A Metamodel of a Multi-paradigm Approach to Smart Cyber-Physical Systems Development

Massimo Cossentino, Salvatore Lopes, Giovanni Renda, Luca Sabatucci, Flavia Zaffora

National Research Council
ICAR Institute, Palermo, Italy
{name.surname}@icar.cnr.it

***Abstract*—This paper illustrates an approach to the development of a shipboard power system reconfiguration as a Smart cyber-physical system (CPS). It is developed by representing it through a meta-model, providing a multi-paradigm approach that exploits the best features of three available frameworks, Jade, Jason and Akka. The resulting developing framework allows the creation of a new composite entity (labelled H-Entity) that exploits the advantages of Jade, Jason and Akka.**

***Index Terms*—CPS, smart, meta-model, multi-paradigm**

I. Introduction

The CPS application fields are nowadays several in all the domains of human activities. Here the paradigm of CPS is presented with the added quality of smartness, trying to find a flexible solution integrating heterogeneous technologies to easily communicate, although formerly not thought to cooperate at all. The specific case study is the Shipboard Power System (SPS) reconfiguration, where the use of a smart CPS is particularly compelling. The proposed solution to the problem adopts a multi-paradigm approach, where the selected requirements of adaptivity, proactivity, reactivity and smartness led to the consideration of different paradigm frameworks, especially referring to Agents and Actors entities. Among the available frameworks, three of the most representative, Jason, Jade and Akka, have been examined. Thus, this approach allowed to use each paradigm's strength points for the most suitable task. The resulting developing framework is proposed in the form of a meta-model, representing the composition of the so-called H-Entity, the core element of the meta-model. It constitutes the central reasoning of the proposal, gathering in itself different entities finally able to exchange information. In the end, an appendix with the definition of all meta-model elements is provided.

II. Theoretical Background

Before entering the proposal of a smart CPS development, it is better to briefly introduce what a CPS is, which are the principal domain applications, and what kind of frameworks can be taken into account. Thus, a short presentation of CPS, Ptolemy, Jade, Jason and Akka is following.

A. Cyber-Physical System (CPS)

A Cyber-Physical System (CPS) is a new generation of systems integrating the physical processes with those of the digital systems [1], [2]. In CPS, components are usually distributed over a network. The term 'cyber-physical' was born around 2006 at the National Science Foundation (NSF) in the United States. This Foundation supports the collaboration of cyber and physical components with the objective to extend the capabilities of the physical layer. The join between the cyber and the physical will create new applications [3]–[6]: industry 4.0, biomedical/healthcare systems, next-gen transportation systems, smart grid and renewable energy.
Since their very start in early 2000, the research and applications on this field increasingly evolved influencing several human activities [1]. Whereas a CPS is by nature adaptive and predictive [7], adding the word "smart" means to define it with a specific quality of "intelligence" that the CPS does not necessarily require. Nevertheless, the smartness of a CPS lies in the property of the reasoning, and in the ability of communication and knowledge-sharing among dissimilar components to take run-time decisions.
Current approaches to the development of smart CPSs encompass several alternative frameworks. Some selected are illustrated in the following.

B. Ptolemy II

Ptolemy II was born around 1996, successor to *Ptolemy Classic*. It is a framework of Java-based components with a graphical user interface called Vergil. As described from the developer [8], "the *Ptolemy* project studies the modelling, simulation and design of simultaneous embedded systems in real time." Its central core consists in the possibility to assemble concurrent components, using well-defined computational models governing the components interactions. A further development of the issue is to mix different computational models for the purpose.

C. Jade

JADE (Java Agent DEvelopment framework), is an Agent-oriented middleware [9]. It is compliant to FIPA standards, implementing the Agent Management specification and the FIPA Agent Communication stack [10]. It allows to realize a distributed system, whose agents are autonomous and proactive [11]. It supports ontologies and content languages and it holds an interaction protocols library [10].

D. Jason

Jason is a framework implementing the AgentSpeak language [12]. It is agent-oriented, and it uses a Java-based platform. As coming from AgentSpeak, it is inspired by the philosophical belief-desire-intention (BDI) model, that gives to it a human-like "reasoning": thus, Agents acquire a kind of intentionality. They are autonomous, proactive, reactive and they show a social ability, so that they are able to coordinate and cooperate. In the framework JaCaMo, the organization of autonomous BDI agents, programmed in Jason, is defined by the Moise framework, and the Agents work in a shared artifact-based environment, programmed in Cartago [13].

E. Akka

Akka is a Java-based framework, containing libraries implemented by the Scala programming language [14], and whose main entity is an Actor. It presents a robust hierarchical structure, made by actors linked each other by a parent-sons strategy. Akka Actors interact by exchanging asynchronous messages.

III. The approach

Smart CPS may be applied in many different contexts: this section will discuss our specific domain of interest, the reconfiguration of shipboard power systems, and an analysis of the approach adopted for developing the proposed framework.

A. The context: smart CPS for Shipboard Power Systems (SPS)

The here-described application context is specifically referred to the use of a CPS for the reconfiguration of Shipboard Power Systems (SPS). An SPS is responsible for vessels power supply, and it governs all the aspects of the electrical system onboard. About the state of the art of the reconfiguration methods, a study has been led [15], and it generated a reflection about the possible use of a smart CPS encountering the requirements this problem would ask for. The case of SPS is particularly relevant because it is an enclosed environment that, in scale, could represent any other application field where the need to make different technologies cooperate is compelling and fruitful to improve the efficiency of the entire system itself. It allows the experimentation on the interaction between the cyber-physical system and human control, for its adaptation after one or several changes occurred. Notably, the implementation and improvement of reconfiguration procedures acting on both the physical (the electrical one) and the cybernetic system (the software side of the problem) imply the challenge to face the occurring faults with a real-time response. Here a general SPS reconfiguration scenario is depicted. The Captain sets the ship's Mission through an Interface, that communicates with the SPSManager and can lay out a change of the Mission. So the SPSManager sets the load priorities (distinguishing vital, semi-vital and non-vital loads in the specific case) and finds a reconfiguration plan responding to the electrical failure perceived and communicated by the sensors through the SensorActor. The solution found has to be validated by the SPSPlanValidator, and the resources must be negotiated with other requirements through the SPS Representative. Now the Captain can decide if approving or not the solution and, if yes, the SPSManager can apply it through the enactor, set by the crew and directly acting on the switches that physically turn off the selected circuits.
By showing this exemplified situation, the smartness is especially required for the central role that the Manager plays and for the need to set run-time plans after deciding the load priorities.

B. The requirements

The general requirements a CPS should fulfill lie in a wide domain, that in [2] is richly exposed. All of these are strictly interrelated each other and the underlying need is the security the system should assure in dealing with trustiness and reliability of its physical and digital components and with the run-time interaction with the real world. Looking closer to the present case, the CPS here developed for a SPS should: be *smart*, i.e. the system should show a reasoning aptitude (agent-oriented paradigm); be *adaptive*, that means the system can efficiently adapt to environmental changes; be *predictive*, i.e. it can foresee changes in its behavior for possible changes in the context; be *distributed*: the system is heterogeneous and its parts are located in different places so it is able to maintain the connection; face *realtime* event, so the system is able to promptly react to changes; provide *feedback loop*, thus it can react to unexpected changes by monitoring itself and adjusting its behavior.

C. The proposed approach: a multi-paradigm strategy

A cyber-physical system is heterogeneous by definition, even just for it couples digital elements with physical ones. The purpose here exposed is to develop a smart cyber-physical system respecting some requirements especially involving the information exchange among heterogeneous systems, an ability we can refer to as "interoperability" [2]. Therefore, interoperability could determine a sort of "translation" among elements, languages, frameworks originally independent, acting on the communication-side of a CPS. A "smart" CPS uses specific frameworks involving some reasoning, and the need for communication has to include a social aspect along with an aptitude to adaptation. These elements explicitly belong to agent-oriented languages, while the necessity of scaling and feedback loop requires to adopt an actor-based language. Thus, at first a selection and a comparison of three agent and actor-based frameworks was done, trying to understand which of them could be more suitable to the purpose. Dealing with the above-mentioned SPS scenario, with the constraint of using different available technologies (the CPS challenge), meant having a multi-paradigm approach.
The term multi-paradigm refers to a domain dealing with complex heterogeneity of models [16]. The Multi-Paradigm Modeling, MPM, comes to solve the contemporary use of different paradigms characterized for not sharing the same language. The more heterogeneous is the relationship the harder is the

issue to address. The problem here exposed couples different paradigms with different underlying languages. Jade, Jason and Akka have been taken into account [11], highlighting their common points and their peculiarities. Although they are all Java-based, their differences are several, and they descend from the concepts behind their main entity, the Agent for Jade and Jason, the Actor for Akka. Moreover, Jade's Agents respect FIPA standards, whereas Jason's are based on a BDI model. In the following lines, their distinguishing traits will be summarized.

- *Jason's reasoning*
  Basing on a BDI (belief-desire-intention) model, Jason's Agent shows a decision-making ability that can be called reasoning [12]. Human-like, it expresses by a sequence of tasks making a plan that is acted to pursue a goal, with the qualities of autonomy, reactivity, proactivity and social ability. With special regards to the last one, the communication among Agents is an actual action [12], that has a Propositional Content and a Performative (e. g. a request).
- *Jade's communication protocols*
  In Jade the interaction between distributed Agents is particularly stressed. As fully adhering to FIPA, the Agent is proactive, so it can react to a state change and decide to pursue a goal [11]. Developing with Jade means using the object-oriented paradigm with specific aptitude to finite state-machine and interoperability due to protocols-oriented communications, to ontologies and to full support for FIPA specifications.
- *Akka's hierarchy*
  The peculiar structure of Akka lies in the father-sons hierarchy of the entities. [14]. Developing a reactive system on Akka means reasoning in terms of parallel, asynchronously communicating processing (the actors), and enfatizing aspects such as efficiency, distribution, scalability and failures control. Unlike Jason and Jade, Akka features allow a safe approach for implementing highly scalable and performant functions; moreover Akka Stream provides a capability of internal coordination to implement pre-defined workflows.

For the above-mentioned aim, it resulted that none of them can be substituted without losing something, so instead of settle with just one, the new objective was to choose all of them, basically using each strength points, i.e.: Jason's reasoning, Akka's hierarchy and scalability, Jade's communication protocols.

IV. The proposed framework

Since it is a compact representation, the use of a meta-model is useful to have an overall view of all the components of the proposal: the more the parts are, the more useful the construction of the meta-model is. Building it up means to clear up the hierarchies and especially the relationships keeping otherwise separated and, most of all, to finally see possible links and how to develop them [17].
In the following section the H-Entity will be introduced. Right after we will have a look at the represented meta-model (Fig. 1). There, a top-down and a bottom-up perspective to deal with it are provided. This means that one could read the meta-model in terms of composition of elements (top-down) or by analyzing the single parts which it is made of (bottom-up).

A. The H-Entity

The so-defined H-Entity, where 'H' stands for *heterogeneous*, holds the entire philosophy here proposed. It can be considered as a polyhedral organism composed of frameworks that cooperate by exploiting their best qualities and winning skills. The H-Entity allows them to share their knowledge among each other if they use different languages, and between them and the external world, making into communication other possible H-Entities. It is conceived as a flexible and versatile structure, defined by Roles played by the most suitable specific Entity (composing the H-Entity).

B. Top-down meta-model perspective

Let's now observe the meta-model. The principle of all lies in the Cyber-Physical World, which is made up of a CPS Organization and of an Environment; they are the organizational side, corresponding to the Moise framework, and the environmental side, belonging to Cartago. The so-called H-Entities also belong to the CPS world: they play the CPS roles, which are Moise roles and make a CPS organization. They constitute the link among the left and right areas. On their turn, the H-Entities are entities (Agents and/or Actors) playing both H-Roles, defined as Manager, Worker, Diplomat and Interface, and Entity Roles, which are Moise roles. This is the Moise Organization, that allows the Entity to pursue a goal by playing a role that determines the behavior of a Group of Entities. The same kind of organization is adopted inside a single H-Entity among its own Entities constituting it, or among different H-Entities, that happen to interact in the Cyber-Physical World. Then, the H-Roles define the distribution of responsibilities in a single H-Entity; the CPS Roles describe the macro-functionalities in the whole external system of CPS, among different H-Entities. The Roles define the structural definition of the organization and allow the Entities to pursue a goal by following a behavioral constraint; the Goal is collective and to be achieved by accomplishing Missions, that are the link between roles and Social Scheme, i.e. between the structural and the functional part of the organization. On the left side, the CPS world has an Environment, both physical and cybernetic. Here the artifacts compose the workspace: the artifacts are made by a Manual, that holds the information about their state, and by some Operations. The latter ones update Observable Properties, that are the states of the artifacts, and generate Observable Events (e. g., in the lower meta-model level, the messages).

C. Bottom-up meta-model perspective

Let's put one aside the other the three frameworks we analyzed. Each of them has its own entity: Jade and Jason Agents and Akka Actor.

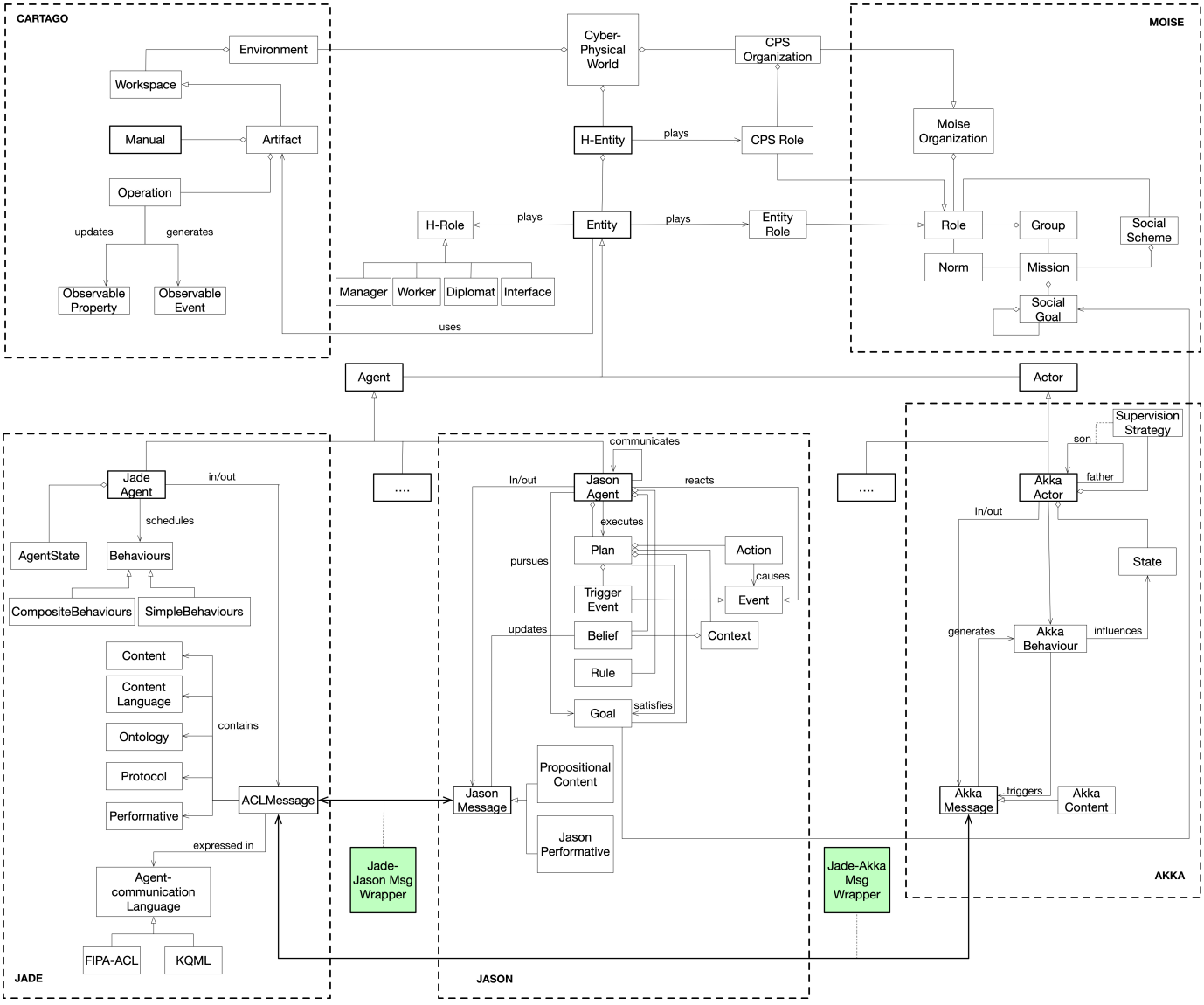

Fig. 1. The H-Entity Meta-Model

a) Jade side: Jade's Agent has an AgentState, performs a behavior through a Scheduler and it sends and receives an ACLMessage, which is FIPA compliant: thus, it contains a Content, a Content Language, an Ontology, a Protocol and a Performative, and it is expressed by an Agent-Communication Language (ACL).

b) Jason side: The Jason Agent "lives" following a BDI model. In pursuing some Goals, it reacts to the Events occurring in the context which trigger some Plans. The Plans have to satisfy these goals. In the meanwhile, Jason Agent communicates through messages, defined as Jason Messages, made by a content (Proposition) and a Performative.

c) Akka side: Finally, in Akka the actor shows a "parental" hierarchy [14] that enacts a Supervision Strategy. The actor performs a Behavior which influences a State. The Behaviour triggers the so-called Akka Message that in turns generates a behaviour.

D. The linking entities

Something is immediately evident: at the bottom of the figure, the message level shows a gradual implementation from the Actor to Jade Agent framework: the Message, containing just a Content in Akka, acquires a Performative in Jason, and then also a Protocol, a Language and an Ontology in Jade (that, it is known, has its strength point in the communication side). Here it comes the key of the interaction among the three frameworks, originally not conceived to communicate at all. The other elements placed outside the known meta-models are those related to the concepts of entities: the upper entities, called H-Entities, are all of those elements who can interact among each other in the CPS world through an upper Moise Organization, that replicates in the lower meta-model level among the Roles of a single entity. Thus, the

Goal can be a Social Goal, when pursued at the upper level, while at the bottom it is the single Goal pursued by Agents. There are still two elements to be analyzed, the so-defined Jade-Jason Message Wrapper and the Jade-Akka Message Wrapper, highlighted in green in the picture. These constitute the practical key for the realization of the inter-communication among the three frameworks. What they do is establishing a "translation" using the Jade Communication Protocol as a channel: whenever an agent or an actor sends a message, Jade intercepts it to re-send it to its recipient, using a wrapper with Jason, by sharing the content and the performative, and a wrapper with Akka, sharing only the content.

V. The solution

The meta-model allowed to understand how and at what level to realize the merging among all frameworks. The following subsections will provide details about the wrappers developed to allow their interactions and the solution to the scenario introduced before.

A. Examples of code: the Wrappers

As introduced in the previous section, the entities are linked together at the message level. This layer is implemented with the *wrappers*: in source code 1 and source code 2, the *wrappers* connect the Jason agent (Manager), with the Akka Actor (root). The communications from the Jason Agent to the Akka Actors are managed through a bridge. This is implemented with a Jade agent named *JadeAkkaActorSystem* (source code 1).

```
1  public class JadeAkkaActorSystem extends Agent {
2     ...
3     protected void setup() {
4        Akka2Jade bridge =new Akka2Jade(this);
5        root = system.actorOf(Root.props(bridge), "root-
             worker");
6        addBehaviour(new ForwardIncomingMessages());
7     }
8
9     private class ForwardIncomingMessages extends
          CyclicBehaviour {
10       @Override
11       public void action() {
12          ACLMessage msg = myAgent.receive();
13          if (msg != null) {
14             String content = msg.getContent();
15             root.tell(content, null);
16          }
17       }
18    }
19 }
```

Source code 1. Jason to Akka *wrapper def*

```
1  public class Akka2Jade {
2     ...
3     public void sendJason(String msg) {
4        if (myAgent != null)
5           myAgent.addBehaviour( new SendToJason(msg));
6     }
7
8     private class SendToJason extends OneShotBehaviour {
9        ...
10       public SendToJason (String content) {
11          this.content = content;
12       }
13       @Override
14       public void action() {
15          ACLMessage msg = new ACLMessage(ACLMessage.INFORM);
16          msg.addReceiver(new AID("manager",AID.ISLOCALNAME))
                ;
17          msg.setLanguage("h-language");
18          msg.setContent(content);
19          myAgent.send(msg);
20       }
21    }
22 }
```

Source code 2. Akka to Jason *wrapper def*

The use of a Jade Agent as a bridge is very useful because, through the extension of the *CyclicBehaviour* class, a type of Behaviour that execute the *action()* method cyclically and never ends, the agent waits for messages from Jason agent, then the Jade Agent sends them to the root actor. With the source code 1, in the *setup()* method (line 5), a *root* actor is instantiated: this actor has the task of initializing all the possible sub-actors useful to the system, and it allows the sub-actors to connect with the Jason Agent. When the Manager (Jason Agent) would like to send a message to an actor, the *JadeAkkaActorSystem* intercepts the message (line 12) and sends it to the *root* actor (line 15), which has the task to send the message to the specified sub-actor. When an actor would send a message to a Jason Agent, uses the class *Akka2Jade* (source code 2). These bridges also exploit a native class named *JadeAgArch* that allows instantiating a Jason Agent into a Jade Agent, sending messages from Jason to Akka Actor through Jade Agent and vice versa. With the source code 2, the actor uses the *sendJason(msg)* method (line 3-21) to send a message to the Manager (Jason Agent).

B. The SPS scenario

In Fig. 2 it is explained the CPS solution for the SPS scenario, with the adoption of the proposed meta-model.
The Captain sets the Mission by using the Interface which is made by a Cartago Artifact; so the Mission is changed and this is communicated to the SPSManager, whose role is played by a Jason Agent. It sets the load priorities with an exchange with the MissionManager, which after detecting the electric failure (by sensors communicating with Akka Actors, through the CircuitMonitor, finds a possible Reconfiguration with the SPS Plan Generator, which is an Actor. This is sent to the Manager which needs to validate the Solution by sending it to a Validator, an Akka Actor, exchanging information with a MatLab platform. So the available resources are negotiated with a "talk" between the Jason Agent Manager and the SPSRepresentative which is a Jade Agent. The solution is finally approved by the Captain who allows the Manager to apply it by enabling the Crew to interact with the Enactor, a Cartago Artifact, passing through a Reconfigurator Enactor, an Akka Actor. It physically switches off the non-vital loads to efficiently deal with the fault. Thanks to the proposed approach, the solution takes advantage of the most relevant features of the three composing frameworks: it shows the capability to adhere to standards, from Jade; it implements a strategy in high levels rules, through Jason, and it is very efficient, due to the Akka contribution.

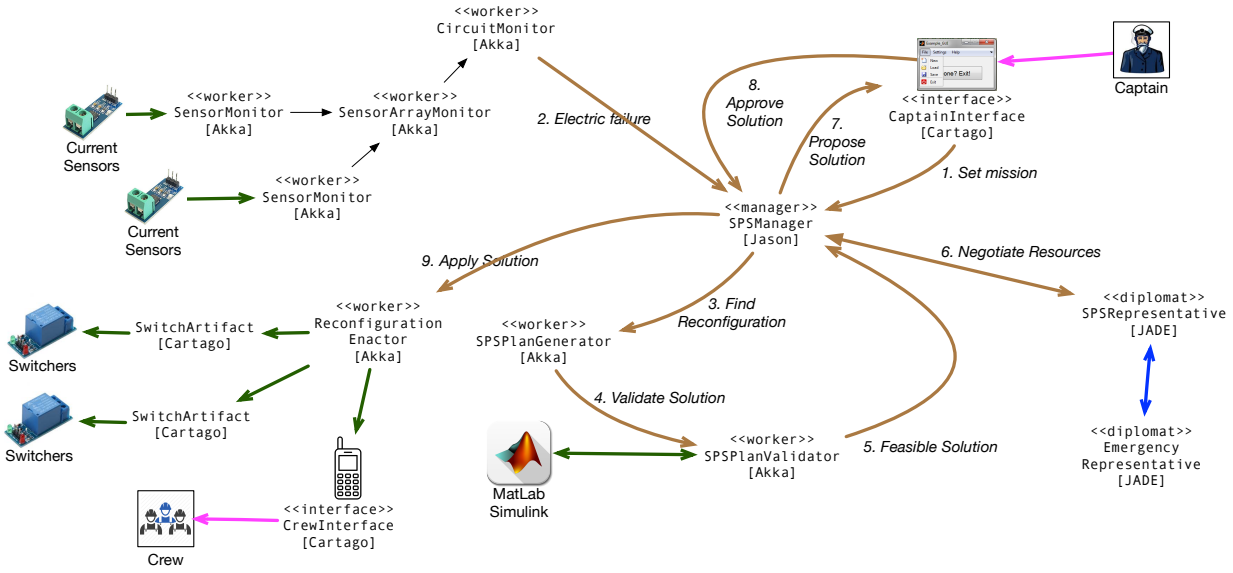

Fig. 2. The solution for a SPS reconfiguration

VI. Conclusions and Future Works

The need to work with smart CPS implies the necessity to make different frameworks and paradigms interact with each other. This issue by nature leads to a multi-paradigm approach, with the theme of communication among not-overlapping languages. The solution we developed is here presented by means of a meta-model including the considered agent and actor-based frameworks (here Jade, Jason and Akka), linked by the so-called H-Entity, a polyhedral structure that allows taking advantage of each framework winner skills to be exploited at best according to the needed roles to be played. The solution adopted for an SPS reconfiguration revealed a possible way to make the different entities interact at the message level, by the use of wrappers.

The SPS reconfiguration system prototype is now under test. The evaluation of the system will be performed by measuring the response time and the interactions among the different parts of the platform. The future advancement will be to extend the use of CPS to all the other ship's facilities: the final purpose is to give form to a Smart Ship solution, as already provided by vendors such as Hyundai [1].

References

[1] J. Shi, J. Wan, H. Yan, and H. Suo, "A survey of cyber-physical systems," in *Wireless Communications and Signal Processing (WCSP), 2011 International Conference on*. IEEE, 2011, pp. 1–6.

[2] V. Gunes, S. Peter, T. Givargis, and F. Vahid, "A survey on concepts, applications, and challenges in cyber-physical systems." *KSII Transactions on Internet & Information Systems*, vol. 8, no. 12, 2014.

[3] H. Li, L. Lai, and H. V. Poor, "Multicast routing for decentralized control of cyber physical systems with an application in smart grid," *IEEE Journal on Selected Areas in Communications*, vol. 30, no. 6, pp. 1097–1107, 2012.

[4] P. Leitão, 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.

[1] IntegrICT, http://www.hyundai-electric.com/elec/en/integrict/integrict1.jsp

[5] J. Lee, B. Bagheri, and H.-A. Kao, "A cyber-physical systems architecture for industry 4.0-based manufacturing systems," *Manufacturing Letters*, vol. 3, pp. 18–23, 2015.

[6] S. Sridhar, A. Hahn, and M. Govindarasu, "Cyber–physical system security for the electric power grid," *Proceedings of the IEEE*, vol. 100, no. 1, pp. 210–224, 2012.

[7] P. J. Mosterman and J. Zander, "Cyber-physical systems challenges: a needs analysis for collaborating embedded software systems," *Software & Systems Modeling*, vol. 15, no. 1, pp. 5–16, 2016.

[8] [Online]. Available: https://ptolemy.berkeley.edu

[9] F. Bellifemine, A. Poggi, and G. Rimassa, "Jade: a fipa2000 compliant agent development environment," in *Proceedings of the fifth international conference on Autonomous agents*. ACM, 2001, pp. 216–217.

[10] F. L. Bellifemine, G. Caire, and D. Greenwood, *Developing multi-agent systems with JADE*. John Wiley & Sons, 2007, vol. 7.

[11] M. Cossentino, S. Lopes, A. Nuzzo, G. Renda, and L. Sabatucci, "A comparison of the basic principles and behavioural aspects of Akka, JaCaMo and Jade development frameworks." 19th Workshop From Objects to Agents (WOA 2018), Palermo, 28-29 June 2018.

[12] R. H. Bordini, J. F. Hübner, and M. Wooldridge, *Programming multi-agent systems in AgentSpeak using Jason*. John Wiley & Sons, 2007, vol. 8.

[13] J. F. Hübner, O. Boissier, R. Kitio, and A. Ricci, "Instrumenting multi-agent organisations with organisational artifacts and agents," *Autonomous agents and multi-agent systems*, vol. 20, no. 3, pp. 369–400, 2010.

[14] M. Gupta, *Akka Essentials*. Packt Publishing, Birmingham, 2012.

[15] L. Agnello, M. Cossentino, G. De Simone, and L. Sabatucci, "Shipboard power systems reconfiguration: a compared analysis of state-of-the-art approaches," *Smart Ships Technology*, pp. 1–9, 2017.

[16] C. Hardebolle and F. Boulanger, "Exploring multi-paradigm modeling techniques," *Simulation*, vol. 85, no. 11-12, pp. 688–708, 2009.

[17] T. Kühne, "Matters of (meta-) modeling," *Software & Systems Modeling*, vol. 5, no. 4, pp. 369–385, 2006.

[18] A. Ricci, M. Viroli, and A. Omicini, "Cartago: A framework for prototyping artifact-based environments in MAS," in *International Workshop on Environments for Multi-Agent Systems*. Springer, 2006, pp. 67–86.

[19] O. Boissier, R. H. Bordini, J. F. Hübner, A. Ricci, and A. Santi, "Multi-agent oriented programming with JaCaMo," *Science of Computer Programming*, vol. 78, no. 6, pp. 747–761, 2013.

[20] C. Hahn, C. Madrigal-Mora, and K. Fischer, "Interoperability through a platform-independent model for agents," in *Enterprise Interoperability II*. Springer, 2007, pp. 195–206.

[21] M. Cossentino, A. Chella, C. Lodato, S. Lopes, P. Ribino, and V. Seidita, "A notation for modeling Jason-like BDI agents," in *2012 Sixth In-*

ternational Conference on Complex, Intelligent, and Software Intensive Systems. IEEE, 2012, pp. 12–19.

VII. Appendix: definition of meta-model elements

CARTAGO [18] [19]

Artifact: it is the basic element of the environment and it provides organizational operations as they are used by Agents.
Environment: it consists in the field of work for the agents' activities.
Manual: it contains the information about the functional description and the observable states of the artifact.
Workspace: it is defined as an open set of artifacts and agents. The agent must exist in a workspace.
Operation: It determines the update of the artifact state or it generates observable events.
Observable Property: it determines the functional behavior of the artifact.
Observable Event: it is an event generated by an operation and sensed and registered by the agent sensors.

MOISE [13] [19]

Group: responsible for one or more Social Schemes that the Agents adhere to.
Social Scheme: it contains several missions.
Mission: contained in the Schemes, it is defined by a set of goals to achieve.
Role: it is a behavioral constraint for an Agent to play in a Group.
Norm: the collecting element between the Role and the Mission so the Group to Role.
Social Goal: the Goal the organization has to achieve.

H-ENTITY

H-Entity: it is the core element of the meta-model. It gathers all the possible heterogeneous entities.
Entity: one of the infinite possible members of the H-Entities. It can be an Agent or an Actor.
H-Role: H-Roles are internal roles played by the Entities.
CPS Role: CPS Roles are played among different H-Entities interacting in the CPS World.
Entity Role: the external roles played by Entities following the Moise Organization.

JADE [10] [20]

Jade Agent: it shows quality of autonomy and proactivity.
Behaviour: It implements the tasks that the Agents can pursue.
ACLMessage: it is based on an asynchronous communication paradigm. It makes the interaction among Agents.
Protocol: as part of the structure of the ACLMessage, it is defined to the purpose of interaction among messages by the specification of sequences.
Content: it defines the information carried by a message. It follows FIPA indications about its expression generally using SL (semantic language).
Content language: it is expressed in SL and it is made up of expressions or propositions.
Ontology: it is a knowledge base the messages refer to.
Performative: it is a type of communication expressed by a message, it is the communicative act.

JASON [12] [21]

Jason Agent: autonomous, reactive, proactive and social entity based on a BDI (belief, desire, intention) model.
Plan: a set of actions to be performed to pursue and satisfy a goal.
Trigger Event: it is the circumstance that activates a plan rather than another.
Action: each task which a plan is made of. It is performed by the agent and can change the environment so becoming an Event.
Event: a change in agent's initial knowledge or goal.
Context: it is a set of conditions determining the execution of a Plan.
Belief: a predicate representing the starting knowledge of the environment the agent has.
Rule: a logic expression made by predicates.
Goal: the final state that the agent wants to achieve by performing a plan.
Jason Message: it has a performative and a content. It allows the Agents to communicate each other and it updates the believes.
Jason Performative: it is of the kind of request, information, etc...
Propositional Content: it is expressed by arguments and it could be true or false.

AKKA [14] [11]

Akka Actor: a reactive entity encapsulating a state and a behavior.
State: it is encapsulated by the Actor.
Behaviour: an encoding of reactions.
Supervision Strategy: the feature based on the hierarchical structure of Akka made of fathers and son. Each father is responsible for its sons.
Akka Message: the interaction key among Actors. It stimulates a reactive action of the Actor. Received messages are stored in a mailbox. It has just a content.