Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 56 (2015) 520 – 525

Procedia Computer Science

International Workshop on Communication for Humans, Agents, Robots, Machines and Sensors (HARMS 2015)

An Analysis and prototyping approach for Cyber-Physical Systems

Samuel Deniaud[a], Philippe Descamps[b], Vincent Hilaire[b,*], Olivier Lamotte[b], Sebastian Rodriguez[c]

[a]IRTES-M3M, UTBM Belfort 90010, FRANCE
[b]IRTES-SET, UTBM Belfort 90010, FRANCE
[c]GITIA, UTN-FRT, Rivadavia 1050, San Miguel de Tucuman, ARGENTINA

Abstract

Sensors and effectors of all sorts are becoming more and more integrated in human everyday lives. Systems built on top of these sensors/effectors adding a cybernetic component in order to assist humans in their everyday life are called Cyber-Physical Systems. Such systems may be difficult to analyze and design, as they are composed of numerous interacting entities immerged in a dynamic and partially predictable environment. The goal of this paper is to propose a methodological approach based upon Multi-Agent Systems that has two main advantages. First it allows the decomposition of CPS structural and behavioral complexity and, second, it enables a prototyping by simulation approach that eases system validation.

Peer-review under responsibility of the Conference Program Chairs

Keywords:

1. Introduction

Sensors and effectors of all sorts are becoming more and more integrated in human everyday lives. One of the goals of this evolution is to build a category of systems, named Cyber-Physical Systems[1,2] (CPS), that aims to help or assist human tasks. CPS cover a wide range of systems which common points are, first, to be a part of human physical existence through sensors/effectors and, second, to take autonomous decisions through a cybernetic part in order to fulfill some goals.

* Corresponding author. Tel.:+33 384 583 009; fax: +33 384 583 342.
E-mail address: vincent.hilaire@utbm.fr

Peer-review under responsibility of the Conference Program Chairs
doi:10.1016/j.procs.2015.07.245

The analysis and design of CPS maybe a difficult task as it frequently requires taking into account many, possibly heterogeneous, components evolving in a dynamic and partially predictable environment. Indeed, human interactions with CPS are by essence non totally predictable. The goal of this paper is to propose a methodological approach based upon agents for the analysis and prototyping of CPS. This approach extends one existing MAS methodology, namely ASPECS[3,4], and integrates within ASPECS specific concepts and activities to deal with the specific challenges of CPS.

The underlying idea consists in defining model elements and activities within the methodology in order to analyze and prototype CPS. The first focus is on the analysis of human-CPS interactions that should be integrated in early analysis activities. Second, the behavior of CPS should be studied by applying a separation of concerns approach in order to define modular, reliable and easy to study systems. Third, as CPS can be categorized as belonging to the complex systems sort due to its inherent features (numerous heterogeneous components interacting in a dynamic environment) a prototyping approach that allows the validation of some requirements can help CPS analysis and design.

More specifically, the contribution proposed in this paper proposes to replace the requirements analysis approach proposed within ASPECS, which recommends use cases for this activity, by a requirement analysis based upon the SysML requirement profile[5] and a part of the Automation profile[6]. Following this requirement analysis activity the subsequent activities refine the requirements in terms of concepts within a problem ontology and organizational structures that, once fully defined, specify the behavior of all parts of CPS and allow their simulations with MAS. Indeed, each component of the system, even humans, can be simulated for system behavior validation. The ASPECS methodology is already based upon such an approach. The contribution resides in defining and using concepts that allow the identification and separation of each component in order to be easily integrated within a simulation.

This paper is structured as follows: section 2 presents some background concerning the ASPECS methodology. Section 3 details the contribution in terms of methodological aspects. Section 4 discuss some related works and section 5 concludes.

2. ASPECS Methodological approach

ASPECS is a step-by-step requirement to code software process for engineering Complex Systems using Multiagent Systems and Holonic Multiagent Systems. ASPECS relies on a set of organization-oriented abstractions that have been integrated into a complete methodological process. The target scope for the proposed approach can be found in complex systems that may, eventually, be described as a hierarchical system. The main vocation of ASPECS is towards the development of societies of holonic (as well as not-holonic) multiagent systems. The ASPECS life cycle consists of three phases. The *System Requirements* phase aims at identifying a hierarchy of organizations, whose global behavior may fulfill the system requirements under the chosen perspective. The second phase is the Agent *Society Design* phase that aims at designing a society of agents whose global behavior can provide an effective solution to the problem described in the previous phase and to satisfy associated requirements. The third and last phase, namely *Implementation and Deployment*, firstly, aims at implementing the agent-oriented solution designed in the previous phase by deploying it to the chosen implementation platform. In this paper we only deal with the first phase: *System Requirements*. The activities of all ASPECS phases are detailed in[3].

The Domain Requirements Description (DRD) activity aims at an initial requirements elicitation. The expected result is a description of the application behavior. Several different approaches can be used. For example, use cases diagrams and documented version to introduce user annotations can specify functional and non-functional requirements. The Problem Ontology Description (POD) activity defines a conceptual overview of the domain concerned. This ontology aims at the conceptualization of experts' knowledge that will provide the application context. Moreover, this ontology helps to understand the problem to be solved and allows requirements refinements. Among the subsequent refinements, the identification of organizations, roles and capacities is one of the most important. The Organization Identification (OI) activity goal consists in assigning to each requirement a behavior, which is not detailed at this level and which is represented by an organization. The meaning of this assignment is that a requirement should be satisfied by an organization. The behavior represented by the organization is the result of the interacting roles within a common context. The latter is conceptualized in the POD defined in the previous activity. This activity starts from the requirements defined in the DRD activity.

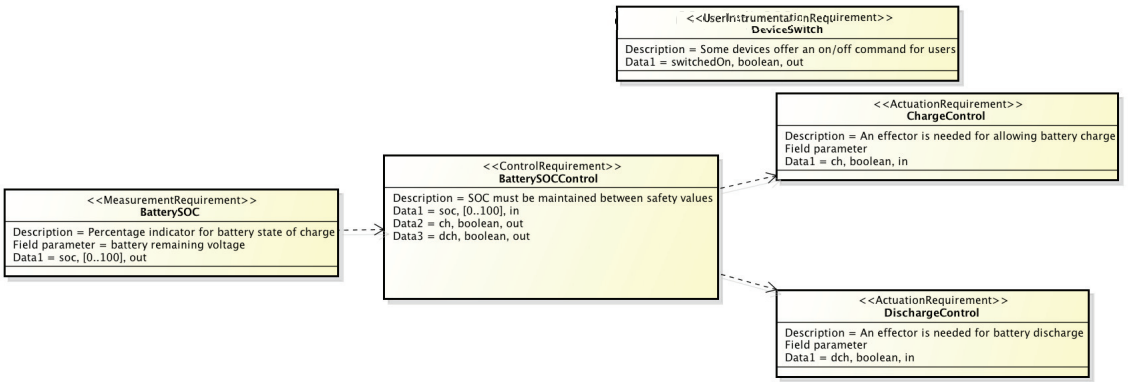

Figure 1 Requirements analysis example

3. CPS dedicated Approach

The following analysis is extracted from a specific MAS dedicated to power management[6]. This system can be qualified as a smartgrid as it autonomously handles power management on behalf human users using specific sensors and effectors. The presented diagrams are simplified from the analysis and must be read as illustration examples.

3.1. Domain requirements description

The requirements analysis of CPS is inspired by the SysML requirements profile and the automation profile (extended to meet our needs). The latter is a profile dedicated to automation systems. Among the stereotypes defined in this profile, three are presented as illustration in figure 1, the interested reader can find more in[6]. The "MeasurementRequirement" stereotype specifies a specific phenomenon to be monitored. This kind of requirement can be further refined by a specific sensor. The "ActuationRequirement" stereotype specifies a specific phenomenon that is influenced by the CPS. This requirement can be further refined by a specific actuator. Both "MeasurementRequirement" and "ActuationRequirement" are characterized by the following attributes: a textual description, a field parameter (what is measured or acted upon) and a set of data named, typed and directed (input or output). Eventually, the "ControlRequirement" stereotype in interaction with "MeasurementRequirement" and "ActuationRequirement" specifies an autonomous control mechanism based upon perception (resp. action) inputs (resp. outputs) information. Each "ControlRequirement" also contains a textual description and the data specifications coming from measurement or going to actuation. In the example provided by figure 1 the system autonomously controls battery state of charge (SOC) in order to both maintain SOC between security bounds and supply a microgrid. The system has thus to monitor battery SOC and can charge or discharge batteries according their SOC. The SOC is measured as a percentage, is taken as input by the control, which outputs two Booleans one for the *ChargeControl* and the other for the *DischargeControl*. The part of the early analysis resulting states a goal to be reached by the system-to-be. This goal, here "ControlRequirement", is specified by a small text and some semi-formal elements such as the required inputs data and outputs provided. The "UserInstrumentationRequirement" represents human users in the system and specifically in the example specifies the possible action (on/off) on microgrid devices.

3.2. Problem ontology description

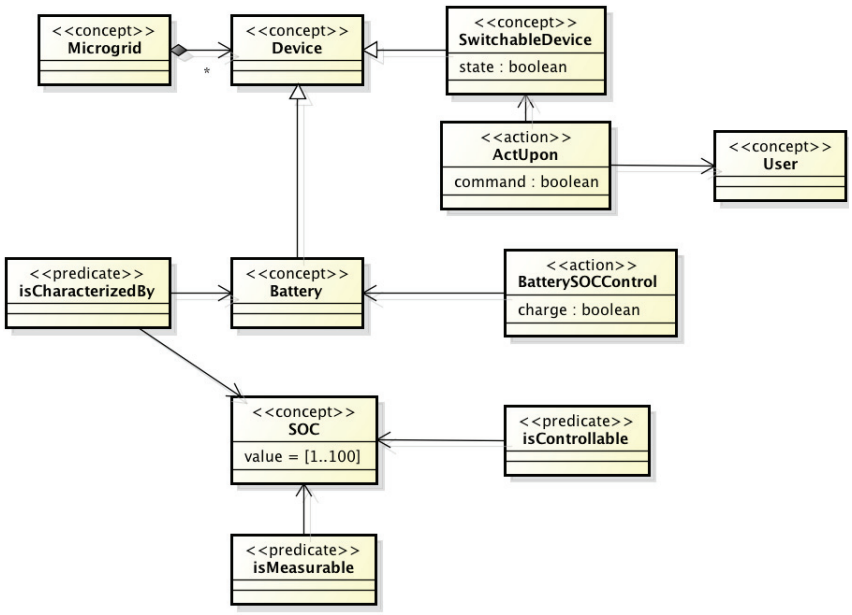

Figure 2 Problem ontology example

The problem ontology description is a conceptualization of the problem domain. This activity can take place before the Domain requirements description, or after, depending on the domain a priori knowledge. A class diagram using a profile defined in FIPA[8] describes the ontology. Each class stereotyped "concept" represents a concept of the domain. The figure 2 presents a simplified example of such ontology. *Microgrid* is a concept of our problem domain that is composed of several concepts named *Device*. The *Battery* and *SwitchableDevice* concepts specify specific kinds of *Device*. The two others stereotypes are: "predicate" which is a specification of a relationships and "action" which denotes an action on a single or several concepts. Starting from the Domain requirements description activity one can deduce some elements of the ontology. In the example of figure 2, the predicate *isMeasurable* can be deduced from the previous "MeasurementRequirement". The problem ontology may further specify the different measured concepts. These description elements can be used later in order to either specify a simulation model of sensors or defined roles and required capacities that will represent sensors in the MAS to be. The same ideas can be applied for "ActuatorRequirements". Each of these requirements defines an interface that may produce an effect on a concept. The action *ControlSOC* can be deduced from the "ControlRequirement". Indeed, a "ControlRequirement" should result in an "action" upon some entities of the domain. The *ActUpon* action and *User* are deduced from the "UserInstrumentationRequirement". Each "UserInstrumentationRequirement" implies an interaction with human users.

3.3. Organization identification

The organization identification activity consists in assigning a global behavior to each requirement. These global behaviors are embodied by an organization. Each requirement is then associated to a unique organization in charge of fulfilling it. An organization may fulfill several requirements. There are several possible guidelines for organization identification. A first possible technique is to use structural relationships and specifically aggregations. The aggregation from *Microgrid* to *Device* identified in the ontology (figure 2) can give birth to the *Microgrid* organization of figure 3. Another technique for organization identification relies on requirements grouping and more specifically "ControlRequirement" and/or "UserInstrumentationRequirement". For our smartgrid example, the *BatterySOCControl* and the *DeviceSwitch* are the only "ControlRequirement" and the behaviors that can be identified through "actions" can be considered at a first glance as belonging to a same organization since they can

not be decomposed without loss of coherence. The *Microgrid* organization is thus chosen and represented as a package surrounding *BatterySOCControl* and *DeviceSwitch.* This organization is thus in charge of these requirements fulfillment.

3.4. Roles and interactions identification

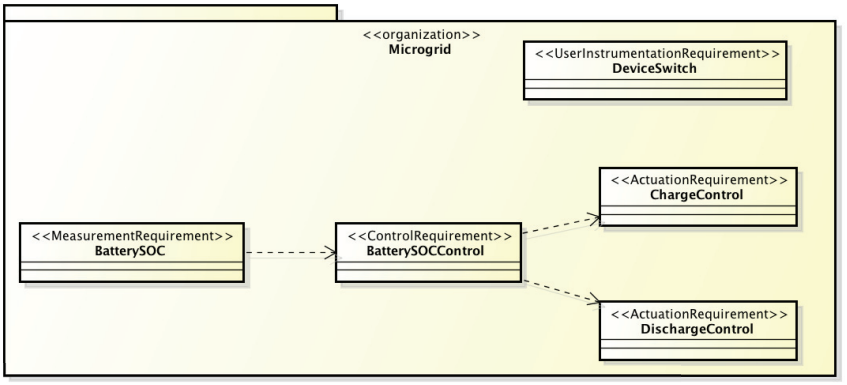

Figure 3 Organization identification example

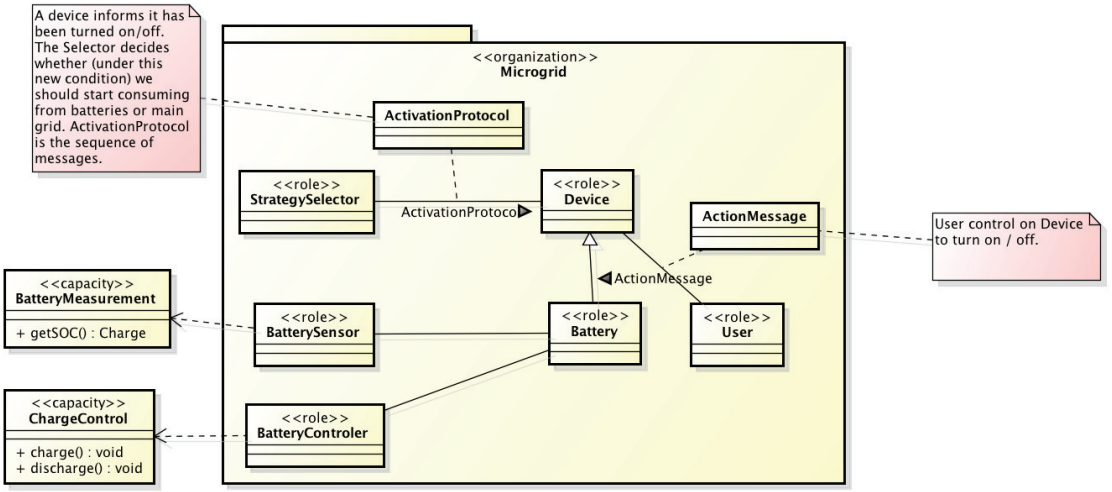

Figure 4 Interactions and role identification example

In this activity, the designer should identify which roles and interactions are required to fulfill the requirements associated to the organization. An example is proposed in figure 4. “ControlRequirement” represent behaviors that the organization should contain, within ASPECS these behaviors can be represented as “Roles”. We can then say that each “ControlRequirement” is mapped to at least one role. In our example, *BatterySOCControl* is divided into two roles: *Battery*, which actually controls the *Device*, and *StrategySelector*, which goal is to decide when it is appropriate to consume energy stored in batteries or energy coming from the main grid.

On the other hand, “MeasurementRequirements” and “ActuationRequirements” represent actions that agents should be able to perform on the physical devices (e.g. turn a device on or off). These actions can be represented by the agent’s capabilities using the concept of “Capacity”. Then, in our approach, a capacity represents one or more Actuation or Measurement requirements. In the example, the *BatteryMeasurement* capacity represents the measurement of batteries’ SOC and *ChargeControl* represents the capabilities of both actuation requirements (i.e. *ChargeControl* and *DischargeControl* in figure 1). This decision of merging two “ActuationRequirements” into a

single capacity may ensure that both actions cannot be performed simultaneously (i.e. we can not discharge and charge the battery at the same time). Additional roles can be obtained from the problem ontology (figure 2), such as *Device* and *User*. Finally, interaction protocols are introduced to ensure proper information exchanges between roles (i.e. *ActivationMessage* and *ActivationProtocol*).

4. Related works

Approaches that reify MAS for giving agents awareness of their environment using specific meta-models and concepts in order to promote the aspects related to interactions and controls are not new. Indeed, artifacts[9], for example, allows the modeling of shared infrastructures with organizational based MAS.

MAS methodologies have already been applied for MAS immersed in real life such as cooperative robotics systems[10]. The main differences with the contribution presented in this paper are based on the use of specific concepts and activities related to sensors/effectors and the focus put on control aspects that are key for the analysis and design of CPS. Moreover, even if not presented in this paper due to the paper space limit, one of the benefits of using ASPECS is that ASPECS can handle holonic based MAS that can be fitted for CPS modeling and deployment and the subsequent ASPECS process models allows: the use of formal verification techniques, the prototyping of MAS models. The SARL platform[11] supporting ASPECS can ease the implementation and deployment of MAS.

5. Conclusion

In this paper, we have presented an approach that can support the analysis and the definition of prototypes of CPS. This approach relies on the definition of several activities that belongs to a methodological process dedicated to MAS. The use of MAS is particularly fitted for CPS as it allows the modeling and simulation of the different components of the CPS and its environment. The different activities that take place in the early analysis phase use and extend SysML/UML profiles in order to represent the different intermediate models that will lead to a complete analysis of the system-to-be. Some guidelines leading the analysis are proposed in this context. An example is used for illustrate the different activities. This example is extracted from the design of a smartgrid simulator[7].

References

1. Garrison Greenwood, John Gallagher, Eric Matson, Cyber-Physical Systems: The Next Generation of Evolvable Hardware Research and Applications, Proceedings of the 18th Asia Pacific Symposium on Intelligent and Evolutionary Systems, Volume 1, Springer International Publishing 2015
2. Ragunathan Rajkumar, Insup Lee, John Stankovic, and Lui Sha. Cyber-Physical Systems : The Next Computing Revolution. In Design Automation Conference, page 1, Anaheim, California, 2010.
3. M. Cossentino, N. Gaud, V. Hilaire, S. Galland, and A. Koukam. aspecs : an agent-oriented software process for engineering complex systems. Autonomous Agents and Multi-Agent Systems, 20(2) :260–304, 2010.
4. Cossentino, M.; Hilaire, V.; Gaud, N.; Galland, S. & Koukam, A. The ASPECS Process, Springer, chapter 2. . 2013
5. Sanford Friedenthal, Alan Moore, Rick Steiner. A Practical Guide to SysML, Third Edition: The Systems Modeling Language. Morgan Kaufmann. 2014
6. T. Ritala, S. Kuikka, "UML Automation Profile: Enhancing the Efficiency of Software Development in the Automation Industry", pp.885-890 5th IEEE International Conference on Industrial Informatics (Volume 2), 2007
7. Basso, V. Hilaire, F. Lauri, D. Paire, and A. Gaillard. A Principled approach for smart microgrids simulation using MAS, in 'MATES/JAWS Workshop - MAS&S'. 2013
8. FIPA, “Fipa rdf content language specification,” Tech. Rep. XC00011B, 2001.
9. Jomi F. Hübner, Olivier Boissier, Rosine Kitio, Alessandro Ricci, Instrumenting multi-agent organisations with organisational artifacts and agents, Autonomous Agents and Multi-Agent Systems, Volume 20, Issue 3, pp 369-400, May 2010.
10. Scott A. DeLoach, Eric T. Matson, Yonghua Li Applying Agent Oriented Software Engineering to Cooperative Robotics FLAIRS-02 Proceedings. Copyright AAAI © 2002
11. Sebastian RODRIGUEZ, Nicolas GAUD, Stéphane GALLAND. SARL: a General-Purpose Agent-Oriented Programming Language In Proc. of International Conference on Intelligent Agent Technology (IAT14), pp. 103-110, Warsaw, Poland, IEEE Computer Science, 2014. DOI: 10.1109/WI-IAT.2014.156.