DEVELOPMENT OF A MULTIAGENT SYSTEM IN THE INDUSTRY 4.0 CONTEXT - USING JACAMO AND APACHE CAMEL

Cleber Jorge Amaral, cleber.amaral@ifsc.edu.br[1]
Stephen Cranefield, stephen.cranefield@otago.ac.nz[2]
Mario Lúcio Roloff, roloff@ifsc.edu.br[3]

[1]Instituto Federal de Santa Catarina (IFSC – Campus São José), R. José Lino Kretzer, 608, São José, Santa Catarina, Brazil
[2]University of Otago, Union St E, North Dunedin, Dunedin 9016, New Zealand
[3]Instituto Federal de Santa Catarina (IFSC – Campus Florianópolis), Av. Mauro Ramos, 950, Florianópolis, Santa Catarina, Brazil

Abstract. *Industry has suffered three big revolutions. Nowadays, in the digital era, the factories are automated by logical controllers, supervisory systems, enterprise management software and robots. However, disruptive technologies are emerging and researches are confident to affirm that a new revolution is going to happen in a few years. In this sense, some emerging technologies will be fundamental to solve the challenges of the factory of future, the Industry 4.0, among those the Cyber-Physical Systems and Internet of Things. To meet this demand, this job presents a proposal of a software platform using JaCaMo, a Multi-Agent Systems (MAS) development framework, able to communicate with Internet of Things devices, supervisory systems and robots through a deployment using Apache Camel framework. The development details of an Apache Camel component and the performed tests of a multi-agent system developed are also presented in this job.*

Keywords: *Smart Factory, Cyber-Physical System, Industry 4.0, Industrial Internet*

1. INTRODUCTION

In history, industry has suffered three big revolutions which systematically led the production from artisan manufacturing to modern automated factories. Nowadays, in the digital era, Programmable Logic Controllers (PLCs) are being used in factories supervised by Supervisory Control and Data Acquisition (SCADA) systems. Top factories are managed by Enterprise Resource Planning (ERP) software, the machines are connected in industrial networks and robots are widely used.

Besides this forefront scenario many researchers agree that the forth revolution is coming, for the first time in an announced way, which is called Industry 4.0. Emerging technologies are being established and driving the creation of the “smart factory”. Cyber-Physical Systems (CPS) and Internet of Things (IoT) devices that spread in many parts of the production process are among the main technologies of this upcoming breakage (WANG et al.; 2016).

Roloff et al. (2016) proposes an approach in this sense, using JaCaMo, a Multiagent System platform to connect to legacy devices over ScadaBR, an OPC server (Figure 1). The main goal of this research was a proposition of a dynamic and flexible solution for Small Series Production (SSP), which is related to the capacity to produce small batch, what leads the reduction of setup time be a critical factor.

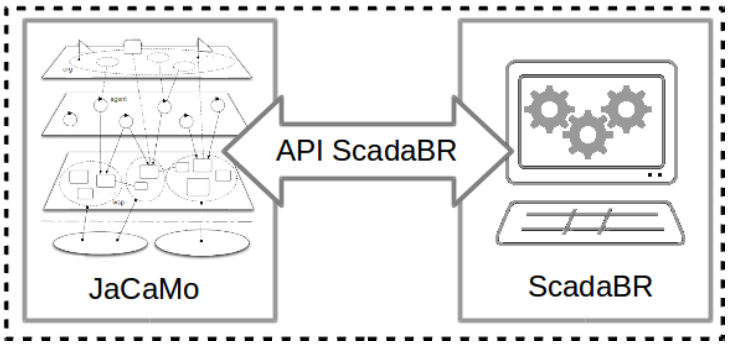

Figure 1. Representation of communication layer of Roloff et al. (2016) research

Besides SSP, the paper here presented is proposing that with some improvements Roloff et al. (2016) concept may be used to accomplish Industry 4.0 requirements, in other words, to be dynamic, flexible, decentralized, autonomous and able to work in real time.

To match these challenges the proposed approach is using JaCaMo and Apache Camel, a mediation framework to improve the integration capability of the MAS (Figure 2). This integration will happen using artifacts, presented in one

of the four dimensions that JaCaMo comprehends. The artifacts are non automata devices that we can find in an environment like sensors, machines, robots, conveyors, software and more. In this sense, JaCaMo is being proposed as a base software for Cyber-Physical Systems with high communication capabilities enabling it to communicate with many kind of devices as PLCs, robots and IoT devices.

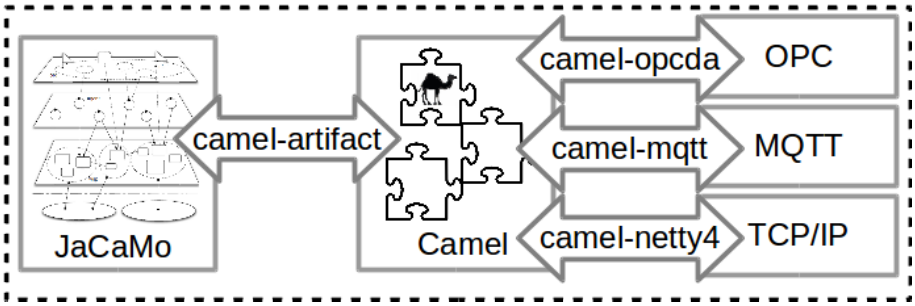

Figure 2. Representation of communication layer of this research

Therefore, this paper is organized to, in session 2, show the evolution of the industry and the requirements to accomplish the "smart factory". In session 3, Cyber-Physical System and Internet of Things concepts are addressed. With this contextualization, it is proposing the use of the frameworks JaCaMo and Apache Camel together in session 4. Then, in session 5, it is explained details of communication layer and the development of a necessary component to make possible the integration of artifacts by Camel Routes. Finally, in session 6, an application that was developed to test this approach is presented followed by session 7 that brings conclusions and final comments.

2. INDUSTRY 4.0 AND ITS REQUIREMENTS

Human being is always improving tools and the way to produce goods. This improvement process had many important inventions, but it was in England in XVIII century with the remarkable invention of steam machines and the development of more productive looms that the first industrial revolution started. After about a hundred years, with electric energy, new chemical products and later, with mass production system a new revolution happened. Finally, few decades ago we get to the digital era. The factories started to use devices connected in industrial networks, repetitive jobs were automated by PLCs and robots, and the companies started managing the inputs and outputs with wide software (Figure 3).

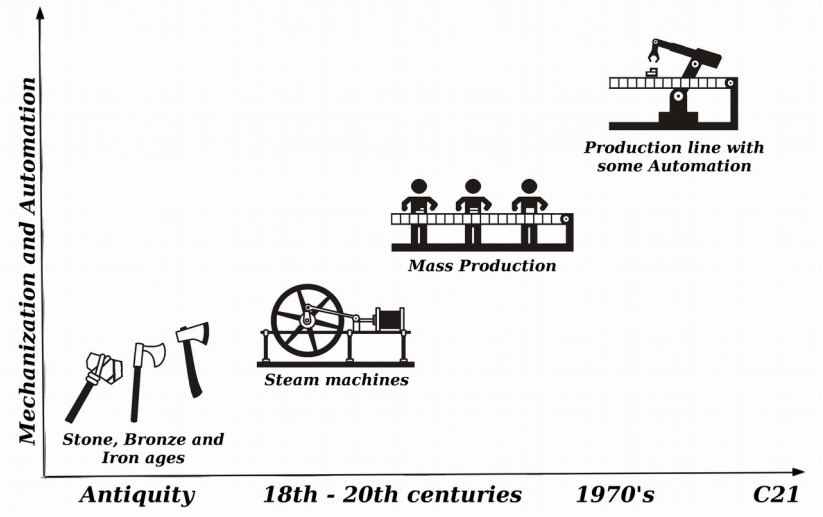

Figure 3. The evolution of Mechanization and Automation

Although, recent advances shows that a new revolution is coming. The term Industry 4.0 appears in Hannover's Industry Fair 2011, a report with recommendations to German government about emerging technologies for industry. It shows that due to disruptive emerging technologies the overall value chain will change dramatically, from inputs acquisition, logistic, and goods production to the delivery and even post sales services. Smart networks with autonomous devices will control trackable products in a modulated, flexible, decentralized and real-time way (LASI e KEMPER, 2014).

For Hermann et al. (2015) the "smart factory" has six fundamental principles:

- Interoperability: the system's capacity to communicate with each other with transparency, which means, in a way that both understand the message and interact about a common subject;
- Virtualization: presumes a computational modeling of the objects of real world. This "virtual copy" of palpable objects allows interaction of computer systems in the same level, in this sense, the physical object will receive commands when the software interacts with its virtual instance and the virtual instance will have its status modified by changes occurring in the physical object;
- Decentralization: in which productive cell, productive sector or production unit have their own plans and specific goals. The achievement of many goals as arrangement of necessary inputs, accomplishment of assembling process, the warehousing, in short, each of these stages must have autonomy to take their own decisions to reach the global mission;
- Real-time response: what is a current problem in the industry will be more critical in the smart factory due to the need of synchronicity of many processes;
- Services orientation: the applications must be available as services, using Service Oriented Architecture (SOA), those have connection interfaces for external systems. This format is very suitable for distributed systems which clients requires operations from service providers;
- Modularization: refers to production flexibility to attach and decouple modules of production plant in a "plug and play" way. With this feature it intents that the factory may easily be adapted to different demands of product models and production volumes.

With the deployment of these principles it is expected to prepare the production process with the capacity to generate updated information about the status of the machines, the process and the products. The information should flow to the interested nodes giving feedback to autonomous parts allowing them to react with the changes in the scenario, making their own decentralized decisions.

3. THE TECHNOLOGY CONCEPTS TO ACCOMPLISH "SMART FACTORY" PRINCIPLES

For Wang et al. (2016) the smart factory is the Cyber-Physical System (CPS) for manufacturing. This system integrates physical objects, logistic components and, managing systems. In this sense, two main concepts should be integrated to build the base of the factory of the future: Cyber-Physical Systems and Internet of Things.

Cyber-Physical Systems are computational models of the real world. Fundamentally, they should have high connectivity, which allows them to get data from physical world in real-time, as well as send commands that will trigger actions in real objects. In addition, they have computational intelligence and capability to assess and manage the data that build the virtual environment (LEE et al, 2015).

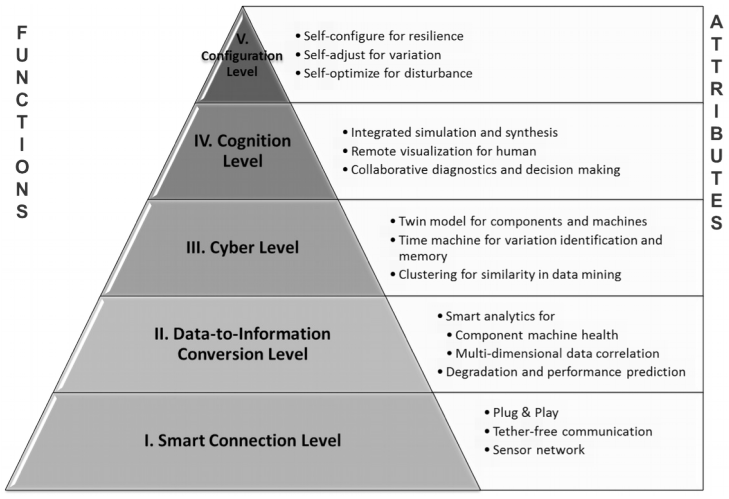

Figure 4. Cyber-Physical system in 5 levels. Lee et al. (2015)

Lee et al. (2015) suggests a structure in five levels to develop and deploy a CPS system (Figure 4). The first layer is about an intelligent connection making the devices get accurate and trustworthy data in an easy way (plug and play). The second level regard to data representation where the machines should be consciousness and react about the communication performance and data degradation. The next refers to the virtualization of the objects, which is the representation of real world. The forth layer is about cognitive capacity to take decisions based on previous knowledge

of the routines and status of the system. Finally, the last level acts over configuration of the machines making them adaptable to the changes.

For widely spread devices, the concept of Internet of Things is used here. This term is being used since 2013 to represent the trend to embed in all types of device communication modules that allows the device to connect in the network, have some processing capacity and being supervised and controlled.

Therefore, in the factories, these devices appear from the already connected machines, specially PLCs, robots and computer servers, to the widest kinds of sensors and actuators that are being used or going to be embedded in components of production, logistic and costumer relationship process. All of this will create a big mass of data, or "big data", that will need to be send by machine to machine protocols and be handled by artificial intelligence.

For PLCs, the primarily used protocol is OPC-DA (Open Platform Communications Data Access) which is mostly used to obtain information in real-time from the devices and deliver it to the supervising interfaces (MAHNKE et al., 2009, p2). To do that, interesting variables are pointed and arranged in items groups in the server (Figure 5). These variables are synchronized with physical devices inputs and outputs.

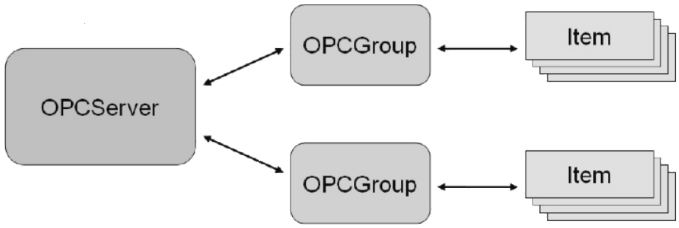

Figure 5. Gathered Items in an OPC Server. Mahnke et al. (2009)

The protocol is supported by most manufacturers and available in a large number of models. The interoperability provided by OPC-DA allows a fast development of supervising interfaces (SCADA). Although, it needs some improvement as a better access through firewalls (Classic OPC-DA uses specific communication ports which needs special firewall permissions), to be multi-platform (OPC-DA is only compatible with Microsoft Windows ™), to be oriented to services and in security. It follows that a new protocol, called OPC UA, was created to solve these issues. But due to a large legacy and the already available converters OPC-DA/OPC-UA, OPC-DA, it is still the most important for industrial shop floor.

For a wide number of sensors and actuators connected in the network, as Internet of Things term is being applied, MQTT protocol is being largely used. The protocol was created to connect some sensors to a server using a satellite link what makes it designed to be simple and lightweight, so it does not need high processing capacity or either large bandwidth.

In this machine-to-machine protocol the main actors of the communication are the producers and consumers of data. But it is also common to have a distributor, the message broker (Figure 6), responsible for storing and forwarding messages published by producers to subscribed consumers in the related topics.

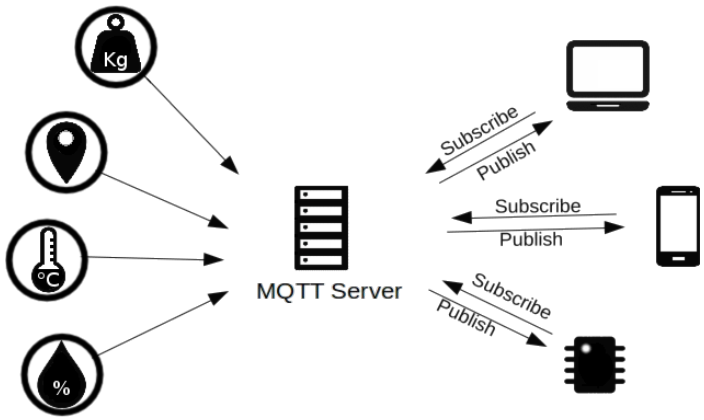

Figure 6. MQTT Broker

The communication with robots is not standardized as with IoT and PLCs devices, since there are a large variety of models and applications. There are simple ones that only do few tasks like push objects and complex ones like anthropomorphic arms with computational vision for welding, precision cuts and painting. So, the integration capability should be designed specifically to the target robot model.

4. A PROPOSAL OF TECHNOLOGICAL PLATFORM FOR THE FACTORY OF FUTURE

Various Artificial Intelligence techniques are being adopted to solve several problems, the use of agents is one those. To Cranefield and Ranathunga (2013), the agents can perform a valuable role to enhance many processes by its adaptability and bringing a behavior oriented to objectives.

Since the production lines have many different kind of devices in the same network, several specialist process and high requirements for flexibility and scalability, it seems to be an ideal application for a multi-agent system.

Multi-agent applications may decide in a decentralized way at same time that they are integrated. The agents play over an environment perceiving signals and acting, under a concept of an organization. The agent programming paradigm allows the design of systems with well defined commands hierarchy in a high level abstraction, an excellent platform to combine other artificial intelligence techniques (BOISSIER et al. 2011).

The agent is a reactive computational system with certain degree of autonomy to execute tasks that were delegated and may set the best way to perform them. The agents are flexible, with some pro-activeness, they react to the environment and cooperate among themselves to achieve goals they were designed to reach (BORDINI, 2007, p. 2).

Rollof (2014) asserts that the adoption of multi-agent systems in the industry was already experienced using the "agentification" approach. This method is more familiar to professionals since it is not far from object oriented paradigm. In this approach a machine, a belt, or a computational software, for example, may be an agent. Besides a complex analysis with a large number of agents, this way restricts the potential of agents technology since there is not a good understanding of roles, capabilities and orchestration. In this sense the risks of a bad distribution of responsibilities, overload of some actors and deadlocks are increased.

As an alternative, the paradigm VOWEL (Figure 7), as explained by Demazeau[1] cited by Roloff (2014:52), splits the MAS in four dimensions: Agents, Environment, Interactions and Organization. JaCaMo platform uses this concept. The name comes from the combination of its internal frameworks: Jason, CArtAgO and Moise+.

In this approach, the components of the system from simple automata to complex systems are modeled as agents. It uses Jason, a specific language based in the model BDI (Belief - Desire - Intention), which is inspired in human behavior. The beliefs are information of the world which the agents have. The desires are the states the agents want to achieve, representing influences in their actions. Intentions are the paths that the agent decided to go through, normally related to the goals that they are committed to (BORDINI, 2007:15-16).

The items featured as resources, tools and other non-automata objects, which include the body of the agent, are treated as artifacts. They are virtual representation of physical and computational non-automata entities. Artifacts live in workspaces and are part of the environment. In JaCaMo they are designed using CArtAgO framework.

Finally, specially in a social point of view, in the organizational dimension the hierarchy, groups and norms are defined. The social schema, mission and general goals are also set in this dimension. It is defined using Moise+ framework (BOISSIER et al., 2011, p. 751; HÜBNER and SICHMAN, 2003).

The communication in the very levels like speak, stimulus and perceptions are part of the interactions.

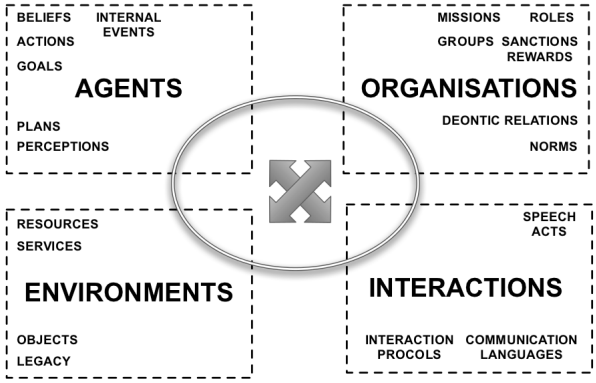

Figure 7. General view of JaCaMo's dimensions. Roloff (2014).

One of the great challenges of a MAS is how to communicate with a large range of technologies and software that can be find in a scenario (CRANEFIELD and RANATHUNGA, 2013). Considering the Industry 4.0 communication requirements, there is a big number of devices producing information all the time, and also critical needs to receive fast and accurate data. Because of this problem, it is proposed to integrate JaCaMo with the *framework* Apache Camel.

The framework Apache Camel is fundamentally a tool for routing, or rather, a communication routes constructor. In this sense, it is possible to define which sources of messages will be allowed, how the process to send the messages to

1 DEMAZEAU, Y. From interactions to collective behaviour in agent-based systems, in: Proceedings of the 1st. European Conference on Cognitive Science. Saint-Malo, pp. 117-132, 1995.

the recipients will work and which rules and message transformations will be applied in each route. Another important feature is that Camel is compatible with many applications in a wide variety of protocols.

Camel allows multiple Domain-Specific Languages (DSL) to define routes, is compatible with Java, Scala, Rest, Groovy and XML. This framework does not require a central data format leaving the endpoint of the route free to define it and allows multiple producers and consumers in the routes.

Camel is modular and compatible with many protocols, in its website there is a list with more than 200 internal components, some of the compatible technologies are: JMS, HTTP, MQTT, FTP, SIP, SQL, among others. Some components are more generic, like Netty4 that implements TCP protocol. There are also open source components that are not in this list, sometimes because of the use of some other licensing terms, or because they are not approved yet.

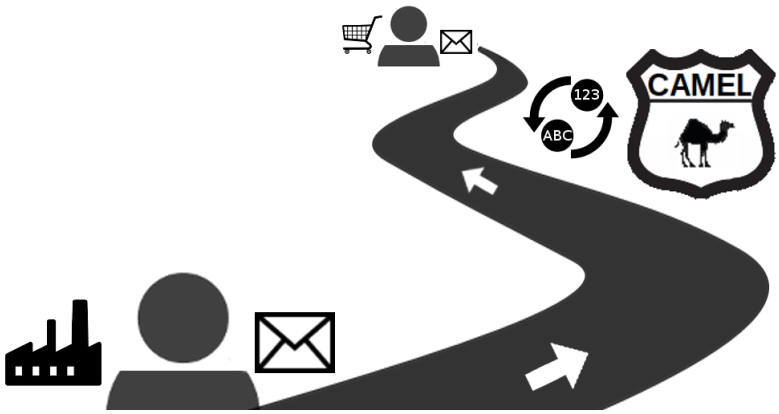

Figure 8. Illustration of a generic Camel route with consumer and producer sides

The route is defined by an entrance that consumes information, sent by a source and produces outputs to one or more recipients (Figure 8). In this process the route can transform the message according to compatibility needs. For industrial application, the devices, machines or softwares that supports the production lines currently, are artifacts for a MAS system. Which means that it is needed to virtualize these objects as artifacts, so in this JaCaMo application agents over an organisation can interact with these components of the factory.

5. THE INTEGRATION LAYER

One side of the route should be a CArtAgO artifact and the other side a device using MQTT protocol, TCP/IP or OPC-DA. There is no CArtAgO compliance component available, so it must be developed. Anyhow, there are many communication protocols already attended by Camel internal components. These projects are maintained by Camel community and the components are seen as native by Camel framework.

For MQTT, the connection among a message broker, producers and consumers is made by “camel-mqtt” component. When camel-mqtt is in “from” URI the route is going to subscribe in a MQTT topic to listen to it. In the other hand, if it is in “to” URI the route is publishing in a MQTT topic. The URI scheme supported by camel-mqtt component is: “mqtt://name[?options]”. Where “name” is any given name to this instance. The directive “options” must follow the set of options supported by this component. For instance, the options “*host*”, “*publishTopicName*” and “*subscribeTopicName*” are mainly used, they regard the address of a MQTT host and a name of a topic to publish or listen.

The integration with TCP/IP devices is also supported by Camel community. This communication uses a basic set of protocols, it is common to have other application protocols over it, usually proprietary protocols like the ones of industrial robots. To build routes with this feature, “camel-netty4” component is the main solution. The URI format of this component is: “netty4:tcp://localhost:port[?options]”. The directive “tcp” sets the used protocol as TCP/IP (it could also be “udp”). Following is an address of a host and a TCP port. Finally, it supports some options as, for instance, “textline” and “sync”, which refer to a plain text content or binary, and if the communication is one-way or bidirectional.

For OPC-DA communication there is no internal component, but there is an open source implementation made by an independent developer available over Apache license version 2.0. This component is called camel-opc. It is a complete implementation, with consumer and producer sides. The component was used in just a few cases, it is not very well tested. The URI format is: “opcda2://opcserver[?options]”, when opcserver is the name of the target instance running in a host. In “options” the “domain”, “host”, “user” and “password” can be defined. It also allows to set “clsid”, which is an application identification used in Microsoft Windows™ systems as well as “delay” and “diffonly”, these last two define polling intervals in milliseconds. They also define if the component will put in the route all OPC items data or only the modified ones. The items are stored as a map, which is a list of data. Each item has a key and a value. Since the component uses as value another map, it allows the store of a list of data related to an item.

Finally, to integrate MQTT devices, a robot using a proprietary protocol over TCP/IP and OPC-DA to a CArtAgO, it needs an artifact component.

5.1. The Artifact Component

Camel was developed to make the creation of new component easy; it is well documented and gives many resources to help this task, as the use of archetype, a template used for creating a new Camel Component, which makes easier the replication of a structure. For the artifact, the modeling choice was to give to it the capacity to create Camel routes.

In order to do that, *CamelArtifact* class was created, which extends artifact, the original CArtAgO class. The common application should be defining routes where the artifact is the sender or the recipient of messages, what uses the new endpoint. The implementation of Camel endpoint is done by *ArtifactComponent*, the factory of producers and consumers for CArtAgO artifact communication.

The class *CamelArtifact* and *ArtifactComponent* share two message lists, one for messages produced by the artifacts that must be send through the route and another for the incoming messages from the route that should be consumed by recipient artifacts. Both lists use *OpRequest* objects to store data. These objects contains an operation name and related parameters

The lists are of *ConcurrentLinkedQueue* model, which is a *first-in-first-out* and "wait-free" implementation. The class *CamelArtifact* has three main methods: *setListenCamelRoute*, *receiveMsg* and *sendMsg*. The first one should be invoked by an agent to start or stop the communication process set in a *CamelArtifact*. The second is making use of *IBlockingCmd*, a CArtAgO resource for blocking functions, in this case used to listen to the route by polling *incomingOpQueue*. Finally, *sendMsg* adds an *OpRequest* to be send through the route.

When a message arrives, by *receiveMsg*, the OpRequest is read by the artifact recipient, which gets the operation name and parameters. If the message is addressed to the *CamelArtifact* itself, a CArtAgo internal function is called (*@INTERNAL_OPERATION*). On the other hand, if it is addressed to another artifact, which may be a CArtAgO primitive one, a CArtAgO linked operation is called (by *execLinkedOp* invoking an external *@LINK* method). This implementation allows messages to be addressed to any artifact visible to *CamelArtifact* instance, which means it can also be used as a router.

6. MAS APPLICATION AND ROUTE TESTS

To test the new component and the integration with a MQTT server, with a robot and with an OPC server, a JaCaMo project called camelJaCaMoRobot was created (Figure 9). It is based in a simple MAS project where there are two agents called "owner" and "robot". The agent owner commands the robot to bring him some resources if available.

The main goal of this application is to implement simple routes to test communication among artifacts and different devices in both directions. Two artifacts are necessary to test internal operations as well as linked operations (router feature). The environment is also composed by the bodies of the agents.

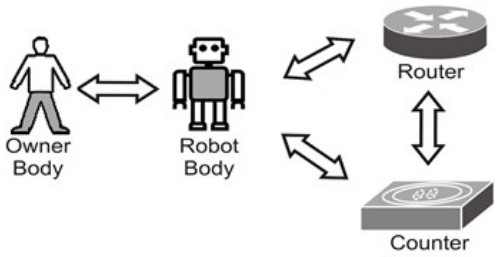

Figure 9. CamelJaCaMoRobot workspace representation

The bodies are the ones automatically created by JaCaMo. Counter is a primitive CArtAgO artifact. It is used in this application as an inventory counter. Router is a CamelArtifact where all routes are set. The Owner agent can interact only with Robot agent, which is the creator of Counter and Router artifacts. The agent Robot runs also the command to set the link between Router and Counter, a configuration that allows them to interact through linked operations (by execLinkedOp method, introduced before).

Counter is an ordinary artifact. The special change is in some methods like "inc2()" which has "@LINK" directive to allow it to be invoked by a linked artifact, what is used by Router when it needs to forward a message.

Router is more elaborate, besides it extends CamelArtifact, it imports camel libraries and other components which allows it to build the routes using internal components, "camel-opcda" and "camel-artifact". It also set an "@OUTPORT" called "out-1" what is used by Robot to link the artifacts Counter and Router.

All artifacts' events and even their existence depend on agents. An agent is necessary to "watch" an artifact, so that the events of the system can be perceived and trigger next actions. The agent Robot is the one to "touch" these artifacts

and who starts Camel mediation process by setting *setListenCamelRoute*, a method of router artifact. From this time on, all messages that arrive to artifacts will be stored in the incoming queue and the artifacts are allowed to send messages through the routes.

The organization of this MAS application is very simple. The chart shows it below. There are two roles for these actors. The *Owner* plays the role "Master" which has authority over the *Robot*, which plays the role "*robot*". The Robot has its cardinality as 10 and there is only one *Owner*.

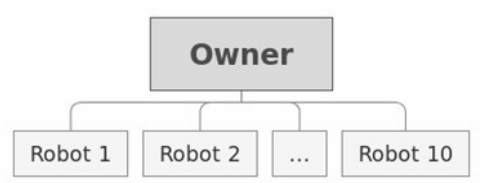

Figure 10. CamelJaCaMoRobot organisation chart

This JaCaMo application was deployed in an Ubuntu 14.04 Server, as a virtual machine using OpenStack infrastructure. A similar virtual server was used to deploy MosQuiTTo service, an MQTT implementation. Another server deploys a firmware of a Robot. In this case it was used a project called Asimov, what will be detailed later. Finally another server, in this case, a Microsoft Windows Server 2012, was used to deploy a MatrikonOPC™ Server for Simulation. These machines were connected in a vlan (Virtual Local Area Network).

For each protocol two routes were created, to test both consumer and producer sides. The tests were played separately by each protocol, this approach was used to simplify the tests since multiple consumers would be treated to avoid sending wrong content to a server. The MAS application ran directly by Eclipse IDE.

The first tests were using MQTT protocol. A route was set using a Camel timer as consumer triggering the route by 2 seconds, sending to the artifact a message addressed to "router" with the operation "inc" invoking *sendMsg*. This method adds a message in the outgoing queue, which will be catch by the consumer side and send through the route. The outcome is a simple message received by MosQuiTTo server, published in the topic "test.mqtt.topic". Another route was set to subscribe to this same topic allowing to test the opposite way, just invoking an artifact function to write in a log.

To test a route between OPC-DA and an artifact, an OPC server was set using a Matrikon™ OPC simulation software. In this test, an integer value was read in the OPC server, then delivered to the MAS application that increments and send it back to the OPC server, updating the item. To start a transaction the consumer side of a route was set with "opc-da" component. This component works by polling, the test was set for 2 seconds, which means that camel asks to OPC server for all items in a group in this interval. The integer value read was sent to the artifact "router". In the other side of the route (producer side) the artifact component will add this message in the *incomingQueue*. The next step should be the router, which is the one that sets the routes and has the queues, will check the incoming queue and invoke the referred method (by the named operation). In the return way, a route with opc-da component in the producer side was set. The trigger to this route is a method *sendMsg* called after the increment.

Finally, for a robot integration, a server ran "Asimov" firmware, which is an emulation of a robot. To start the communication, similarly as MQTT routes, there was the creation of a route triggered by a camel timer. Every 2 seconds a message addressed to the artifact "router" invokes the method *goRobot*. This method adds a message to be sent to Asimov. The commands "Go Right", "Go Left" and "Stop" were tested, what virtually makes the robot to do these movements. The robot sends back the same message its received as an acknowledgement what was just put in a log. By doing all these tests, both sides, consumer and producer, were tested as well as messages with and without arguments.

7. CONCLUSIONS

The changes of this forth revolution are so comprehensive and are coming so fast that cannot be ignored or barred. Factories that work within the current model may not survive in a near future. The relations among labor, commerce and environment will also suffer dramatic moves. Studies and movements to adapt to this trend are critical for any company, industrial organization and government.

The study here presented is proposing a platform and it discussed an application in this context. This solution can be used for multiple applications, by the use of Apache Camel, the integration can easily go further than OPC-DA, MQTT and Robots. The communication with OPC-DA devices is especially interesting, taking account all the legacy that are enough modern and robust. It is considered that the connectivity of a MAS is a key factor for its evolution and wide application of this technology.

Comparing with CPS in 5 levels proposed by Lee et al. (2015), this approach attends requirements of connectivity, virtualization and cognitive capacity. But, improvements to make it plug and play, to have performance and degradation supervising and self configuration may need more studies.

In face of the six fundamentals of Industry 4.0, it becomes clear that this approach satisfactorily covers at least four requirements: interoperability, once JaCaMo framework solves well communication issues among agents and, by Camel, among external systems; Virtualization, since the framework CArtAgO is developed to solve this topic creating representation of resources, tools and even the agents, allowing perceptions and actuation over the virtual environment; Decentralization, specially by the agents property acting in an autonomous and orchestrated way; and finally, Modularization, by the adaptive capacity of a Multi-Agent System and Camel, for new routes creation.

However, two items are not well covered: Real Time Response, once JaCaMo was not developed with this feature and some preliminary studies showed that synchronous process may be a great challenge for this platform since the platform is relatively heavy; and, Service Orientation, although Camel can easily integrate with SOA systems, JaCaMo does not fit this model.

About the application and tests, uses of both sides of integrations were presented. Future expansions, more complex applications and stressing use seems to work well. The use of multiple Artifact consumers may bring extra challenges since a way to filter recipients was not created. The use of some strategy for different Camel context may be needed, or alternatively, the use of multiple instances of *CamelArtifact*.

Another limitation of these tests is related to other concepts of CArtAgO artifacts such as multiple workspaces, manual, and observable properties and events. The developed component focused in small applications using only actuation commands.

In this sense, taking into account how wide the subject is, it is stated that this simple arrangement of technologies has great potential to be a well-succeeded combination in the context of Industry 4.0.

8. ACKNOWLEDGEMENTS

For their support and attention in this research, the authors would like to thank to Instituto Federal de Santa Catarina (IFSC) and University of Otago and the researches Jomi Fred Hübner, Justin Smith and Maria Teresa Collares.

9. REFERENCES

BOISSIER, Olivier; BORDINI, Rafael H.; HÜBNER, Jomi F.; RICCI, Alessandro; SANTI, Andrea. Multi-agent oriented programming with JaCaMo. Science of Computer Programming (2011), doi:10.1016/j.scico.2011.10.004, accepted 6 October 2011.

BORDINI, R. H.; HÜBNER, J. F.; WOOLDRIDGE, M. Programming Multi-Agent Systems in AgentSpeak Using Jason. Wiley Series in Agent Technology. John Wiley & Sons, 2007.

CRANEFIELD, S., & RANATHUNGA, S. Embedding Agents in Business Processes Using Enterprise Integration Patterns, 97–116, 2013.

HERMANN, M.; PENTEK, T.; OTTO, B. Design Principles for Industrie 4.0 Scenarios: A Literature Review. Working Paper No. 01 / 2015.

HÜBNER, J., SICHMAN, J. Organização de Sistemas Multiagentes. Campinas, JAIA 2003.

LASI, Heiner, KEMPER. Hans-Georg. Industry 4.0. Business & Information Systems Engineering. 2014.

LEE, J.; BAGHERI, B.; KAO, H. A Cyber-Physical Systems architecture for Industry 4.0-based manufacturing systems. Manufacturing Letters, 2015.

MAHNKE, W., LEITNER, S., DAMM, M. OPC Unified Architecture. Springer. 2009.

ROLOFF, Mário Lucio. Uma nova Abordagem para a Implementação de um Sistema Multiagente para a Configuração e o Monitoramento da Produção de Pequenas Séries. 218 f. Tese (Doutorado) - Curso de Doutorado em Engenharia de Automação e Sistemas, Engenharia de Automação e Sistemas, UFSC, Florianópolis, 2014.

ROLOFF, Mário Lucio; AMARAL, Cleber Jorge, STIVANELLO, Maurício Edgar; STEMMER, Marcelo Ricardo. MAS4SSP: a Multi-Agent Reference Architecture for the Configuration and Monitoring of Small Series Production lines. 12Th IEEE/IAS International Conference on Industry Applications, 2016.

WANG S., WAN J., ZHANG, D., LI, D., ZHANG, C. Towards smart factory for industry 4.0: a self-organized multi-agent system with big data based feedback and coordination. ELSEVIER. Computer Networks 101 (2016) 158–168. 2016.

10. RESPONSIBILITY NOTICE

The authors are the only responsible for the printed material included in this paper.