ENHANCING DEPENDABILITIES OF SMART UNITS IN INDUSTRY 4.0 CONTEXTS: A MULTIAGENT SYSTEM APPROACH

[1]HERMANN KUHNLE, [2]HESSAMEDIN BAYANIFAR

[1,2]Otto Von Guericke Universität Magdeburg, Germany
E-mail: hermann.kuehnle@ovgu.de, hessamedin.bayanifar@ovgu.de

Abstract- Dependabilities of smart manufacturing units' interactions in cyber contexts may be multiplied by multiagent system setups and their implementation into the manufacturing context. The adoption of revolving decision cycles in complex networks with the specifications of the agents demonstrates advantages but has still number of limitations and restrictions to overcome. The introduction of specific FMEA modes facilitates the specification and implementation of multiagent systems under an open architecture. Failure mode components in the manufacturing context may lay the grounds for a fast learning system.

Index Terms- Industry 4.0, Cyber-Physical Production Systems, Dependability, Security, Multi-Agent Systems, FMEA

I. INTRODUCTION

Smart distributed manufacturing systems, consist of a large number of widely dispersed loosely-coupled yet collaborating heterogeneous components, that are vastly connected to and communicating with cyber spaces. To enhance their capabilities, these systems try to exploit smart properties through enhancing their own intelligence and processing power, or via accessing the internet and its vast options to enhance these properties. On the one hand, using these properties and enhancing capabilities can apport a plethora of opportunities and strategic advantages to manufacturing enterprises. On the other hand, however, vast dispersity and exposure to cyber spaces as well as versatility of processes and systems' structures raise major vulnerabilities; so, dependability and data security issues may diminish the readiness to rely on such enormous capabilities. Hence, to harness all capabilities, not just through implementation, but also as real-time dependabilities and with utmost security must be assured. Moreover, an adaptable, learning tool to analyze and steadily improve all conditions that accommodate all dependability and security needs, would represent an important contribution to make smart manufacturing units' applications more attractive.

Smart Distributed Manufacturing systems, enabled by Cyber-Physical Systems, have major structural similarities, as they typically expose three main layers: a physical layer, a cyber layer, and a data communication and integration layer. Each of these layers has its own concerns with regards to dependability and security. Accordingly, many studies tried to point out these issues or suggest adequate measures [1-5]. Considering these studies and recommendations released by the Federal Office for Information Security on industrial control system security [6], the main issues there are namely: Distributed Denial of Service (DDoS) attack, social engineering and phishing, control system compromising (masquerading, repudiation, manipulation, etc.), Man-In-The-Middle attack, malware infection, intrusion, as well as stability and controllability issues, such as loss of connectivity, observability, breakdown or failure, Quality of Service (QoS), to be seen among the major possible risks. In [7], the author proposes a design and an implementation approach to robust cyber physical systems focusing on security, stability, and systematicness (seamless integration of sensors and actuators) [8] choses to model ontology-based dependability in CPSs using FMEA techniques. Moreover, the use of FMEA has also been taken into account in [9-10] as a probabilistic method for analyzing failures in the design phase of improving dependability of self-optimizing entities. Other approaches [11] make use of systems' context awareness to increase security in information access by asking the questions, e.g. who wants the information, how, what information, from where, and when and develop an information system that verifies accesses.

This paper proposes a generic model, and an agent-based implementation structure along with a FMEA real-time approach to develop an intelligent and autonomous distributed dependability and security supervision and control that is broken down throughout the system, and is to be carried out by smart components themselves in Industry 4.0 environments. To this aim, it adopts and harnesses smart manufacturing systems' capabilities and properties that are elaborated in [12], (i.e. interoperability, autonomy, scalability, modularity, heterogeneity, reconfigurability, and context-awareness), for maximizing its performance and versatility. The paper outlines the model and its relationship to smart systems' properties, then presents multi-agent systems as an adequate toolset in combination with the FMEA approach with an example for illustrating the achieved functionalities. And the paper ends with conclusions and related future works.

II. THE DEPENDABILITY AND SECURITY MODEL

To assure maximum dependability throughout an enterprise, the adopted approach must cover all incorporated components, all information flows among them as well as the addressed cyber areas, networks, databases and servers. To this goal, a distributed

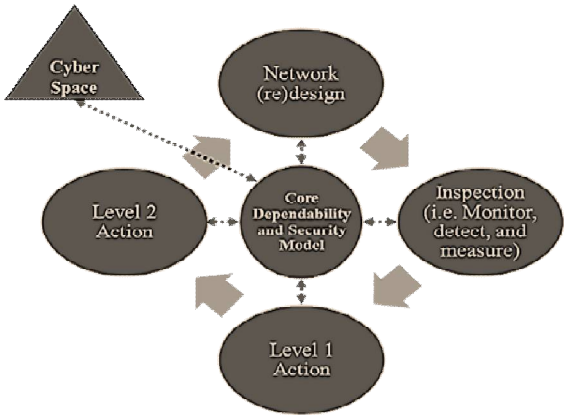


Figure 1. Smart Dependability and Security architecture

Dependability and Security Model (Fig. 1) is proposed based on the decision cycle (elaborated in [13]) for covering the entire network, every unit and component down to all levels of detail (LoD). Furthermore, the model includes a core model, a control loop, and a connection to the virtual world.

The model core consists of two main parts: object description and risk model; the former focuses on objects' context and self-awareness, and imports data about object's environment, collaborations, functions and modules, objectives, application and task description, etc. that are needed to develop risk model. The latter accordingly covers all Dependability and Security parameters, vulnerabilities and risks, and the ways of measuring and treating them. The core model feeds the improvement process done by the control loop. The relevant data for object description section are imported from the cloud or sensed as a part of the object's self-/context-awareness. The risk model contains a scalable feature for considering possible risks, assessing their criticality and their possible effects on the object or the system in total and the countermeasures associated with the risks (e.g. Failure Mode, Effect and Analysis (FMEA) or Fault Tree Analysis (FTA)). Self-optimization occurs via sharing knowledge with all other smart units, and updating its own structure and database through continuous feedbacks (control loops).

The Control Loop invokes the Inspection process (i.e. Monitoring, Detecting, and Identifying and Measuring), and the Reactions (i.e. giving Alarms, taking Action, and doing the Reconfiguration afterwards) in real-time. All steps will be carried out fully- or semi-autonomously by smart objects applying the model core, which is located in the cyber space and is in collaboration with all other models. This gives the components all abilities to collaborate with the common objective of raising and maintaining the dependability and security of the total system.

III. MULTI-AGENT SYSTEMS (MAS) FOR MODEL ARCHITECTURE

The dependability and security model with the properties shown, offers a toolset capable of backing such properties. Accordingly, Multi-Agents Systems (MAS) are top candidates, since intrinsically, intelligent agents (IA) demonstrate responsiveness, proactiveness, goal-orientation, social-ability, scalability, flexibility, robustness, self-configuration, adaptability/ re-configurability, along with their decentralized architecture and learning capabilities [14, 15]. After determining the tools, the model is translated into an architecture composed of interacting agents. The first step splits the model task-wise onto single agents. Doing so, the following lists the agents that are defined to specify the model, and their task description. Fig. 2 shows their overall structure and collaborations' relations [16].

Agents supporting the core model:

Status manager: Updates the status of components as main part of context-awareness. Knows all approved contexts, authorities, topography of the system, etc.

Database: Stores components' models and risk data. Data are stored in modules for each type of risk to be accessed by analyzers and assessment agents. Other components when authorized, have access to some of the data during negotiation or when they are new to the system, to get updated with vulnerabilities and measures, etc.

Assessment Agent: Receives data from the analyzer and assesses the risks by negotiating with other components' assessors, and by receiving context data from TS manager.

Interface: For negotiations between agents of other components; Updating the topography and the context information, more accurate and global risk assessments, provides access to databases of other components.

Agents supporting the control loop:

Data filter: Filtering out redundancies. Looking for useful data among loads of data.

Monitoring: Looking for anomalies and risks, by comparing the current-state data with current-state approved contexts. Then sends the detected cases to the (level 1) analyzer.

Data-analyzer level 1: For simpler problems/quicker responses; The Data analyzer does the identification and measurement of risks. Level 1 analyzer analyses and communicates with the assessment agents to provide proper input for DM level one1. It then provides the feedback to the core.

Data analyzer level 2: For more complicated problems; analyzer level 1 sends the case to analyzer level 2 with more abilities. If needed this analyzer practices negotiations with other components agents to

provide best global data of the risk to feed the DM level 2.

Decision-making level 1: For simpler reactions/ quicker responses. After supplied with proper risk information, releases alarm to right entities and send proper commands to actuators to apply right corrective actions, and reconfiguration when needed. Feedback is then being provided to the core model.

Decision-making level 2: Provides higher level reactions, and reconfigurations, and a more advanced alarming system. If needed asks for collaboration of other agents and components' resources to solve a problem. Feedback is then being provided to the core model.

IV. MAS AND FMEA TO DELIVER SMART SYSTEMS' PROPERTIES

The model is adopting smart systems' properties to enhance its performance. The following table 1 lists the contributions of these properties in each phase of the dependability control loop, and the methods and the requirements delivered.

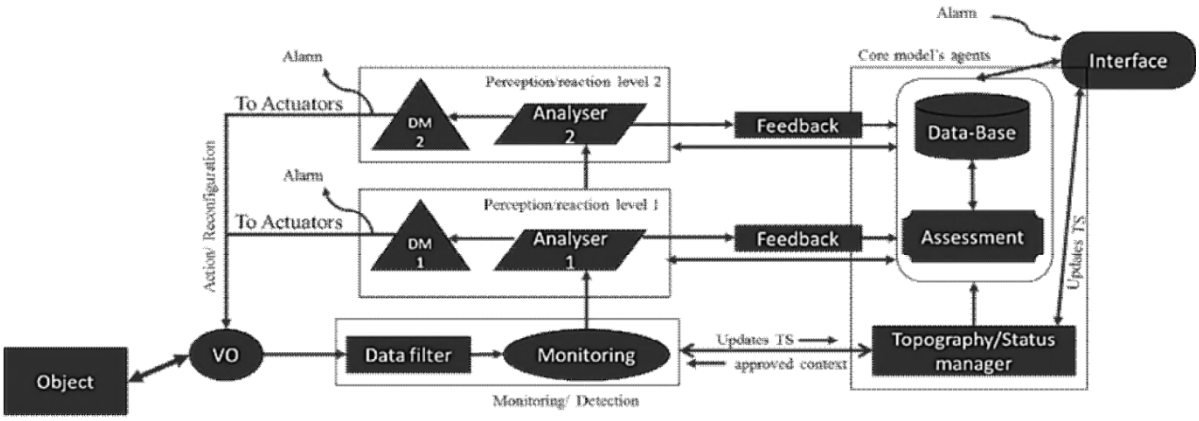


Figure 2. Proposed Agent Architecture for components' intelligent dependability and security

Table 1. Control loop steps and the relationship with smart manufacturing systems' properties

	Contribution	Requirements / method
Monitoring: constantly checking the related parameters defined in the core model to find risks		
Context-awareness	Parameters are derived from the context	Via sensors/ cloud
Interoperability	Hierarchical/heterarchical collaboration	Shared semantic/ ontology
Autonomy	Autonomously done by Intelligent Agents	Core model/ sensors
Modularity	Resources to be used in various setups	Modular resources
Scalability	Resources to be added or removed	Registering mechanism
Heterogeneity	Mechanism differs based on object type	Via core model
Detecting: Finding anomalies/risks by comparing real-time status with approved parameters		
Context-awareness	Current vs Approved context comparison	Via sensors/ core model
Interoperability	Hierarchical/heterarchical collaboration	Shared semantic/ ontology
Autonomy	Done by agents, and aided by core model	Agents/ data base
Modularity	Data-base and model to be re-useable	Modular risk Data-base
Scalability	To be seen data-base / detection methods	Updateable core model
Heterogeneity	Methods differs based on object type	Set in core model
Measuring: When detected, identifying the risk type, its severity, impacts, occurrence frequency, using assessments methods such as FMEA/FTA		
Context-awareness	Finding global measure based on context	Updating context data
Interoperability	Objects negotiating to find global measure	Via agents/ model
Autonomy	Done by agents and by using risk model	Risk model in the core
Modularity	Measurement resources to be shared reused in various setups	Risk categories to be modularly saved
Scalability	Criteria/ data-base to be changed/ updated	Scalable risk model
Heterogeneity	Criteria and risk model differ object-wise	Risk model definition
Alarm: Alarming right entities, i.e. people, or components that may be affected by the risk		
Context-awareness	Right entities are known through context	Context update in model
Interoperability	In carrying alarm to various entities	Semantic definition
Autonomy	Done by agents after measuring risks	Agent collaboration
Modularity	Agents/functions to be used in new setups	Modular alarm resource
Scalability	methods/agents to be added or removed	Scalable alarm resource

Heterogeneity	Actions/resources differ by object type	Via model/ resources
Reconfiguration: providing feedback to the model and preparing the component to be reused		
Context-awareness	To be done based on context requirements	Via the model
Interoperability	Providing understandable feedbacks to others/ receiving required data	Via model and semantic definition
Autonomy	Semi/fully autonomous and done by agents	Model/ Agents
Modularity	Components may be reused in new setups	Modular components
Scalability	Extended/reduced structure in new setup	Scalable model/object
Heterogeneity	Done based on object type. Same feedback mechanism	Via model Same agent functionality

The first step to be continuously carried out is Monitoring. As the distributed nature implies, this task will be broken down and distributed onto agents that subscribe to the main monitoring agent. The main monitoring agent has a list of agents registered. If a threshold is exceeded or an anomaly detected, a message will be created as from the sub-agent to main monitoring agent, and an information from the monitoring agent will be sent to the next level for identification and measurement. This way, the monitoring will have maximum scalability and heterogeneity as the task is broken down and distributed among sub-agents that are in charge of e.g. checking one or a small group of sensors. Each of the sub-agents checks the sensed data against the approved value.

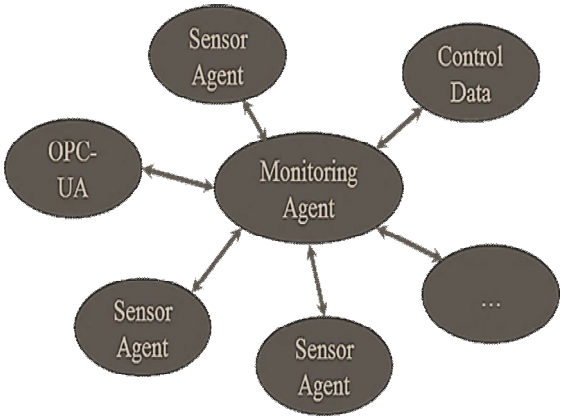


Figure 2. Breakdown of monitoring mechanism

In cases, where the system has OPC-UA (Open Platform Communication-Unified Architecture) implemented, the monitoring agent subscribes to it as a client to receive the sensor data required. Earlier It has an event-based alarming mechanism that can inform the client (i.e. here the client can be the monitoring agent) if a certain condition in a certain part of the system is not met [17]. This mechanism in OPC-UA is configurable by the client, and is scalable. The monitoring agent can register various entities of any structure and format in real-time (fig. 3), can receive their signals and locate arising issue/s. Such structure reduces the complexity and in turn increases the robustness and flexibility of the monitoring mechanism.

The next step, which is the analyzer agent's job, is about identifying and locating the risk, based on the sensors anomalies and location observations, and by having access to the system topology and retrieving the risk from a FMEA model, defined for components in assessment agent.

System topology is provided by the topology/status manager. it is formed through ontology definition, where components are introduced with basic ontological information, to be parsed into a meaningful map of connections and hierarchical position in the system. Main topology data can be as: $Component_i$ = {is part of, is made of, receives from, sends to}.

System topology can be imported from AML (Automation Markup Language), if the system has already Automation ML installed. AML can model all the components, their location in a hierarchical tree, their connections and relations [18]. It uses XML language for communication and can send the required data with any desirable applied filter to its client (here the topology manager agent). Then the topology manager feeds the other agents, i.e. assessment agent, analyzer agent, etc. The FMEA model is defined to evolved the system in real-time by providing risk assessment data for the analyzer agent and measures for the decision-making agents. It is in general a bottom-up tool for identifying failure modes, their causes, their effect on the system and their severity, plus detection and countermeasure suggestions in a tabular form, and is being used as one of the main tools in dependability engineering. The elements to be received and stored for identifying and measuring failure i of component j are: mode, severity, cause, detection, effect.

Modes are defined and categorized by a taxonomy (fig. 4) for components (e.g. Input/Output, task fulfilment or functionality, etc.), while ontology is well established among agents to relate each failure mode to their task.

Then each mode will be profiled with associated causes, ways of detection (sensors involved/type of anomaly). Anomalies are to be well defined and categorized in a list, while each failure mode has a group of anomalies indicating it. These data are to be accessed by right entities (e.g. other new components) to update their database if needed. The analyzer agent receives anomaly/s array from the monitoring agent and identifies the mode by sending a request to assessment agent which has the FMEA knowledgebase to retrieve failure mode after comparing the

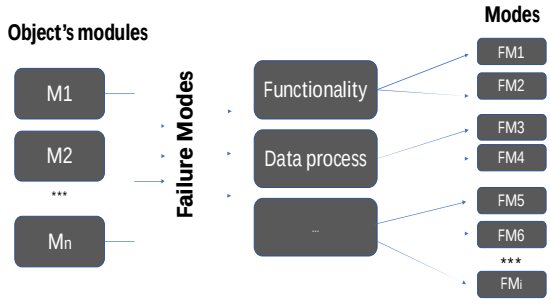


Figure 3. FMEA categorization of failure modes

detected anomalies with items in the list of anomaly arrays associated with each failure stored in FMEA database to send a message containing the identified failure mode to the Decision-Making agent. In cases of the anomalies unknown to the FMEA database, anomaly/s data will be sent to the higher-level analyzer for identification (e.g. by negotiating with other components). Effects are defined based on the topology data, i.e. which higher level components might be affected, or what other object might be compromised, given their relationship with the failed component. The severity accordingly will be determined as follow:

```
probability [ ]     p = {P0, P1, …, Pi};
Severity [ ]        s = {S0, S1, …, Si};
Double              totalSeverity = 0;
for (int i = 0; i < p.length; i++) {
totalSeverity = totalSeverity + p [i] * s [i]; }
```

Where Pi and Si respectively indicate the probability and the severity of losses to each of the other components, that are collaborating with the compromised component (in cases where there is a possibility of risk propagation or a failure causing other failures to other components in collaboration with it). Other information to record after each risk it the time, and its new occurrence value (OccurrenceCount ++).

Moreover, for each specific mode i of failure of/in component j, there would be information on measures/ fixing, and reconfiguring. In addition, each component is registered in the model with its task in the topology/status manager, which also includes information about its functionality. These data are going to be parsed by the decision-making agent.

Table 2. List of failure modes and action requirements

Failure Modes	Countermeasure / fixing	Requirements	Other corresponding components
FM_i	C_F_i	REQ_i [ ]	C_{ji} [ ] [ ]
FMi is a given failure mode, C_Fi is a description of the countermeasure process or fixing mechanism, REQi [ ] is an array of requirements (can be defined as codes and also use companies coding system for materials and spare parts) for a given failure mode, and Cji [ ] [ ] is Component j that is needed for, or has the Requirement i.			

The Decision-making agent uses the abovementioned data to carry out the proper actions. measures/ fixing process will be performed by triggering the right entities/actuators listed in the FMEA model database for each action, based on sets of criteria (e.g. cost, time, etc.). At the same time a substitute component must be found to deal with the task that was being done by the currently failed component. To do this, an adequate information about the component and the task is required. As previously noted, components will be registered with various ontology information, among which are the capabilities and the job

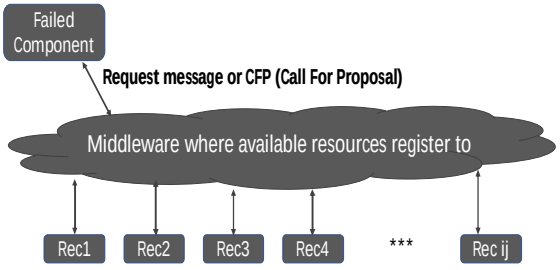


Figure 4. Middleware for resource enquiry by failed component

specifications (i.e. functionalities, capacity, I/O specification, etc.). Here, a communication mechanism presents all available resources by other entities. Agent-based control systems provide this information through a middleware where availabilities are shared. Components in their idle state are to be presented/ registered in a middleware with their capabilities. At the time of need, the decision-making agent sends a request to the middleware to see if there is any resource available (fig. 5). If yes, the negotiation starts between the two components on utilizing the resource, through the tools that the agent platform provides for agent communications.

Details about the negotiation can vary based on the offer provider (other components that have the free and proper resource to accomplish the task), and/or the type of the component and the task that is to be performed. Also, it is up to the developer to how to

break down tasks between the level one and level 2 decision-making agent.

The end of the loop is marked by the reconfiguration and reporting process. Ontologies and task descriptions about the component to be reused can be retrieved from data base and if necessary modified since all parts of the model is broken down into modules and objects. Furthermore, a report of the process, failure mode occurrence, countermeasures etc. will be provided to update the model.

An example to demonstrate this mechanism is a job-shop unit as an intelligent component with several automated robots, along with multi-agent systems implemented for controlling the processes. The monitoring agent, through its subscribed agents that monitor various sensors data, constantly checks the behaviour of the sub-components and the process. A possible failure can be an accidental breakdown of one of the robot's arms, causing it to stop working properly. Taking the assumption that the robot was being monitored through 5 sensors namely S1, …, S5, and an anomaly is detected in sensors S2 and S4 indicates that the arm of that robot is broken-down, the analyzer identifies the issue after checking anomaly data from the monitoring agent against its library of failure modes and their associated anomaly cases. The next step would then be to measure the total severity of this failure in the system. The arm causes the robot not to work well, and malfunctioning of the robot may bring delay to production process for the unit in which it is performing. Also, in its collaboration with other entities this failure may cause damage to components next to it. Based on the severity and probability of each of these cases, the total severity would be calculated as the sum of the product of the probability of each case and its severity if that case occurs. The next step would be to trigger the decision-making process. The Analyzer sends the failure ID and the severity to decision-making agent which sends a request to the assessment agent and database to retrieve the required data for releasing alarms to right entities (e.g. higher level component, other robots to stop if necessary, and so on), performing countermeasure/fixation (e.g. requirements and other components that provide them), in this example parts are needed to change the broken arm, inventory units that have them, maintenance teams, and perhaps transportation units. If several options are available, the decision maker choses the most favorable one (i.e. cheapest, fastest, etc. depending on the priorities). The next step is to find a substitute (robot/ a unit that has the same robot or can perform the same task). A request as a call for proposal (CFP) will be sent to get offers among available components with desirable resources to perform the task. When found, the decision-making agent will start negotiations with those components to make decisions on the best deal seen its priorities. After locating the alternative resource, ways of transportation (e.g. conveyors, AVGs, trucks, etc.), will be chosen to send the parts (that had to be processed by the broken robot) to the alternative resource. A Feedback of this process will be sent back for updating the model.

CONCLUSION

Multiagent system approaches for orchestrating local failure management of manufacturing units with network properties in the context of Industry 4.0 largely profits from the incorporation of FMEA. As the setup is the outcome of UA systems architecture, the models can be extended for covering more complex manufacturing systems. The plethora of failures occurring in on the shopfloor and in manufacturing areas imposes to start with a wider approach and to lay ground for learning algorithms and interactions of the agents. The first failure consolidations that have been taken into account are still simple and one-dimensional. However, corresponding to the network character of manufacturing and smart manufacturing processes, the problem sizes increase already with 2 stage processes enormously. The performance of the model can thereby expand through various ways, such as utilizing sophisticated reasoning algorithms in decision making to reach closer to globally optimized solutions or in adopting data mining approaches to predict possible coming failures from the patterns observed in the monitored data; taking precautions accordingly, providing the model with richer data about the context in real-time, using of more capable computational resources and communicational platforms, etc.

In the next phase, the model will be implemented on more cases including several security threats and dependability cases similar to the example described above to evolve its capabilities and to provide proper measures on its performance within given computational resources.

REFERENCES

[1] S. Huang, C.-J. Zhou, S.-H. Yang, and Y.-Q. Qin, "Cyber-physical system security for networked industrial processes," International Journal of Automation and Computing, vol. 12, pp. 567-578, 2015.

[2] L. J. Wells, J. A. Camelio, C. B. Williams, and J. White, "Cyber-physical security challenges in manufacturing systems," Manufacturing Letters, vol. 2, pp. 74-77, 4// 2014.

[3] G. Wu, J. Sun, and J. Chen, "A survey on the security of cyber-physical systems," Control Theory and Technology, vol. 14, pp. 2-10, 2016.

[4] L. Zhang, Q. Wang, and B. Tian, "Security threats and measures for the cyber-physical systems," The Journal of China Universities of Posts and Telecommunications, vol. 20, Supplement 1, pp. 25-29, 8// 2013.

[5] S. Karnouskos, "Chapter 6 - Industrial Agents Cybersecurity," in Industrial Agents, ed Boston: Morgan Kaufmann, 2015, pp. 109-120.

[6] F. O. f. I. Security, "Industrial Control System Security: Top 10 Threats and Countermeasures 2016," 2016.

[7] F. Hu, Y. Lu, A. V. Vasilakos, Q. Hao, R. Ma, Y. Patil, et al., "Robust Cyber–Physical Systems: Concept, models, and implementation," Future Generation Computer Systems, vol. 56, pp. 449-475, 2016.

[8] T. Sanislav, G. Mois, and L. Miclea, "An approach to model

dependability of cyber-physical systems," Microprocessors and Microsystems, vol. 41, pp. 67-76, 2016.
[9] R. Dorociak and J. Gausemeier, "Methods of Improving the Dependability of Self-optimizing Systems," in Dependability of Self-Optimizing Mechatronic Systems, J. Gausemeier, F. J. Rammig, W. Schäfer, and W. Sextro, Eds., ed Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pp. 37-171.
[10] R. Dorociak, T. Gaukstern, J. Gausemeier, P. Iwanek, and M. Vaßholz, "A methodology for the improvement of dependability of self-optimizing systems," Production Engineering, vol. 7, pp. 53-67, 2012..
[11] K. Wan and V. Alagar, "Context-Aware Security Solutions for Cyber-Physical Systems," Mobile Networks and Applications, vol. 19, pp. 212-226, 2014.
[12] H. Kühnle and G. Bitsch, "Smart Manufacturing Units," pp. 55-70, 2015.
[13] H. Kühnle and G. Bitsch, "Core Models, Principles, and Networks' Structuring," in Foundations & Principles of Distributed Manufacturing: Elements of Manufacturing Networks, Cyber-Physical Production Systems and Smart Automation, ed Cham: Springer International Publishing, 2015, pp. 27-53.
[14] R. Unland, "Chapter 1 - Software Agent Systems A2 - Leitão, Paulo," in Industrial Agents, S. Karnouskos, Ed., ed Boston: Morgan Kaufmann, 2015, pp. 3-22.
[15] "Agent Systems," in CIRP Encyclopedia of Production Engineering, L. Laperrière and G. Reinhart, Eds., ed Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pp. 33-33.
[16] H. Bayanifar and H. Kühnle, "Enhancing Dependability and Security of Cyber-Physical Production Systems," in Technological Innovation for Smart Systems, ed: Springer International Publishing AG 2017, 2017. "to be published"
[17] W. Mahnke and S.-H. Leitner, "Mapping of COM OPC to OPC UA," in OPC Unified Architecture, ed Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 283-292.
[18] N. Schmidt and A. Lüder, "AutomationML in a Nutshell " 2015.

★★★