The third way to react on sensor failures is a dynamic reconfiguration of the system during runtime using redundant devices or information. However, the backup solution is constrained by several requirements:

- Backup devices are expensive. A machine for pressing fibreboards has up to 200 sensors of the same kind. A hardware backup for each sensor would double the cost of the sensor equipment; it would extend the necessary number of inputs at the PLC and increase the effort for wiring the additional devices.
- A solution can be the use of virtual devices, which is a model based calculation of a measuring point. These calculations base on analytical dependencies to neighbouring sensors. Software solutions are cheap, easy to duplicate and they do not require any additional cabling. The quality of these calculations is depending on the quality of the model.
- The automation software should switch to backup devices at runtime. Dynamic reconfiguration at runtime requires more than having a number of suitable solutions. It also needs a decision, which solution is an ap-propriate compensation for the existing failure.
- The automation system has to be reconfigured in real time. E.g. the material of a fibreboard moves more than one meter per second. The material could jam very quickly if a failure is not handled immediately. Therefore, the decision to replace defective sensors by an alternative must be part of the control software.

3. Predefined alternatives

Traditional imperative programming languages offer different strategies to combine virtual sensor devices, replacement techniques and quality checks. Established design principles are based on modular approaches. Functional coherences are capsulated in modules. The programming languages that are used in PLC do not contain such constructs to assign the replacement functions at runtime. Sensor inputs are read directly from input variables. A solution may only be realized by hard coded, nested “if - then - else” clauses. The decision has to be retrieved each time; an actual sensor value is needed. Such decisions have to cover every possible failure and link to a corresponding strategy to handle it. In case of sensor failures, a virtual value will be calculated using a predefined calculation rule. To handle cases, which are combinations of sensor failures, every possible sequence has to be regarded and implemented. In these classical approaches, each element and its relations in between is predefined and described in a static way. Furthermore a high dependability between these elements arises from a low level of abstraction by describing them (e.g. function calls) [17].

The object orientation has methods (late binding), which allow the creation of new structures at runtime [9]. However, the object-oriented approach uses the same low abstraction level. The first available object oriented IEC 61131-3 platform doesn't support this concept of late bindings [4].

To react on changes of the system-structure at run

time, it would be necessary, to handle all possible changes, which can occur during design-time of a system and define the appropriate behaviour. This is particularly difficult when the behaviour of the system is connected to real-time requirements.

This is possible for systems with a limited number of elements, dependencies and a limited behaviour variation. For systems, which consist of many elements affected by many factors, the overall behaviour grows by the number of possible dependencies between the different elements much faster than the number of elements. [9] The attempt to describe all possible states of the system a priori, leads to an extraordinary effort designing the software.

Agent based approaches promise a real dynamic reconfiguration at runtime, as decisions can be concluded from rules.

4. An agent approach for rule based decisions

A well-suited approach for developing decentralized, complex and dynamic software systems 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 [16].

Unlike a static approach in an agent oriented approach the entirety of the structure and its behaviour has not to be fully specified at design time. The behaviour is generated dynamically at runtime regarding the current situation and within defined variations.

Strategies for faulty sensors for example, are determined by a set of rules, instead of predefining a calculation as replacement for a faulty sensor. The agent retrieves the best alternative from a set of possible references at run-time. Decisions, concerning the best reaction to the current situation, are moved to run time. This leads to a reduction of complexity at design-time [17].

The lack of suitable methods for the design of agents in industrial applications is actually recognized by several working groups. The national project AgentAut [7] as well as the European projects Pabadis [6] and Pabadis promise [10] work on an integrated method for distributed control systems, but focus only on the integration of PPC/MES and control level. The European projects SOCRADES [12] and RI-MACS [11] 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 optimised planning of production program at runtime and not to increase the dependability of the system regarding real time requirements. However, neither methods nor tools that are adapted to design agents for industrial real-time applications on PLC basis exist [8]. An exception is the project AVE [1], which developed such a method to support a systematic design of an agent-based system for embedded real-time systems in terms of safety and real-time requirements.

The difficulty in developing an agent system for real-time applications is to define the action space of an agent precisely enough to ensure the requirements regarding the availability of the system, the required performance and the product quality.

In [15] a SysML [5] based approach was presented by us that supports developers defining the requirements and constraints of an automation system. This model is used as a template for the definition of the action space as the main part of the agent's knowledge base. A main part of the agents duties in the context of increasing availability of an automation system, is to detect, analyse and handle faults. By now, the agents just focus on instrument fault detection by using analytical redundancy between different measurement-points. For every real sensor, which has functional dependencies to other sensors, we calculate additional virtual sensors using values of neighbouring real sensors. The virtual sensors are used to validate the corresponding measurements and detect faults (parity space approach). If there is more than one virtual or real sensor available at one measurement-point, it is possible to detect a single fault (isolation). Principally the virtual sensor values will never be as precise as a real sensor values. In case of diagnosing a fault, this implies the risk of false alarms or on the other hand the sensitive to faults. In case of substituting a real sensor by a virtual one, this loss of precision is not only relevant for closed loop controls but for the whole control strategy of the process.

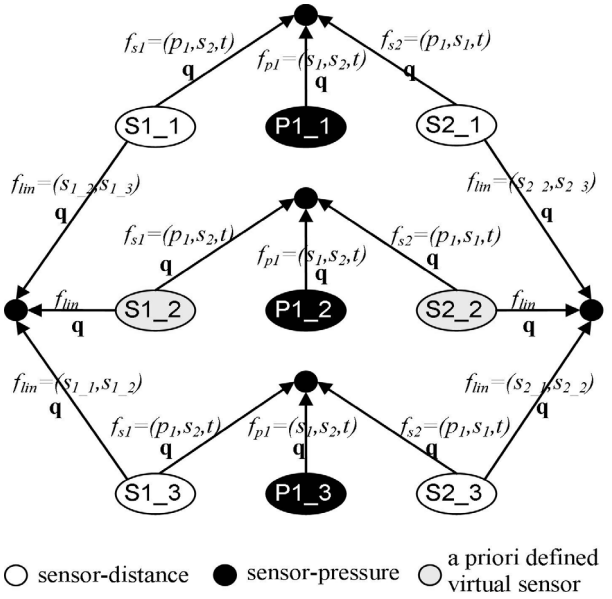
Therefore, it is insufficient just to calculate virtual sensors and to substitute faulty real-sensors. Additionally, the consequences of such substitutions and constraints of the automated system have to be taken into consideration. An agent knowledge base contains two main components - constraints and knowledge. Constraints define the margin of the activity space that is used by the agents to take decisions. Knowledge is the possible alternatives at a certain point inside the activity space. Both aspects require an exact orientation of the agents in the action space.

Next, a knowledge base, which allows detecting sensor failures, calculating a surrogate value and estimates the resulting precision at runtime, will be introduced. One important point for the design of such a knowledge base is, that it is easy to design and implement in a PLC environment. A very simple and powerful notation for this purpose, which is well known in the domain of automation, is the directed graph [3].

In this graph, each node represents a measurement point. It is equipped with a value source that can be either a real or a virtual sensor. A quality-value at each node describes the accuracy of the measured or calculated value. The quality value ranges continuously from 0 to 1.

The edges of the graph describe functional correlation (f, Fig. 4) between the measurement-points and represent the analytical dependencies, which are used to calculate virtual sensor-values at runtime. The direction of the arrows indicates sensor values that are appropriate for a substitution (Fig. 4). The black dots are used, if more than one sensor is required to calculate a virtual sensor. For example: Sensor “S2_1” can be calculated, using the function “f_{s2}” and the sensor values “P1_1” and “S1_1”. The function “f_s2” expresses the dependency between the thickness of the incoming material “s_1” the

pressure of the hydraulic cylinder "p_1" and the thickness of the outgoing material using a spring model. The spring constant (C, Fig. 4) represents the elasticity of the material and depends on the actual temperature, density and humidity of the wood. The time-delay (t_v), which is causes by the moving material and the distance between the sensors, is considered by using recorded values for "S1_1" and "P1_1". The functions "fs1" and "f_p1" are transformation of the same equation. The function "f_{lin}" is a linearization between two measurement points (x_1,y_1; x_2,y_2) which value is calculated at the position of the virtual sensor (x).



$$f_{s2} = s_1(t - t_v) - \frac{p_1(t - t_v)}{C} \qquad f_{s1}(t - 2t_v) = s_2(t) + \frac{p_1(t - t_v)}{C}$$

$$f_{p1} = C \cdot (s_1(t - 2t_v) - s_2) \qquad f_{lin}(x) = \frac{(y_2 - y_1)}{(x_2 - x_1)} x + y_1$$

Fig. 4. Analytical dependencies of sensor values (P-pressure, S-distance) using the material property (C).

The substitution of a real sensor by a virtual one will not change the structure of the graph. It is possible to use virtual sensors as source for other virtual sensors and the probability for that rises with the number of failures and corresponding substitutions.

The precision of virtual sensor-values is possibly reduced by inaccurate models and time aspects, e.g. dead time or delays because of the underlying measurement, the field bus or the calculation of virtual sensor values in the PLC. Reduced precision lowers the quality of the virtual sensors compared with original measurements. This loss of precision is considered by a so-called quality factor (q, Fig. 4), which is bound to every arrow of the graph and described by values between 0 and 1. In addition a quality-value (Q, Fig. 4) at every node represents the precision of a sensor measurement. Real sensors get a quality-value, which is initially determined by vendor specifications. The quality-value of a virtual sensor is determined by a product of quality-factor multiplied with the quality-value of its source. It indicates the coherency between the model and the reality. A low quality-value shows a high uncertainty about the calculated or measured value.

While each replacement impairs the accuracy of calculated values, the quality values represent the estimation of uncertainty. The described strategy prefers real measurements, as they have the lowest divergence or the highest quality value. It cuts of cyclic calculations, as their quality value would tend towards zero. This prevents complete virtual process images that are coherent but not validated.

In case of a sensor failure, its quality-value would tend towards zero. Following the arrows of the graph and using the corresponding functions, it is possible to calculate a virtual sensor-value as well as its precision with the quality-factor and quality-value of the source. Every arrow that leaves the node represents a virtual sensor, which can be benchmarked at runtime by comparing their quality-values. The virtual sensor with the highest quality-value will replace the defective sensor without changing the model structure but changing the quality value. References to other sensors still remain. If a virtual sensor is used as a backup of a sensor, which is the source of other virtual sensors, the quality of these derived sensor values will be recalculated. The agents, which pre-assigned the defective sensor as a source, have to compare the available alternatives again. Doing so, the agent system optimises the use of the remaining real sensor values. Local decisions of agents lead to an optimal quality of controller input values over the entire system. The agent system optimises the use of measurements dynamically at runtime.

The correctness of values can be reasoned by combining the assigned quality values of all possible virtual sensors. Discrepancy of measured value and a virtual sensor of low quality value do not stringently implicate a sensor fault. Only a very high discrepancy would implicate that either the measured or the source of the virtual sensor is faulty. In contrary, a sensor is detected as defective if all possible substitute values diverge in the same way with high quality values. The threshold, which defines the agent's decision, is oriented at user defined safety requirement for the specific part of process or the technical system.

Furthermore the agents use the quality value of a virtual sensor to determine the effect on the availability of the plant operation and to compare it with the given requirements and constraints. The reliability of sensor values is evident for processing automated production systems. While the substitution of real sensors with calculated virtual sensors increases the readiness in case of partial faults, it risks the accuracy of the process flow. While the correctness of possible alternative strategies for static systems is determined during development time, an agent based dynamic system decides this during runtime. Both have to decide, whether the production process can be continued with replaced, calculated values or if it has to be suspended. The loss of a sensor reconfigures the control behaviour automatically. The automated result may be a single parameter adjustment or an immediate shut down of an entire plant and is characterized as dynamic reconfiguration at runtime.

The effort to calculate virtual sensor values depends on the complexity of the related mathematical terms and on the number of dependant sensors. As long as virtual

sensors are used only for diagnostic purposes, the effort may be reduced by extending the calculating period of virtual sensors.

5. Implementation and evaluation

The introduced concept was evaluated by applying it to the thermo-hydraulic press. A Matlab model of the process was used to simulate the real industrial application. The agent system was implemented using a classical PLC [13], which is programmed according to the IEC 61131-3 standard. Each agent is coded in IEC 61131-3 programming languages and is assigned to a separate function block with duties for control, messaging, and diagnosis. The agents are linked together by a common process image and the option to communicate via messages. The agents can either bound to different tasks or PLC's or just run all in the same task on one PLC.

All control agents are identical. Each agent controls one of the 23 frames of the thermo-hydraulic press. This architecture is oriented at the structure of the technical systems that composes the mechanical elements of a plant. Agent parameters are set according to the requirements of the local sub-processes. They depend on the location of the agent's controlled elements in the press. One supervisor agent stores this knowledge. On request of the control agents the supervisor submits the settings and constraints.

As part of all control agents, the common knowledge base represents relations between sensors on one frame and its neighbouring frames.

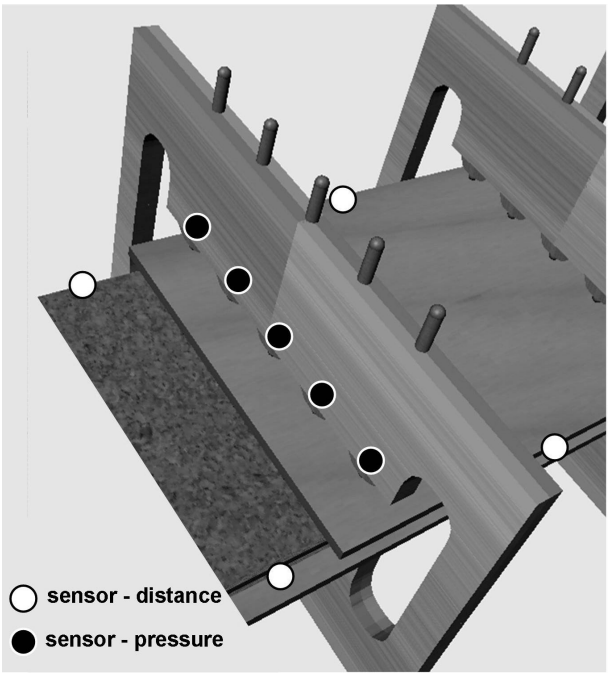


Fig. 5. Distribution of sensors in a frame.

Due to their identical structure all agents use a nearly congruent part of the knowledge base. Differences only concern the parameters of their calculation terms and result from the transformation of material properties. The processing of wood fibre and glue influences the dependency of measured values between different sensors significantly.

	S1,1	S1,2	S1,3	P1,1	P1,2	P1,3	S2,1	S2,2	S2,3
S1,1	x_s1,1	x_virt_ s1,1 (1)		x_virt_ s1,1 (2)			x_virt_ s1,1 (3)		
S1,2	x_virt_ s1,2 (1)	x_s1,2			x_virt_ s1,2 (2)			x_virt_ s1,2 (3)	
S1,3		x_virt_ s1,3 (1)	x_s1,3			x_virt_ s1,3 (2)			x_virt_ s1,3 (3)
P1,1				x_p1,1			x_virt_ p1,1 (1)		
P1,2					x_p1,2			x_virt_ p1,2 (1)	
P1,3						x_p1,3			x_virt_ p1,3 (1)
S2,1				x_virt_ s2,1 (1)			x_s2,1	x_virt_ s2,1 (2)	
S2,2					x_virt_ s2,2 (1)		x_virt_ s2,2 (2)	x_s2,2	
S2,3						x_virt_ s2,3 (1)		x_virt_ s2,3 (2)	x_s2,3

Fig. 6. Matrix of as representation of the knowledge base.

The graph is mapped to a matrix (Fig. 6) that is implemented on an IEC 61131-3 runtime system. The columns and rows are labelled with references of all nodes. The main diagonal is periodically updated with the values and qualities of real measurements. All other intersections of rows and columns refer to tuples of quality factors and values. The column id is the source and the row id is the target. Replacement for sensor in column id can be identified through the row id. These table cells represent the marked edges of the graph. Table cells, which do not represent an edge, are set to the tupel (0,0).

Using this mapping, an agent can get the results of real measurements and all alternative calculations. This simple access allows an easy diagnosis and straightforward reconfigurations. All table cells contain references to variables, which are written by function blocks, with measured or calculated values. The function blocks of real measurements are used to pre-process the measured input values. The function blocks of virtual sensors are more complex. They have to compensate delays that are caused by the material transport in the press. This requires measured values to be recorded. The calculation of virtual sensor values is done using this recorded data. As long as a related sensor is operating, the difference between measured and calculated value is recorded. This information is used to quantify the quality value. Additionally, this information helps to improve the precision of a simple initial model.

The application example uses three different types of virtual sensors. The first sensor type calculates the back-up values by interpolation of neighboured sensor values. The second one uses physical relations of the process and calculates the resulting material thickness at a frame from the incoming material and the current pressure. The third virtual sensor type calculates the pressure from the difference of incoming and outgoing material thickness. The function blocks deliver a sensor value and a quality value independently from a measured or calculated source. A status variable indicates the operation status of a sensor. This is used for local diagnosis of sensors.

Although pointers are not part of the IEC 61131-3 language specification, they are supported by some of the leading manufacturers of automation equipment, for example TwinCat by Beckhoff or Step7 by Siemens.

The use of pointers was helpful to build references to the measured and calculated values. It was possible to

build one knowledge base that could be used by all agents without recopying the values. Additionally, pointers simplified the software structures as reconfiguration was realized by changing pointers from references to real sensors to references to virtual sensors.

The agents continually checked the plausibility of current values using its alternatives. In case of a determined fault the current sensor was replaced by the best alternative sensor. Therefore, the address of the replacement substitutes the address of the predecessor. The new current sensor is available to all other functions, function blocks and programmes at the accustomed place, delivering new values for sensors and quality. The detection of faults is done at the beginning of every PLC-cycle and the replacement is done immediately, so that further access to this measurement-point is redirected to the new calculated value and no delay occurs.

The exchange is represented by using the values of the virtual sensor instead of the real in the corresponding node (measurement-point). All relations of the structure remain valid. A decreasing quality value causes further decreasing quality values at the derived virtual sensor. All related agents include this new information in their decision for the selected virtual sensor and dampen the negative effect of a sensor break down on the entire system.

6. Conclusions

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 detect and react on failures, in order to increase the availability or to improve the efficiency by adapting to changing requirements. Compared with a classical approach the self-adaptation of the application during run time was flexible and easy to implement. Decision rules on basis of the knowledge base facilitate the definition of constraints and dependent requirements. 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 agent's knowledge base and replacement mechanism allows reaction based on changes in environmental conditions. By that it handles changed precision of sensors and/or actuators during run-time. The solution adapts the actual values with appropriate changes instead of using worst-case values in predefined replacements. 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 availability of the production line.

The current work evaluated, that it is possible to set-up an agent based decision system that adapts data sources for sensor values automatically and shows that complex decisions are possible with rules defined in easy structures that can be implemented in an IEC 61131-3 environment. Future work will also evaluate the usability of this approach. The required effort and skills for modelling such agent-based solutions will be compared with conventional procedures. The usability of creating, understanding and modifying agent based or conventional solutions will be regarded separately.

The self-adapting agent approach gives the perspective that modular systems may be composed easily. Agents promise a better reuse because they negotiate their relations instead of being bound strongly and inflexible. The benefit of this approach will be regarded at different classes of applications, like hybrid combination of process and production systems.

AUTHORS

Andreas Wannagat* - Chair of Embedded Systems, University of Kassel, 34121 Kassel, Germany. E-mail: wannagat@uni-kassel.de.

Birgit Vogel-Heuser - Chair of Embedded Systems, University of Kassel, 34121 Kassel, Germany. E-mail: vogel-heuser@uni-kassel.de.

* Corresponding author

References

[1] AVE, "Agents for flexible and reliable embedded systems", *DFG Project: VO 937/5-1*, 2005-2007.

[2] Braun E., *Technology in Context - Technology Assessment for Managers*, Routledge: New York, 1998.

[3] Chartrand G., "Directed Graphs as Mathematical Models", in: *Introductory Graph Theory*, Daver: New York, 1985, pp. 16-19.

[4] CoDeSys 3, 3S-Smart Software Solutions GmbH, Kempten, www.3s-software.com.

[5] Hause M., "The SysML Modelling Language", *Fifth European Systems Engineering Conference*, Edinburgh, 2006.

[6] Klemm E., Lüder A., "Agentenbasierte Flexi-bilisierung der Produktion bei Verwendung von vorhandenen Steuerungssystemen", *atp Automatisierungs-technische Praxis*, vol. 45, 2003.

[7] Lüder A., Peschke J., Sanz R., "Design Patterns for Distributed Control Applications", *atp international*, vol. 3, 2006, pp. 32-40.

[8] Mubarak H., Göhner P., Wannagat A., Vogel-Heuser B., "Evaluation of agent oriented methodologies", *atp international*, vol. 1, 2007

[9] Palsberg J., Schwartzbach M., "Object-Oriented Type Inference", *ACM SIGPLAN Sixth Annual Conference on Object-Oriented Programming Systems, Languages and Applications,* 1991.

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

[11] RI-MACS, "Radically Innovative Mechatronics and Advanced Control Systems", *European Project: FP6 NMP-IST Joint Call 2,* 2005-2008.

[12] Socrades, "Service-oriented cross-layer infrastructure for distributed smart embedded devices", *Integrated Project, European Commission, Information Society Technologies*, *Frame Programme 6,* 2006-2009.

[13] TwinCat, Beckhoff Automation GmbH, Verl. www.beckhoff.de.

[14] Vogel-Heuser B., "Automation in wood and paper industry", *Nof, S. Y. (Ed): Handbook of Automation*, Springer-Verlag, New York, 2009; to appear.

[15] Wannagatn A., Vogel-Heuser B., "Agent oriented soft-

ware-development for networked embedded systems with real time and dependability requirements in the domain of automation", *IFAC World congress, 2008*.

[16] Wooldridge M.J., Jennings N.R., "Intelligent agents: Theory and practice", *The knowledge Engineering Review*, vol. 10, issue 2, 1995, pp. 115-152.

[17] Jennings N.R., "On agent-based software engineering", *Artificial Intelligence 117*, 2000, pp. 277-296.