CAPS: Architecture Description of Situational Aware Cyber Physical Systems

Conference Paper · April 2017

CITATIONS
0

READS
38

2 authors:

Henry Muccini
Università degli Studi dell'Aquila
157 PUBLICATIONS **1,715** CITATIONS
SEE PROFILE

Mohammad Sharaf
Università degli Studi dell'Aquila
18 PUBLICATIONS **50** CITATIONS
SEE PROFILE

CAPS: Architecture Description of Situational Aware Cyber Physical Systems

Henry Muccini
University of L'Aquila
DISIM Department
L'Aquila, Italy
Email: henry.muccini@univaq.it

Mohammad Sharaf
University of L'Aquila
DISIM Department
L'Aquila, Italy
Email: massharaf@yahoo.com

***Abstract*—This paper proposes CAPS, an architecture-driven modeling framework for the development of Situational Aware Cyber-Physical Systems.**

Situational Awareness involves being aware of what is happening in the surroundings, and using this information to decide and act. It has been recognized as a critical, yet often elusive, foundation for successful decision-making in complex systems. With the advent of cyber-physical systems (CPS), situational awareness is playing an increasingly important role especially in crowd and fleets management, infrastructure monitoring, and smart city applications. While specializing cyber physical systems, Situational Aware CPS requires the continuous monitoring of environmental conditions and events with respect to *time* and *space*. New architectural concerns arise, especially related to the sense , compute & communication paradigm, the use of domain-specific hardware components, and the cyber-physical space dimension.

This work illustrates the CAPS modeling languages used to describe the software architecture, hardware configuration, and physical space views for a situational aware CPS.

***Keywords*-Situational Awareness, Cyber-Physical Systems, Cyber-Physical Spaces, MDE**

I. Introduction

According to Eoin's Wood article [1], the 2020s will be the age of *intelligent connected systems*. They are characterized by *things* connected through ubiquitous networks to our systems, and providing not only services but intelligent assistance, that is, some algorithmic manipulation of big data to foresee and manage future needs.

In the context of intelligent connected systems, our team is involved in the design and implementation of *situational awareness systems*, that are, cyber-physical systems that by using sensors, CCTV cameras, and other environmental data gathering devices (e.g., smart readers, RFID tags, beacons), as well as social media information, support Situational Awareness (or simply SiA). By transforming sensed data into actionable intelligence, SiA has the ability to observe the (user's) surroundings and make detailed assessments about his environment. SiA is essential for decision making, and its lack is recognized to be one of the primary factors in accidents attributed to human errors [3].

Architecting SiA cyber-physical systems (SiA-CPS) requires the description of novel architectural views, were the physical environment, as well as its cyber dimension, plays a key role. Software components, implemented on top of SiA-specific hardware IoT devices (e.g., sensors and actuators, CCTV cameras, beacons), are required to interact in a prescribed open or closed physical space (e.g., a parking lot, a classroom, a hallway) under observation. A new view, typically referred as cyber physical space [4], [5], [6] has to be explicitly modelled, since design decisions will have to be taken accordingly.

To support the engineering of SiA-CPS, we present the CAPS architectural description, designed according to the IEEE/ISO/IEC 42010 standard [7]. The framework is aimed at supporting the architecture description, reasoning, and design decision process. This research contributes with a modeling and analysis platform to support an architecture-driven development of Sia-CPSs. The platform is called CAPS[1] and is based on a multi-view architectural approach [7] based on three (plus two) modeling languages to describe a SiA-CPS from different viewpoints, and namely: (i) software components and their interactions, (ii) the hardware specification of situational awareness equipment, (iii) the physical environment where hardware equipment is deployed.

Two subsidiary views are used to define the correspondences among the identified views.

The whole CAPS platform is realized by exploiting advanced Model-Driven Engineering (MDE) techniques, such as metamodelling, model weaving and model transformation.

The main contributions of this paper can be summarized as follows:

- an architecture description of SiA-CPS with a focus on viewpoints and views is introduced;
- a multi-view modelling platform for engineering SiA-CPS is presented in details (including a specification of the software architecture, low-level and hardware specification, and physical space);
- the application of the modeling languages to a case study is outlined.

The rest of this paper is organised as follows. Section II provides background information on the 42010 standard and on Situational Awareness. The SCUNA smart card

[1]It stands for **A**rchitecting platform for **C**yber **P**hysical **S**pace

978-1-5090-5729-0/17 $31.00 © 2017 IEEE
DOI 10.1109/ICSA.2017.21

case study is presented as well. Section III describes the proposed architecture description for SiA-CPS. Sections IV, V, VI provide a detailed discussion of the modeling languages associated to the identified views, with their application to the SCUNA case study. The two auxiliary modeling languages, MAML and DEPML are presented in Section VII. Section VIII presents related work, while the paper concludes in Section IX.

II. Background and Case Study

This section provides background information on the ISO/IEC/IEEE 42010 standard, some more information on situational awareness, and an overview of the SCUNA case study.

A. Background on 42010 Architecture Description

In this paper, we build upon the conceptual foundations of the ISO/IEC/IEEE 42010:2011, *Systems and software engineering — Architecture description* [7] standard, to investigate the essential elements of architecture description for situational aware cyber-physical systems (SiA-CPS).

The standard addresses architecture description (AD), that is, the practices of recording software, system, and enterprise architectures so that architectures can be understood, documented, analyzed and realized. Architecture descriptions take on many forms, from informal to rigorously specified models.

The content model for an architecture description is illustrated in Figure 1. *Architecture viewpoint* is a fundamental building block representing common ways of expressing recurring architectural concerns reusable across projects and organizations. It encapsulates *model kinds* framing particular *concerns* for a particular audience of system *stakeholders*. The concerns determine what the model kinds must be able to express: e.g., security, reliability, cost, etc. A model kind determines the notations, conventions, methods and techniques to be used. Viewpoints, defining the contents of each architecture *view*, are built up from one or more model kinds and *correspondence rules* linking them together to maintain consistency.

B. Situational Awareness

Situational Awareness (SiA) refers to a person's awareness of what is going on in her surroundings, the meaning of these surroundings, and using this information to decide and act.

A theoretical model of SiA has been presented by Endsley in [8]. His model describes SiA states and illustrates three stages of SiA formation: perception (i.e., the perception of the status, attributes, and dynamics of relevant elements in the environment), comprehension (including pattern recognition, interpretation, and evaluation), and projection (i.e., the ability to project the future actions of the elements in the environment). SiA is also framed within the OODA (Observe, Orient, Decide, Act) loop [9].

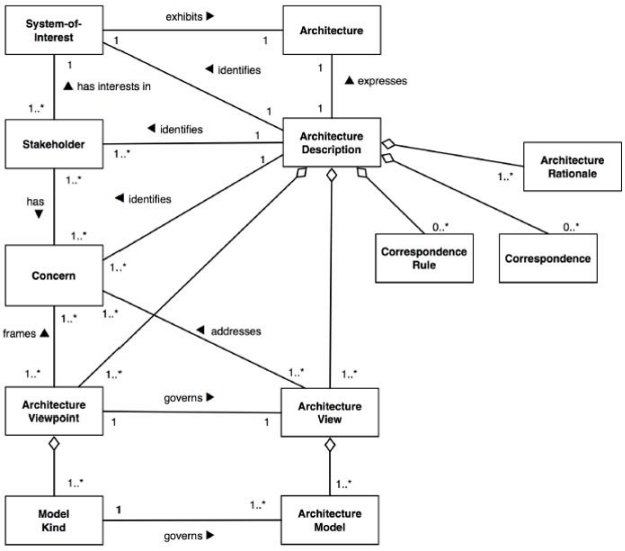

Figure 1. Content model of an architecture description (following ISO/IEC/IEEE 42010)

The growth of the market is described in the report "World Situational Awareness Systems (SAS) - Market Opportunities and Forecasts, 2015-2022"[2], where the market is projected to garner USD 32,6 billion by 2022. The reasons are many, ranging from governments initiatives for disaster management, to increasing interest in public safety, new technologies for data monitoring and collection, and the evolution of the Smart Cities concept and related.

C. The SCUNA case study

The SCUNA (Smart Card at the UNiversity of l'Aquila) project consists in using a smart card and other IoT technologies to improve the quality of the working life of students, professors, and staff. The main services are related to the access control to rooms, laboratories, and parking lots; to the monitoring of open and closed spaces; to the emergency exit management in case of safety cases such as earthquakes and fire; to the support to people with disabilities. The full list of services can be found at https://goo.gl/zyGpe1. We here focus, instead, on a few SiA services that we will use in the rest of this paper:

- conditional access due to roles: the access of an actor to certain rooms is conditional to the presence of other actors in the room (e.g., a student may enter a laboratory only if a technician is already in);
- university space monitoring: the internal and external spaces are monitored for temperature, load, available seats, and other variables;
- conditional access due to the space state: the access to a room is granted only if the room state (e.g., the number of available seats) are granted (e.g., while a student may

[2]https://www.alliedmarketresearch.com/situation-awareness-SAS-market

have access to a classroom, if the latter is too full, the student will not be granted access). A number of people in the room can be monitored through people counters, cameras, or other IoT devices.

III. The CAPS Architecture Description for SiA-CPS

This section reports the main elements comprising the CAPS the architecture description: the stakeholders and concerns, the software, hardware, and space viewpoints, and the 3+2 architecture views.

The CAPS system stakeholders and concerns

CAPS offers a fruitful modeling framework implementing separation of concerns among different modeling views, enhancing reuse, abstracting low-level details,the ability to model space and time, and representing different types of the control loop [10].

In order to provide a valuable instrument to model situational awareness, we first briefly define the main stakeholders and their concerns. This arrangement of concerns depends on our involvement in the SiA-CPS development, and on the systematic studies that we performed [11][12].

The main stakeholders are: software developers, system integrators, sensor network IoT experts, and smart building/space modelers. Overall, different interdisciplinary skills are required to develop a SiA-CPS, ranging from software production, to hardware/IoT integration, and environment modeling.

Time and *space* place an important role in SiA-CPS. Accordingly, the main concerns we found related to SiA-CPS are: ▷ *data exchange* related to the organization of data and how it is collected, managed and transformed within the SiA-CPS; ▷ *space coverage* denoting how well an environmental area is monitored or tracked; ▷ *networking and communication* denoting different networking aspects that may affect the SiA-CPS; and ▷ *energy consumption* concerns how much power is consumed by the application running on the hardware element considering the used batteries or harvested energy sources.

The CAPS viewpoints

Three are the main architectural viewpoints we found of extreme importance when describing a SiA-CPS: the software, hardware, and space viewpoints.

The software viewpoint deals with the organization of and interaction among software components. This viewpoint looks exclusively to software elements, with a specific focus on the software structure and its behavior. The typical stakeholders are software engineers and developers, and the concerns it frames are energy consumption and data exchange.

The hardware viewpoint frames the energy consumption, networking , and communication concerns. It focuses on the SiA IoT devices used to monitor, sense, and act. The associated stakeholders are system integrators and sensor network IoT experts.

The space viewpoint focuses on describing the environment (open or closed space) the user wants to become aware of. It frames the space coverage, and is associated to the smart building/space modeler stakeholders.

The CAPS views

The CAPS architecture description for Situational Aware CPS (SiA-CPS) comprehends three main views: the software architecture structural and behavioral view, the hardware view, and the physical space view. In order to create a combined software, hardware, and space view of SiA-CPS, the three proposed modelling views are linked together via two auxiliary views, denoted as Mapping Modelling Language (*MAPML*) and Deployment Modelling Language (*DEPML*).

An overview, the modeling language meta-model, and an example application to the SCUNA example for the three main views are presented in the sections below.

Our proposed approach is implemented by extending the Eclipse platform. Our metamodels are defined by using the Eclipse modeling Framework (EMF). The concrete syntax of the modelling languages has been defined by using the Graphical modeling Framework (GMF), a model-driven approach to generate graphical editors in Eclipse.

IV. The Software Architecture View Modeling Language (*SAML*)

1) Overview: The *software architecture view* allows designers to define the software architecture of the SiA-CPS application through the *SAML* modeling language. Software `components` exchange messages through `message ports`. Each component can declare a set of `application data` manipulated by actions defined in the behavior of the component. The behavior of each component is represented by a list of `events`, `conditions`, and `actions`. `Modes` can be defined as well.

2) Meta-Model: By defining the SAML underlying metamodel, we can formalize the structure and constructs of the SAML language. Figure 2 and Figure 3 show the parts of the SAML metamodel related to its structural and behavioral concepts, respectively.

The **software architecture** of a SiA-CPS is defined as a collection of software components and connections. A **component** is a unit of computation with internal state and well-defined interface [13]. The internal state of a component is denoted by the current behavioral mode of application data and its values.

An **application data** can be shown as a local variable declared in the component scope; application data are deployed by *actions, events, and conditions* defined in the component behavior, such as **StoreData**, **ReceiveMessage**, etc.

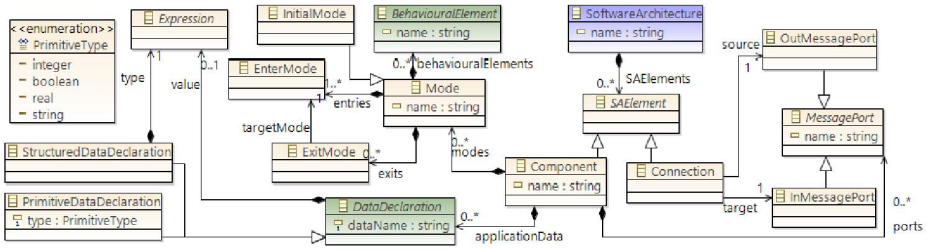

Figure 2. SAML Metamodel: structural concepts (external metaclasses in green)

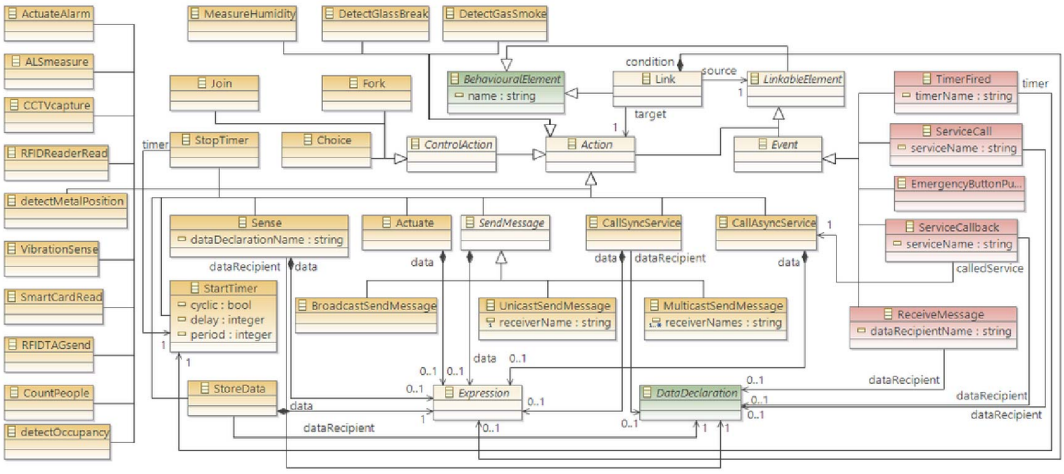

Figure 3. SAML Metamodel: behavioral concepts (actions in orange, events in red)

A **mode** indicates a specific status of the component. Examples of modes can be, energy saving mode, sleeping mode, etc. The component can only have one active mode at a time. Mode transitions occur by passing from an action called **ExitMode** to a distinct type of event called **EnterMode**. Thus, entry and exit modes can be associated with actions and events, that formalize among modes a continuous behavioral flow. Every single mode can have a set of **behavioral elements** denoted by actions, conditions, and events that all together depict the control flow within the component.

Components can interact by passing messages through **message ports**. For receiving incoming messages input message ports are used while output message ports are used for sending outgoing messages. Actual communication methods of a message (i.e., broadcast, multicast or unicast) are shown in Figure 3. In this context, a **connection** represents unidirectional communication channel between two message ports of two different components.

An **action** is a special kind of behavioral element which represent an atomic task that can be performed by the component. The action can be executed when it is triggered by an event or when a previous action in the behavioral flow has been achieved. An action can be for example: get data from a specific sensor (e.g, occupancy sensor), start or stop of a timer, send a message from a specific port, and so on. Precise types of actions can be followed : the *fork* action splits the control flow into multiple parallel executions, the *join* action merges (previously split) control flows, and the *choice* action specifies a branch in the control flow after which one and only one outgoing control flow will be executed.

An **event** is triggered in response to either some internal appliance of the component (e.g., a timer fired) or an external stimulus of the component (e.g., the message reception on a input message port). Examples of events: entering a specific mode, receiving a message at a given port, an activation of a timer, the receiving of a call from an external service, etc.

Finally, connections between events and actions that represent the control flow among them are called **links**. An architect can benefit from these links in deciding the order in which actions can be performed and the actions that must be executed when an event is triggered.

3) Application of SAML to the SCUNA case study: Figure 4 shows the SAML model of the CAPS, simply representing a partial example of SCUNA case study introduced in Section II-C. It is important to note that this figure is actually a screen-shot of our real tool CAPS. From

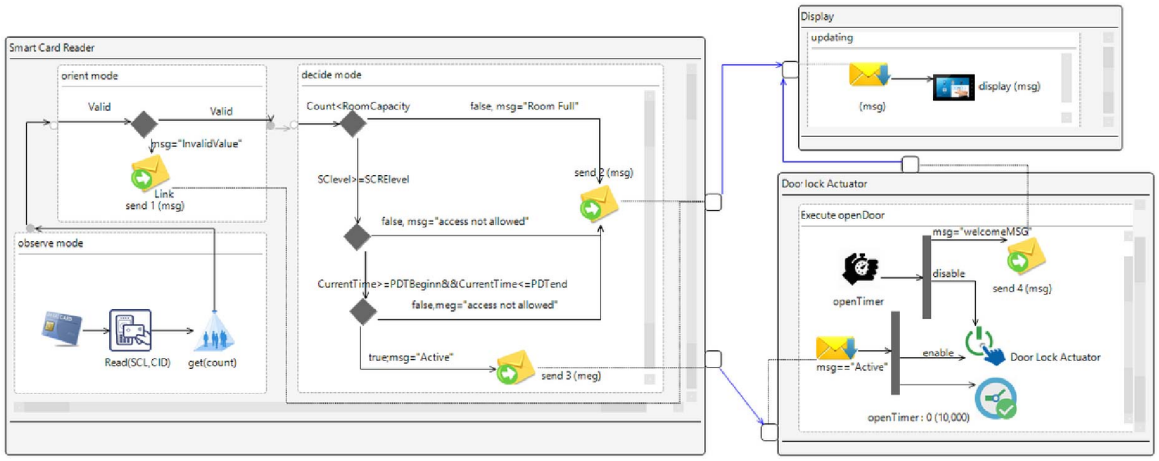

Figure 4. The Software architecture of simple scenario in SCUNA case study

a structural point of view, the shown SiA-CPS model is composed of three main components.

The *Smart Card reader* component is responsible for controlling the access to a room. It includes three modes:

1) Observe mode: the smart card reader reads the scanned smart card ID and Level. The ID is the card unique identifier. The smart card level identifies the accessibility privileges given to the card holder. For example, level one is given to undergraduate students, level two to graduate students, level three to teachers. This mode is also responsible for monitoring the number of people in the room through a people counter feature (the get(count) in the figure).
2) Orient mode: the smart card reader performs a validation check for the scanned card. If the card number is approved, the card is valid for use and the system can enter the decide mode. Otherwise, a message is sent through the output message port of the smart card reader component to the input message port in the display component to show that this card is not valid.
3) Decide mode: predefined rules are set for entering the room. This example focuses on three entering rules. First, the count of persons inside the room must be lower than the room capacity. Second, the smart card Level must be higher than or equal to the smart card reader level. Third, the time must lie within the opening and closing hours for the room (for simplicity we have just indicated a beginning time and ending time). Thus, if all the rules have been satisfied, the card holder is allowed to enter the room, and a message is sent through the output message port of the smart card reader component to the input message port of the door lock actuator component. Otherwise, if any of the rules fails, a message is sent from the output message port of the smart card reader component to the input message port of the display component to show that the access is not allowed.

The *Door lock actuator* component is responsible for opening the door. It has one mode called execute openDoor. When this mode receives an active message to the input message port of its component, the door lock actuator is enabled for ten seconds to allow the entry. This period (ten seconds) is defined by countdown timer, which has been activated when the active message has been received. After 10 seconds this event will disable the door lock actuator; as a result, the door is closed. Simultaneously, this event will trigger a sending message action, and will send a welcome message through the output port of this component to the input port of the Display component.

The *Display component* displays messages for the smart card holder. It has only one mode that is, the updating mode. It is responsible for displaying messages, that are received from other components through the input port of its component.

V. The Hardware View Modeling Language (*HWML*)

1) Overview: The *hardware view* describes the hardware characteristics of each `hardware element` to be used within a SiA-CPS. A model in HWML encompass specific low-level information of the hardware. The associated HWML model contains exclusively low-level, node-specific information, like its `memory`, `energy source`, `processor`, installed `sensing units` and `actuating units`.

2) Meta-Model: An abstract description of the HWML metamodel is depicted in Figure 5. A concise description of the main concepts defined in the HWML metamodel is given below.

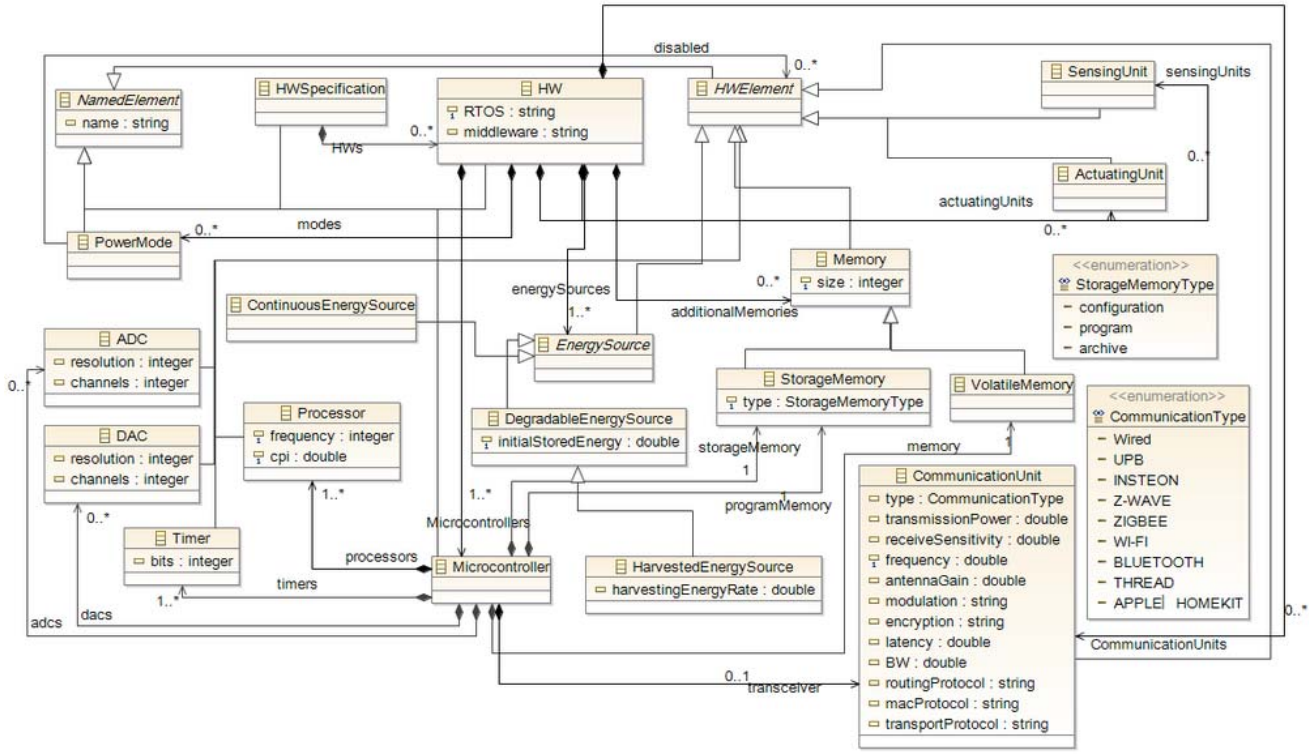

Figure 5. HWML Metamodel

The root element of a HWML model is **HWspecification**. It is a container of instances of the *HW* metaclass. A HW represents a specific HW type that can be used within the SiA-CPS. A HW can have a name (inherited from the *NamedElement* metaclass), its operating system (e.g., TinyOS), *middleware* (e.g., TeenyLIME).

A SiA-CPS HWML encompass one or more microcontroller (i.e., the component mainly devoted to computation and memory management), energy sources (e.g., batteries), a set of additional memories representing external storage memories of the element, a set of power modes in which the hardware can be at any given time, a set of sensing units, a set of actuating units, and a set of communication units to communicate with other hardware within the SiA-CPS.

An **energy source** denotes any device or equipment used to supply the node with electrical energy. Here we have three types of energy that are supported by HWML:

- **ContinuosEnergySource** that potentially never runs out. AC/DC supply is an example of this type of energy source.
- **DegradableEnergySource** that can run out at any time. As an example, batteries can be denoted as a degradable energy source in HWML.
- **HarvestedSource** that can run out at any time and harvest additional energy in some way. As an example, batteries coupled with a solar panel can be denoted as a harvested energy source in HWML.

The hardware component performing the actual readings from the environment is the **sensing unit** in HWML. It is simply the unit that measures a physical quantity and converts it into a signal which can be manipulated and analyzed by a microcontroller; zero or more sensing unit can be added to the SiA-CPS element. Our CAPS has a set of sensing units, for example, a sensing unit can get information about the lighting conditions of the environment, its current temperature, a presence of smoke or gas, count of people existing or entering a facility or zone.

An **actuating unit** is the hardware component that operate physically on the environment. Zero or more actuating units can be added to the SiA-CPS element. Examples of actuating units include: lights, motors, electronic door locks, water sprinklers, etc.

A **communication unit** is a component that allows communications with other CPS elements.

In HWML a communication unit could be RF transceiver that can operate on specific bands, and are compliant with some IEEE standard which specifies their physical layer and media access control, such as the IEEE 802.15.4 [14].

In HWML, a communication unit has some attributes related to the capabilities of the device such as transmission power, frequency, latency, bandwidth, *transportProtocol* (e.g.TCP), *macProtocol* (e.g. T-MAC), *routingProtocol* (e.g.

LEACH), and Communication type that can be of different types: *wired*, *UPB*, *INSTEON*, *Z-WAVE*, *ZIGBEE*, *WI-FI*, *BLUETOOTH*, *THREAD*, and *APPLE HOMEKIT* according to how the SiA-CPS elements have been linked.

Usually, a SiA-CPS element needs **memory** and our HWML supports two types of memories: storage memory or volatile memory.

A **micro-controller** is an electronic device integrated into a single chip and it is commonly used in embedded systems(e.g.ATMega128). In our HWML, a micro-controller can contain one or more processors, zero or more ADCs (Analog-to-Digital Converter) , zero or more DACs (Digital-to-Analog Converter), one or more timers, a volatile memory, a program memory, a storage memory, and an optional radio transceiver. A storage memory can be *configuration*, *archive*, and *program* memory. depending on how the application uses it.

Timers are devices that trigger periodically the clock of the SiA-CPS element. In CAPS a timer can be implemented either as a hardware or software component.

Different **power modes** can be defined, each one identifying a set of hardware elements (such as memory, DAC, etc.) and deciding which elements are *disabled* (all the other elements of the node are supposed to be active). For example, a given SiA-CPS node can have a *Sensing* power mode in which all the radio transceivers are disabled and hence saving all the energy needed to perform networking operations, or a given SiA-CPS node can have all the sensing devices disabled, and thus focusing only on networking operations, and so on.

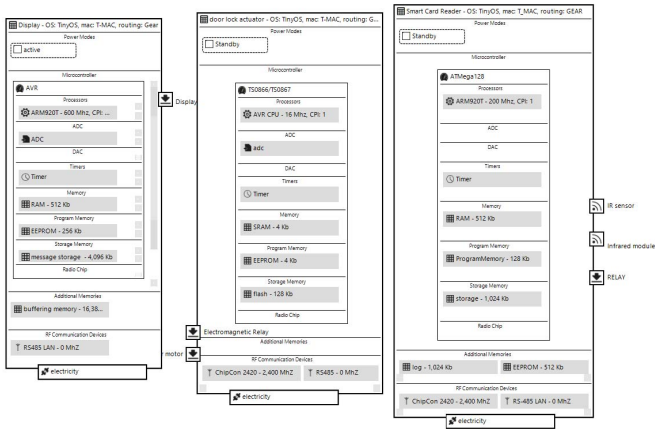

Figure 6. Hardware modeling of simple scenario in SCUNA case study

3) Application of HWML to the SCUNA case study: Figure 6 shows the HWML model developed for our SCUNA scenario. three hardware configurations have been defined:

- *smart card reader* is equipped with an *IR receiver module* for sensing the signals coming from a smart card and counting people crossing the door.This equipment is powered by a classical electrical plug connected to the main electrical system of the university. The micro-controller used is the low-power Atmel AVR ATmega128 equipped with an ADC ,different type of memory such as program memory (EEPROM).The smart card reader is always on standby power mode which will be changed to activate mode by smart card interrupt. smart card reader equipped with RS-485 for communication and USB-Port.
- *Door lock actuator* has a single actuator device which will send the active signal to the door relay and continues sending this signal for 10 seconds depending on equipped timer in the microcontroller. The micro-controller used is the low-power Atmel AVR ATmega128 equipped with timer.
- *Display* has a single actuator device for graphically showing the messages received by smart card reader component or door lock actuator on a digital display.

VI. The Physical Space View Modeling Language (*SPML*)

1) Overview: The *physical space view* describes the physical space involved in situation awareness. The SPML modeling language defines an `area` with its `coordinates`, as well as `rooms` with associated `walls`, `ceiling`, and `floor`.

2) Meta-model: CAPS provide support for developing 3D model editors and simulators, which are essential tools for engineering of SiA-CPS. Developing a 3D editor from scratch is a challenging task that requires special (and expensive) skills. CAPS framework enables engineers to specify 3D syntaxes for SPML in a declarative way which will reduce the amount of effort and need for low-level expertise. It aims to support the standardization of a language for declarative specification of 3D concrete syntaxes for SiA-CPS. This view is especially useful for developers and system engineers when they have to consider the network topology, the presence of possible physical obstacles (e.g., walls, trees) within the network deployment area, and so on.

In future work, CAPS will provide scalable repository to enable the reuse of 3D models. In this way, SiA-CPS designers can depend on the availability of modeling repository, likewise the development of complicated software systems, which are typically developed by reusing or extending existing software libraries.

Figure 7 shows the metamodel underlying the SPML modeling language.

The SPML language is about the site in the real world where the SiA-CPS equipment will be deployed. The central class in the SPML is the **CyberPhysicalSpaces** class.The **CyberPhysicalSpace** represents the overall environment in the 3D space in which the SiA-CPS nodes will be deployed.

Any kind of SiA-CPS element **(cyber element or physical element)** can be placed in the environment. Each element is characterized by the name, the abscissa and the ordinate of

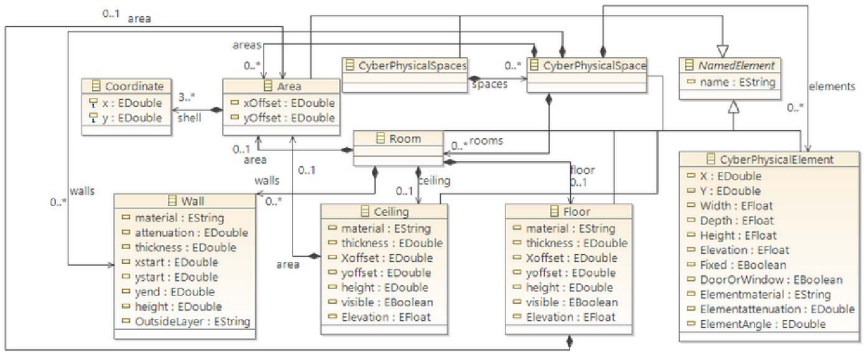

Figure 7. SPML Metamodel

the center of this element, dimensions (width, depth, height), elevation, fixed or movable, door or window, the angle, material type, and its attenuation coefficient of this material. The attenuation coefficient is a decimal number ranging from zero (when it is totally irrelevant when considering radio signal attenuation, e.g., a sheet of paper), to one (when it totally blocks radio signals, e.g., a panel made of lead). The attenuation coefficient is one of the most important parameters when considering path loss models, network connectivity and coverage, and so on. The element is given by a sequence of **coordinates** representing the perimeter of the obstacle in the 3D space.

In SPML an **area** identifies a portion of physical space in which element can be distributed . The shape of this area is given by its *shell*: a sequence of **coordinates** representing the perimeter of the area in the 2D space. A **Wall** is a class characterized by name of the material, attenuation coefficient, thickness, and the abscissa and the ordinate for the beginning and the end of this wall. To avoid unnecessary repetition of effort, we used existing extensible 3D modeling environments (and specifically, Sweet Home 3D) to represent the space model generated by CAPS. Sweet Home 3D supports the implementation of new plug-in files to develop new features [15].

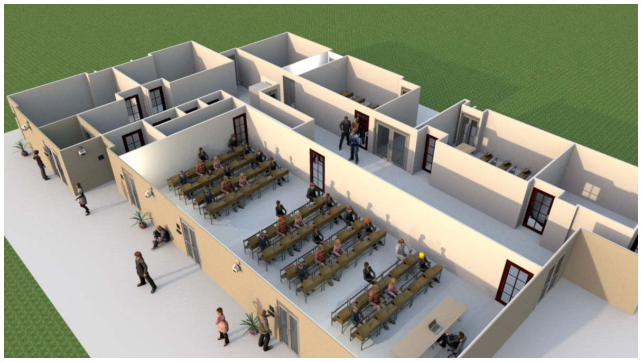

Figure 8. Space modeling of simple scenario in SCUNA case study

3) Application of SPML to the SCUNA case study: Figure 8 shows the SPML model representing the physical environment of our SCUNA scenario. It is the real ground floor of the Alan Turing building in university of LAquila .It contains many kinds of obstacles that are concrete walls dividing the whole building into rooms and corridors, a main class door on the left, and a glass door for each room. Each physical element is represented by a unique name, its attenuation coefficient, and the coordinates of all the points of its perimeter, and many other attributes. The physical environment of our SCUNA scenario contains many deployment areas. At each door, we have smart card reader and screen and connect it to door lock actuator. Also from the figure, we can see number of SiA-CPS elements deployed in the environment.

VII. The auxiliary views: the Mapping View Modeling Language (MAPML) and the Deployment View Modeling Language (DEPML)

1) Overview: The *mapping* and *deployment* views are used to create a combined software, hardware, and space view of SiA-CPS. The former view uses the MAPL modeling language to link together SAML to HWML models in order to assign software components to the corresponding hardware configuration in which they will be executed on. The latter view uses the DEPML language to link together HWML to SPML for virtually deploying SiA-CPS elements into precise space within the physical environment.

2) Meta-models: MAPML model represents the typical notion of deploying software components into hardware resources[13]. MAPML is an intermediate model between SAML and HWML model and its presence help in clearly separating the application layer from all other lower levels of SiA-CPS. Accordingly, architects can focus on the application from a functional point of view in SAML, while other designers can focus on low-level aspects of the SiA-CPS in HWML. The metamodel for MAPML is depicted in Figure 9.

Referring to MAPML metamodel, **mapping** element is

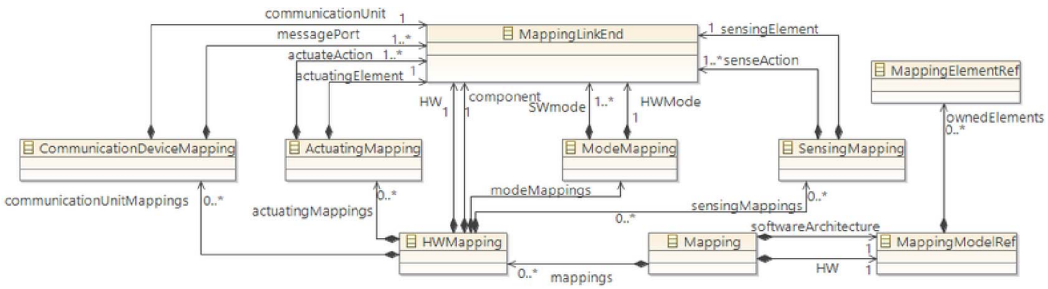

Figure 9. MAPML Metamodel

the root of MAPML model that reference the linked SAML and HWML models through *softwareArchitecture* and *HW* containment references, respectively.

A **Mapping** element is also made of a set of **HW mappings**, each of these HW mappings links a HW definition from the HWML model and a component from the SAML model, that means the linked component in the SAML model will be physically deployed on the linked HW in the HWML model. A group of secondary links can be encompassed within HW mapping , each of these links can be seen as a refinement of the HW mapping. Secondary links are:

- **SensingMapping** this link allows designers to specify to which physical sensing unit does a sense action refer to. It maps a *Sense* action in a SAML component to a *Sensing Unit* hardware in a HWML.
- **ActuatingMapping** It is similar to the *SensorMapping* concept, but it refers to actuating unit, rather than sensing unit. It maps an *Actuate* action in an SAML component to an *Actuating unit* hardware in a HWML.
- **CommunicationDeviceMapping** It maps an SAML message port of the component linked by the parent *HWMapping* to a HWML communication unit.
- **ModeMapping** this link maps a mode defined in an SAML component to its corresponding mode in a HWML. It allows designers to disjoin the mode concepts in SAML and HWML. Designers can have modes definitions specified in pure software world that can be independent of the modes definition specified in the hardware physical world.

An overview of the metamodel underlying the DEPML language is shown in Figure 10.

A DEPML model encompasses only one type of link, called **DeploymentLink**, that links together a hardware configuration in HWML and a position in SPML. The meaning behind the deployment link is that the linked hardware configuration will be instantiated and deployed virtually multiple times in the linked position. DEPML allows designers to consider each equipment configuration defined in a *HWML* model and to *instantiate* it in a specific position within the physical space defined in a *SPML* model. Therefore,designers are allowed to concentrate on generic

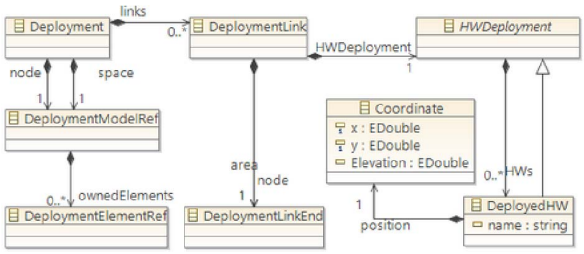

Figure 10. DEPML Metamodel

components and hardware types in SAML and HWML. The DEPML modeling editor is similar to the MAPML one. It is composed of three panels given a tree-based demonstration of the HWML, SPML and deployment links of DEPML, respectively.

VIII. Related Work

The modeling languages presented in this work are inspired by the A4WSN framework presented in [16]. A4WSN is a Model-Driven Engineering framework to support an architecture-driven development and analysis of wireless sensor networks. Since our aim with CAPS is to describe SiA-CPS (instead of wireless sensor networks), we extended and adapted the SAML behavioral modeling language, created the hardware modeling language reflecting SiA-CPS hardware, and focuses on 3D modeling of a cyber-physical space.

Bhave et al [17] depicts an architectural approach to thinking about relations between heterogeneous system models. They proposed an architectural view framework to bind system models from different domains. The run-time base architecture of the system is utilized as a binding together portrayal to compare the structure and semantics of the related models. Their focuses are on guaranteeing consistent connections between different models which are an imperative part of an incorporated design procedure.Their approach is generalized to address CPS.

Bertran et al [18] introduced, described, and developed a tool suite dedicated to the development of SCC (Sense/-Compute/Control) applications, called DiaSuite. They have

developed a SCC design language, in which a compiler produces customized support for each development stage: implementation, testing, and deployment. DiaSuite has been used to develop applications in a broad range of domains like: home/building automation, avionics, telecommunications, software monitoring, and robotics.CAPS is specific for SiA-CPS and use the graphical tool instead of coding. CAPS provides 3D modeling for the physical environment ,and supports the implementation of new plug-in files to develop new features.

IX. Conclusions and Future Work

In this paper we proposed an architecture description and associated modeling platform for the model-driven engineering of Situational Aware Cyber-Physical Systems (SiA-CPS).

SiA-CPS deals with specific concerns, such as space and time. For this reason, we proposed an architecture description comprising three main viewpoints, and 3+2 views. Software, hardware, and space are three fundamental concepts in SiA-CPS engineering, and this paper focuses on those three. The modeling framework is outlined with its use on the SCUNA case study.

While this is an initial effort, our plan for future work is quite rich. First, we want to complete the modeling tool so to be able ourselves to use it in our SiA-CPS development. This will allow us to apply the CAPS approach to the smart city and cultural heritage SiA-CPS systems we are currently developing. Then, we are currently working on model-driven analysis with a specific focus on energy consumption and data traffic management.

The ultimate goal for CAPS would be to allow stakeholders to perform analysis for effective architectural decision making in earlier stages of the SiA-CPS development life cycle.

ACKNOWLEDGMENTS

The authors would like to thank Ivano Malavolta who contributed to a previous version of the framework, and the reviewers for their valuable comments.

References

[1] E. Woods, "Software architecture in a changing world," *IEEE Softw.*, vol. 33, no. 6, pp. 94–97, Nov. 2016. [Online]. Available: https://doi.org/10.1109/MS.2016.149

[2] S. Jajodia, P. Liu, V. Swarup, and C. Wang, *Cyber situational awareness*. Springer, 2010, vol. 14.

[3] "Train engineers loss of situational awareness led to amtrak derailment; ntsb says technology could have prevented fatal accident," NTSB Press Release National Transportation Safety Board Office of Public Affairs organization.

[4] V. Menon, B. Jayaraman, and V. Govindaraju, "The three rs of cyberphysical spaces," *Computer*, vol. 44, no. 9, pp. 73–79, Sept 2011.

[5] I. Malavolta and H. Muccini, "A survey on the specification of the physical environment of wireless sensor networks," in *2014 40th EUROMICRO Conference on Software Engineering and Advanced Applications*, Aug 2014, pp. 245–253.

[6] C. Tsigkanos, T. Kehrer, and C. Ghezzi, "Architecting dynamic cyber-physical spaces," *Computing*, vol. 98, no. 10, pp. 1011–1040, 2016. [Online]. Available: http://dx.doi.org/10.1007/s00607-016-0509-6

[7] ISO/IEC/IEEE, "ISO/IEC/IEEE 42010:2011 Systems and software engineering – Architecture description," 2011.

[8] M. R. Endsley, "Toward a theory of situation awareness in dynamic systems," *Human Factors*, vol. 37, no. 1, pp. 32–64, 1995. [Online]. Available: http://dx.doi.org/10.1518/001872095779049543

[9] F. P. Osinga, *Science, Strategy and War: The Strategic Theory of John Boyd*. Routledge.

[10] H. Muccini, M. Sharaf, and D. Weyns, "Self-adaptation for cyber-physical systems: a systematic literature review," in *Proceedings of the 11th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*. ACM, 2016, pp. 75–81.

[11] I. Crnkovic, I. Malavolta, H. Muccini, and M. Sharaf, "On the use of component-based principles and practices for architecting cyber-physical systems," in *2016 19th International ACM SIGSOFT Symposium on Component-Based Software Engineering (CBSE)*, April 2016, pp. 23–32.

[12] I. Malavolta, H. Muccini, and M. Sharaf, "A preliminary study on architecting cyber-physical systems," in *Proceedings of the 2015 European Conference on Software Architecture Workshops*. ACM, 2015, p. 20.

[13] C. Szyperski, *Component Software. Beyond Object Oriented Programming*. Addison Wesley, 1998.

[14] D.-M. Han and J.-H. Lim, "Smart home energy management system using ieee 802.15.4 and zigbee," *Consumer Electronics, IEEE Transactions on*, vol. 56, no. 3, pp. 1403 –1410, aug. 2010.

[15] H. SWEETHOME-SWEET, "3d (2015)," *Sweet Home 3D*.

[16] K. Doddapaneni, E. Ever, O. Gemikonakli, I. Malavolta, L. Mostarda, and H. Muccini, "A model-driven engineering framework for architecting and analysing wireless sensor networks," in *Third International Workshop on Software Engineering for Sensor Network Applications, SESENA 2012*, pp. 1–7. [Online]. Available: http://dx.doi.org/10.1109/SESENA.2012.6225729

[17] A. Bhave, B. Krogh, D. Garlan, and B. Schmerl, "Multi-domain modeling of cyber-physical systems using architectural views," 2010.

[18] B. Bertran, J. Bruneau, D. Cassou, N. Loriant, E. Balland, and C. Consel, "Diasuite: A tool suite to develop sense/compute/-control applications," *Science of Computer Programming*, vol. 79, pp. 39–51, 2014.