Architectural Issues in Implementing a Distributed Control System for an Industry 4.0 Prototype

Puscasu Samuel, Voju Radu Alexandru, Matei Alexandru, Zamfirescu Bălă Constantin
Lucian Blaga University
Sibiu, Romania
samuel.puscasu@ulbsibiu.ro, radu.voju@ulbsibiu.ro, alex.matei@ulbsibiu.ro, constantin.zamfirescu@ulbsibiu.ro

Abstract — This paper introduces the concepts of an architecture for a distributed control system of an industrial production line. The system is based on a multi-agent system that controls the hardware components using a network of low-level controllers. The low-level controllers are programmed using the new standard IEC61499. This architecture will be tested on a preexisting demo production line.

Keywords—distributed control; multi-agent system

I. INTRODUCTION

According to [1], considering the dynamics of industrial automation these days, it would be better to increase the flexibility and adaptability of the production systems. Another direction would be to reduce the system design costs by using fewer engineering tools to model each system or sub-system. As stated in a review by Thramboulidis K. [2], the applications developed under IEC61499 are not welcomed yet, as it should be, in the world of integrators from system automation environment in production industry from different parts of the world. When IEC61131-3 standard appeared more than 20 years ago, it has created a real development of industrial automation taking into consideration latest trends. The real objective of the new IEC61499 standard is to overcome the technical limitation of the old standard by introducing new concepts that are suitable for multi-agent architectures that are capable to adapt to new processes and constraints changes [2].

When comparing the new IEC61499 and the older IEC61131-3, the main three advantages it has are based on three principles [3]:

- portability - the possibility to import elements developed under different software platforms,
- configurability - regardless of software tools used to have the opportunity to configure and parameterize different hardware,
- interoperability - the possibility to collaborate under a distributed application.

It was demonstrated that the old IEC61131-3 was unable to sustain these three principles. Therefore, the new standard had to follow these principles [3].

The results from applying this standard in cyber-physical production systems show that agent-based control is the most suitable approach [4] [5] [6]. An agent is described as an autonomous computational entity, with social skills, capable of efficient behavior, reactive or/and proactive, that has the purpose of solving specific problems inside dynamic open environment [7]. All the actions that an agent takes are aimed to achieve his own goals. Based on [8], from the set of characteristics that an advanced agent must have, we can mention autonomy, reactivity, proactivity, goal-directed behavior, social skills and learning capabilities. In multi-agent systems, the agents can be classified into two types of agents, in respect to their behavioral characteristics: deliberative agents (cognitive) and reactive agents [9], [1].

A multi agent system (MAS) is a structure of multiple agents, fully or semi-autonomous, which will collaborate to reach the global objective by pursuing each agent own objective. Solving complex problems from real world usually requires many agents that must collaborate to be able to solve a problem much easier and more efficient then a centralized control solution. In order to make it work, basic principles must be taken into consideration such as communication, collaborative structure, negotiation and delegation of responsibility, all based on its own rational and social intelligence of the involved agents [10]. The global behavior of a MAS is defined by the emergent actions between agents, resulting that the capabilities of a MAS will follow the capabilities of all agents that compose that MAS.

The rest of this paper is structured as follows. Section 2 introduces the architectural issues considered in the DiFiCIL project. Section 3 describes the Industry 4.0 prototype, while Section 4 presents the current software implementation for both high-level MAS and low-level distributed control system based on IEC61499. The final section concludes this research and proposes further research directions.

II. ARCHITECTURE PROPOSAL

A domain model is an abstract representation of the behavior and data present in a specific domain, including the important knowledge of that domain. This is a representation of important concepts from the real world, that will also be used in software modelling and development. These concepts are provided by real world data involved in the system including their rules of functioning. A domain model is using a domain specific vocabulary. With the help of the domain

This work is supported by the DiFiCIL project (contract no. 69/08.09.2016, ID P_37_771, web: http://dificil.grants.ulbsibiu.ro), co-funded by ERDF through the Competitiveness Operational Programme 2014-2020.

978-1-7281-6870-8/20/$31.00 ©2020 IEEE

specific vocabulary, it is possible for other persons that are not familiar with the domain to understand the model and its representation. Model creation does not imply details of technical implementation, like databases or software components.

The architecture proposed in Fig. 1, is developed in the DiFiCIL project and it is composed by three distinctive parts: Client, Orders and Production Line.

Client side – consists of a software application that runs on a mobile platform and represents the user interface with the human client. Using this application, the client can create an order, customize the product to its own taste, modify an existing order, monitor the order status or it can cancel an order.

Orders – it is composed by Order Management Agent (OMA), Order Tracking Database, Order and Order to Task Translator.

- **OMA** – it is an agent that is used to manage orders, and will create an Order Agent for each unique order,
- **Order Tracking DB** – represents a network resource used to save and monitor the status of each order.
- **Order** – it is a data structure, created by **OMA**, instantiated for each order created.
- **Order to Task Translator** – it is an agent that divides the orders into smaller tasks, using a recipe provided by Knowledge Management Agent (KMA). These tasks need to be simple, so they can be executed by working agents.

Production Line – is composed by:

- **Yellow Pages Agent** – it is used to facilitate communication among agents, as there is no direct communication between worker agents. In this way, a safe and reliable communication is assured between all the other agents on the line, by using FIPA Interaction Protocol [11] that is similar to a publisher-subscriber communication model.
- **Agent Platform** – represents a virtual environment that connects all agents together. The architecture is designed to run on a distributed system, but the physical structure is compact enough to assure that multi-agent system will run on a single computing device.
- **Augmented Entity** – each physical component from the line must have a digital model. The two entities, digital and physical form together the Augmented Entity.
- **Worker Agents** – represent the digital entity of the Augmented Entity (Testing station, Transporter, Warehouse, Assembly station)
- **Task** – it is a data structure instantiated and managed by Task Management Agent (TMA) in collaboration with Order to Task Translator.
- **Client Agent** – it is an interface agent for each human user that interacts with the multi agent system
- **KMA** - Knowledge Management Agent will assure management of the entire knowledge (production recipes, predicates, etc.) organized into ontologies. This agent is the main component in the domain of information organization.
- **Order Agent** – it is an agent that is instanced each time an order is placed and it is destroyed whenever the order is fulfilled.
- **TMA** – Task Management Agent it is an agent, which will handle the tasks created by Order to Task Translator, after they are stored into the database (Task DB).
- **Interface** – Are interfaces, which will assure communication in between working agents and devices from lower level, that controls the physical system.

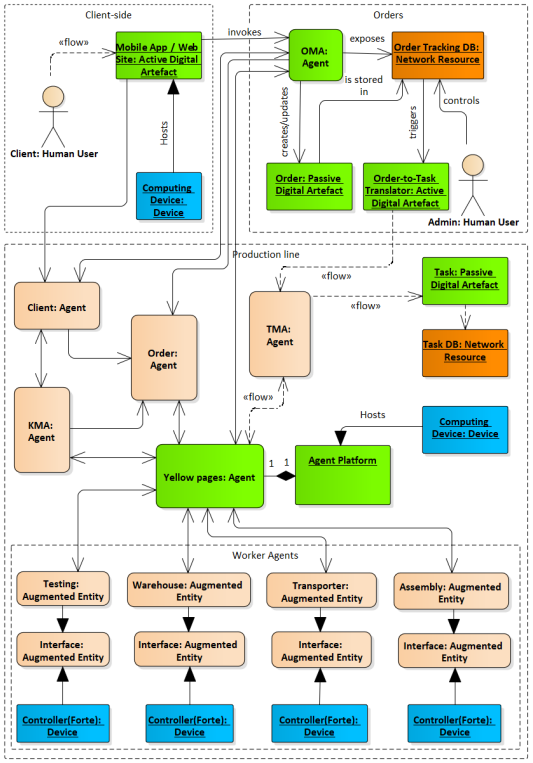

Fig. 1 The basic architecture employed in the DiFiCIL project

- **Controller** – Are running under Linux, which will permit a safe run of Forte runtime [12]. 4diac (Framework for Distributed Industrial Automation and Control) software tool is providing an open source software tool developed under IEC 61499 standard and allows easy programming of low-level controllers using Function Blocks (FB).

Because of a strong partnership between academic and industrial cooperation, there are several solid results as open and free tools to use: 4diac Runtime Environment (suits embedded devices), 4diac Integrated Development Environment (4diac IDE – as an engineering tool) and 4diac LIB as a library of FBs to be used for different industry branches.

Interaction diagram from Fig. 2 is describing in a simplified way, the steps needed to have an order placed and fulfilled in the proposed system. In this way, the interface agent with client (Client Agent) is launching an order directly to order manager (OMA). OMA will create an Order Agent for each order placed. The Order Agent is responsible to

generate the task list, using the order to Task Translator, which will later be published to notify all working agents.

Firstly, a check with the Warehouse Agent is made to see if the order can be fulfilled using the current stock. If there is enough stock available for the current order, a request for a Transporter Agent is made. With at least one Transporter Agent available, another check is made with all the other Worker Agents to see if they can execute the tasks from the task list. Only if all Worker Agents can execute the tasks, the Client Agent is notified that the order can be processed.

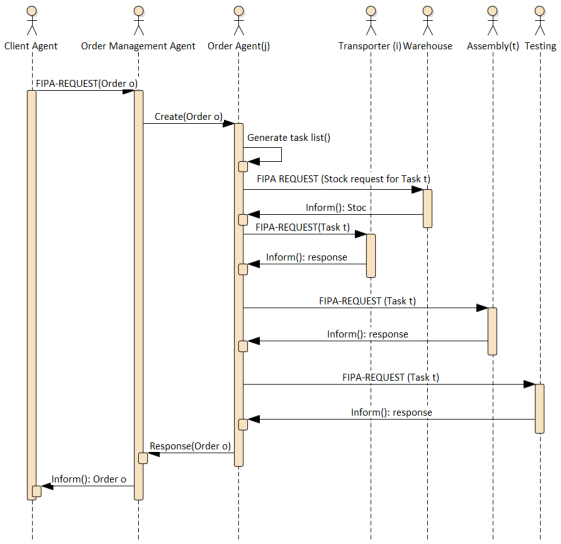

Fig. 2 Interactions diagram

III. CURRENT STATUS OF HARDWARE PROTOTYPE

The proposed architecture will be tested on an existing production line prototype - Fig. 3. The product (memory stick manufactured from three subcomponents) has to be assembled, afterwards transported, tested and in the end delivered to the final customer.

Another important aspect of the product that is assembled is that before confirming an order, the customer has to choose what configuration to be assembled. There are several colors available for the subassemblies of the product, which the system has to take into account, in order to deliver the customized product chosen by the customer.

Warehouse will hold the manufacturing part needed for assembly. There, is a buffer for each type of part, where a human operator will need to provide each type of part in the specific buffer with different colors of their housing. Pneumatic cylinders, controlled by the low-level software, are sorting the parts on a random-access temporary buffer, based on the recipe of assembly sent by the customer. Parts sorting was done using a TS20 Staubli robot.

Transfer of the manufactured part from assembly nest to transport units is done by the high speed Scara 4-axis robot manufactured by Staubli Robotics (model TS20), which will make the connection between the Warehouse and the transporting units installed on the closed loop circuit that is used to guide the transporter units to stopping stations. Transporter units - see Fig. 4 - are composed by their housing, a Raspberry Pi embedded underneath, a main power unit for the whole assembly and electrical motors to enable movement across the circular circuit of the manufacturing prototype. For full productivity purposes, there were three transporting units manufactured.

Fig. 3 Overview of the manufacturing prototype

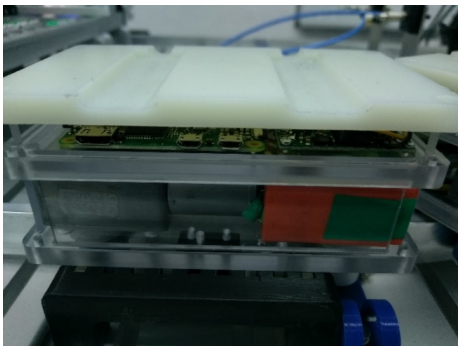

Fig. 4 Transporting unit

Last unit is the testing station that checks the full manufacturing process quality and that the parts are assembled correctly according to the specific customized order.

From the software point of view, the production line currently runs on 4diac for the transporting units and the 4-axis robots implemented. Warehouse low-level control system is implemented in 20sim, a modelling and simulation software that can generate the code to be deployed on the Raspberry Pi board including the code for the I/O extension board for assembly commands and color sensor inputs. A complete description of this implementation is given in [13] [14]. For the DiFiCIL project, this solution is not fully compatible with our desired objectives (distributed control, open source software, stability, reliability). The solution to solve this issue is to replace 20sim with 4diac while preserving the same advantages on our manufacturing machine (simplified communication, open source, stability).

This will solve the actual unstable communication to other systems merged inside this physical concept. Even though 4diac solution is not entirely mature yet, unlikely to be seen as a commercial solution sold into automation control world, but it would rather have an advantage to multi-agent platform designed here.

In order to integrate the proposed solution to the actual concept, two modifications must be made:

- A PC Server, to host the entire MAS.
- Electrical changes for Warehouse, in order to use 4diac.

IV. CURRENT STATUS OF SOFTWARE COMPONENTS

Currently, the transporter software is functional and it is developed in 4diac, accordingly to the new IEC61499 standard. Fig. 5 shows the functional blocks of the program.

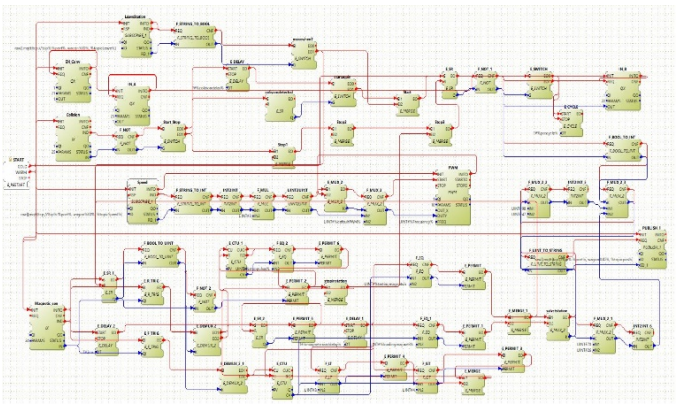

Fig. 5 4diac implementation of transporting unit

Interaction between each instance of the wagon and agents is achieved using messages. By far, the most important message is the "Speed" parameter, which is controlling the motion flow for each wagon. When the wagons are in motion to a specific point, the distances between each other are monitored constantly as a safety measure to avoid collisions. Considering a more client-oriented approach with customization enabled, the system allows to select one of the two levels of speed for wagons movement (normal mode or fast). Start of the transportation sequence will be fired up from the order interface and sent by the central system. Also, the order will be sent with the level of customization, which will select the level of speed of the wagons. After that command, motion is triggered and the actual position of the wagon is approximated taking into consideration two factors: if distance measured in front if the wagon is under safety distance or if a station is reached. If one of these two conditions are met, the stop motion command will be triggered. If the safety distance is reached, then the wagon will remain still until the safety mode is deactivated and the motion is being restarted until a station is reached. A more detailed explanation of the transporter system can be found in [13].

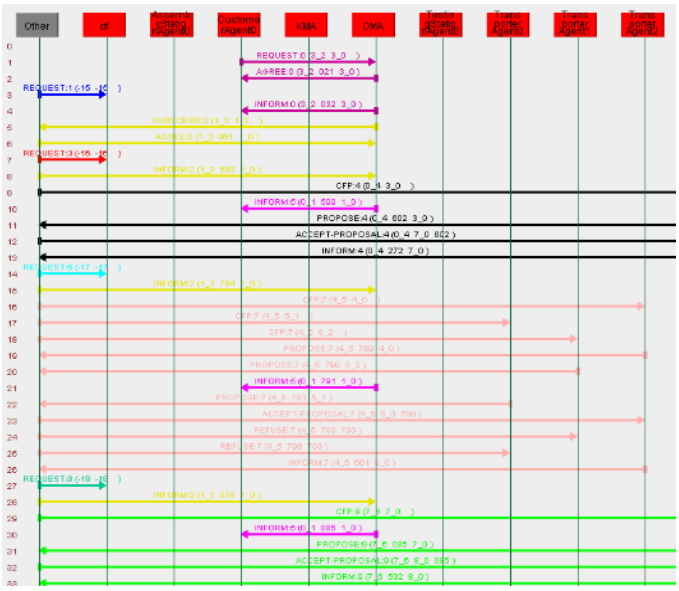

Fig. 6 Multi-agent interaction diagram in Jade

The MAS is currently developed up the simulation stage – the interfaces to the low-level controller that interacts with the hardware are missing. This simulation includes the following agents: OMA, KMA, Assembly Station, Transporter, Customer, Warehouse and Testing. The agents are using Fipa ACL messages to communicate among them. In Fig. 6 it can be seen an example of the sequence of messages exchanged in the MAS when an order is placed by a customer. Fig. 6 shows only the first messages because the completion of an order would generate, in average, about 200 exchanged messages.

V. CONCLUSION

This article presents a proposal of a distributed control architecture based on two layers of control, applicable to Industry 4.0. This architecture has been briefly explained, from software, hardware and human interaction point of view. Even if the physical system presented here is not a complex one, it is still a starting point for learning the stages of developing a real distributed system, with high complexity, where the only viable solution is the distributed control approach.

REFERENCES

[1] A. Zoitl, T. Strasser and G. Ebenhofer, "Developing Modular Reusable IEC 61499 Control Applications with 4Diac," 2013.

[2] K. Thramboulidis, "IEC61499 vs. 61131: A Comparison Based on Misperceptions," 2013.

[3] C. Sunder, A. Zoitl, J. Christensen, H. Steininger and J. Fritsche, "Considering IEC 61131-3 and IEC61499 in the context of Component Frameworks," Daejeon, South Korea, 2008.

[4] K. Kruger and A. H. Basson, "Multi-agent Systems vs IEC 61499 for Holonic Resource Control in Reconfigurable Systems," *Procedia CIRP,* vol. 7, pp. 503-508, 2013.

[5] L. Paulo, A. W. Colombo and S. Karnouskos, "Industrial automation based on cyber-physical systems technologies: Prototype implementations and challenges," *Computers in Industry,* vol. 81, pp. 11-25, 2016.

[6] W. Dai, V. N. Dubinin, J. H. Christensen, V. Vyatkin and X. Guan, "Toward Self-Manageable and Adaptive Industrial Cyber-Physical Systems With Knowledge-Driven Autonomic Service Management," *IEEE Transactions on Industrial Informatics,* vol. 13, no. 2, pp. 725-736, 2017.

[7] M. Wooldridge, An Introduction to Multiagent Systems, 2009.

[8] G. Weiss, Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence, London: The MIT press Cambridge, 1999.

[9] G. Mevludin, "Agents and Multi-Agent Systems: A Short Introduction for Power Engineers," University of Liege, Liege, BELGIUM , 2006.

[10] M. Vladimir, F. Martyn and P. Michal, "Holons & Agents: Recent Developments and Mutual Impacts," in *Part of the Lecture Notes in Computer Science book series (LNCS, volume 2322)*, Cambridge UK., 2002.

[11] "fipa.org," [Online]. Available: www.fipa.org. [Accessed 20 02 2020].

[12] "https://www.eclipse.org/4diac/," ECLIPSE, [Online]. Available: https://www.eclipse.org/4diac/en_help.php. [Accessed 11 02 2020].

[13] N. Mihai, C. B. Zamfirescu, P. G. Larsen, K. Lausdahl and K. Pierce, "Multi-paradigm discrete-event modelling and co-simulation of cyber-physical systems," *Journal Studies in Informatics and Control,* vol. 27, no. 1, pp. 33-42, 2018.

[14] M. Neghina, C. B. Zamfirescu and K. Pierce, "Early-stage analysis of cyber-physical production systems through collaborative modelling," *Software and Systems Modeling,* 2019.