Towards a Framework for Applying the Visualization of Smart Monitoring Architectures to a Distributed Ubiquity Mobility Platform

Djamel Khadraoui and Christophe Feltus
Luxembourg Institute of Science and Technology (LIST)
E-mail: djamel.khadraoui@list.lu

***Abstract*—Smart Mobility is proved to be a high priority topic in regard to arising European societal challenges. Deploying smart mobility required both technological and monitoring knowledge, and one important key features of the initiative stay in the multiplicity of the final users. Its goal is, depending on the type of users, to provide the required accurate data through a dynamic monitoring application. This implies to collect data coming from physical sensors deployed in all the parking areas of a region. Those sensors are simple, meaning that the information that they can collect is limited to an entry or exit signal of a vehicle. This paper presents an architecture for applying the visualization of smart monitoring architecture to a distributed ubiquity mobility platform and show a deployment in the frame of a use case. The later has been developed in a European region and consists in a smart mobility monitoring project.**

Keywords-Mobility; Visualisation; Model; Self-adaptability; Self-management; Monitoring; Automatic Context-aware system.

I. INTRODUCTION

Smart Mobility is proved to be a high priority topic in regard to arising societal challenges. Deploying smart mobility required both technological and monitoring knowledge. In the frame of a use case, which has actually been developed in a European region, and which consists in a mobility monitoring project (actually implemented), one important key feature of the initiative stay in the multiplicity of the final users. Its goal is, depending on the type of users, to give the wanted data through a dynamic monitoring application. This implies to collect data coming from physical sensors deployed in all the parking areas of a region. Those sensors are simple, meaning that the information that they can collect is limited to an entry or exit signal of a vehicle. Another data that has to be collected in this scenario is the live traffic data from the same geographical region; once again, the type of data is simple; the number of passing vehicles for each road of the region is collected in a predefined and fixed period of time. The need of monitoring is not a new challenge in computer science since a lot of solutions are proposed until this day. The fact is that the monitoring can be effective for a project only if it is completely applied for the problem while it should give the right information to a specific user (physical or not). For our case, a famous delivery company needs an effective and complete monitoring for all its parking spots around a big urban center. So, in collaboration with the city administration, this company needs a platform to handle a large amount of data and transform it into valuable information for their daily operations, optimizing their routines. This platform aims at monitoring the trips of their employees around the urban area and to give them an exact live situation. The solution responds to the business needs of an organization and provides to the different users a dynamic monitoring of the data combined with specific business rules. Using one deployed platform, collected and analyzed data are accessible from different final users with distinctive needs. In parallel to the monitoring of their employees, the targeted system is also able to provide important information to the city administration around live traffic levels and parking availability. Another view of the system could also be the notification to citizens about the roads congestion of the city. However, the interfaces must be readjusted for each case and administration solutions have to be adapted to the user and his rights among the system. Therefore, it is obvious that the current solution must be extended with new functionalities that should be able to be added without any new implementation of the gathering platform.

Smart monitoring systems consists in solutions which monitor, control and support the decision making related to security issue of complexes and critical systems (and information systems) spread out over disseminated areas. Hence, smart monitoring architecture seems to be the most relevant approach for the monitoring and decision making provided that they are designed to deal with increasingly sensitive and crucial situations for an economy or country (like the healthcare, the power distribution, the telecom, etc.) and consists in complex, sophisticated and integrated systems which support people in governing and monitoring a plethora of knowledge generated by critical infrastructures (CI – in military, energy, transport, industries, and healthcare) [1]. In our previous work, we have first defined a metamodel for the components of the smart monitoring architecture [2]. This metamodel has been elaborated acknowledging traditional enterprise architecture metamodel (EAM) and it allows modelling each component according to a similar structure. Afterwards, we have proposed a complement [2] to explore the enterprise architecture model ArchiMate® and to redesign its structure in order to comply with component software actors'

 ISBN: 978-1-61208-555-5

characteristics, specificities and domain constraints. The principal focus of this paper concerns the design and the consideration of the policies that are centric concepts related to the activation of component's comportment. Our new contribution consists in the modeling of the monitoring system platform and the definition of the policies according to these models.

The paper is structured as following: next section presents the OCTOPUS platform (model and software) that we designed, Section III presents the OCTOPUS platform augmented with a smart monitoring solution. Section IV illustrates the monitoring interface for Smart Mobility in the frame of the augmented OCTOPUS platform and discusses the approach. Section V presents related works and last section concludes the paper and presents futures works.

II. OCTOPUS PLATFORM

OCTOPUS is a multi-agent platform; all the technologies related to agents are combined to provide a system for solving a data gathering and monitoring problem in an adaptive way. Being a multi-agent system, OCTOPUS has basic MAS characteristics as autonomy, local view and decentralization.

All the agents are autonomous and partially independent: the shutdown of an agent does lead to a platform's deactivation. Furthermore, the agents can continue their execution if the system has to reboot for any reason in order to ensure that their behavior is unchanged and that the data gathering is operational even if the remote communication is temporary deactivated.

No local agent has a global view of the platform and the main behavior aside from data gathering of the agents is to communicate to remote agents. In this way, a decentralization of the processes is effective; all the agents collect specific data and spread information to a controlling component of the system. This controlling component and its particular communication with the remainder of the system is the main defining characteristic of OCTOPUS.

OCTOPUS defines several components to achieve the deployment of an adaptive multi-agent system with different views of monitoring data and a particular communication routine to implement the constraints of the problem. Those constraints are the rules that each agent has to follow and depending on them, each agent changes his behavior.

OCTOPUS platform presents a hierarchy between the sub-platforms; containers grouping agents that are remotely connected. Throughout this hierarchy, the system defines types of agents that have a specific role. Each agent's sub-platform has an implemented behavior and specific role. A *Brain* agent is implemented, which is the management component, connected to all the agents of the sub-platform. All data gathering agents are waiting for rules from this *Brain* agent and are sending feedback in return. When necessary, *Brain* agents can also be part of a global hierarchy, in which a *Super-Brain* takes care of their organization and management. This way, each *Brain* can provide a view for a specific level of work; a main, administration view of the entire system is provided by the *Super-Brain* (see Figure 1). The P_n components represent agent's sub-platform containing *P* type agents while the T_n components represent implemented *T* type agent's sub-platform. In this case, the system is composed of a single *Brain*, a communication and organizing instance sub-platform.

This *Brain* is a sub-platform containing agents which remotely connects all the agents existing in its network. These agents are waiting for information collected from the *T* and *P* sub-platforms. The Brain is able to send this data to a monitoring interface through messages. The selection of the view and the type of data to be sent to the monitoring component remains at the sole discretion of the *Brain*. The main purpose of the *Brain* is to send rules to the connected sub-platforms of agents and receiving data from them. This way, the untreated data is sent from lower levels (*T* and *P* sub-platforms) to higher (*Brain*). Finally, this system example is a lower level of OCTOPUS itself; it is only one of the "*tentacles*" of the final architecture.

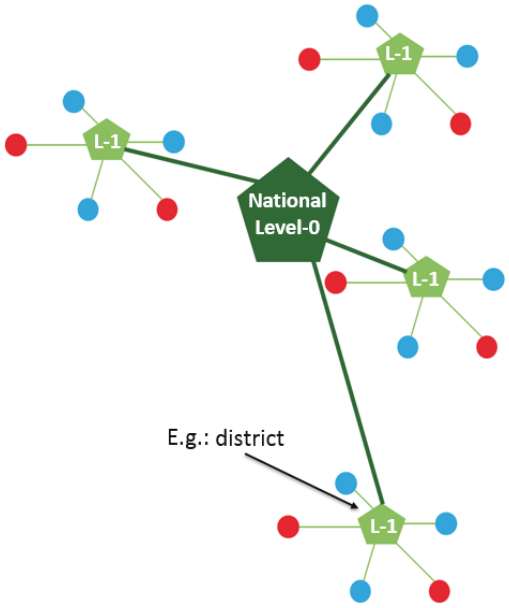

Figure 1. OCTOPUS global architecture

Obtaining a complete OCTOPUS instance is possible with adding one hierarchical level to the previous example. Figure 1 presents an architecture, in which all the *Brains* (with their *T* and *P* connected sub-platforms) are linked to a *Super-Brain*. Such a component has the same role as a *Brain*, but the collected data is coming from *Brain* sub-platforms. In this case, a global monitoring of the platform is possible and the rules are sent to the *Brains* of the network. The architecture is typically the same but with one higher level of hierarchy.

Finally, such architecture introduces a two-way data and rules flow: data collected from *T* and *P* type agents is sent to

 ISBN: 978-1-61208-555-5

Brain sub-platforms and after analysis, forwarded to the *Super-Brain*.

In return, rules are pushed from *Super-Brain* to the others *Brains* and their establishment inside *T* and *P* sub-platforms (through the agents). This bi-directional data flow is yet another key functionality of OCTOPUS platform. The generic examples presented in this section are only possible instances of OCTOPUS, the system can be adapted to different scenarios, data type and number of agents following the same hierarchical architecture. The *T* and *P* agents' type is an example of generic implementation of agents gathering data. Specific characteristics of agents and their types are descripted in next sections.

III. OCTOPUS PLATFORM AUGMENTED WITH A MONITORING SOLUTION

This section introduced the monitoring approach proposed by our OCTOPUS framework.

A. Smart monitoring platform metamodelling insights

The smart monitoring platform metamodel has been largely, and with many details, presented in [2]. This section recalls and summarizes the theoretic foundation and premise of our research in this area. The goal in modelling the monitoring system into a layered architecture metamodel is to furnish CI actors with solutions for governing the platform (monitoring and decision making support mechanism). In our previous work [3], extended smart monitoring platform metamodel using the *ArchiMate*® metamodel was elaborated to provide and support the use of a multiple layered approach of a monitoring component based on dynamic and autonomous policies.

To generate the OCTOPUS platform, we realized a specialization of the original *ArchiMate*® metamodel for the monitoring components. First, we redefined and structure the *Core* of the metamodel in order to figure out the semantic of the *Policy* [14] [17] (see Figure 2). The *Core* represents the handling of *Passive Structures* by *Active Structures* along the realization of *Behaviors.*

Concerning the *Active Structures* and the *Behavior*, the *Core* differentiates between external concepts which represent the way, in which the architecture is being perceived by the external elements (as a *Sub-Brain of a type T or P* attainable by means of an *Interface* or communicating with the *Brain*), and the internal elements which is composed of *Structure Elements* (*Roles*, *Components*) and linked to a *Policy Execution* concept. *Passive Structures* contains *Object* (e.g., *data or organizational object)*, which represents architecture knowledge. Secondly, the concept of *Policy* has been defined in accordance to the platform metamodeling approach. The proposed representation is composed of three elements which allow defining the *Policy structure*: (1) the "Event" that is defined as a trigger generated by a *Structural component* that generates the realization of a *Policy*, (2) the "Context" whish symbolizes a configuration of *Passive Structure* that allows the *Policy* to be realized. In the case of Octopus, the context includes the sub-region environment specificities (3) the "Responsibility" [4][5][12][13][16] which is the more rich semantic concept and which is defined as a state assigned to a component (human or software) to specify obligations and rights in a specific context (Feltus et al., 2014).

Thereby, the responsibility corresponds to a set of behaviors that have to be realized by means of *Structure Elements*. That behavior may also use *Objects* of y type *Passive Structure* or modify values. With these three elements, we generate an auxiliary *Policy artefact* that mirrors the fulfilment of a set of *Responsibilities* [2] in a specific monitoring *Context* and in response to a predefined *Event*. Through the *Policy Concept*, we show that each operation done by the monitoring components can be transferred into a *Policy Execution*.

Although there is a clear semantic difference in *ArchiMate*® between the business user (human or machine) which exploits an application, and the application itself, in the smart monitoring field, we consider that actors and roles are played by components that we define as being a specific *Structure Elements* acting in Critical Infrastructure environment. As a result, three level are necessary to structure the metamodel for the monitoring domain: (1) The *Organizational Layer* offers services and products to external customers that are represented in the organization by organizational processes performed by *Organizational Roles* according to *Organizational Policies*. (2) The *Application Layer* supports the *Organizational Layer* with *Application Services* which are realized by *Applications* according to *Application Policies*. (3) The *Technology Layer* which offers *Infrastructure Services* needed to run applications, performed by system software, computer and communication hardware.

Concepts and colors were taken from the original *ArchiMate*® language, except for *Organizational Function* and the *Application Function* which were switched with the *Organizational Policy* component and the *Application Policy* component. Based on the following analysis, we have defined the *Organizational Policy* as "*the rules which define the organizational responsibilities and govern the execution of behaviors, at the organization domain, that serve the product domain in response to a process domain occurring in a specific context, which is symbolized by a configuration of the information domain"*

And we have defined the *Application Policy* as "*the rules that define the application responsibilities and govern the execution, at the application domain, of behaviors that serve the data domain to achieve the application strategy.***"**

B. Smart monitoring system metamodel layers

The three layers which structure the smart monitoring platform metamodel (see Figure 2) are from down to top: the *technical level*, the *applicative level* and the *organization* or *business level*.

The *Technical Layer* is used to represent the structural aspect of the system and highlights the links between the *Technical Layer* and the *Application Layer* and how physical pieces of information called *Artefacts* are produced or used. The main concept of the *Technical layer* is the *Node* which represents a computational resource, on which *Artefacts* can be deployed and executed. The *Node* can be accessed by other *Nodes* or by components of the *Application Layer*. A *Node* is composed of a *Device* and a *System Software* [6]. *Devices* are physical computational resources where *Artefacts* are deployed when the *System Software* represents a software environment for types of components and objects. Communication between the *Nodes* of the *Technology Layer* is defined logically by the *Communication Path* and physically by the *Network*.

An *Organizational Object* defines unit of information which relates to an aspect of the organization. At the *Application layer*, this is used to represent the *Application Components* and their interactions with the *Application Service* derived from the *Organizational Policy* of the *Organizational Layer*. The concept of the components in the metamodel is very similar to the components concept of *UML* (UML 2) and allows representing any part of the program. Components use *Data Object*, which is a modelling concept of objects and object types of *UML*. Interconnection between components is modelled by the *Application Interface* in order to represent the availability of a component to the outside [3] (implementing a part or all of the services defined in the *Application Service*). The concept of *Collaboration* from the *Organizational Layer* is present in the *Application Layer* as the *Application Collaboration* and can be used to symbolize the cooperation (temporary) between components for the realization of behavior. *Application Policy* represents the behavior that is carried out by the components.

The *Organizational Layer* highlights the organizational processes and the associations with the *Application Layer*. Firstly, the *Organizational Layer* is defined as an *Organizational Role* (e.g.: *Alert Detection Concept*). This role, accessible from outside the monitoring behavioral structure through an *Organizational Interface*, performs behavior based on and according to organization's policy (*Organizational Policy component*), which are associated with the role. Afterwards, the components are able (depending on their roles – but also *function* is some cases) to interact with other roles to perform behavior; this is symbolized by the concept of *Role Collaboration* outside.

Organizational Policies are behavioral components of the organization whose goal is to achieve an *Organizational Service* to a role following *Events*. *Organizational Services* are contained in *Products* accompanied by *Contracts*. *Contracts* are formal or informal specifications of the rights and obligations associated with a *Product*. *Values* are defined as an appreciation of a *Service* or a *Product* that the *Organization* attempts to provide or acquire. The complete smart monitoring platform metamodel is the union of the three layers. As shown below, new connections between the layers have appeared.

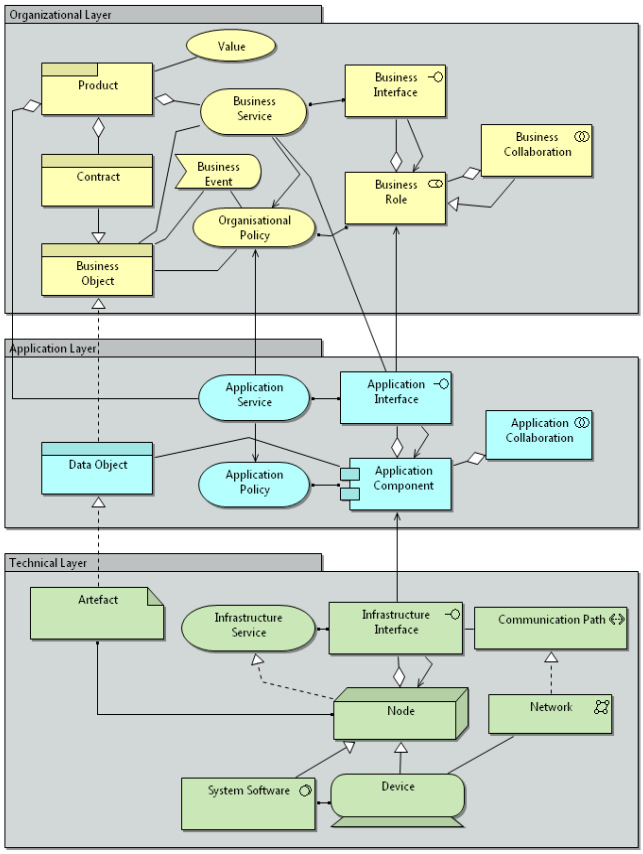

Figure 2. Smart monitoring platform metamodel

For the *Passive Structure*, we observe that *Artefact* of the *Technical Layer* realizes *Data Object* of the *Application Layer* which, itself, realizes *Organizational Object* of the *Organizational layer*.

The *Behaviour* concept association shows that the *Application Service* uses the *Organizational Policy* to determine the services that it sustain. In the same manner, the *Technical Layer* bases its *Infrastructure Service* upon the *Application Policy* of the *Application Layer*. Concerning the *Active Structure* connections, the *Role* concept determines, together whit the *Application Component*, the *Interface* provided in the Application layer. The Interface of the Technical Layer is also based on the components of the Application Layer. The modelling language related to the above artefact is available in The Open Group [19].

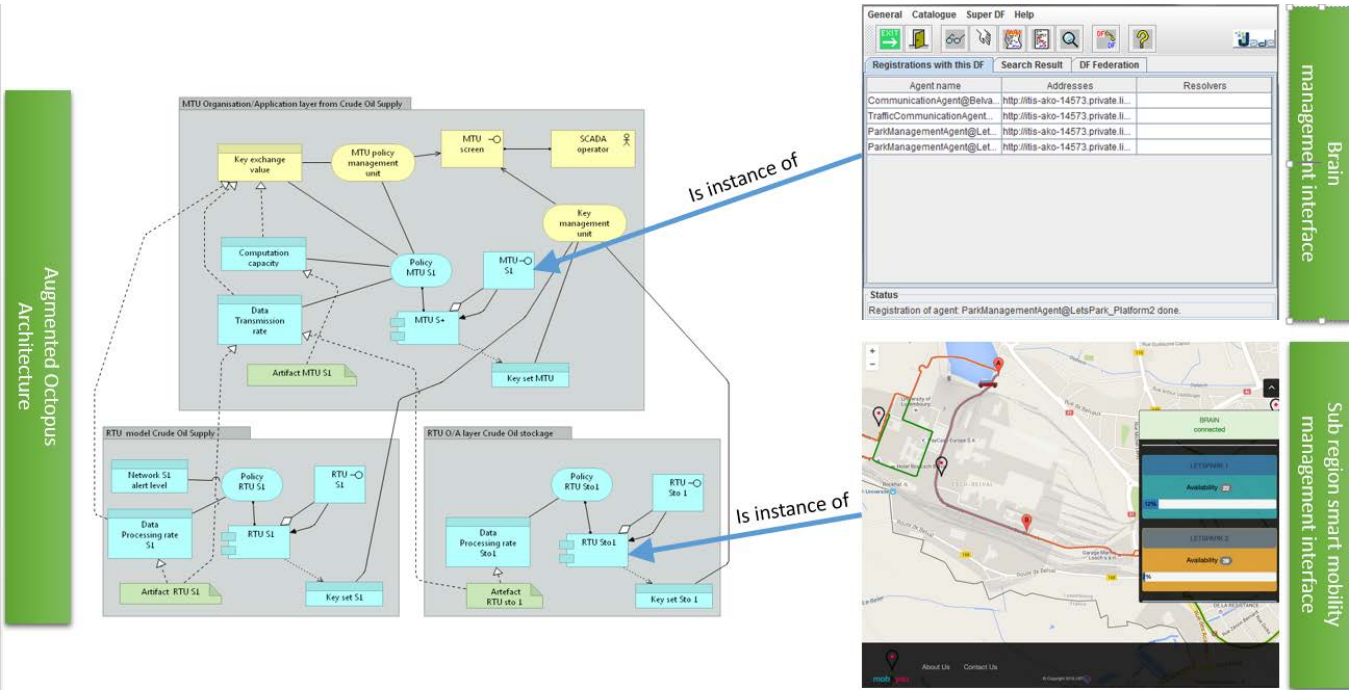

Figure 3. Smart Mobility management and sub-region interfaces

IV. VALIDATION IN THE FRAME OF A MONITORING INTERFACE AUGMENTED FOR SMART MOBILITY

This section aims at reporting and evaluating how OCTOPUS augmented has been designed for a specific Mobility management steering interface. Therefore, we review and validate the advantages and the improvements provided by the implementation that has been specifically required for this use case in the mobility domain.

As explained earlier, the monitoring architecture is defined based on generic agents easily instantiable for whatever cases, but steering interface is always dependent of the type of monitoring developed. Figure 3 shows the interface for the implementation of the smart mobility solution in the region. This interface content a monitoring frame including static information (e.g., map of the region, frames for the parking monitoring, etc.) and dynamic information (e.g., level of traffic jam on specific road, amount of places available at each parking, etc.)

Aside the monitoring interface, additional management functionalities are also available. These functionalities are not presented in the paper. They concern the management of the users of the solution, the creation of specialized viewpoints for each type of user requirements, the dynamic definition of "business rules" in order to configure the behavior of the different agents and hence, to suggest user mobility decisions.

This needs to be put in parallel with the three constraints related to the key management broadcasting mechanism related to the smart monitoring platform architecture have been defined by Bailey et al., 2003 [7] and need to be considered along the modelling of the policies: (1) the computational capacity limit, which may be represented as an artefact of a type data object at the application layer of the MTU, (2) the low data transmission rate which is also a concept related to the MTU by means of a data object, and (3) the real-time processing that needs to be consider to prevent data processing delay and which may be represented as a data object from the RTUs structures.

The definition and the exploitation of the proposed augmented OCTOPUS framework in the mobility area has demonstrate to what extend the solution offers flexibility and usability to the business administrators. Indeed, most of the manipulations (e.g., traffic decisions, road optimization, informed communication, etc.) performed by the platform operators has been realized more intuitively and with more accuracy than with previous version of the frameworks.

V. RELATED WORKS

Literatures explain methodologies to model Multi-Agent System (MAS) [18] and their environments as a one layer model and give complete solutions or frameworks. Gaia [8] is a framework for the development of agent architectures based on a lifecycle approach. AUML (http://www.auml.org), and MAS-ML [9] are extensions of the UML language for the modelling of MAS but do no longer exist following the release by the OMG of UML 2.0 supporting MAS. Prometheus defines a metamodel of the application layer and allows generating organizational diagrams, roles diagrams, classes' diagrams, sequences diagrams and so forth.

 ISBN: 978-1-61208-555-5

The Prometheus approach permits hence to generate codes but does not provide links between diagrams and therefore makes it difficult to use for alignment purposes or with other languages (e.g., MOF, DSML4MAS [10]). CARBA provides a dynamic architecture for MAS similar to the middleware CORBA based on the role played by the agent. Globally, we observe that these solutions aim at modelling the application layer of MAS [11]. CARBA goes one step further introduces the concept of Interface and Service. This approach is closed to the solution based on *ArchiMate*® that we design in our proposal but offers less modelling features. As we have notice that agent systems are organized in a way close to the enterprises system, our proposal analyses how an enterprise architecture model may be slightly reworked and adapted for MAS. Therefore, we exploit *ArchiMate*® which has the following advantages to be supported by The Open Group. It has a large community and proposes a uniform structure to model enterprise architecture. Another advantage of *ArchiMate*® is that it uses referenced existing modelling languages like UML.

As a conclusion of the related work, we may consider that our approach may be used in parallel to existing solutions while, in the same time, complete their added value in a set of business driven dimensions like the visualization of the system or the elaboration of integrated and self-contain two types of policies. The evolution of our approach may also be regarded following the performance generated at the metric level. Indeed, contrarily to solutions presented through the state of the art, our proposal fit fully with the measurement theory requirement and, hence, may be more pragmatically devoted to performance based design of critical and highly sensitive infrastructures.

VI. CONCLUSIONS AND FUTURE WORKS

Monitoring systems are important solutions to secure critical infrastructures against traditional and cyber-attacks threats. Those systems need to be accurately managed and protected in terms of interconnection, homogeneity and real time reaction. Therefore, the paper proposes an integrated approach for modelling the monitoring architecture based on the enterprise architecture modelling language and more specially ArchiMate® which has been particularly tailored for smart monitoring systems.

Based on a dedicated metamodel, the paper has demonstrated how technical, application and organization policies could be designed and metamodeled, especially regarding the policy management for interconnected monitoring systems for two of its functions. All along the modelling of the platform model and the definition of the policies according to these models, we have illustrated the theory with a business case study related to the petroleum supply chain, and more specially the specific functions of crude oil supply and crude oil storage and distribution.

REFERENCES

[1] L. Briesemeister, S. Cheung, U. Lindqvist, and A. Valdes, "Detection, correlation, and visualization of attacks against critical infrastructure systems," In Privacy Security and Trust (PST).

[2] C. Feltus and D. Khadraoui,."Designing security policies for complex SCADA systems management and protection," International Journal of Information Technology and Management, 15(4), pp. 313-332.

[3] C. Feltus, M. Ouedraogo, and D. Khadraoui, "Towards cyber-security protection of critical infrastructures by generating security policy for SCADA systems," in Information and Communication Technologies for Disaster Management (ICT-DM), 2014. IEEE.

[4] G. Neumann and M. Strembeck, "A scenario-driven role engineering process for functional RBAC roles," In Proceedings of the seventh ACM symposium on Access control models and technologies, 2002, pp. 33-42, ACM.

[5] C. Feltus and M. Petit, "Building a responsibility model including accountability, capability and commitment," in Availability, Reliability and Security, 2009. ARES'09. International Conference on, pp. 412-419. IEEE, 2009

[6] G. Beydoun, C. Gonzalez-Perez, G. Low, and B. Henderson-Sellers, "Synthesis of a generic MAS metamodel". In ACM SIGSOFT Software Engineering Notes, 30(4), 2005, pp. 1-5.

[7] D. Bailey and E. Wright, "Practical SCADA for industry". Elsevier, 2003, Newnes, 288 pages.

[8] L. Cernuzzi, T. Juan, L. Sterling, and F. Zambonelli, "The gaia methodology. In Methodologies and Software Engineering for Agent Systems", 2004, pp. 69-88.

[9] V. T. da Silva, R. Choren, C. J. De Lucena, "A UML based approach for modeling and implementing multi-agent systems," in Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems-Volume 2, 2004, pp. 914-921. IEEE Computer Society.

[10] S. Warwas, C. and Hahn, C, "The DSML4MAS development environment," in Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems-Volume 2, 2009.

[11] J. J. Gomez-Sanz, J. Pavon, and F. Garijo, "Metamodels for building multi-component systems," Proceedings of ACM symposium on Applied computing. ACM, New York, NY, USA,2002 pp. 37-41.

[12] C. Feltus, M. Petit, and E. Dubois, "Strengthening employee's responsibility to enhance governance of IT: COBIT RACI chart case study". In Proceedings of the first ACM workshop on Information security governance, 2009, pp. 23-32. ACM

[13] J. Zachman, "The zachman framework for enterprise architecture". Zachman International, 2002.

[14] C. Feltus, D. Khadraoui, B. de Rémont, and A. Rifaut, "Business Gouvernance based Policy regulation for Security Incident Response," In IEEE GIIS 2007 Global Infrastructure Symposium, Vol. 6, 2007.

[15] G. Guemkam, C. Feltus, P. Schmitt, C. Bonhomme, D. Khadraoui, and Z. Guessoum, "Reputation based dynamic responsibility to agent assignement for critical infrastructure, " in Proceedings of the IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology-Volume 02, 2011, pp. 272-275.

[16] C. Feltus, M. Petit, and E. Dubois, "ReMoLa: Responsibility model language to align access rights with business process requirements, " in Research Challenges in Information Science (RCIS), 2011.

[17] A. Rifaut and C. Feltus, "Improving Operational Risk Management Systems by Formalizing the Basel II Regulation with Goal Models and the ISO/IEC 15504 Approach", in ReMo2V, 2006.

[18] B. Gâteau, D. Khadraoui, and C. Feltus, "Multi-agents system service based platform in telecommunication security incident reaction," in Information Infrastructure Symposium, 2009, pp. 1-6.

[19] http://pubs.opengroup.org/architecture/archimate2-doc/ (last access: February 2017)

 ISBN: 978-1-61208-555-5