A metamodel to guide a requirements elicitation process for embedded systems

Tarcísio Pereira[13], Aêda Sousa[1], Reinaldo Silva[1], Deivson Albuquerque[13] Fernanda Alencar[12], Jaelson Castro[1]
Universidade Federal de Pernambuco[1]
Universidade de Pernambuco[2]
Recife, Brazil
Instituto Federal do Sertão Pernambucano[3]
Petrolina, Brazil
Email: {tcp,amcsb,ras12,mdac,jbc}@cin.ufpe.br, fernanda.ralencar@ufpe.br

***Abstract*—[Context] In the embedded systems (ES) area, more than 50% of problems occur at system delivery and are related to misconceptions in capturing requirements. Therefore, it is necessary to address what should be considered by requirements engineers in the elicitation and specification phases of embedded systems. However, understanding embedded systems and their environment is a strenuous activity. Hence, requirements engineers need to know the core concepts related to ES and also a systematic guide to consider these concepts in the development process. [Goal] This paper presents a metamodel based on a systematic literature review (SLR) that can support the elicitation of ES concepts. Additionally, we used the metamodel as a guide to the development of a requirements elicitation process. [Method] We used the studies of an SLR as a basis for the metamodel development and their concepts as input for an investigation of actions that are suitable to be used in our elicitation process. The proposed metamodel concepts were evaluated by a domain expert, and we applied the actions of the process to elicit requirements of an infusion pump system. [Results] The metamodel has 118 entities that represent the RE concepts for ES, and the elicitation process has 49 actions to guide requirements identification and definition. [Conclusion] The metamodel can be used by practitioners to check if they are eliciting the appropriate information for the development of embedded systems. Moreover, it can assist future research. Besides, the process can help organizations in improving their requirements practices to support completeness and correctness of ES elicitation.**

***Keywords*—embedded systems, requirements engineering, requirements elicitation, process, metamodel.**

I. Introduction

In everyday life, people are dependent on several critical services supported by software, many of them are ubiquitous nature. Examples of such systems include traffic air control, road monitoring, telecommunication system control, medical equipment control, automation technology, and automotive control systems. Broy [1] define Embedded System (ES) as a system that regulates a physical device by sending control signals to actuators in reaction to input signals provided by its users and by sensors capturing the relevant state parameters of the system.

Frequently, an embedded system is a component of a larger system embedded in a product and transparent to the final user [2], [3]. Furthermore, such systems are designed to repeatedly carry out a specific function, keeping its operation under different constraints from the ones of general purpose systems [2], [4]. According to Berger [5], different from a Personal Computer, embedded systems have real-time, and power constraints and they are cost-sensitive. Besides, they often operate under extreme environmental conditions, and software failure causes severe damages.

Embedded systems are known for their high complexity, caused by the increasing number of functions and the growing number of interactions among different functions. Due to their complexity, the risk of undetected errors and deficiencies increases considerably. In the embedded system domain, more than 50% of the problems occur when the system is delivered [1]. However, several problems are not related to the correctness of implementation but due to misconceptions of requirements.

These shortcomings are the result of inappropriate Requirements Engineering (RE) tasks, resulting in incomplete requirements, incorrect elicitation and specification, high complexity, and economic or human loss. However, to understand embedded systems and their environment is a strenuous activity. Hence, requirements engineers need to know the core concepts related to ES and also an elicitation process.

The requirements for software products and systems are gathered, analyzed, documented, and managed through the Requirements Engineering (RE) process, which focuses on the development of processes, methods, techniques, and tools to help in the conception of software and systems. It covers the activities of elicitation, analysis, specification, validation, and management of requirements [6]. Hence, a requirements engineering process is crucial in order to meet time, cost, and quality goals [7].

There are some studies that investigated the concepts that should be considered during Embedded Systems development [8]–[10]. However, these studies did not capture the core concepts and appropriate evidence. Moreover, they did not perform any systematic investigation of the literature. In fact, there is much confusion among requirements engineers and stakeholders due to the different kinds of information that need to be managed. In this context, a well-defined metamodel that captures ES concepts with their interrelations would constitute a significant step forward to improve the requirements quality of embedded systems.

In this paper, we extend our previous work [11]. We have updated and evaluated a Metamodel for Embedded Systems (MM4ES). It was based on an SLR whose aim was to support the elicitation of ES concepts in RE process. Our metamodel consists of 118 entities and relationships among them that should be considered in the development of embedded systems. Given that the metamodel was based on an extensive literature review that addressed concerns from academic and industry context, it may be a means for requirements engineers to improve the completeness and correctness of the embedded system elicitation and specification.

Therefore, we used the metamodel concepts to develop the Requirements Engineering Process for Embedded Systems (REPES). Organizations can use the process as a guide for improving the requirements elicitation for ES by adopting selected actions supported by the process. In this paper, we focus on requirements elicitation actions of the REPES process. As a proof of concept, we use the proposed process to elicit requirements for a Patient Control Analgesia (PCA) pump. Finally, we provide guidelines for further research.

The main contributions achieved by the development of the metamodel are the followings: (i) a knowledge-based metamodel which serves as a resource model for integrating characteristics of the significant concepts appearing in embedded systems; (ii) a reference model that can be used as input for model transformation if model-driven development is intended. It can also be used to develop a domain specific language; (iii) the metamodel can be used as a guide for further research; and (iv) a requirements elicitation process for embedded systems that can contribute to eliminate errors and requirements problems from the beginning.

This paper is structured as follows: The related works are presented in Section II. In Section III we describe the research methodology adopted in this study. In Section IV we present our metamodel. Then in Section V we present the requirements elicitation process and an example of its application. Finally, in Section VI we provide some conclusion and indications for future work.

II. Related work

Requirements engineering for embedded systems is a challenging activity since these systems require the specification of several issues related to software, hardware, environment, and multiple stakeholders [12]. However, due to the complexity of ES, the traditional RE methods are inappropriate [8]. Hence, the specification of ES should be carefully conducted to avoid incorrect requirements.

A metamodel allows the concepts of a particular domain to be structured. It captures abstraction and relationships of the target domain concepts [13]. However, according to our previous work [11], there is no ES metamodel based on an SLR. Besides, there is no study that proposes a requirements process for the embedded systems domain guided by a metamodel. Thus, this paper presents a metamodel whose aim is to support the elicitation of ES concepts. The focus relies on software, hardware, environment, business, and contextual information. Our goal is to define a generic metamodel for the embedded systems domain and to present a requirements elicitation process guided by the proposed metamodel.

Several approaches addressing requirements engineering for embedded systems have already been presented. Among these, the closest to our work are [13]–[15]. Arpinen et al. [13] present a metamodel for requirements management which focuses on software and embedded system domains. Their approach addresses concepts such as stakeholders, product description, functional and non-functional requirements, and risks. As shown in this paper, the context of an embedded system is more than software. In our metamodel, we also consider concepts that are related to business, software, hardware, environment, and context information. By contrast, these concepts were extracted from the results of an extensive literature review.

The work presented by Li et al. [14] is related to ours because it proposes a metamodel for embedded real-time systems requirements variability. They are only interested in the representation of functional, physical, and non-functional variability. They provide a separate view of the system requirements to represent those concepts. However, Li et al. [14] do not specify any of them. Embedded systems engineering is a complex discipline. It includes consideration of domain knowledge, process infrastructure requirements, manufacture, and hardware / software components. In our metamodel, we consider these issues as well as the ones proposed in [14]. Moreover, we specify the concept of 58 non-functional requirements (NFRs) and the concept of Hardware Device in 12 devices. Thus, our metamodel provides a much more information than those presented in [14].

Dubois et al. [15] propose a metamodel for requirements traceability. They have explored the concepts described in [16] and [17] to construct their model which focuses on system requirements. Thus, there is only one requirement class, and this class is decomposed into a set of other requirements such as project requirement, system requirement, and process requirement. The system requirement, in turn, is divided into functional and non-functional requirements, and constraints. Different from the study presented in [15], we have looked for concepts in 84 studies. Besides, our metamodel covers a broader range of requirements concepts than the one described in [15] such as business, environment, design, NFRs, software, and hardware.

Despite the number of studies that have reported requirements engineering approaches for embedded systems domain, no previous study has had such a focus, to systematically develop a metamodel to represent the core RE concepts for ES based on a systematic literature review. Additionally, these studies do not consider the use of a metamodel to guide the development of a requirements process.

III. Research methodology

In this section, we present the process followed by us to develop and provide a standard representation for the design and development of embedded systems. The process was based on our experience on metamodeling [18] and on the method for developing taxonomies proposed by Usman et al. [19]. In order to evaluate the result obtained through the execution of the process, we included an additional phase called *Evaluation*. The process is illustrated in Fig. 1. In the next sections, we present the details of each step of the process.

A. Planning

This work belongs to research whose the objective is to improve the activities of requirements engineering in the embedded systems domain. In this research line, we are working on the development of a well-defined RE process to support practitioners in the development of embedded systems.

In the planning phase, we defined requirements engineering as the software engineering knowledge area for the new metamodel. We also defined its objectives and scope (task 1). The development of the metamodel was driven by the following research question: *"What is the core set of requirements engineering information that should be specified by requirements engineers in the development of embedded systems?"*.

As source data for the metamodel, we initially considered the studies captured by our previous SLR [11]. The studies are from academic and industrial context and also from different domains to get a broad coverage of the state of practice. Additionally, we have also updated the period of time covered by the SLR.

B. Identification and extraction

In this step, we read all articles of our systematic literature review to look for concepts that make part of the domain knowledge of an embedded system (task 2). Then, we structured the domain knowledge in a glossary of terms of the domain vocabulary identified in the previous specification activity (task 3). The majority of the concepts definitions were obtained from the IEEE Standard Glossary of Software Engineering Terminology while the remaining were retrieved from articles, books, and websites[1]. The core concepts of our metamodel are depicted in Fig. 2. These concepts were extracted from the Glossary of Terms[2] developed in Task 3 of the process.

C. Design and construction

In this step, we performed the identification and description of the concepts and relationships in a *relation cardinality document* (RC). It consist of the name of the classes, relationships, and cardinality between them. Table I shows the results of tasks 4 and 5. For example see in Fig. 2, the *EmbeddedSystem* class has a relationship called *hasDomainKnowledge* with the *DomainKnowledge* class; The *Software Requirements* class has a relationship *defines* with the *Software* class.

We used the *relation cardinality document* to implement the metamodel (task 6). Each line of the document was mapped to a UML class with its relationships.

TABLE I
Relation cardinality document excerpt

Class name	Relation	Cardinality	Class name
Embedded System	hasDomainKnowledge	1..1	Domain Knowledge
Embedded System	hasAbstractionLevel	1..0*	Abstraction Level
Embedded System	hasSoftware	1..1*	Software
Embedded System	hasStakeholder	1..1*	Stakeholder
Embedded System	hasContext	0..1	Context
Embedded System	hasHardware	1..0*	Hardware
Embedded System	hasAction	1..0*	Action
Embedded System	hasBusiness	1..1	Business
Embedded System	hasEnvironmentRequirements	1..0*	Environment Requirements
Software Requirements	defines	1*..1*	Software
Functional Requirements	is-a		Software Requirement
Non-Functional Requirements	is-a		Software Requirement
Hardware	hasHardwareRequirements	1..1*	Hardware Requirements
Hardware Requirements	defines	1..1*	Hardware Device
Sensor	is-a		Hardware Device
Sensor	monitors	1..0*	Control Variables
Actuator	is-a		Hardware Device
Actuator	generates	1*..1*	Action
Environment	hasEmbeddedSystem	1..0*	Embedded System

D. Evaluation

In the evaluation phase, we considered two ways to evaluate our metamodel (task 7), both rely on an Infusion Pump example to demonstrate its utility.

The first version of the metamodel is described in our previous work [12]. As a first evaluation, the metamodel was analyzed three times by a domain expert to check if the concepts were correct from his point of view in the context of medical device development.

In the second evaluation, we tried to demonstrate the utility of the metamodel by tracing the results of the elicitation process actions to the embedded systems concepts using the metamodel as a basis. In Section V, we describe the process and also the usage example. Whenever the results are presented, its number is highlighted in bold. Thus, we can check the number in the metamodel to see the concept the results are related.

IV. Metamodel for embedded systems - MM4ES

We hope that the MM4ES will contribute to requirements engineers express their knowledge about the domain and consolidate the information that should be elicited and specified during the beginning of RE. Thus, it may help to meet schedule time, reduce cost, and quality goals.

In order to provide an update state of the art about RE for embedded systems, we updated our SLR (1970 - September 2016) to consider studies from 1970 to March 2018. Thus, was necessary to improve our metamodel performing for the second time the metamodel development process depicted in Fig. 1. The goal was to provide an updated metamodel and identify possible missing concepts and perceived adjustments. It is important to note that our previous SLR presented in [11] had 75 papers, while in this updated study considered 84 papers. All references can be downloaded at https://bit.ly/2ljIGqO.

[1]http:whatis.techtarget.com/glossary

[2]The complete glossary can be downloaded at https://bit.ly/2K5UasW

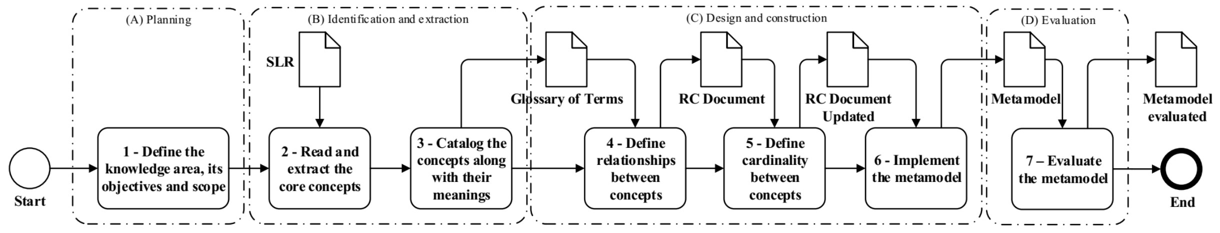
(A) Planning
(B) Identification and extraction
(C) Design and construction
(D) Evaluation
SLR
Glossary of Terms
RC Document
RC Document Updated
Metamodel
Metamodel evaluated
Start
1 - Define the knowledge area, its objectives and scope
2 - Read and extract the core concepts
3 - Catalog the concepts along with their meanings
4 - Define relationships between concepts
5 - Define cardinality between concepts
6 - Implement the metamodel
7 – Evaluate the metamodel
End

Fig. 1. Metamodel development process

In our previous work, the metamodel has 41 concepts, while 118 concepts are presented and discussed in this new study (see Fig. 2, 3, and 4). The green boxes are new concepts, and the blue boxes were updated with a set of attributes. What remains the same from our previous work are the yellow classes.

The main concepts in Fig. 2 related to an *Embedded System* are *Stakeholder*, *Business*, *Domain Knowledge*, *Environment*, *Environmental Requirements*, *Hardware*, *Software*, *Action*, *Context*, and *Risk*. These concepts should be considered during the requirements engineering process for embedded systems development.

Stakeholder is a person, group or organization that has interest or concern in an organization or (system). Requirements engineering for embedded systems is a multi-disciplinary approach. It requires domain experts from several areas, such as mechanical engineers for physical context, electrical engineers for the hardware context, as well as Human-Machine-Interface (HMI) experts for the usability aspects, in addition to requirements engineers and software developers, testers, and maintainers with a computer science background [20]. Thus, *Stakeholders* can play one or multiple *Roles*.

The *Business* defines the *Market Pressure*, *Suppliers Relationships*, and *Migration of Standards*. The *Market Pressure* involves the consumer changes, environmental, and regulatory needs. All of these can impact the requirements engineering for ES since it is necessary to move fast and change requirements to facilitate the agility and flexibility needs of its container [20]. The market is shared by suppliers, manufacturers and tool vendors which need to improve processes exchangeability among them and the reuse of software in different product lines. This issue raises the challenge of the reduction of development time [21]. Associated with *Business*, we have the *Business Rules*, which aims at description of a business policy or procedure. *Business Rules* are usually expressed at the atomic level, i.e., they cannot be broken down any further.

An embedded system requires that the *Domain Knowledge* should be understood. In general, domain is an area of control or a sphere of knowledge. It defines the *Execution Environment* of the system, *Certification Process*, *Process Infrastructure*, *Product Infrastructure* and *Manufacture*.

Execution Environment requirements should capture the essential properties of the environment such as temperature, humidity, the area of operation, and electric current in which the embedded system under development will operate [22]. *Certification Process* requirements aim the application of a particular development process, guidelines, documentation, and steps for the product under development. These issues will depend on the embedded system domain, such as avionics, automotive, industrial automation, and medical. *Process Infrastructure* requirements and standards are related to the way the results of RE activities and tasks will be documented, managed and which development tools must be used [23].

Product Infrastructure requirements and standards aim the definition of which implementation technology (programming language, code pattern) must be used or which standard components have to be reused [23]. Finally, *Manufacture* requirements aim the production of relevant information/ documentation for fabricating the product [24]. The *Environmental Requirements* should define the environmental assumptions/constraints such as human users, temperature, and local of operation required by the system to its correct operation.

Hardware is the physical equipment used to process, store, or transmit computer programs or data. *Hardware Requirements* specify the *Hardware Devices* characteristics that should be provided to the embedded system. The requirements for hardware should include information such as hardware functions, user interaction, hardware characteristics (temperature range, humidity range, battery, e.g.), action buttons, memory specification, e.g., [25].

The main *Hardware Devices* are *Sensor*, *Actuator*, *External Interface* and *Hardware Adapter*. A *Sensor* is any system providing up-to-date information about the context where the system is running. *Sensors* monitor *Control Variables* and *Environment Variables*. Besides, it can send messages to the *Software*. *Control Variable* is one that the system can control, maintaining or changing its value to keep the system properly running. On the other hand, *Environment Variables* are a set of dynamic named values that can affect the way embedded systems will behave.

An *Actuator* is any actuator in the environment which can receive commands from the system to act on the environment context. *Actuators* generate *Actions*, that can be a *Software Behavior* or *Hardware Behavior*. An *Action* is any activity performed by the system, and *Software Behavior* is a function performed by software while *Hardware Behavior* is a function performed by hardware.

External Interface provides external communication in different ways such as *USB*, *CAN*, *Serial*, *Serial Peripheral*

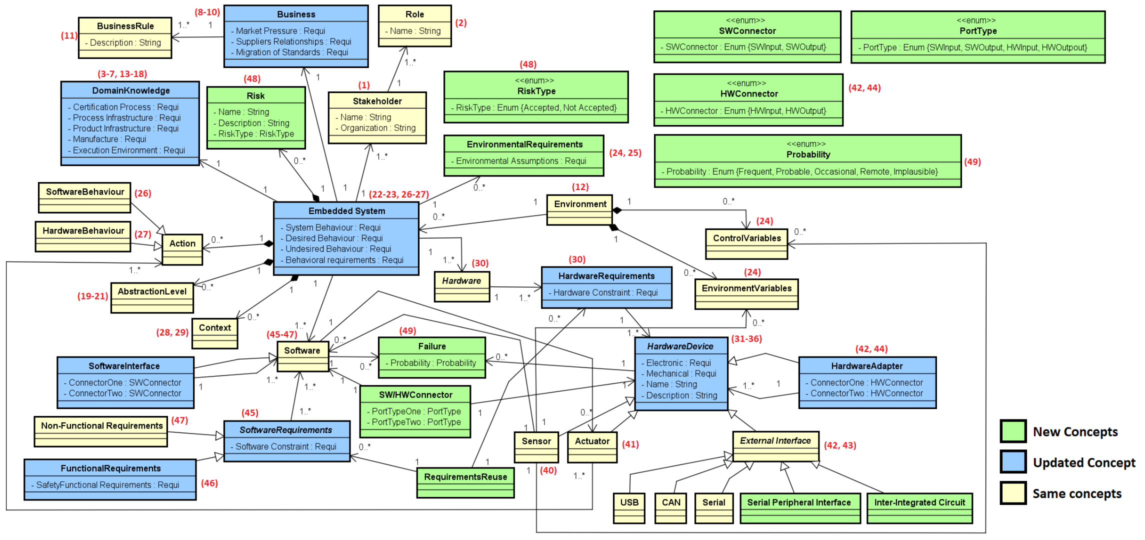

Fig. 2. Metamodel for embedded systems - MM4ES

Interface, and *Inter-Integrated Circuit*. A *Hardware Adapter* is a physical device that allows one hardware or electronic interface to be adapted (accommodated without loss of function) to another hardware or electronic interface. The *HWConnector* of a *Hardware Adapter* is an enumeration with the following options: *HWInput* (hardware input), and *HWOutput* (hardware output).

Fig. 3 presents a set of 16 *Hardware Devices* identified in our study (*Controller*, *Memory RAM*, *Memory ROM*, *Processor*, *Microprocessor*, *Microcontroller*, *Digital Signal Processor*, *Legacy Hardware*, *Seven-Segment Display*, *LCD Display*, *Graphical Display*, *Trimmer Potentiometer*, *Keypad*, *Buttons*, *Storage Device* and *Power Supply*).

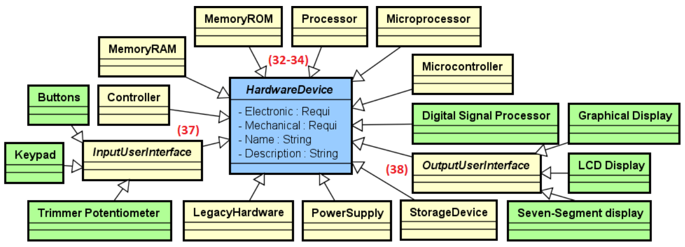

Fig. 3. Hardware devices for embedded systems

Software Requirements defines the services that the *Software* should provide and they set out constraints on the system's operation. Besides, they should address the following issues: functionality (what is the software supposed to do?), external interfaces (how does the software interact with people, system hardware, other hardware, and other software?), performance (e.g. what is the speed, availability, response time, recovery time of various software functions?), attributes (e.g. what are the portability, correctness, maintainability, security, considerations?), and design constraints imposed on an implementation [26].

The *Software Requirements* is classified into (i) *Functional Requirements*, i.e., a functionality of the system, and (ii) *Non-Functional Requirements*, i.e., a feature or characteristics that affect an item quality. A *Software Interface* is a software responsible for connecting one or more softwares. The *SWConnector* of a *Software Interface* is an enumeration with the following options: *SWInput* (software input), and *SWOutput* (software output). *SW/HWConnector* is a special communication component. It works as a bridge for the communication between software and hardware [27]. *Port Type* of an *SW/HWConnector* is an enumeration with the following options: *SWInput* (software input), *SWOutput* (software output), *HWInput* (hardware input), and *HWOutput* (hardware output).

An embedded system can be associated with several *Contexts*. A *Context* is a state of the world that is relevant to an actor goal. Contextualization can be used as a way to allow for changes at both design and run-time. *Abstraction Level* defines a view of an object that focuses on the information relevant to a particular purpose and ignores the remainder of the information.

A *Failure* is an event where a software or hardware component does not exhibit the expected behavior. It has an enumeration called *Probability* which can be *Frequent*, *Probable*, *Occasional*, *Remote*, and *Improbable*. The possibility that something bad will happen is a *Risk*. *Risk Type* is a property that defines if the *Risk* is acceptable (*Accepted*) or not (*Not Accepted*).

Fig. 4 presents a set of 58 *Non-Functional Requirements* and their refinements, which characterizes specific issues that must be considered to achieve such quality requirement. Depending on the ES domain, all refinements or part of them can be implemented. The requirements with refinements are *Security*, *Performance*, *Safety*, *Time*, *Privacy*, *Resource*, and *Temporal*. For example, Safety has five refinements, such as *ReactionToViolationOfPrevention*, *PreventionOfHazards*, *Pre-*

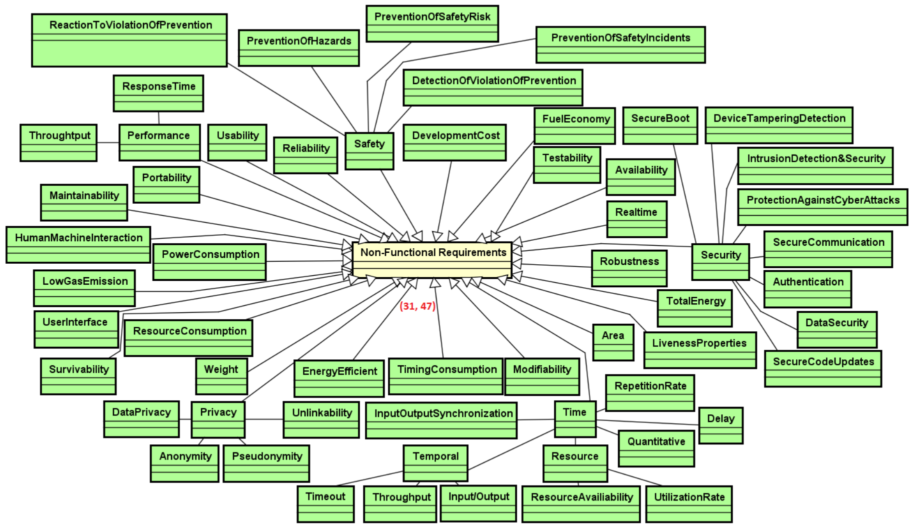

Fig. 4. Non-functional requirements for embedded systems

ventionOfSafetyRisk, *PreventionOfSafetyIncidents*, and *DetectionOfViolationOfPrevention*. The rest of the requirements have no refinements. As ongoing work, we are using the NFRs presented Fig. 4 to develop a catalog to support requirements engineers during the elicitation process.

V. Requirements elicitation process

The contribution presented in this section is part of an on going Ph.D Thesis that consists on the development of a requirements engineering process for embedded systems. In this paper, we focus on its requirements elicitation actions.

A. Process development methodology

We have followed 6 steps to develop the requirements elicitation process.

1) Knowledge acquisition: In this first step, we investigated the literature available about RE for embedded systems to become familiar with the domain. To accomplish this goal, we performed an SLR to evaluate and synthesize the evidence available in the literature to answer research questions on the use of approaches, methods, techniques, and processes to support the RE in the ES domain.

2) Problem definition: After answering a set of questions regarding RE for embedded systems, we investigated the problem relevance, i.e. (i) the lack of a requirements engineering process. Therefore, we conclude that there was a need for proposing the process according to the goals and scope of our work. This issue lead us to another problem, (ii) what should be considered to develop the RE process?

3) Metamodel development: To overcome question mentioned before, we followed the steps described in this paper to develop a metamodel that captures ES concepts with their interrelations.

4) Identification of information sources: In this step, we took the concepts of the metamodel and looked for them in the studies of the SLR (84 studies) and the main RE standards to identify and select the information sources for the requirements engineering sub-processes, and actions/practices. We considered the following RE standards: IEEE Std 1233:1998, IEEE Std 830:1998, ISO/IEC 12207, ISO/IEC 29158, ISO/IEC 15289, ISO/IEC 15288, INCOSE Handbook, SE-CMM, and CMMI-DEV. In this paper, we focus on the requirements elicitation activity.

5) Definition of process design: After the analysis of the information sources, we chose the design of the process. We followed the structure of Uni-REPM, since it is an universal lightweight model to evaluate the maturity of a RE process [28].

The model is structured in three levels, namely: Main Process Area (MPA), Sub-Process Area (SPA) and Action (ACT). On the top level of the model, there are four Main Process Areas (Business Requirements, System Requirements, Software Hardware Requirements and Security Requirements). Each MPA is further broken down into several SPAs and, on the bottom level, an Action denotes a particular activity that should be executed or a specific item that should be present. In the context of this paper, we focus on the activities that should be performed to elicit/identify requirements.

6) Development of the process: In this step, we developed the process. Hence, we have defined in total 4 process areas, 24 sub-processes, and 89 elicitation actions. See Tables II and III.

B. The REPES process

To save space, we are objectives by describing the elicitation actions. Thus, we do not present them in the structure defined for the process (step 5). Note that the number of each action is traced to the concept that supported its development (see Figures 2, 3, and 4). Using the process, ES engineers can manage the elicitation activity in an organized way. The process is divided into four process area, named **Business Requirements (BR)**, **System Requirements (SR)**, **Software**

TABLE II
REPES Elicitation Actions - Part 1

#	ID	Description
	BR	Business Requirements
	BR.ISS	Identification of System Stakeholder
1	BR.ISS.a1	Identify the stakeholders to be involved in the system life cycle
2	BR.ISS.a2	Define their roles and responsibilities
	BR.DPI	Define Process Infrastructure Requirements
3	BR.DPI.a1	Describe the way the results of the requirements engineering activities will be documented
4	BR.DPI.a2	Define one or more tools, or an environment, required to support the activities towards the RE
	BR.CPR	Define Certification Process Requirements
5	BR.CPR.a1	Define and document a specific development process for the company
6	BR.CPR.a2	Define a step by step for the product development considering the chosen development process
7	BR.CPR.a3	Establish a set of items/artifacts that must be produced and documented by the requirements engineering
	BR.ASP	Acquire and Supply Products or Services
8	BR.ASP.a2	Identify and document a set of supplier candidates for the project that is being carried out
	BR.MMP	Management of Market Pressure
9	BR.MMP.a3	Document and share with the project team the applicable legislation, organizational constraints, industry standards, and regulatory needs
	BR.BEE	Business and Environmental Elicitation
10	BR.BEE.a1	Elicit and include in the business documentation a set of business processes in which the embedded system will operate
11	BR.BEE.a2	Elicit and include in the business documentation a set of business rules
12	BR.BEE.a3	Elicit and document a set of environmental information
	BR.DAN	Domain Analysis
13	BR.DAN.a1	Elicit information about system domain restrictions
14	BR.DAN.a2	Elicit information about technical infrastructure of the system
15	BR.DAN.a3	Elicit information about operational domain of the system
16	BR.DAN.a4	Elicit information about system boundaries
	BR.PIR	Product Infrastructure Requirements
17	BR.PIR.a1	Define a set of implementation technology
18	BR.PIR.a2	Establish one or more case tools, or an environment to support the activities towards the embedded systems development
	BR.ALD	Abstraction Level Definition
19	BR.ALD.a1	Identify and define different kinds of representations, such as models, to be produced by the sub-processes
20	BR.ALD.a2	Identify and associate the stakeholders who have different skills with each model to be produced
21	BR.ALD.a3	Establish the levels of detail of the representations based on the different skills of the people involved after the stakeholder's identification
	SR	System Requirements
	SR.DSG	Define System Goals
22	SR.DSG.a1	Provide short statements describing what the system must accomplish
23	SR.DSG.a3	Elicit a set of system goals from stakeholder's needs
	SR.IEA	Identification of Environmental Assumptions
24	SR.IEA.a1	Establish and document the monitored and controlled variables
25	SR.IEA.a5	Establish a list of devices that monitor the monitored and controlled variables
	SR.DBR	Definition of Behavioral Requirements
26	SR.DBR.a1	Provide a set of software behaviors to document the actions the software should perform
27	SR.DBR.a2	Provide a set of hardware behaviors to document the actions the hardware should perform
	SR.MCI	Management of Contextual Information
28	SR.MCI.a1	Analyze the input documents to identify the contexts that can affect the system operation
29	SR.MCI.a2	Identify and document statements, facts, and variables
	SHR	Software and Hardware Requirements
	SHR.HRC	Hardware Constraints
30	SHR.HRC.a1	The constraints for each hardware device previously identified are defined and documented
	SHR.HRD	Hardware Requirements Definition
31	SHR.HRD.a2	Elicit and specify a set of non-functional requirements that the hardware devices must fulfill
32	SHR.HRD.a3	Provide an overview of the hardware components to be used in the development of an embedded system
33	SHR.HRD.a4	Elicit the mechanical requirements
34	SHR.HRD.a5	Elicit electrical requirements

TABLE III
REPES Elicitation Actions - Part 2

#	ID	Description
35	SHR.HRD.a6	Identify and document the microcontroller of the embedded system based on the results of previous actions
36	SHR.HRD.a9	Elicit a set of manufacture requirements
	SHR.DIO	Define Input and Output User Interface
	SHR.DIO.a1	Define and document interface standards
37	SHR.DIO.a2	Define and document a set of input user interface
38	SHR.DIO.a3	Define and document a set of output user interface
	SHR.DSA	Define Sensors and Actuators
39	SHR.DSA.a1	Define and document a set of sensors and actuators standards
40	SHR.DSA.a2	Identify and document a set of sensors
41	SHR.DSA.a5	Identify and document a set of actuators
	SHR.DEI	Define External Interfaces and Hardware Adapter
42	SHR.DEI.a1	Identify and document a set of external interfaces and hardware adapters standards
43	SHR.DEI.a2	Identify and document a set of external interfaces
44	SHR.DEI.a3	Identify and document the hardware adapters
	SHR.SFC	Software Constraints
45	SHR.SFC.a1	Identify and document the software constraints
	SHR.SRD	Software Requirements Definition
46	SHR.SRD.a1	Elicit and document a set of functional software requirements
47	SHR.SRD.a5	Elicit and document a set of non-functional requirements
	SER	Security Requirements
	SER.RMG	Risk Management
48	SER.RMG.a1	Identify and document potential risks
	SER.FMG	Failure Management
49	SER.FMG.a1	Identify, classify, and document potential software and hardware failures

Hardware Requirements (SHR) and **Security Requirements (SER)**. These process areas are organized in 24 sub-processes that have 89 actions, which 49 of them are related to requirements elicitation. However, some of them can also be associated with others RE activities (example: elicit and specify non-functional requirements).

All actions are responsible for generating artifacts that will be part of the ES requirements. Additionally, the actions of BR and SR process areas help engineers to move from high-level requirements to more specific technical requirements. The process steps are not meant to be strictly sequential, apart from the first step; they are only given for guidance. A summary of the REPES process with focus on elicitation actions is shown in Tables II and III.

C. Process usage

A typical scenario in which the REPES can be used is when an organization decides to introduce requirements engineering activities in its embedded systems development process. Applying the process in a specific domain, we can perform the actions in a sequential way to get a document with the requirements elicited.

As a proof of concept, we applied the process provided in this paper to elicit requirements from a patient control analgesia pump. The goal is to demonstrate the utility of the process by presenting examples as results of the actions. See Tables II and III to verify the action that was used to elicit the requirements. Note that we can check the number in the metamodel to see the correspondent embedded system concept. This cross-referenced is also a way to demonstrate the utility of the metamodel.

Due to space limitation, it is difficult to provide in this paper a detailed description and to include all the results of the elicitation actions. As a usage example, we performed actions from all process areas, resulting in 13 actions (26.5%) of the 49 ones presented in the elicitation process. Hence, this can be a limitation of our work.

We used the documents provided by the SAnToS research group at Kansas State University (KSU) and the National Science Foundation US Food and Drug Administration Scholar-in-Residence (NSF FDA SIR) as input for the elicitation actions execution. The documents are available at https://rtg.cis.upenn.edu/gip/.

The real world scenario used is related to a medical device. The following description was retrieved from Pennsylvania Patient Safety Authority [29]: Patient-controlled analgesic (PCA) infusion pumps allow patients to give themselves pain-relieving medication within certain limits as prescribed by a doctor or other licensed professional. PCA therapy is used for patients after an operation, obstetric patients, terminally ill patients and patients who have a serious injury. PCA pumps deliver medication through a needle (e.g., intravenously) and allow patients to give themselves the medication by the push of a button. Next, we present the actions we have performed and their results, i.e., the elicited requirements.

First, we started executing actions from the *Business Requirements* process area. For example, performing action 1, *(BR.ISS.a1 - Identify the stakeholders to be involved in the system life cycle)* we can identify the Clinician and the Patient. Executing action 2, *(BR.ISS.a2 - Define their roles and responsibilities)* we can get following results: (i) clinician - initialization, attachment, basal infusion, detachment; and (ii) patient - extra dose upon patient-determined need. Continuing with the execution of action 9, *(BR.MMP.a3 - Document and share with the project team the applicable legislation, organizational constraints, industry standards, and regulatory needs)* we can obtain the (i) ASTM International F2761-09 Medical Devices and Medical Systems; and (ii) IEC 60601-1-8 Medical electrical equipment.

The action 14 *(BR.DAN.a2 - Elicit information about technical infrastructure of the system)* is responsible for the identification of external software that interacts with the system such as the Integrated Clinical Environment (ICE). A system which allows clinician's to remotely monitor the operation of the pump. The result of the action 15 *(BR.DAN.a3 - Elicit information about the operational domain of the system)* is the following description: The PCA pump will operate in a hospital room or similar clinical setting: controlled ambient temperature, assured power, lighting, infection-control procedures and equipment, regular electromagnetic fields and particles.

Actions from the *System Requirements* process area were executed as follows: Performing action 22, *(SR.DSG.a1 - Provide short statements describing what the system must accomplish)* we can determine that a PCA infusion pump infuses narcotic, liquid pain-killer at a prescribed basal rate plus any bolus doses that the patient may request to alleviate their pain, or be commanded by an attending clinician, most often, a registered clinician. Executing action 23, *(SR.DSG.a3 - Elicit a set of system goals from stakeholder's needs)* we can define the following goals: (G1) the pump will safely infuse drugs intravenously for pain relief, and (G2) the patient should receive enough drug to reduce his pain.

Through the execution of action 26, *(SR.DBR.a1 - Provide a set of software behaviors to document the actions the software should perform)* we can define that (i) after the start button has been pushed, a timer counter shall be displayed, and (ii) when the infusion is in progress, a boolean signal shall be displayed. Carrying out action 27, *(SR.DBR.a2 - Provide a set of hardware behaviors to document the actions the hardware should perform)* we can determine that (i) after the button A has been pushed, a red light shall be lit, and (ii) after the dose button has been pushed, two beeps shall be sounded, and the pump will begin delivering the demand dose.

Next, we present two actions from the *Software Hardware Requirements* process area. Executing action 43, *(SHR.DEI.a2 - Identify and document a set of external interfaces)* we can identify the delivery tube and needle. The drug is conveyed from the pump to a needle to be infused into the patient. The needle is placed into a vein, usually in an arm or hand. The results of action 47 *(SHR.SRD.a5 - Elicit and document a set of non-functional requirements)* are: (i) response time to a button press shall be less than 200 msec (performance), and (ii) software access shall be password protected (security).

Two actions belong to the *Security Requirements* process area. Performing action 48, *(SER.RMG.a1 - Identify and document potential risks)* we can define (i) the execution environment of the system, while executing action 49, *(SER.FMG.a1 - Identify, classify, and document potential software and hardware failures)* we can identify mechanical failure and classify it as deterioration and/or shock.

Analogously, the same can be done for the rest of the actions.

VI. Conclusion and future work

According to our Systematic Literature Review (SLR) [11], no evidence explicitly depicts how an embedded system must be elicited and specified. Despite all information collected, analyzed, and interpreted, some issues remain: (i) the studies did not present what should be considered by requirements engineers during the elicitation and specification of embedded systems, and (ii) there is a need for a specific RE process for embedded systems.

In this paper, we propose a metamodel for embedded systems to describe the concepts that should be elicited and specified by requirements engineers. We focused on the representation and relationships between the identified concepts in our SLR and the feedback provided by a domain expert. Then, we used the metamodel to guide the development of a requirements elicitation process. The process provided in this paper describes the actions to be performed for the identification and definition of embedded systems requirements. We aim to help organizations towards a more mature process execution.

The main contributions of this paper are the followings: (i) a knowledge-based metamodel which serves as a resource model for integrating characteristics of the significant concepts appearing in embedded systems; (ii) a reference metamodel that can be updated in future works to consider more embedded systems concepts; (iii) a metamodel that can be used as input for model transformation if model-driven development is intended, allowing the development of a domain specific language from scratch or to extend an existing language; (iv) a metamodel that can help practitioners to improve the traceability of the requirements specification; (v) an elicitation process that was guided by a metamodel that covers ES concepts identified by studies applied in academy and industry; and (vi) the process can be used as a guide to assess the requirements activities of organizations.

As limitations of this paper, we can highlight: (i) RE for embedded systems is a difficult activity. Thus, only a metamodel can not address all characteristics of an ES for a specific domain; (ii) the metamodel and the requirements elicitation process was not evaluated in the industry yet; and (iii) a threat may be the selection of actions that are included in the REPES process since they were based on the metamodel concepts and RE standards.

As future work, we intend to execute an evaluation of the process in the industry to get more realist results. Moreover, this study has generated some research directions that we intend to address in future works: (1) How can we evaluate the completeness of the proposed metamodel? (2) How can we extend the metamodel to represent the specific characteristics of the different domains of an ES? (3) How can we evaluate whether the process has sufficient coverage of actions? (4) How can we validate the usefulness and ease of use of the process? (5) How can we develop a CASE tool to support the process? (6) How can we conduct a comparative analysis with other studies using the same case study (PCA)?

ACKNOWLEDGMENT

The following Brazilian institutions have supported this work: CNPQ and FACEPE.

REFERENCES

[1] M. Broy and T. Stauner, "Requirements engineering for embedded systems," *Informationstechnik und Technische Informatik*, vol. 41, pp. 7–11, 1999.

[2] S. Haldar and A. Aravind, *Operating systems*. Pearson Education India, 2010.

[3] F. Vahid and T. Givargis, "Embedded system design: A unified hardware/software approach," *Department of Computer Science and Engineering University of California*, 1999.

[4] M. Wolf, *Computers as components: principles of embedded computing system design*. Elsevier, 2012.

[5] A. Berger, *Embedded Systems Design: An Introduction to Processes, Tools and Techniques*. CMP Books; 1st edition, 2001.

[6] I. Sommerville, *Software Engineering*, 9th ed. Addison-Wesley, 2011.

[7] E. Sikora, B. Tenbergen, and K. Pohl, "Industry needs and research directions in requirements engineering for embedded systems," *Requirements Engineering*, vol. 17, no. 1, pp. 57–78, 2012.

[8] J. Ossada, "Gerse: Requirements elicitation guide for small and medium size companies, in portuguese: Gerse: Guia de elicitação de requisitos para pequenas e médias empresas," Piracicaba, Sao Paulo, Brazil, 2010.

[9] G. Kainz, C. Buckl, S. Sommer, and A. Knoll, "Model-to-metamodel transformation for the development of component-based systems," in *13th International Conference on Model Driven Engineering Languages and Systems*. Springer, 2010, pp. 391–405.

[10] H. Fennel, S. Bunzel, H. Heinecke, J. Bielefeld, S. Fürst, K.-P. Schnelle, W. Grote, N. Maldener, T. Weber, F. Wohlgemuth *et al.*, "Achievements and exploitation of the autosar development partnership," *Convergence*, 2006.

[11] T. Pereira, D. Albuquerque, A. Sousa, F. M. R. Alencar, and J. Castro, "Retrospective and trends in requirements engineering for embedded systems: A systematic literature review," in *Proceedings of the XX Iberoamerican Conference on Software Engineering*, 2017.

[12] T. Pereira, D. Albuquerque, A. Sousa, F. Alencar, and J. Castro, "Towards a metamodel for a requirements engineering process of embedded systems," in *VI Brazilian Symposium on Computing Systems Engineering*, 2016.

[13] T. Arpinen, T. Hämäläinen, and M. Hännikäinen, "Meta-model and uml profile for requirements management of software and embedded systems," *EURASIP Journal on Embedded Systems*, vol. 2011, no. 1, p. 592168, 2011.

[14] M. Li, F. Batmaz, L. Guan, A. Grigg, M. Ingham, and P. Bull, "Model-based systems engineering with requirements variability for embedded real-time systems," in *Fifth Model-Driven Requirements Engineering Workshop at 23rd IEEE International Requirements Engineering Conference*, 2015.

[15] H. Dubois, M.-A. Peraldi-Frati, and F. Lakhal, "A model for requirements traceability in a heterogeneous model-based design process: Application to automotive embedded systems," in *15th ieee international conference on engineering of complex computer systems*, 2010.

[16] A. Albinet, S. Begoc, J. Boulanger, O. Casse, I. Dal, H. Dubois, F. Lakhal, D. Louar, M. Peraldi-Frati, Y. Sorel *et al.*, "The memvatex methodology: from requirements to models in automotive application design," in *4th European congress of Embedded Real Time Software*, 2008.

[17] M. Glinz, "On non-functional requirements," in *Requirements Engineering Conference*, 2007, pp. 21–26.

[18] T. Pereira, "Bvccon-tool: A modelling tool to support a dynamic business process configuration approach, in portuguese: Bvccon-tool: Uma ferramenta para apoiar uma abordagem de configuração de processos de negócios dinâmicos," Recife, Pernambuco, Brazil, 2014.

[19] M. Usman, R. Britto, J. Börstler, and E. Mendes, "Taxonomies in software engineering: A systematic mapping study and a revised taxonomy development method," *Information and Software Technology*, 2017.

[20] I. Krüger, C. Farcas, E. Farcas, and M. Menarini, "7 requirements modeling for embedded realtime systems," in *Model-Based Engineering of Embedded Real-Time Systems*, 2010.

[21] J.-L. Boulanger *et al.*, "Requirements engineering in a model-based methodology for embedded automotive software," in *IEEE International Conference on Research, Innovation and Vision for the Future*. IEEE, 2008, pp. 263–268.

[22] C. J. Fidge and A. M. Lister, "Disciplined approach to real-time systems design," *Information and Software Technology*, pp. 603–610, 1992.

[23] M. Broy and O. Slotosch, "From requirements to validated embedded systems," in *First International Workshop on Embedded Software*. Springer, 2001, pp. 51–65.

[24] K. H. Pries and J. M. Quigley, *Project Management of Complex and Embedded Systems: Ensuring Product Integrity and Program Quality*. CRC Press, 2008.

[25] L. Martins Galvao, J. Ossada, A. Belgamo, and B. Ranieri, "Requirements elicitation guide for embedded systems: An industry challenge," *The Eighth International Conference on Software Engineering Advances*, 2013.

[26] IEEE, "Ieee recommended practice for software requirements specifications," *IEEE Std 830-1998*, pp. 1–40, Oct 1998.

[27] X. F. Zha, S. J. Fenves, and R. D. Sriram, "A feature-based approach to embedded system hardware and software co-design," in *Computers and Information in Engineering Conference*. American Society of Mechanical Engineers, 2005, pp. 609–620.

[28] G. Tony and T. Kaarina, "Universal requirements engineering process maturity model," =http://www.gorschek.com/doc/REPM_Project_files/uniREPM_v09CR.pdf, 2011, acessed: 2018-06-19.

[29] B. R. Larson and J. Hatlicff., "Open patient-controlled analgesia infusion pump system requirement," http://santoslab.org/pub/open-pca-pump/artifacts/Open-PCA-Pump-Requirements.pdf, 2018, accessed: 2018-06-04.