Test-Driven Agent-Oriented Software Development

Munir Merdan*, Pavel Vrba** and Martin Melik-Merkumians*
*Vienna University of Technology — Automation and Control Institute
Gusshausstrasse 27-29/E376
1040 Vienna, Austria
**Czech Technical University — Department of Cybernetics, Faculty of Electrical Engineering
Technick 2
166 27 Prague 6, Czech Republic
E-mail: merdan@acin.tuwien.ac.at

Abstract

Lack of awareness about the potentials of agent technology, as well as concerns regarding stability, scalability, and survivability especially in unpredictable environments of attacks and system failures are identified as major reasons for missing trust in the idea of delegating tasks to autonomous agents. Therefore, one of the key research issues is how the quality of Multi-Agent Systems can be assured, while at the same time reducing the development efforts to enhance the acceptance of agent technology in the industry. In order to ensure the industrial relevance, we focus on development of methods and techniques for the evaluation and tests of agent-based control applications. Our research covers the development of demonstration framework, including pilot implementations in industrial-like settings.

1. Introduction

Manufacturing control systems are facing global trends towards small and medium lot sizes and are forced to produce customized products in a short time at a low price and under dynamic conditions. This demands concepts and open architectures that support easy reconfiguration, extension and parallelization of production lines e.g. if changes in throughput demands occur without the need of a complete re-design of the production line. Due to their limited capability for agile adaptation, current systems respond weakly to such demands and are mostly not capable to react effectively on sudden changes or unpredictable events such as failures and disruptions [3, 29]. The application of Multi-Agent Systems (MASs) is recognized as a convenient way to handle the dynamics in large systems reducing the complexity, increasing flexibility, and enhancing fault tolerance [14, 15]. However, although confirmed as a promising approach and deployed in a number of different applications throughout the last few years the widespread adoption of agent-based concepts by industry is still missing. Lack of awareness about the potentials of agent technology [22] and paradigm misunderstanding [7] due to the lack of real industrial applications, missing trust in the idea of delegating tasks to autonomous agents [26] as well as concerns regarding the stability, scalability, and survivability especially in unpredictable environments of attacks and system failures [10] are identified as a major reasons for that.

There are some works which address the problem of building confidence in the agent-based systems with particulars techniques based on testing and monitoring, others are based on debugging and others on simulation. However these works are still at very early stage. Actual formal methodologies provide validation tests that are applicable in only few and mostly irrelevant cases though [8]. The main reason of this lack of applicability is that activities, which should assure that the program performs satisfactorily, are very challenging and expensive since it is quite complicated to automate them. Besides, the proposed modeling languages and methodologies still remain incomplete or at least they are not able to fully capture and/or communicate the underlying meaning and complexity of MAS. In particular, up to now, the study of issues connected with the MAS implementation phase has not been stressed enough [5] and looking at the testing levels most of the approaches need to be consolidated and evaluated on realistic case studies to provide evidence of their usability [20]. Introduction of tools, techniques and methodologies that will ensure development of agent systems mature enough for industrial deployment is of vital importance for making a progress in the development and adoption of agent technologies [22].

To facilitate the design of such multi-agent control systems, we present an approach for development of the MASs, based on the simulation and real system application. In the first stage single agents and the resulting system are tested in the provided simulation environment. The final target is the worldwide unique testbed at the Au-

tomation and Control Institute (ACIN), where the developed system will be implemented and permanently tested in a real environment.

The paper is structured as follows: In the next section the methodological approach is introduced. The third section is concerned with the MAS design phases. In section four, we describe the diagnostic capabilities of agents controlling physical components of an automation system, gained through our design approach. Section five presents the MAS testing framework, and the sixth section describes the procedure of system configuration. Finally, section seven concludes this paper and provides an outlook.

2. Methodological Approach

MAS are systems composed of multiple software agents that interact with one another in order to achieve their intended goals and the goals of the systems as a whole. The specific nature of software agents, which are designed to be distributed, autonomous, and deliberative makes it difficult to apply existing software testing, monitoring, and diagnostics techniques to them. For instance, agents operate synchronously, in parallel and concurrently, what can lead to non-reproducible effects [20]. It is not ensured that two executions of the systems will result in the same state, even if the same inputs are used. As a consequence, looking for a particular error is difficult, as it is usually impossible to reproduce in a systematic fashion [13]. Besides, agents communicate primarily through message passing, each owning its own data, and since systems can be made up by hundreds of agents also the amount of data can be a serious issue [12]. For these reasons agent testing and monitoring systems are needed that allow developers to systematically and automatically test agents, the functionality of the agent network, and to monitor the agents after they are deployed to ensure that they continue to operate correctly [24].

Several methodologies have been proposed for building MAS. The GAIA methodology has been presented by Wooldridge et al. [32], MESSAGE by Caire et al. [6], MaSE by Wood and DeLoach [31], Prometheus by Padgham and Winikoff [21] and Tropos by Bresciani et al. [2]. Our approach is fundamentally based on the Designing Agent-based Control Systems (DACS) methodology [3]. DACS is of particular interest for us since it provides a method for analyzing the production control problem bridging the gap between the domain of production control and agent-based systems. It covers all agent-oriented design steps from analyzing the production problem, through identifying the control agents, to re-using existing interaction protocols.

Building scalable and reconfigurable transportation systems is a big challenge, since its efficiency is highly influenced by the information flow between the heterogeneous agents within such systems. In our approach, we apply a MAS approach to a transportation system in order to investigate efficiency, fault tolerance, and stability of the system. The system should be able to transport pallets between different destinations in the system and robustly react to dynamic changes (e.g., failures of single entities) in a way that it does not affect the system functionality and/or significantly decrease its performance. When a component (e.g., conveyor, intersection, etc.) breaks down, the transported parts are usually blocked and have to necessarily wait for the component to be repaired. Additionally, if the destination machine is broken, the system should be able to autonomously reroute the parts to other, still functioning, machines. If the system is inflexible and therefore not able to adequately reroute these parts, additional expenses are caused that again reduce the profitability of the production process [28]. Furthermore, balancing the workload between parallel machines, considering the current system state, has been recognized as a potential way to flexibly schedule appropriate operations. Particularly in complex environments, this approach can maximize the overall system throughput, minimizing the work in process, flow time, and makespan [23]. It can increase resource utilization, hence improve productivity, and at the same time can help to avoid bottlenecks and to schedule unavailable resources. The production system should be able to change quickly and cost-effectively from its current configuration to another configuration without being taken off-line and balance the workload between available resources avoiding preventable limits and improving system output. The influence of the transportation time and machine utilization on the system output was investigated separately from this research [19].

The core components of the proposed transportation system are the Automation Agents (AAs), which are responsible for controlling the physical resources [27]. These agents have to fulfill their tasks under specific conditions, as applied in most existing MAS deployments, in order to meet diverse requirements. The essential knowledge about the domain is made available to the agents through an explicitly defined ontology. The agent control architecture clearly separates the control software into two layers: the high level control (HLC) and the low level control (LLC). On the one hand the so-called HLC layer is responsible for the control of the global behavior of the agent in order to ensure both the achievement of its own goals and the coordination with other agents of the system. High-level decisions are influenced by the present and past observed states in the environment, can be calculated in a longer time period and usually have a certain time buffer for execution. On the other hand LLC layer is responsible for directly controlling the physical component by making use of a limited set of reactive behaviors thus directly supervising the components actions and informing the high level about its state. The description of the simulation and real testbed will be presented in the Section 5.

3. Multi-Agent System Design Phases

Agent-oriented software development involves multiple disciplines, for example software engineering, cognitive science, social science, and artificial intelligence. In this project, we adopt the software engineering and software testing perspective on MAS. Software testing is a software development phase to evaluate the product quality and enhancing it by detecting errors and problems by executing the system and its components under specified conditions. The test results are observed or recorded and compared against specifications or intended results [11]. In this paper, we are presenting a monitoring and testing approach for software agents and MASs able to specify and handle their peculiar nature. The technique tends to be effective and adequate to evaluate agent's autonomous behaviors and build confidence in them. Our development technique is based on monitoring and testing a MAS and it is divided into four phases: phase one is aimed at particular behaviors of an agent, the second one focusing on the functionality of a single agent as a whole; the third one dealing with the entire MAS; and the fourth one aiming at validation in real-world environments. In other words, at first we use simulation for our testing approach and then we make investigations in a real-life environment. The key advantage of our approach is the ability to test the MAS in a real system, contrary to the current practice for agents development based only on simulation tests.

In the ***first phase***, after we identified the number and types of the agents, we proceed with creation of the agent structure and knowledge for each specific agent type in the system. We also specify the inter-agent relations and the required interaction protocols. The next step in the development of the agent test and verification framework is to define a test specification language, which should be an easy to learn but nonetheless a formalized and unambiguous language. In this basic phase, we perform monitoring and testing of each single behavior of the individual agent. We test whether results of the behaviors corresponds with to design objectives, in a similar way as classical unit tests used in software development are designed and performed. Agent behavior testing is a basic part of agent development, especially when test-driven development methodology [1] is employed.

In the ***second phase*** testing of the functionality of the entire agent we validate if the design is adequate to achieve the agents aims. We search for the errors coming from the agent interactions, because even if the individual agent is working properly, hidden errors could still show up during the interaction of two or more agents. In this phase, we also perform the analysis of the knowledge structure, preparation of validation scenarios, and at the end application of test scenarios to evaluate the software. At the beginning of the validation by simulations, the problem solving and interaction abilities of the agents are investigated. After the instantiation of specified agents and setting related parameters, simple tasks are used to check for incompleteness of the agents knowledge. The resolving of potential inconsistencies, redundancies, and conflicts in the agents knowledge base enables to prevent possible future occurrences. Furthermore, these tests are used for testing and validation of the simulation environment itself. Figure 1 shows a simple test scenario for the testbed, as well as the prose and the formalized form of the test case specification. The test case consists of an "initial condition" section as well as a "goal" section. The MAS is in charge to transit from the initial state defined by the initial conditions to the goal state. Non-defined system variables are optimized by the MAS and are not explicitly defined in the goal state. The planning language used to define the test case is a simple subset of classic planning languages used in goal-driven agent planning [25]. The proposed language only consists of five predicates, two actions, and one percept, but it is sufficient to describe the transportation processes and the palette waiting times due to product processing at the handling stations. For the definition of the use case shown in Figure 1b) only the predicates are used for scenario description. In the second step a formal action sequence verification system will be explored, evaluated, and adapted for the validation of single agents. Goal-driven agents will be charged with a specific problem as shown in the test case in Figure 1. The solution, a resulting action sequence, will be validated, whether its execution leads to the desired goal state. Therefore the action/percept-sequence must be expanded from the initial test case description by the test system and compared to the result of the MAS operation, if the expected goal state is reached. Another issue is the evaluation of the solution-sequence quality (e.g., goal time, energy-consumption). In agent planning the usage of a relaxed problem based on the original problem as automatically derived utility functions for solving planning problems is a well-established approach [25]. This approach has to be evaluated for assessment of the generated action sequences.

The ***third phase*** constitutes testing the entire MAS. Although we may find the individual agents as well as their interactions error-less, the whole system could still fail. On this level, performance problems and bottlenecks of system hubs, and vulnerability to mass collapses of agents could appear. In the system test phase, real life examples as well as routing trainings are used to test and validate the distributed systems functionality. The agility and performance of the architecture are studied simulating different scheduling scenarios as well as the failure of system components, particular agents or entire system parts for example conveyer loops. The results of the simulation are used as an input for the MAS improvement before its deployment in real-life control. The knowledge of agents is improved constantly, demonstrating the extensibility of the system. Such testing leverages the diagnostics features (see Section 4) designed with the aim of providing the detection and explanation of the incorrect system behavior. The simulation also offers the possibility to test different

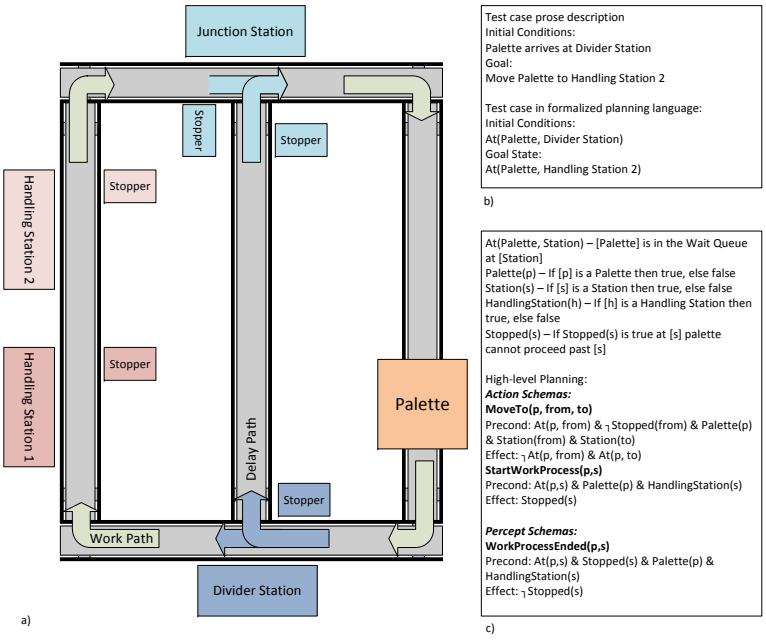

Figure 1. MAS based transportation system a) Schematic Test Scenario on the Testbed b) Test Case description c) Simple test definition language

strategies much faster, due to the possibility of accelerated execution of the MAS, than it would be possible in the real testbed. Moreover, the relations between different manufacturing parameters, such as throughput, tardiness, capacity, complexity of products assembled, and so on, could be established much more effectively, at lower costs and what should not be underestimated safer.

The ***fourth phase*** is the most challenging since according to our best knowledge only two real life experiments with MASs were performed until now [4, 16, 3]. Besides, there is no guarantee that experiments in the simulation are directly comparable with those using real world equipment. MASs are very rich in topics, such as distributed control, scheduling, adaptation, cooperation, inter-agent communication and negotiation, and so forth. Put simply, we cannot know or predict everything that agents might encounter in performing their tasks. The simulation model might not exactly correspond to the real system as some of the parameters might not be set properly or there might be slight deviations among real components of the same type, which are not considered during simulation. Thus, in order to validate the simulation study a real testbed is indispensable. In this phase, we will test the integration of the different modules inside an agent, testing agents capabilities to fulfill their goals, and to sense and effect the real environment. Furthermore, collective behaviors will be tested to make sure that a group of agents and environmental resources work correctly together. At the end, we will test the expected emergent and macroscopic properties of the system as a whole (in our case transporting of goods) testing also the quality properties that the intended system has to reach, such as adaptation, openness, fault-tolerance, performance. Finally, our approach concentrates on evaluating MAS quality focused tests regarding stability and scalability. Considering that a lot of theoretical work has been done in this field [9], we are making permanent real life long-run tests (10-20 days), where the agents and system are going to be confronted with different stressful situations (e.g., component break, system disturbance, introduction, and reduction of agents numbers) in order to test this MAS capabilities.

4. Diagnostics Framework

A significant innovation of the proposed approach is the design of a general framework for monitoring and diagnostics of the agent-based control system. The aim of the diagnostic system is to detect deviations from typical patterns of behaviors based on analysis of the current event flow. The general assumption is that the detected unusual sequence of events could signify a possible error or failure. In the example of the transportation system, certain sequences of sensors activation simply cannot happen, because of the system topology that physically prevents it to happen. For instance, the output sensor of a divider is activated and thus the palette should be detected in the subsequent index station, but suddenly appears somewhere else in the system. The proposed system is able to detect the occurrence of a deviation and possibly classify the type of the error. In other words the system should be able to identify types of errors which already occurred in the past. The diagnostics features are achieved with the design of internal diagnostics modules embedded in each agent as well as with the proposal of specialized di-

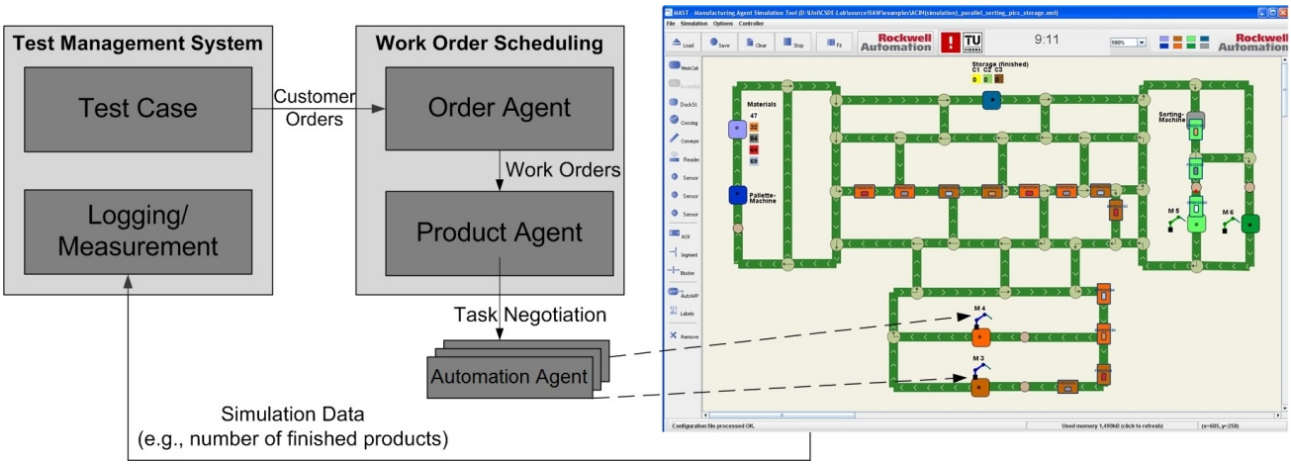

Figure 2. Overview System Architecture[17]

agnostic agents that concentrate data from single agents in order to grant the fault detection in the wider context of the controlled system. The key element of the proposed solution is combination of statistical layer, based on Hidden-Markov Models (HMM) capturing regular behavior and the employment of semantic techniques in order to give the agents a better tool for creating, handling and exchanging the information [9]. The ontologies, used to capture semantics, will be used for expressing the topology of the transportation system (how the components are connected with each other), for semantic description of the events (e.g., input of the palette, output of the palette), and for the semantic description of the deviations types, which will be used for further classification.

The natural modular character of MASs provides the opportunity for modularization of the diagnostic layer as well. In this case each agent with diagnostic module has its own HMM reflecting the agent model of the world. Agents of target distributed system have different diagnostic roles and this difference among agents is utilized during compilation of topology at the beginning of diagnostic system lifecycle. Hence, components can be divided into three categories agents providing diagnostic relevant data and owning HMM, agents providing diagnostic data, and agents without relevant data. Agents owning HMM are the components ensuring dynamically compilation of neighborhood. Neighborhood is set of surrounding agents and is defined by size of neighborhood and by a metrics on the agent system. Neighborhood N of agent x can be described as follows:

$$N(x, \epsilon) = \{y : d(x, y) \leq \epsilon\}$$

where d is distance between agents (e.g. Euclidian distance) and ϵ is size of neighborhood. Concept of neighborhood defined in this way gives us ability to affect its shape due to different rating of distance among agents. Compilation process begins by sending a message with request to neighborhood compilation from the central component to its successors. Subsequently, the successors are forwarding the request to its successors until the predefined size of

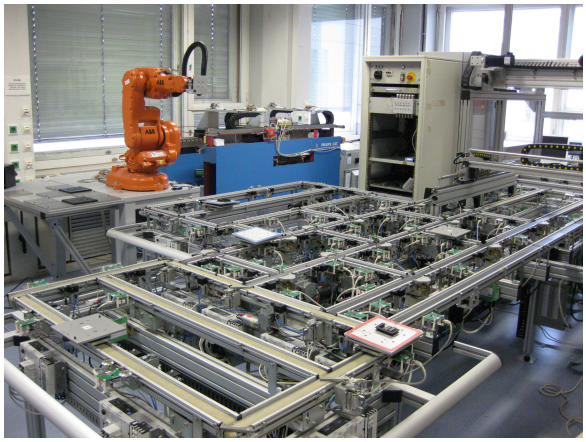

Figure 3. Testbed

the neighborhood is bigger than actual. Each agent send back to the central agent information about provided diagnostic data after allocation of neighborhood. These data are needed to estimate parameters of model (e.g. number of hidden states, the data dimensions, limitations in transition probabilities in due to character of system parts) therefore central agent is able to establish a HMM from these collected data.

The visualization system complements the diagnostic system in order to notify the operator about detected errors or deviations from normal operation. It shows schematically the components and the context in which they operate, the producer-consumer relations and the flow of notifications among them. The key functionality is to show where the error happened and to provide a playback of the events flow that led to this error. The visualization takes the advantage of the semantic description of events and failures, abstracting from the low-level data coming from sensors and actuators.

5. Framework for Testing Multi-Agent Systems

In our MAS architecture for manufacturing control of transportation system (see Figure 3), each system component of the pallet transfer system such as a robot, divider, junction, conveyor, pallet, and handling station is represented and supervised by a corresponding AA. We also introduce the Order Agent (OA), which is responsible for receiving product orders from customers and guarantee their accomplishment. The OA decomposes the product order into work orders, where each work order step represents a single machine operation (e.g., welding, drilling, painting) which has to be done to finish the product. Each work order is forwarded to the corresponding Product Agent (PA) that negotiates with the related AAs that provide particular operations. The overall system architecture consists of three main parts: the work order scheduling, the MAS simulation and the Test Management System (Figure 2).

5.1. Test Management System

The Test Management System (TMS) [18] is an extension of the simulator providing means for automatic execution and evaluation of a predefined set of simulation experiments. Each experiment tests a specific scenario with a different set of input parameters (e.g., the number and type of products to be transported, the work-flow scheduling strategy to be applied, and the number of pallets to be used). In order to execute the set of evaluation scenarios, the simulation system is reset into a well-defined initial state – the configuration of the workshop is loaded from a configuration file and the components of the workshop as well as their controlling agents are created. As a next step the XML file containing the description of evaluation scenarios is loaded and the first scenario is injected into the system. The TMS uses the input parameters of the scenario to automatically run the simulation experiment. All relevant events like finalization of a product or occurred failure are logged to an output XML file. Once the experiment is finished the agent system automatically is reset into the initial state, the next scenario is selected, and the corresponding simulation experiment is conducted.

5.2. Simulation

The Manufacturing Agent Simulation Tool (MAST) provides a unique combination of multi-agent based manufacturing control system, and a simulator of the manufacturing environment used to verify the functionality of the agent-based control system.

The agent subsystem provides the run-time environment for particular agents, where each agent instance presents and controls the behavior of a specific manufacturing component. There is no central decision making authority in the system – all the control logic and knowledge related to the current state of the production process is fully distributed among the agents. Each agent is aware only of its immediate neighbors and uses message sending as form of information exchange and cooperation. The programming language for the agents as well as for the rest of MAST application is Java. The agent's run-time environment is provided by the open source agent platform JADE[1]. The visualization module displays the status of the simulation in a GUI. The user can watch the transportation of products and check the decisions of agents concerning the alternative routing in case of artificially introduced failures. The real-life control capabilities of MAST agents have been verified on the physical palette transfer system located in the Odo Struger Laboratory at the Vienna University of Technology (VUT)s ACIN [30].

5.3. Logistics and Transportation System (Testbed)

The testbed (Figure 3) architecture consists of a pallet transfer system with redundant paths, the industrial robot ABB IRB140 as well as a portal robot (see Figure 3). Robots and the handling unit are considered as autonomous entities able to perform certain operations. The pallet transportation system provides a complex logistic system consisting of several redundant paths. This pallet transfer system already employs adaptive distributed control paradigms performed by 120 controllers including RFID units for the pallet identification (current typical distributed testbeds have about five controllers). Experiments with this transport system have shown the advantages of our distributed control approach. To deploy our agents, we use control modules for CPX valve terminals of the type CPX-CEC-C1 by Festo[2] This type of controller hosts an XScale-PXA255 agile Intel microprocessor with 400 MHz, 28 MB Flash and 24 MB RAM. Each agent was placed on a separate controller. The 4DIAC[3] FORTE[4], a portable C++ implementation for small embedded control devices, was employed as the IEC 61499 [33] run-time environment allowing the execution of the LLC, which is an IEC 61499 function block (FB) network. The 4DIAC-IDE engineering environment was used for modeling the IEC 61499 application, which was then downloaded to the CPX-CEC-C1 controller.

6. Further Work

To offer facilities for better understanding of MAS and provide the foundations for their broader application in the industry following research issues (RIs) should be tackled in the future:

RI 1: Investigation of functional behaviors and evaluation of the actions, considering both the individual AA and the entire MAS. There is the necessity to test components in an offline and in an online mode. The offline tests are testing the agents/system, during phases when it does not produce or transport goods for the daily business. Online tests are needed to test the components/system during

[1] Java Agent DEvelopment framework

[2] Festo AG & Co. KG., http://www.festo.com

[3] 4DIAC-Consortium, 4DIAC - Framework for Distributed Industrial Automation and Control, open source initiative, www.fordiac.org

[4] 4DIAC Run-Time Environment

the production or transport process. Only the combination of online tests and offline tests brings the best result. Consequently all the results of a test run have to be evaluated. Hence, a question is if the goal-state is reached and how can agent oriented software testing methods verify these characteristics? Further questions are how to judge if agents and the resulting emergent system behavior is correct and how to check the mutual relationship between macroscopic behaviors and agent's individual behaviors?

RI 2: What is a suitable way for specifying test cases of an agent-based system so that the inputs for agents/system test execution system can be generated? The language for specifying test cases for agents should provide as much abstraction as possible, as tests will be specified by the appropriate domain experts. Therefore the test specification method should be easy to use for non-experts in testing and agent-based development. The test specification process should be fast with less effort, because it is an indicator if tests will be more used in the development process. The question is which methods should be developed to specify these tests and which language is appropriate to use?

RI 3: Identification of different methods and benchmarks for testing agents in a simulation environment vs. a real-world environment. At the end of the day the real-world system should be tested successfully. However, the real system is not always available and therefore the implementation has to be tested on a simulation environment. There is a need for a test framework including a test environment to check the implementation in both application domains, within the simulation environment and on the real system. As a result several testing methods should be designed and developed to support the test process in the simulation environment and as well as on the real system. The questions here are how to verify the MAS from a human tester's subjective perspectives and which benchmarks should be defined to assess the qualities of the MAS under simulation/real system tests, such as safety, efficiency, stability, scalability, etc.? Further question related to the results of these tests is how to use them to build confidence of the developers and the end-users in autonomous agents?

RI 4: Investigation the effectiveness and limitations of the agent control system when controlling the subsystems involved in the production process. This includes the analysis of effectiveness (i.e., ability to autonomously accomplish particular operations), robustness (i.e., correctness of the processes, e.g., reliability in improper or stressful environments when applying the system to a set of evaluation scenarios) and efficiency (i.e., steps or resources needed to achieve a given process result) of the approach. Here is the identification of constraints that subsystems (e.g., sensors, actors) set on the agents world model and its representation (i.e., how to identify and specify particular equipment activities and states such as moving or busy) as well as the determination an optimal control configuration for a specific task and calibration the related components important.

RI 5: Visualization of the agents behavior in the testing process and in real-world application. Through the difficult understanding of the MAS's emergent behavior a visualization and interaction with the operator can be the solution. It should visualize the planned activities and the results of the execution. This visualization is not only useable for monitoring of the systems behavior, but also applicable for supporting the test process and the diagnosis functionality. If the diagnostic system detects a deviation from the common behavior pattern, the human operator's expertise can be brought into the process to indicate the seriousness of the detected situation. Furthermore, it is of vital importance to set the preconditions and offer the human user the complete overview in the agents activities as well as a possibility to supervise its actions through an adequate HMI. This would improve the safety and on another side enable a user to follow and understand the current state of the process. This can further improve the missing trust in the idea of delegating tasks to autonomous agents, especially considering emergent behaviour of the overall agent control system.

However, visualizing overall system behavior in a distributed control system is a notoriously difficult task, since each agent in the system has only a local view of the organization and the user has to integrate the large amounts of information provided by individual agents. The questions are how to represent the dynamic nature of software agents and MASs, as well as to visualize their behaviors and actions?

7. Conclusion

The multi-agent approach has been widely recognized as an enabling technology for designing and implementing the next generation of distributed intelligent manufacturing systems. Nevertheless, the applications of agent-based control systems developed in the industry are very rare and the implemented functionalities are usually limited. We presented an integrated approach for the development, testing, and validation of MASs in simulations and real-world applications. Furthermore, this development approach enhances the agents diagnosis capabilities, and appropriate graphical user interfaces will support the plant operators in understanding the current state and behavior of the systems — and by thus further increasing confidence in MASs.

Acknowledgments

The research leading to these results has received funding from the European Union's Seventh Framework Programme (FP7/2007-2013) under *grant agreement* n° 284573.

References

[1] Beck. *Test Driven Development: By Example*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA,

2002.

[2] P. Bresciani, A. Perini, P. Giorgini, F. Giunchiglia, and J. Mylopoulos. Tropos: An agent-oriented software development methodology. *Autonomous Agents and Multi-Agent Systems*, 8(3):203–236, 2004.

[3] S. Bussmann, N. Jennings, and M. Wooldridge. *Multiagent systems for manufacturing control: a design methodology*. Springer series on agent technology. Springer, 2004.

[4] S. Bussmann and K. Schild. An agent-based approach to the control of flexible production systems. In *Proc. 8th IEEE Int Emerging Technologies and Factory Automation Conf*, volume 2, pages 481–488, 2001.

[5] G. Caire, M. Cossentino, A. Negri, A. Poggi, and P. Turci. Multi-agent systems implementation and testing. In *In Fourth International Symposium: From Agent Theory to Agent Implementation*, pages 14–16, 2004.

[6] G. Caire, W. Coulier, F. Garijo, J. Gmez-Sanz, J. Pavn, P. Kearney, and P. Massonet. *The Message Methodology*, volume 11 of *Multiagent Systems, Artificial Societies, and Simulated Organizations*. Springer US, 2004.

[7] G. Candido and J. Barata. A multiagent control system for shop floor assembly. In V. Mavrk, V. Vyatkin, and A. W. Colombo, editors, *HoloMAS*, volume 4659 of *Lecture Notes in Computer Science*, pages 293–302. Springer, 2007.

[8] M. A. de Cerqueira Gatti and A. von Staa. Testing & debugging multi-agent systems: A state of the art report. Technical report, Departamento de Informatica, PUC-Rio, Rio de Janeiro, 2006.

[9] P. De Wilde and G. Briscoe. Stability of evolving multiagent systems. *IEEE Transactions on Systems, Man, and Cybernetics - Part B: Cybernetics*, 41(4):1149–1157, 2011.

[10] A. Helsinger, M. Thome, and T. Wright. Cougaar: a scalable, distributed multi-agent architecture. In *Proc. IEEE Int Systems, Man and Cybernetics Conf*, volume 2, pages 1910–1917, 2004.

[11] Z. Houhamdi. Multi-agent system testing: A survey. *International Journal of Advanced Computer Sciences and Applications*, 2:135–141, 2011.

[12] Z. Houhamdi and B. Athamna. Structured system test suite generation process for multi-agent system. *International Journal on Computer Science and Engineering*, 3:1681–1688, 2011.

[13] M.-P. Huget and Y. Demazeau. Evaluating multiagent systems: a record/replay approach. In *Proc. IEEE/WIC/ACM Int. Conf. Intelligent Agent Technology (IAT 2004)*, pages 536–539, 2004.

[14] N. Jennings and S. Bussmann. Agent-based control systems: Why are they suited to engineering complex systems? *Control Systems, IEEE*, 23(3):61 – 73, june 2003.

[15] P. Leitao. Agent-based distributed manufacturing control: A state-of-the-art survey. *Engineering Applications of Artificial Intelligence*, 22(7):979 – 991, 2009.

[16] F. P. Maturana, P. Tichý, P. Slechta, F. Discenzo, R. J. Staron, and K. H. Hall. Distributed multi-agent architecture for automation systems. *Expert Syst. Appl.*, 26(1):49–56, 2004.

[17] M. Merdan, T. Moser, P. Vrba, and S. Biffl. Investigating the robustness of re-scheduling policies with multi-agent system simulation. *The International Journal of Advanced Manufacturing Technology*, 55:355–367, 2011.

[18] M. Merdan, T. Moser, D. Wahyudin, S. Biffl, and P. Vrba. Simulation of workflow scheduling strategies using the mast test management system. In *Proc. 10th Int. Conf. Control, Automation, Robotics and Vision ICARCV 2008*, pages 1172–1177, 2008.

[19] M. Merdan, M. Vallee, W. Lepuschitz, and A. Zoitl. Monitoring and diagnostics of industrial systems using automation agents. *International Journal of Production Research*, 49(5):1497–1509, 2011.

[20] C. D. Nguyen, A. Perini, C. Bernon, J. Pavón, and J. Thangarajah. Testing in multi-agent systems. In *AOSE*, pages 180–190, 2009.

[21] L. Padgham and M. Winikoff. Prometheus: A methodology for developing intelligent agents. 2585:174–185, 2003.

[22] M. Pechoucek and V. Mark. Industrial deployment of multi-agent technologies: review and selected case studies. *Autonomous Agents and Multi-Agent Systems*, 17:397–431, 2008.

[23] S. Rajakumar, V. Arunachalam, and V. Selladurai. Workflow balancing strategies in parallel machine scheduling. *The International Journal of Advanced Manufacturing Technology*, 23:366–374, 2004.

[24] C. Rouff. A test agent for testing agents and their communities. In *Proc. IEEE Aerospace*, volume 5, pages 5–2638, 2002.

[25] S. J. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Pearson Education, 2003.

[26] K. P. Sycara. Multiagent systems. *AI Magazine*, 19(2):79–92, 1998.

[27] M. Vallee, H. Kaindl, M. Merdan, W. Lepuschitz, E. Arnautovic, and P. Vrba. An automation agent architecture with a reflective world model in manufacturing systems. In *Systems, Man and Cybernetics, 2009. SMC 2009. IEEE International Conference on*, pages 305 –310, oct. 2009.

[28] M. Vallee, M. Merdan, W. Lepuschitz, and G. Koppensteiner. Decentralized reconfiguration of a flexible transportation system. 7(3):505–516, 2011.

[29] P. Vrba and V. Marik. Capabilities of dynamic reconfiguration of multiagent-based industrial control systems. *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, 40(2):213–223, 2010.

[30] P. Vrba, V. Marik, and M. Merdan. Physical deployment of agent-based industrial control solutions: Mast story. In *Proceedings DHMS 2008*, pages 133–139, 2008. Vortrag: IEEE SMC International Conference on Distributed Human-Machine Systems (DHMS), Athens, Greece; 2008-03-09 – 2008-03-12.

[31] M. Wood and S. DeLoach. An overview of the multiagent systems engineering methodology. 1957:1–53, 2001.

[32] M. Wooldridge, N. R. Jennings, and D. Kinny. The gaia methodology for agent-oriented analysis and design. *Journal of Autonomous Agents and Multi-Agent Systems*, 3:285–312, 2000.

[33] A. Zoitl. *Real-Time Execution for IEC 61499*. ISA, 2008.