A Model-based Approach for Managing Criticality Requirements in e-Health IoT Systems

Ch. Kotronis, M. Nikolaidou, G. Dimitrakopoulos, D. Anagnostopoulos
Department of Informatics & Telematics,
Harokopio University of Athens, Athens, GREECE.
{kotronis, mara, gdimitra, dimosthe}@hua.gr

A. Amira, F. Bensaali
College of Engineering,
Qatar University, Doha, QATAR
{abbes.amira, f.bensaali}@qu.edu.qa

***Abstract*—Internet-based solutions, enhanced by Internet of Things (IoT) and Cloud computing, are constantly driving revolutionary approaches in multiple domains, including healthcare. Indicatively, telemedicine, real-time diagnosis and remote monitoring of patients, are expected to transform the healthcare domain. These systems may offer several services of different criticality, necessitating safety-/mission-critical core components and non-critical peripheral components; in other words, they are complex, mixed-criticality System-of-Systems (SoS). To understand and design such systems, engineers must be facilitated with the appropriate modeling tools.**

In this work, we explore the application of model-based design, using the Systems Modeling Language (SysML), of IoT e-Health systems, emphasizing criticality requirements. We focus on the Remote Elderly Monitoring System (REMS) use case, combining IoT technologies with classic healthcare practices, to demonstrate the potential of the proposed approach. Requirements comprise a basic concept of a systematic model-driven methodology that enables the successful management of the criticalities in system design, implementation and deployment. In the REMS use case, identified criticalities are modeled as SysML requirements, while SysML constraints and parametric diagrams are employed to describe and verify quantitative criticality requirements.

***Index Terms*—Internet of Things, Healthcare, Model-Driven Engineering, SysML, Remote Elderly Monitoring, Criticalities, Requirements**

I. Introduction

The advent of the Internet of Things (IoT), the rapid expansion of the Cloud and on-demand computation and storage as well as the proliferation of sensors and ubiquitous wireless communication, are expected to drive revolutionary approaches in healthcare activities, such as: real-time diagnosis of medical issues, telecare and telemedicine as well as remote monitoring of patients.

Such approaches can be implemented via new types of System-of-Systems (SoS), which are mainly composed of hardware (e.g., sensors, smartphones) and software (e.g., specialized operating systems, Cloud services, etc) that can execute several applications of different criticality [1]. These systems can be modeled, studied and analyzed in the context of model-based engineering, using well-defined modeling languages such as the Systems Modeling Language (SysML) [2]. A model-based methodology can assist the engineer to define and evaluate the system components, the interconnection of these components, as well as the requirements that need to be satisfied. Furthermore, it is important to identify and understand the criticality restrictions of an IoT system and its subsystems.

Healthcare is traditionally a domain requiring mostly safety-critical core systems and functionalities, since human lives can be jeopardized by a faulty implementation. Moreover, the complex Information and Communication Technology (ICT) involved may also include mission-critical [3] components, whose failure may put organizations (such as hospitals) at great risk, or non-critical peripheral components. Managing the criticality of a specific IoT component, application or service in complex smart healthcare systems and the way it might affect others, is thus of significant importance for the effective implementation and support of such systems.

In [4], the identification of criticalities in healthcare IoT systems was explored, as a first step to effectively manage them in system implementation and deployment. In that paper, we recognized fundamental mixed-criticality characteristics of such systems, through two principal use cases, namely the (i) Remote Elderly Monitoring System (REMS) and the (ii) Smart Ambulance System (SAS).

As such systems can be treated as SoS, in this work, we explore the potential of model-based design of healthcare IoT systems using SysML. As a first step, we focus on REMS as a case study. REMS provides remote monitoring and diagnosis for the sensitive demographic of elderly subjects, dealing with the real-time diagnosis of medical incidents. Its architecture can be conceived as having a self-adapting structure, since its subsystems (described in Section III) can self-organize, based on specific requirements. Thus, the REMS, as a SoS, allows operational and managerial independence of each subsystem, evolutionary development, emergent behavior and geographic distribution [5]. Model-based design of such healthcare IoT systems should not only explore their structure as a hierarchy of system components, that may be effectively supported by the SysML, but rather enable the description and management

of criticalities, crucial for their operation. These criticalities, as for example the privacy of the exchanged data, are associated with IoT components (e.g., sensor devices) and should be resolved in different levels of abstraction.

To this end, we extended SysML in order to:

i model the REMS infrastructure (Block Definition Diagram (BDD) - Section III-A), and the interconnection (over which data flow) of its subsystems and components (Internal Block Diagram (IBD) - Section III-B),

ii represent and describe the REMS criticalities, identified in our previous work, as SysML requirements (Section IV),

iii extend the system design by employing operational quantitative requirements, which govern SysML parametric execution (Parametric Diagrams - Section V-B).

The rest of the paper is structured as follows: In section II related work is briefly discussed. REMS design using SysML, as a component hierarchy, is presented in Section III. The adoption and extension of SysML requirements to effectively model REMS criticalities is introduced in Section IV. The Home component of REMS is emphasized as an example in Section V, where the proposed concepts are used to describe and verify criticalities as SysML requirements. Conclusions reside in Section VI.

II. Related Work

A review of the relevant literature reveals the ongoing and increased interest in the IoT and IoT-based technologies and solutions. In [6], an overview of the IoT is presented.

Healthcare is becoming one of the most attractive applications fields, where the IoT can offer improved access to care, increased quality and efficiency and reduced costs. Ambient Assisted Living (AAL) systems have the potential to meet healthcare challenges, exploiting ICT and IoT [7]. Such systems comprise medical sensors, wireless networks and software applications for healthcare monitoring [8], helping an individual remain independent from medical facilities, using ICT [9]. As the technology for collecting, analyzing and transmitting data in the IoT continues to grow and evolve, more IoT-driven healthcare applications, services and systems emerge [10]. In [11], the need of an integration of IoT technologies (e.g., RFID [12] or wearable devices) and e-Health solutions is addressed. Focus is given on an integrated system for the continuous monitoring of students at risk to high blood pressure as well as a quick treatment and consultation from medical experts from a distance. Islam et al. [13] analyze a variety of more medical IoT applications, such as remote health monitoring, fitness programs or elderly care. The aforementioned applications were combined with architectures and platforms, like the Internet of Things Healthcare Network (IoThNet).

Relevant to the notion of remote monitoring and care, few approaches propose the use of a mixture of embedded sensors as diagnostic tools, Cloud-based architectures and data analytics that would improve the quality of life of a patient [14], [15]. In addition, the authors in [16] experimented with portable devices and different communications protocols and models for the creation of e-Health applications, like the CardioNet. Moreover, in [17], a tested real-time monitoring platform that uses IoT gateways and medical devices, introduces the need of *Edge Computing* and shows the effectiveness of IoT in real-time e-Health services. In a similar context, the exploitation of *Fog Computing* in healthcare IoT systems is proposed in [18], implementing a Smart e-Health Gateway (UT-GATE) suitable for the deployment of health monitoring systems especially in clinical environments.

As the elderly have become one of the main target groups that need e-Health applications and solutions, there are frameworks that focus on making healthcare technologies more accessible to them. The Home Health Hub Internet of Things (H3IoT) is presented in [19] as a novel architectural framework for elderly monitoring. Furthermore, an IoT-based remote elderly monitoring study [20] introduces and analyzes an architecture where vital signs of elderly individuals are collected via sensors and monitored in real-time by a remote facility (e.g., a hospital).

The aforementioned systems were proposed and designed based on custom empirical methodologies, specific to the healthcare context. The need for model-driven methodologies for healthcare services and applications is advocated by Eldabi et al. [21]; this is exactly the need that this work covers.

We approach the problem of the analysis of healthcare systems via a general, formal model-driven methodology, based on the SysML standardized modeling language. SysML is an extension of the Unified Modeling Language (UML) which is widely known and used, both in academia and industry. We use SysML concepts and adjust them to the specific healthcare use case [22].

Beyond the healthcare domain, in [23], manufacturing systems are modeled in UML and SysML as IoT configurations of their mechatronic components. The IoT is important in the development process of manufacturing systems and IoT technologies integrated with modeling techniques required for the specification of the systems' components, provide an effective approach for the automation of the development process.

In general, using a UML/SysML-based approach, systems engineers are able to effectively represent, specify and describe the technical aspects and structure of systems as well as their requirements. However, it might be necessary to extend SysML entities, such as requirements, to effectively depict specific system properties [24], in this case Healthcare IoT.

III. Using SysML to model REMS as a SoS

Our proposed system, namely the REMS, falls within the AAL domain, and acts as a platform, enabling the remote real-time monitoring of physiological signs and health parameters (e.g., heart rate) of an elderly, from a healthcare personnel (e.g., doctor(s)) located at a remote facility (e.g., hospital) [25]. Therefore, the REMS infrastructure comprises the following subsystems: (i) the *Home*, i.e. the place where the elderly patient resides, (ii) the *Data Repository*, where the collected

data is stored, processed and analyzed, and (iii) the *Remote Facility*, where the healthcare personnel is located.

In the following subsections we apply a model-driven approach, using the SysML, in order to describe the structure of the REMS with respect to its individual components (BDD - Section III-A) and their interconnection (IBD - Section III-B).

A. REMS Individual Components

The BDD, on which we focus in this Section, illustrates the system hierarchy and component classifications. The basic structural element of the BDD and the fundamental modular unit in SysML for describing a system is the *block* [2]. Blocks can be associated with each other, or be generalized; such block relationships are also illustrated in a BDD.

A high-level approach to create BDDs is the following: (i) identify all systems, subsystems or components of the SoS under consideration and represent them as blocks, (ii) annotate the aforementioned blocks with relevant properties and operations, and (iii) draw *association* or *generalization* relationships between the blocks.

Specifically for the REMS, we developed a BDD (see Fig. 1), consisting of: the *REMS*, the *Home Subsystem*, the *Data Repository Subsystem*, the *Remote Facility Subsystem*, the *Sensor*, the *Gateway*, the *Cloud Database*, the *Medical Database*, the *Healthcare Monitoring System* and the *Ambulance Dispatch System* blocks. Each of those REMS-related subsystems/components will be described below in detail, via a top-down explanation.

The REMS is located at the top of the system hierarchy and connects via *composite associations* with other blocks. This conveys that the REMS comprises the Home, the Data Repository and the Remote Facility subsystems.

Each subsystem block contains specific *parts*, *value properties*, *operations* and *flow-ports* that are shown in different compartments of the block. Here, only the parts compartment, containing all relevant parts which compose a block, and the flow-port compartment, containing the interaction points of the blocks where specific data can flow in or out, are shown.

At the Home, medical data from the elderly patient, stemming from Sensors, is electronically streamed/transmitted via secure channels to a Gateway, and then to a remote server for further analysis. Thus, the Home Subsystem is connected via two composite associations with the Sensor and the Gateway blocks, respectively. The potential multiplicity of instances of system parts (e.g., the Home Subsystem may contain numerous Sensors and Gateways) is depicted accordingly on the composite associations using *multiplicity factors*.

The Data Repository is used for the storage of the data, transmitted from the Home, as well as their processing and analysis. The parts of the Data Repository are the Cloud Database and the Medical Database. Note that data can be stored and processed directly on the Medical Database(s); in fact, the *0..** multiplicity factor for the Cloud Database indicates that a Cloud-assisted solution is optional. In general, the REMS Data Repository can be one of these databases or a combination of them. If the latter solution is chosen, patients' Sensor-collected data that are stored in the Cloud and patients' medical information that already exists in the Medical Database, can be integrated and combined into a comprehensive patient data folder. Moreover, stored data can be transmitted and shown to end-users, such as the healthcare personnel, at a Remote Facility.

In the Remote Facility Subsystem, patients' vital signs and general data are monitored and remain under medical supervision by the healthcare personnel. In case of an emergency, actions like: communication between doctor and patient while the patient is at home, dispatching an ambulance, support and medical advice to the patient, e.g., “increase dosage of a medicine”, consultation and care, are taken from the personnel's side to help the elderly at Home. The Remote Facility contains the Healthcare Monitoring System and the Ambulance Dispatch System parts.

Having described the subsystems of the REMS, we delve one layer below into the specific parts that form each subsystem. The Sensor and Gateway blocks compose the Home. The main functions of the Sensor is to monitor, record, collect and push measured values and data to the Gateway. In Figure 1, the generalization relationship between the Sensor and the *ECG* block conveys that an Electrocardiogram (ECG) is a Sensor [26].

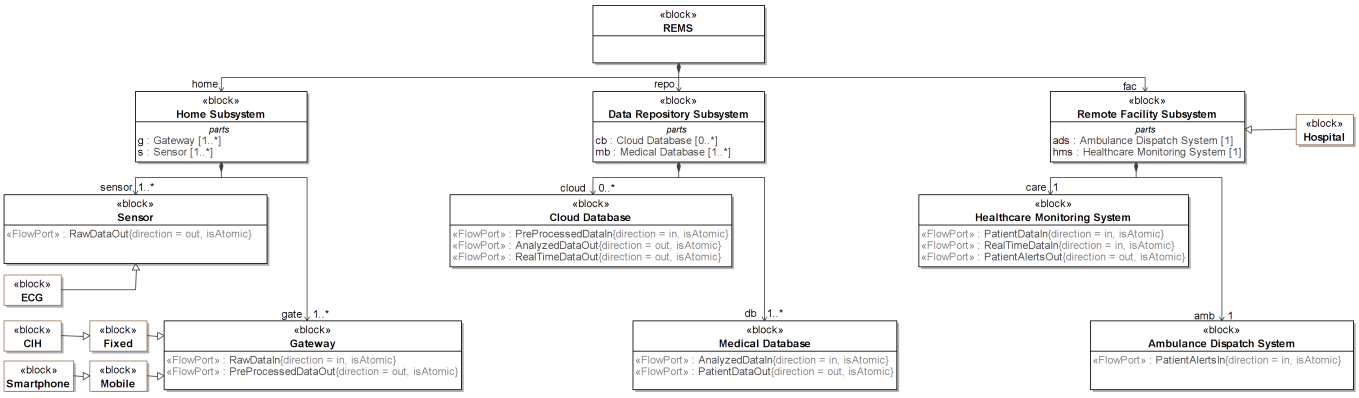

Fig. 1. Modeling REMS structure as a SysML BDD

The Gateway operates as a *Central Information Hub (CIH)* or *Central Home Hub*, communicating with the Sensor(s) and receiving data. Moreover, its functionalities may include the secure delivery of these data and their pre-analysis (e.g., a primitive categorization of data types or their compression) just before their transmission to the Data Repository. The generalizations that are used here create a classification tree that shows that the *Smartphone* and CIH are mobile and fixed types of Gateways, respectively.

The structural decomposition of the Data Repository results in the Cloud and the Medical Database blocks. The Cloud Database will receive, process, analyze and host the Sensor-generated data, while the Medical Database will host the patient medical folder. In real-time, the Medical Database will sync with the Cloud and update the patient's folder with the fetched data and inferred characteristics/alarms. These data will be already processed (e.g., classified and analyzed by automated sub-routines) before they are integrated and displayed, so that the healthcare personnel can extract meaning out of the dataset quickly. Processing and analyzing methods include reasoning, human-assisted classification and inference, machine learning algorithms or pattern recognition [27].

Finally, a *Hospital* can be a Remote Facility, based on the generalization relationship with the Remote Facility block. In the following, the parts that compose it are briefly described.

The Healthcare Monitoring System receives all the data (patients' vital signs) from the integration of the Cloud Data Repository and the Medical Database, and shows them to the healthcare personnel in a clear, understandable format. Moreover, it is designed to visualize the alarms and urgently brief the doctors in case of emergency.

The Ambulance Dispatch System operates in the REMS as the emergency service for immediate and out-of-premises patient care. It can provide fast and accurate medical dispatching, proper care and treatment to a patient, increasing the patient's chances of survival when in critical condition. We note that the treatment of the patient on the ambulance until the arrival at the Remote Facility is out of the scope of this work; this system deals only with the dispatching phase.

Both the Monitoring System and the Ambulance System receive an alert *signal* in case of a patient emergency.

B. REMS Components Interaction

The IBD describes the internal structure of a system's block in terms of its properties, parts, flow-ports and connectors. For a detailed description and specification of these terms and SysML in general, see [2].

The REMS IBD, shown in Figure 2, illustrates how each of the subsystems is structured as well as how these and their parts are connected with each other. Having described the basic functionality of each block in the BDD (see Section III-A), here we focus more on *how the data flows* between the REMS internal blocks using the flow-ports. Note that while we go into more detail in the text on how the connections are practically formed, the diagram is on purpose kept simple in order to illustrate *what* and over *which ports* is exchanged, irrespectively of the particular technology used to implement the corresponding communication channel.

Starting from the Sensor, the *RawDataOut* flow-port is used for pushing the patient's vital signs in raw format to the Gateway. The Sensor transmits these data using fast, low-power and short-range wireless communication protocols, like Bluetooth Low Energy (BLE) [28] or ZigBee [29]. After reception, the Gateway transmits pre-processed data to the Data Repository via WiFi; in practice a secure communication channel (e.g., based on IPSec [30]) from the Gateway to the Data Repository is formed over the Internet, independently from the underlying transmission technologies employed (WiFi, Digital Subscriber Line (DSL), optical cables, etc.).

The Cloud Database of the Data Repository sends the data to the Medical Database for permanent storage and integration with other –relevant– data for a complete medical patient folder. *AnalyzedDataOut* and *AnalyzedDataIn* flow-ports are used for the transmission and reception of the data between the Cloud Database and the Medical Database. Moreover, the analyzed data can be seen directly from the Cloud, in real-time, by the healthcare personnel at the Remote Facility. In detail, the integration of the Cloud-based storage system and the on-site Medical Database of the Remote Facility is an important aspect of the REMS. The Cloud will host the massive analyzed data, while the Medical Database will host the sensitive per-patient medical folder as described above. In real-time, the

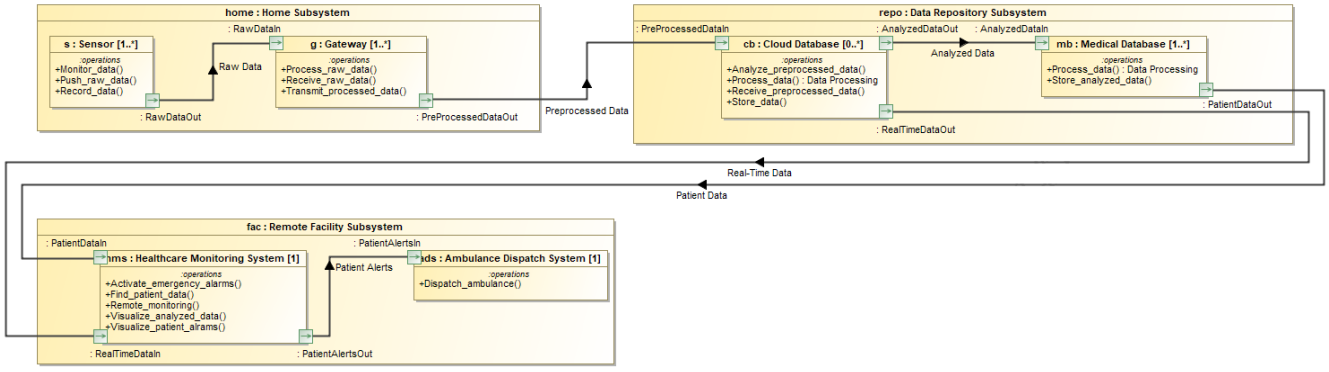

Fig. 2. Modeling REMS component interaction as a SysML IBD

combination of analyzed data from the Cloud, as well as the patient's folder and inferred characteristics/alarms, extracted from the Medical Database, is transmitted to the Healthcare Monitoring System and shown to the healthcare personnel. The healthcare personnel will have direct access both to the Cloud and the Medical Database via administrative accounts; data access can occur via a Smartphone, a tablet or a computer.

The final component interactions take place at the Remote Facility where the Healthcare Monitoring System is connected with the Ambulance Dispatch System, sending potential patient alerts to immediately dispatch an ambulance to the elderly's home in case of an emergency; the emergency is inferred via the aforementioned analysis of the received data and the corresponding alarms that the databases generate.

IV. Modeling Criticalities as SysML Requirements

In our previous work [4], three criticality categories were identified for Healthcare IoT systems, i.e. *safety-critical*, important for the human life, *mission-critical* [3], essential to business operations or organizations, and *non-critical*.

According to Object Management Group (OMG) [31], "a requirement specifies a capability or condition that must (or should) be satisfied, or a function that a system must perform or a performance condition a system must achieve". Thus, criticalities may be modeled utilizing the SysML requirement concept.

The description of the SysML requirement is restricted by two properties: a unique *id* and a textual-formed self-*description*. A requirement is assigned to a model element through the *satisfy* relationship, while the *verify* relationship defines how the element verifies the requirement [2]. Since requirement description is restricted by two properties, the *refine* relationship, associating requirements to other model elements, may contribute to their analytical description.

Requirements can be extended through the *stereotype* concept [32]. Stereotypes allow the definition of new types of SysML model elements, derived from existing ones, that have additional properties, suitable for the examined system [33]. The stereotypes also enable the addition of *constraints*, restricting the types of model elements the corresponding element may be related to [33] (e.g., an operational requirement may be constrained so that it can only be satisfied by a specific SysML block).

In the case where requirements reflect operational or performance constraints (e.g., "energy consumption $<$ 5 watt-hour"), they may be refined by the *constraint*. When this applies, the requirement should be verified by an expression resulting in either "true" or "false", analytically described in a Parametric Diagram (PD) [34]. The PD is used to integrate the design model (i.e. a BDD) with engineering analysis models (e.g., calculation of a device's energy consumption), including the usage of *constraint blocks* to specify constraints representing mathematical or logical expressions [35], constraining the properties of other blocks [36]. Moreover, this specific diagram is suitable for specifying assertions about valid system values in an operational system.

Table I illustrates how the identified criticalities [4] can be modeled using the SysML and its requirements. To explore such potential, we focus on the Home Subsystem of the REMS.

TABLE I
Home Subsystem criticality requirements

Stereotype	SysML entity	Constraint
Safety-critical	Requirement	Satisfied by any block.
Time Req.	Safety-critical	Satisfied by Home block. Satisfied by Sensor block. Satisfied by Gateway block. When satisfied by Sensor or Gateway block, refined by a constraint, verified by a corresponding Parametric Diagram.
Mission-critical	Requirement	Satisfied by any block.
Security Req.	Mission-critical	Satisfied by Home block. Satisfied by Gateway block.
Non-critical	Requirement	Satisfied by any block.
EnergyConsumption Req.	Non-critical	Satisfied by Home block. Satisfied by Sensor block. When satisfied by Sensor block, refined by a constraint, verified by a corresponding Parametric Diagram.
SWaP Req.	Non-critical	Satisfied by Sensor block. When satisfied by Sensor block, refined by a constraint, verified by a corresponding Parametric Diagram.

The *Time Req*, *Security Req*, *EnergyConsumption Req*, and *SWaP Req* are custom SysML stereotypes, defining basic Home Subsystem requirements-criticalities.

Time is a safety criticality for both Sensors and Gateways, and the Home in general. The measurements and recordings have to be generated and transmitted within a given time interval, otherwise the real-time behavior of the system is jeopardized and considered faulty.

The *Security* is mission-critical as it may affect the credibility of the Home Subsystem. There is a need of communications between devices (Sensors to Gateway, etc.) that ensure the confidentiality and integrity of transmitted data without any fault or modification by an adversary.

The *Energy Consumption* of all devices can be considered as non-critical, and must remain low. Although, the problem with low energy consumption protocols, like BLE, is that the security criticality might be affected in a negative way [37].

The *Size, Weight and Power (SWaP)* [38] is also a non-criticality that helps ensure that devices are easier to carry and have larger autonomy.

The aforementioned stereotypes are applied to their respective Safety-, Mission-, or Non-critical SysML *Requirement*.

The constraints column depicts the restrictions related to the requirement stereotypes. For example, the Safety-critical Time Req stereotype can only be satisfied by the Home block, as well as its parts, i.e. a Sensor or a Gateway. Due to the fact that it is an operational quantitative requirement, when the Time Req is satisfied by the Sensor or the Gateway, it is refined by a constraint, and thus, verified by a corresponding PD. Other, non-operational requirement stereotypes, e.g., the Security Req, are constrained to only be satisfied by specific blocks, e.g., Home and Gateway, without any PD presence.

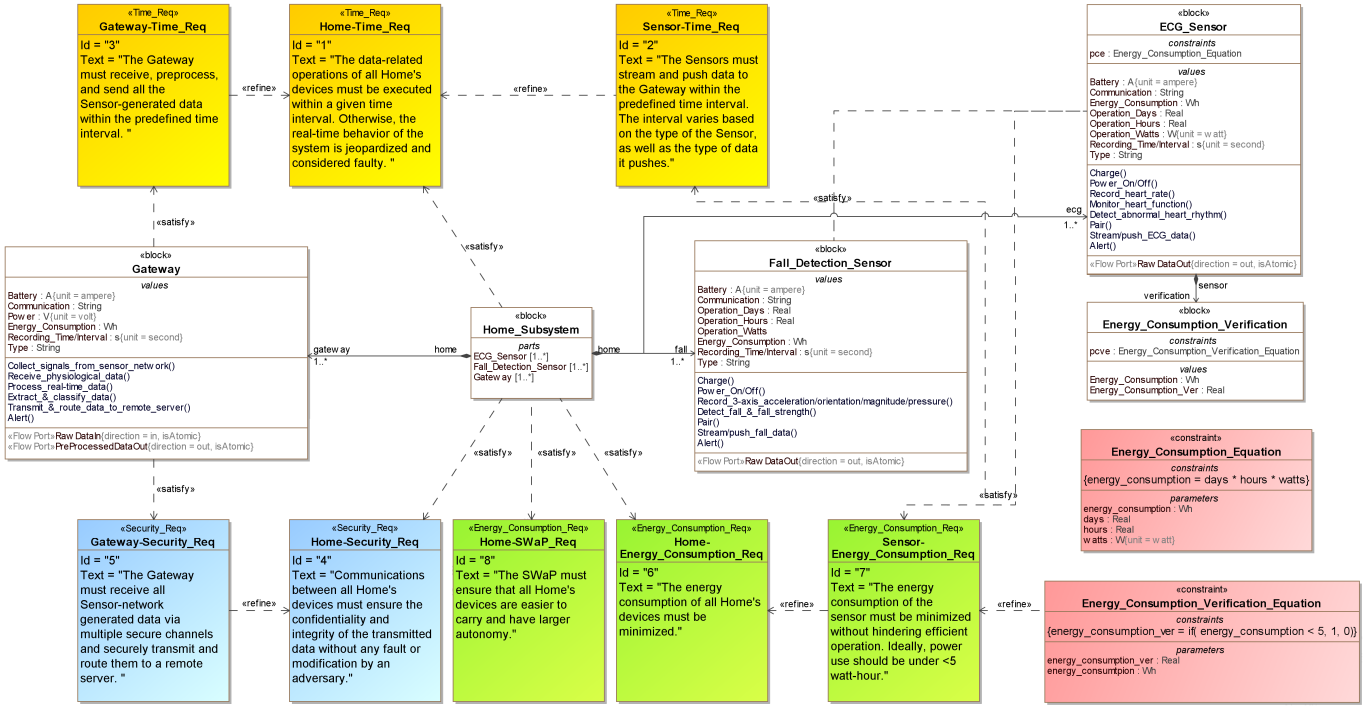

Fig. 3. Modeling REMS Home Subsystem components and criticality requirements using SysML

V. Modeling Home Subsystem using SysML

A. Home Block Definition Diagram

Applying the proposed SysML criticality requirements to the REMS, we aim to achieve the following systems engineering activities: structure design, requirements definition and performance analysis based on respective constraints. Figure 3 illustrates the assignment of criticality requirements to REMS components –white-colored–.

As a case study, we focused on the Home Subsystem, containing two basic Sensors, i.e. an *ECG* and a *Fall Detection* Sensor, as well as a central Gateway. These components are connected with the Home via *directed compositions* and, as parts of a larger system –Home–, are shown at the Home Subsystem block's parts compartment, along with their corresponding multiplicity factor (e.g., one or more Sensors).

The *ECG_Sensor*, the *Fall_Detection_Sensor*, and the *Gateway* blocks contain specific value properties, operations and flow-ports. The value properties specify the quantitative properties of their containing block. For example, the ECG_Sensor holds its type, battery capacity, energy consumption and other operational parameters that describe it.

The operations compartment includes all the functions of the respective entity, e.g., the Fall_Detection_Sensor powers on/off, records acceleration or orientation of the patient, pairs/-connects with the Gateway and pushes measured (fall) data.

The ECG_Sensor block contains an extra compartment, namely *constraints*, showing that a property of this block (here, the energy consumption) is calculated parametrically, as described in the following Section (Section V-B).

Each requirement, illustrated at Figure 3, is explicitly marked as one of the four requirement types (i.e., Time –dark yellow-colored–, Security –blue-colored–, Energy Consumption and SWaP – green-colored–), according to the SysML stereotyping mechanism described in Table I.

The Home block satisfies the *Home-Time_Req*, *Home-Security_Req*, *Home-Energy_Consumption_Req*, and *Home-SWaP_Req*. Home's components, namely the Sensors and the Gateway, are satisfying more specialized requirements.

Both Sensors are connected via satisfy relationships to the *Sensor-Time_Req* requirement that constrains the Sensors' operational time. In addition, the operation of a Sensor should be adjusted, minimizing the energy consumption without hindering efficient operations. Therefore, the Sensors must also satisfy the *Sensor-Energy_Consumption_Req*. In particular, this requirement states that the Sensor' s energy must be under 5 watt-hour for a continuous efficient operation. The *Energy_Consumption_Verification* block belonging to the ECG_Sensor is used to check a verification value property, indicating “true” or “false” (“1” or “0”), against the energy consumption requirement. For the *Sensor-Energy_Consumption_Req*, we have also created a constraint block, described in detail in the following Section (Section V-B).

The Gateway component satisfies a *Gateway-Time_Req* and a *Gateway-Security_Req* requirement. The latter is used to highlight the security levels that the Gateway's operations must conform to.

Note that the aforementioned components' requirements refine the upper level Home requirements, e.g., the Sensor-, and

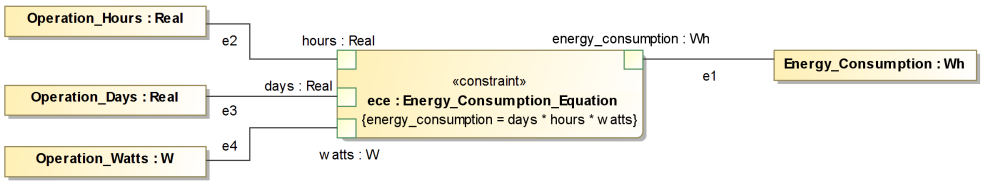

Fig. 4. Sensor block equation as a SysML Parametric Diagram

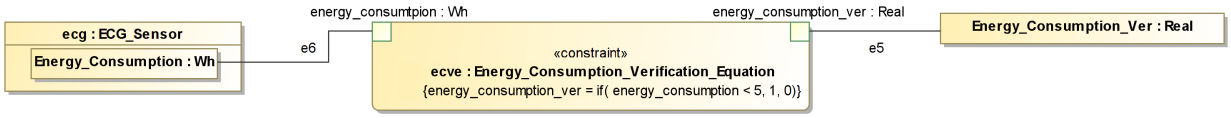

Fig. 5. Verification block expression as a SysML Parametric Diagram

Gateway-Time_Req requirements refine the Home-Time_Req by placing it in their respective specialized context. In addition, the Gateway-Security_Req is connected with the Home-Security_Req, while the Sensor-Energy_Consumption_Req refines the Home-Energy_Consumption_Req requirement.

The logic behind the illustration is similar for all the REMS subsystems; the Home and its components are only shown here for brevity.

Using SysML, a consistent component-requirement model can be defined, forming the examined system's infrastructure. Modeling the hierarchy of the components and the satisfied and refined requirements as well as explicitly mapping the relationships between them, helps in dealing with the system's complexity. The advantage is that in large, complex systems, having a hierarchy of components and requirements and organizing them into various levels, the complexity of systems is managed from the early beginning of their development.

B. Parametric Diagrams

Parametric execution added to SysML design models can help evaluate the performance (and other parameters) of a system design as well as verify its requirements.

The constraint block comprises two compartments, i.e. constraints and *parameters* [39]. The constraints compartment contains an equation, expression or rule that combines the parameters from the parameters compartment. *Constraint properties* –specifying the constraints of other properties in their containing block–, constraint parameters, as well as block value properties are displayed inside a PD. The constraint parameters provide connection points, connected via *binding connectors*, to specific block properties or to other constraint parameters on the same or other constraint properties.

In Figure 3, we have created two constraint blocks –red-colored–; the *Energy_Consumption_Equation* is suitable for the calculation of the energy consumption of a Sensor (here, the ECG Sensor), while the *Energy_Consumption_Verification_Equation* is used for expressing the Sensor-Energy_Consumption_Req textual requirement as a conditional mathematical expression.

The Energy_Consumption_Equation contains a simple energy consumption equation:

$$energy\_consumption = days * hours * watts \quad (1)$$

based on its parameters.

The PD illustrated in Figure 4, was created inside the ECG_Sensor block. This block's value properties, i.e. *Operation_Hours*, *Operation_Days*, *Operation_Watts*, and *Energy_Consumption* are connected to corresponding constraint parameters, i.e. *hours*, *days*, *watts*, and *energy_consumption* of the constraint property. Note that a block property and the respective constrained parameter must have the exact same *value type* (e.g., “Wh” for watt-hour).

Figure 5 shows the PD, created inside the Energy_Consumption_Verification block, in order to evaluate the energy consumption against the desired energy usage, returning “true” or “false”, “1” or “0”. The defined expression is the following:

$$if(energy\_consumption < 5, 1, 0) \quad (2)$$

In this PD, the constraint relationship is complex, since one of the parameters in the relationship is tied to a value property owned directly by an other block. For example, parameter *energy_consumption* is tied to value property Energy_Consumption, owned by the ECG_Sensor block (i.e. ECG_Sensor.Energy_Consumption). The *energy_consumption_ver* is related to the *Energy_Consumption_Ver* property of the verification block.

C. SysML Tools

The BDDs, illustrating structural information of the examined system, the IBD, representing the interconnections between the system's components, and the PDs, showing the mathematical relationships among the elements of the system, were developed in the MagicDraw UML tool [40]. In particular, the ParaMagic plugin [41], imported in MagicDraw, is suitable for the parametric execution, using OpenModelica [42] as math solver.

VI. Conclusions

Our case study, namely the REMS, is a novel IoT-based SoS for the remote monitoring of elderly subjects; this system belongs in the expanding category of mixed-criticality healthcare systems that are enhanced by ICT technologies. We employed and extended SysML in order to apply a model-based design approach to illustrate the: (i) structure and interconnection of the REMS, in terms of individual subsystems and components, (ii) the diverse criticalities of the REMS Home Subsystem in the form of –manageable– SysML requirements, and (iii) mathematical relationships and validation expressions among the components and operational requirements of the examined system. This is the first concrete step towards formal SysML models for IoT healthcare systems. Based on reasonable system-wide abstractions, it can be quite useful for systems engineers, seeing that it can shed light to the design and management of complex mixed-criticality healthcare systems. This is necessary to understand these systems (via comprehensive models), before implementing and deploying them in the real world. Additional novel healthcare oriented applications can be investigated, through the proposed mixed-criticality model-driven approach.

VII. Acknowledgement

The authors wish to acknowledge the Qatar National Research Fund project EMBIoT (Proj. No. NPRP 9-114-2-055) project, under the auspices of which the work presented in this paper has been carried out.

References

[1] A. Burns and R. Davis, "Mixed criticality systems-a review," *Department of Computer Science, University of York, Tech. Rep*, pp. 1–69, 2013.

[2] S. Friedenthal, A. Moore, and R. Steiner, *A Practical Guide to SysML: The Systems Modeling Language*, 2014.

[3] F. Ciccozzi, I. Crnkovic, D. Di Ruscio, I. Malavolta, P. Pelliccione, and R. Spalazzese, "Model-driven engineering for mission-critical iot systems," *IEEE Software*, vol. 34, no. 1, pp. 46–53, 2017.

[4] C. Kotronis, M. G., G. Dimitrakopoulos, M. Nikolaidou, D. Anagnostopoulos, A. Amira, F. Bensaali, H. Baali, and D. H., "Managing criticalities of e-health iot systems," in *Proceedings of IEEE International Conference on Ubiquitous Wireless Broadband (ICUWB '17)*, 2017.

[5] A. Pyster, D. H. Olwell, N. Hutchison, S. Enck, J. F. Anthony Jr, D. Henry *et al.*, "Guide to the systems engineering body of knowledge (sebok) v. 1.0. 1," *Guide to the Systems Engineering Body of Knowledge (SEBoK)*, 2012.

[6] K. Rose, S. Eldridge, and L. Chapin, "The internet of things: An overview," *The Internet Society (ISOC)*, pp. 1–50, 2015.

[7] A. Dohr, R. Modre-Opsrian, M. Drobics, D. Hayn, and G. Schreier, "The internet of things for ambient assisted living," in *Information Technology: New Generations (ITNG), 2010 Seventh International Conference on*. Ieee, 2010, pp. 804–809.

[8] A. N. Belbachir, M. Drobics, and W. Marschitz, "Ambient assisted living for ageing well–an overview," *e & i Elektrotechnik und Informationstechnik*, vol. 127, no. 7-8, pp. 200–205, 2010.

[9] K. Spitalewsky, J. Rochon, M. Ganzinger, P. Knaup *et al.*, "Potential and requirements of it for ambient assisted living technologies," *Methods of information in medicine*, vol. 52, no. 3, pp. 231–238, 2013.

[10] G. Carnaz and V. B. Nogueira, "An overview of iot and healthcare," 2016.

[11] T. Takpor and A. A. Atayero, "Integrating internet of things and ehealth solutions for students healthcare," vol. 1, 2015.

[12] R. Want, "An introduction to rfid technology," *IEEE pervasive computing*, vol. 5, no. 1, pp. 25–33, 2006.

[13] S. M. R. Islam, D. Kwak, M. H. Kabir, M. Hossain, and K. S. Kwak, "The internet of things for health care: A comprehensive survey," *IEEE Access*, vol. 3, pp. 678–708, 2015.

[14] Z. M. Kalarthi, "A review paper on smart health care system using internet of things," *International Journal of Research in Engineering and Technology*, vol. 05, no. 03, p. 8084, 2016.

[15] M. P. Bharathan, M. V. Nadar, and M. S. Wayal, "Remote health monitoring using iot," *International Journal of Advance Research, Ideas and Innovations in Technology*, 2017.

[16] G. Sebestyen, A. Hangan, S. Oniga, and Z. Gál, "ehealth solutions in the context of internet of things," pp. 1–6, 2014.

[17] H. Moustafa, E. M. Schooler, G. Shen, and S. Kamath, "Remote monitoring and medical devices control in ehealth," pp. 1–8, 2016.

[18] A. M. Rahmani, T. N. Gia, B. Negash, A. Anzanpour, I. Azimi, M. Jiang, and P. Liljeberg, "Exploiting smart e-health gateways at the edge of healthcare internet-of-things: a fog computing approach," *Future Generation Computer Systems*, vol. 78, pp. 641–658, 2018.

[19] P. P. Ray, "Home health hub internet of things (h3iot): An architectural framework for monitoring health of elderly people," pp. 1–3, 2014.

[20] I. Azimi, A. M. Rahmani, P. Liljeberg, and H. Tenhunen, "Internet of things for remote elderly monitoring: a study from user-centered perspective," *Journal of Ambient Intelligence and Humanized Computing*, vol. 8, no. 2, pp. 273–289, 2016.

[21] T. Eldabi, G. T. Jun, J. Clarkson, C. Connell, and J. H. Klein, "Model-driven healthcare: Disconnected practices," in *Simulation Conference (WSC), Proceedings of the 2010 Winter*, 2010, pp. 2271–2282.

[22] A. Tsadimas, M. Nikolaidou, and D. Anagnostopoulos, "Extending sysml to explore non-functional requirements: The case of information system design," in *Proceedings of the 27th Annual ACM Symposium on Applied Computing*, ser. SAC '12, 2012, pp. 1057–1062.

[23] F. C. K. Thramboulidis, "Uml4iot - a uml-based approach to exploit iot in cyber-physical manufacturing systems," *Computers in Industry*, vol. 82, pp. 259–272, 2016.

[24] A. Tsadimas, M. Nikolaidou, and D. Anagnostopoulos, "Handling non-functional requirements in information system architecture design," in *2009 Fourth International Conference on Software Engineering Advances*, Sept 2009, pp. 59–64.

[25] M. Bujnowska-Fedak and U. Grata-Borkowska, "Use of telemedicine-based care for the aging and elderly: promises and pitfalls," *Smart Homecare Technology and TeleHealth*, p. 91, 2015.

[26] E. Nemati, M. J. Deen, and T. Mondal, "A wireless wearable ecg sensor for long-term applications," *IEEE Communications Magazine*, vol. 50, no. 1, 2012.

[27] C. Ji, Y. Li, W. Qiu, U. Awada, and K. Li, "Big data processing in cloud computing environments," in *Pervasive Systems, Algorithms and Networks (ISPAN), 2012 12th International Symposium on*.

[28] C. Gomez, J. Oller, and J. Paradells, "Overview and evaluation of bluetooth low energy: An emerging low-power wireless technology," *Sensors*, vol. 12, no. 9, pp. 11 734–11 753, 2012.

[29] J.-S. Lee, Y.-W. Su, and C.-C. Shen, "A comparative study of wireless protocols: Bluetooth, uwb, zigbee, and wi-fi," in *Industrial Electronics Society, 2007. IECON 2007. 33rd Annual Conference of the IEEE*, 2007.

[30] C. R. Davis, *IPSec: Securing VPNs*. McGraw-Hill Professional, 2001.

[31] S. Friedenthal, A. Moore, and R. Steiner, "Omg systems modeling language (omg sysml) tutorial," in *INCOSE Intl. Symp*, 2006.

[32] R. Guillerm, H. Demmou, and N. Sadou, "Safety evaluation and management of complex systems: A system engineering approach," *Concurrent Engineering*, vol. 20, no. 2, pp. 149–159, 2012.

[33] J. Holt and S. Perry, *SysML for systems engineering*. IET, 2008, vol. 7.

[34] E. Huang, R. Ramamurthy, and L. F. McGinnis, "System and simulation modeling using sysml," in *Proceedings of the 39th conference on Winter simulation: 40 years! The best is yet to come*.

[35] L. Delligatti, *SysML distilled: A brief guide to the systems modeling language*. Addison-Wesley, 2013.

[36] M. Hause *et al.*, "The sysml modelling language," in *Fifteenth European Systems Engineering Conference*, vol. 9. Citeseer, 2006.

[37] M. Ryan, "Bluetooth: With low energy comes low security." *USENIX WOOT*, vol. 13, pp. 4–4, 2013.

[38] I. THALES DEFENSE & SECURITY, "Design considerations for size, weight, and power constrained radios," *2006 Software Defined Radio Technical Conference and Product Exposition*, 2006.

[39] T. A. Johnson, J. M. Jobe, C. J. Paredis, and R. Burkhart, "Modeling continuous system dynamics in sysml," in *ASME 2007 International Mechanical Engineering Congress and Exposition*, 2007, pp. 197–205.

[40] "MagicDraw UML," http://www.magicdraw.com/.

[41] "Paramagic plugin," https://www.nomagic.com/product-addons/magicdraw-addons/paramagic-plugin.

[42] "Openmodelica," https://openmodelica.org/.