Requirements Engineering Process Maturity Model for Embedded Systems

Tarcísio Pereira[1], Sthéfanie Dal Magro[1], Mozart Melo[2], Jaelson Castro[1] and Fernanda Alencar[1]

[1] Universidade Federal de Pernambuco, Recife-PE, Brasil
[2] Universidade de Pernambuco, Recife-PE, Brasil

tcp@cin.ufpe.br, sdm2@cin.ufpe.br, mozart-kmf@gmail.com, fernandaalenc@gmail.com, jbc@cin.ufpe.br

Abstract. Requirements engineering for embedded systems is challenging since they have unique properties that make it complex, expensive and error-prone as compared with other system categories, such as information systems. It is well known that organizations with high process maturity level reduce errors in system development. There are some requirements standards and maturity models. However, they are not complete and detailed enough to guide organizations and practitioners in the Requirements Engineering for Embedded Systems. In this paper a maturity model is developed which can be used to assess the maturity of requirements engineering process for embedded systems. We propose an Embedded System Module, which consists of an enhancement of the Unified Requirements Engineering Process Maturity model (Uni-REPM). This module is based on international standards, a metamodel, and an extensive systematic literature review. It will provide a quick assessment tool through which a company that develops embedded systems would be able to know what the strengths and weaknesses of their requirements are engineering process.

Keywords: Embedded Systems, Requirements Engineering, Maturity Models,

1 Introduction

An Embedded System (ES) can be defined 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 [4].

Requirements engineering for embedded systems is challenging since it has unique properties that make it complex, expensive and error-prone as compared with other software categories, such as information systems. Examples of unique properties: (i) Embedded Systems are usually tightly coupled to their physical environment (eg. burglar alarm, microwave, printer, brakes, and climate control), (ii) the context of ES requires an extensive range of stakeholders with different roles (eg. hardware, electrical, and requirements engineer), (iii) the interaction interfaces are mostly hardware

components (eg. buttons and LCD display), (iv) Hardware Requirements Specification (HRS) is as essential as Software Requirements Specification (SRS), and (v) most of the embedded software functions are performed regularly and repetitively.

Inappropriate Requirements Engineering (RE) practices may result in incomplete requirements, incorrect elicitation and specification, high complexity, and economic or human loss. In fact, software plays a critical role in ES. Hence, RE for embedded systems is a problem that needs to be investigated to avoid inadequate or misunderstood requirements [18, 22].

A maturity model can be defined as a set of stages through which an organization progresses from an initial to the desired stage [25]. Organizations with high maturity levels minimize errors in system development and requirements engineering. Thus, companies can avoid fixing problems after system delivery. Additionally, there are organizational benefits such as gains in productivity, improved product quality, and more accurate and predictable schedules and budgets [7].

There are standards that cover requirements engineering such as IEEE Std 1233 [10], IEEE 830 [11], ISO/IEC 12207 [12], and ISO/IEC 29148 [14] as well as some maturity models such as SE-CMM [1], Uni-REPM [23], and CMMI-DEV [21]. However, these standards and maturity models are general and do not include practices related to the peculiarities of the embedded system like hardware, mechanical, electrical, and environmental requirements. Additionally, some requirements maturity models like Requirements Capability Maturity Model (RCMM) [2], and Market Driven Requirements Engineering Process Model (MDREPM) [9] apply to a limited type of RE process and application domain or exist in draft form. Hence, they are not complete and detailed enough to guide organizations and practitioners in the requirements engineering for embedded systems.

We argue that the standards and models discussed above seem unable to help embedded systems organizations in assessing and improving their RE process capability and maturity. Thus, the goal of this paper is to propose an embedded system maturity module for organizations to use it as a guide to assess their current practices and processes, as well as improve their processes by adopting selected practices suggested by the module. It is named Uni-REPM ES.

The maturity module proposed in this work is an extension to Uni-REPM. We opted to extend it since Uni-REPM is a known universal lightweight model to evaluate the maturity of RE processes [23]. Additionally, it has been used by companies, covers the main requirements engineering phases, and allows the requirements engineers to evaluate by themselves a RE process in few hours. Moreover, it was the basis for the Uni-REPM SCS extension [24], which define a maturity model for Safety Critical Systems (SCS).

The goal of the new Uni-REPM ES module is to offer to organizations that develop embedded systems a way to evaluate and improve their RE practices. Using the Uni-REPM ES module, organizations can identify the strengths and weaknesses in their requirements engineering process and improve them from basic to advanced level. Besides, the module can be used in different embedded system domains, and it does not require the use of specific techniques.

The remainder of this paper is organized as follows. Section 2 presents the research methodology. In Section 3 we present the structure and contents of the Uni-REPM ES module. The limitations are discussed in Section 4. Finally, conclusions and future works are shown in Section 5.

2 Research Methodology

The construction of the Uni-REPM ES was based on 8 steps:

1. Knowledge acquisition: In this first step, we extended a systematic literature review that was conducted to identify and analyze the requirements engineering approaches for embedded systems [19], in order to cover the period from 1970 to June 2019.
2. Problem definition: After a comprehensive investigation of the domain and answering a set of questions regarding RE for embedded systems, we identified, i.e (i) the lack of requirements engineering maturity model for ES, (ii) the need for specific RE practices for embedded systems. These issues lead us to another problem, (iii) what should be considered when creating a RE maturity model for embedded systems?
3. Metamodel development: We developed a metamodel to serve as a resource model for integrating characteristics of the significant concepts of embedded systems [20].
4. Identification of information sources: In this step, we took the concepts of the metamodel and looked for them in all 92 studies of the SLR and the main RE standards to identify and select the information sources for the requirements engineering subprocesses, and actions/practices. We considered the following several RE standards to knowledge acquisition, including: IEEE Std 1233:1998[10], IEEE Std 830:1998[11], ISO/IEC 12207[12], ISO/IEC 15289[15], ISO/IEC 15288[13], SE-CMM[1], and CMMI-DEV[21].
5. Definition of module design architecture: After the analysis of the information sources, we proposed the structure and organization of the module components such as its process areas, sub-process areas, and actions.
6. Development of a draft model - process dimension: In this step, we developed a draft model. Hence, after the literature review and RE standards analysis, we proposed 15 new sub-processes areas and 89 actions to be addressed by companies that develop embedded systems. In Figure 1 we present the sub-process areas, their respective actions, and how they are connected to the Uni-REPM model.
7. Development of a draft model - maturity dimension: In this stage, we updated the draft module of the previous step considering the maturity dimensions. Thus, we assigned a fixed number of maturity levels for the module actions. We opted to use the same measurement mechanism as Uni-REPM. The Likert scale with three levels (Basic, Intermediate, and Advanced). According to Vilela et al [24], three options reduce user difficulties and improve their interpretation. Thus, it makes it easier for practitioners to understand the maturity model evaluation
8. Consolidate the model: After several discussions and refinements, we consolidate the model in an understandable way.

3 The Uni-REPM Embedded System Module

We propose an embedded system maturity module (Uni-REPM ES) for Uni-REPM. The goal is to include requirements engineering practices for embedded systems and to support companies to improve the maturity of their processes. In the next sections, we describe the sources of sub-processes, module structure, its contents and how to use it to evaluate the maturity level of an organization.

3.1 Source of Sub-Process

The module consists of several sources of information such as systematic literature review [19], requirements engineering standards [10,11,12,13,14,15], and concepts of a metamodel [20]. We examined requirements engineering standards to knowledge acquisition since organizations have used them. These documents provide guidance to gather, evaluate, document, and manage requirements for software development. Note that the complete description of the module is available at: http://tiny.cc/Uni-REPMES

3.2 Overall Structure of Uni-REPM ES

The Uni-REPM ES module is structured in the same way as Uni-REPM [23]: A Process Area view and a Maturity Level view. The purpose of the process area view is to provide a way to navigate the model easily and find actions/practices of the same group. On the other hand, the maturity level view specifies sets of practices that belongs to a consistent and coherent RE process, and where the practices in one level supports each other as well as the more advanced practices on the next level [23].

Process area view. The embedded system module follows the same hierarchy of Uni-REPM. It defines three levels: Main process area (MPA), Sub-process area (SPA) and Action (ACT). On the top level of the model, there are seven Main Process Areas: organizational support (OS), requirements process management (PM), requirements elicitation (RE), requirements analysis (RA), release planning (RP), documentation and requirements specification (DS), and requirements validation (RV). Each MPA is further broken down into several SPAs and, on the bottom level, an Action denotes an activity that should be executed or a specific item that should be present [23].

Each action has a level that goes from 1 to 3, corresponding to "Basic," "Intermediate," and "Advanced" level. It depends on the difficulty to perform the action; how essential it is for the RE process and dependencies between actions. This assignment is necessary for the Maturity Level View development. Additionally, Uni-REPM ES has a measurement instrument for practitioners to identify the status of their RE process.

Figure 1 presents the Embedded Systems module and its relationship with Uni-REPM. The embedded system module extends the Uni-REPM model by adding new SPAs highlighted through dashed lines. The original module and processes were not altered, and none were removed.

Our ES module has seven main processes, fifteen sub-processes, and 89 embedded systems actions describing practices to improve processes maturity.

Considering our goal to integrate embedded systems practices in RE process, we preserve the seven process areas of Uni-REPM. It is important to highlight that Uni-REPM was defined according to well-adopted processes such as Kontonya and Sommerville [16]. The Main Process Areas (MPAs) are described below.

(1) Organizational Support (OS): it is responsible for assessing the amount of assistance given to RE practices from the surrounding organizations.
(2) Requirements Process Management (PM): it contains a set of activities to manage, control requirements change as well as to ensure the organization about the process execution.
(3) Requirements Elicitation (RE): it deals with the discovery and understanding of customer needs to communicate the information to other stakeholders such as system developers.
(4) Requirements Analysis (RA): encompasses a set of activities to detect errors, to create a detailed view of requirements as well as to estimate risk and priorities.
(5) Release Planning (RP): covers a set of actions to determine the optimal set of requirements for a specific release to achieve a defined/estimated time and cost goals.
(6) Documentation and Requirements Specification (DS): it handles the organization of requirements gathered from elicitation process into consistent, accessible and reviewable documents.
(7) Requirements Validation (RV): the goal is to check if the requirements agree with quality standards and the needs of stakeholders. Additionally, it contains actions to assure that the documented requirements are complete, correct, consistent, and unambiguous.

Sub-process area (SPA) contains closely recommendations or permissible actions intended to support the achievement of a bigger goal [23]. The unique identifier assigned to each SPA is composed of the MPA identifier to which the SPA attaches and its abbreviation. For example, "OS.PR" represents a sub-process called "Process Requirements" which resides under "Organizational Support".

We purpose fifteen SPAs to be connected to the several MPAs as described below:

(OS.PR) Process Requirements: covers the identification of system stakeholders and the definition of a development process and guidelines to define the way the requirements engineering activities and task are documented, managed and which requirements engineering tools must be used.

(OS.ER) Environmental Requirements: it has actions to understand and describe the application domain, the business and business rules required by the system and the environment in which the system will operate.

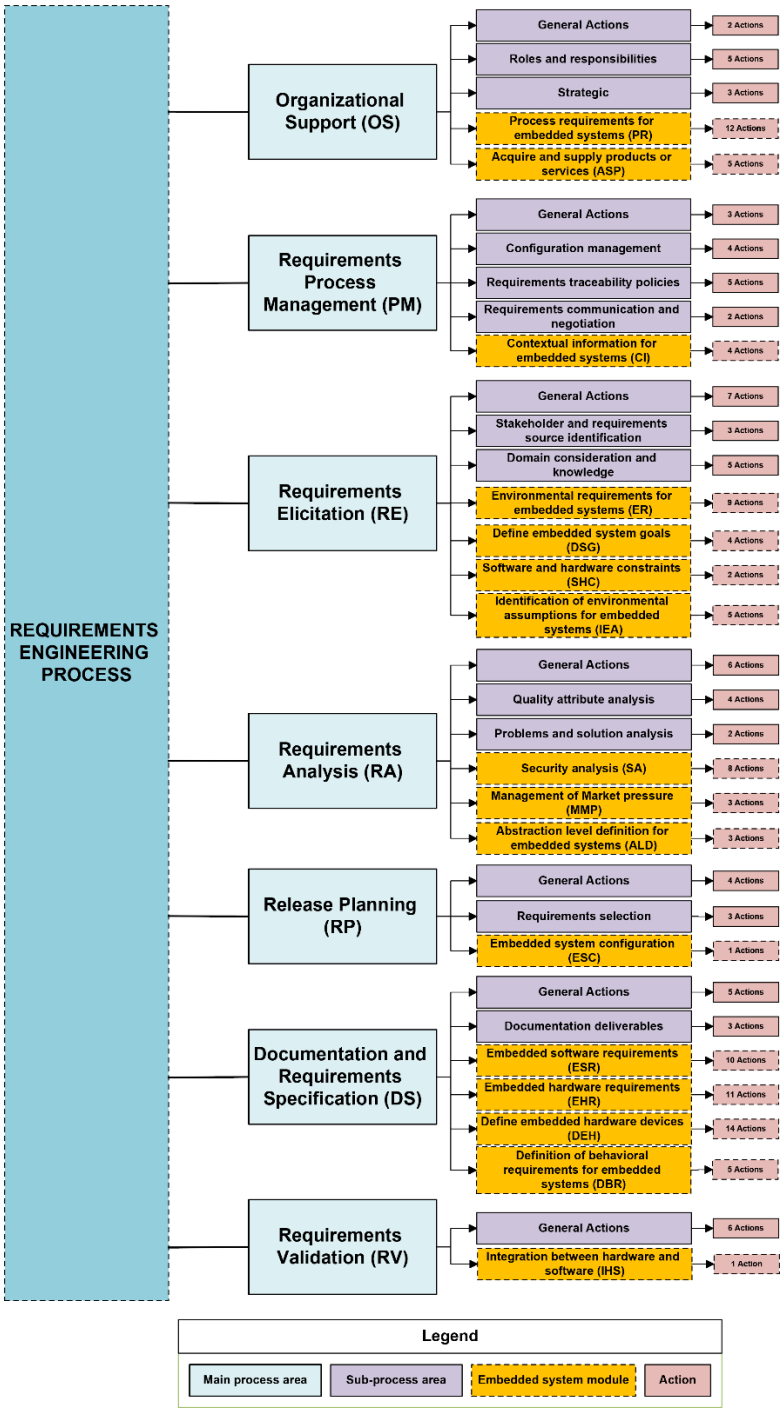

Fig. 1. Embedded system module and its relationship with Uni-REPM

(OS.ASP) Acquire and supply products or services: is responsible for managing the acquisition and/or supplying products or services from/for other companies.

(PM.CI) Management of contextual information: handles the identification and specification of the contexts that can affect the system operation.

(EL.DSG) Define System Goals: has practices defining and describe the embedded system goals.

(EL.SHC) Software and Hardware Constraints: contains actions to provide a set of software and hardware constraints to ensure that the software and hardware components will be in accordance with the architecture, design, or implementation constraints.

(EL.IEA) Identification of Environmental Assumptions: addresses the identification of assumptions about the environment in which the system will operate.

(RA.SA) Security Analysis: handles the identification and management of risks and failures to maintain product integrity and reliability.

(RA.ALD) Abstraction Level Definition: aims at the definition of the model abstraction level that will be produced at each sub-process.

(RA.MMP) Management of Market Pressure: has actions to investigate and manage the consumer changes, environmental, and regulatory needs.

(DS.ESR) Embedded Software Requirements: handles the specification of software requirements for embedded systems.

(DS.EHR) Embedded Hardware Requirements: addresses the identification and specification of a set of hardware devices to be used in the development of an embedded system.

(DS.DEH) Define Embedded Hardware Devices: has actions to identify and define a set of hardware devices required by the embedded system such as sensors, actuators, external interfaces, and hardware adapter.

(DS.DBR) Definition of Behavioral Requirements: contains practices to identify and specify the software and hardware behaviors of the system.

(RV.IHS) Integration Between Hardware and Software: the purpose relies on the system integration, which verifies the interaction among software components and the communication interfaces between the software and hardware.

Maturity level view. There are three levels of maturity: Basic, Intermediate, and Advanced. These levels represent how mature the requirements engineering process is at an organization. The benefits are achieved from moving from one level to another one.

Each action has a maturity level considering its essentiality and required skill/cost to carry out. Dependencies among actions were also considered. When an action A depends on action B to be performed, its level must be the same or higher than action B. The maturity levels are described below:

(1) Basic: it contains basic actions in an established and repeatable Requirements Engineering Process.
(2) Intermediate: the RE process at this level involves different roles and responsibilities. Additionally, the process is driven by product strategies and goals.
(3) Advanced: it is the most mature RE process. The procedures are well established, and concern about future work is essential (e.g., reusable materials, evaluation terms, etc.).

Module Contents. Uni-REPM ES module has seven process area, fifteen sub-process area, and eighty-nine actions. In Table 1, we present a process area of the Uni-REPM ES module called Documentation and Requirements Specification (DS). For the sake of space, it is not possible to include all sub-processes and actions of the module. Hence, we only outline the actions of the Embedded Hardware Requirements (HER) sub-process.

Table 1. Uni-REPM ES module overview

ID	Description
DS	Documentation and Requirements Specification
DS.EHR	Embedded Hardware Requirements
DS.EHR.a1	Specify and document a set of hardware architecture standards for the domain of the embedded system that will be developed
DS.EHR.a2	Elicit and specify a set of non-functional requirements that the hardware devices must fulfill
DS.EHR.a3	Provide an overview of the hardware components to be used in the development of an embedded system
DS.EHR.a4	Elicit the mechanical requirements (e.g., dimensions and materials) for each identified hardware component
DS.EHR.a5	Elicit electrical requirements for each identified hardware component
DS.EHR.a6	Identify and document the microcontroller of the embedded system based on the results of previous actions
DS.EHR.a7	Specify and document a set of additional hardware devices based on the results of previous actions
DS.EHR.a8	Perform hardware analysis. The goal of this analysis is to select the hardware devices based on the business goals correctly, and the quality attributes, also known as non-functional requirements, to be achieved by the embedded system
DS.EHR.a9	Elicit a set of manufacturing requirements
DS.EHR.a10	Perform traceability between the functionalities of each hardware device and the definition of behavioral requirements described in process area DS.DBR

Module Usage. The goal of Uni-REPM ES is to assess the maturity of the requirements engineering process for embedded systems. Thus, it can be used by stakeholders who are involved in the process, deeply understand it, and oversee process improvement in general. Examples of users are requirements engineer, software engineer, hardware engineer, mechanical engineer, electrical engineer, quality assurance engineer, project manager, and product manager.

The user should perform a mapping from the actions present in the model to the activities in his/her real process to assess the maturity of the RE process. For each action, the evaluator can select one of the three options: (IC) Incomplete - the action was deemed vital but was performed partially or not all in this RE process; (C) Complete -

the action was completed in this RE process; and (IA) Inapplicable - the action was not necessary or possible to be executed in this process.

The Uni-REPM ES maturity level is akin to the ones adopted by Uni-REPM [23] and Uni-REPM SCS [24] : for the MPA to achieve a specific level, all applicable actions of the level must be completed.

3.3 Example of Definition of Action

In the Uni-REPM ES module, actions also follow the same format assigned to sub-processes to define their unique identifiers. Actions are identified by the MPA/SPA under which they reside, followed by an "a" which stands for "action" and their position in the group. For example, "RV.IHS.a1" means the first action under MPA "Requirements Validation (RV)" and SPA "Integration between hardware and software (IHS)". Besides the description of each action, there can be examples. They give practitioners ideas on what the action is requesting.

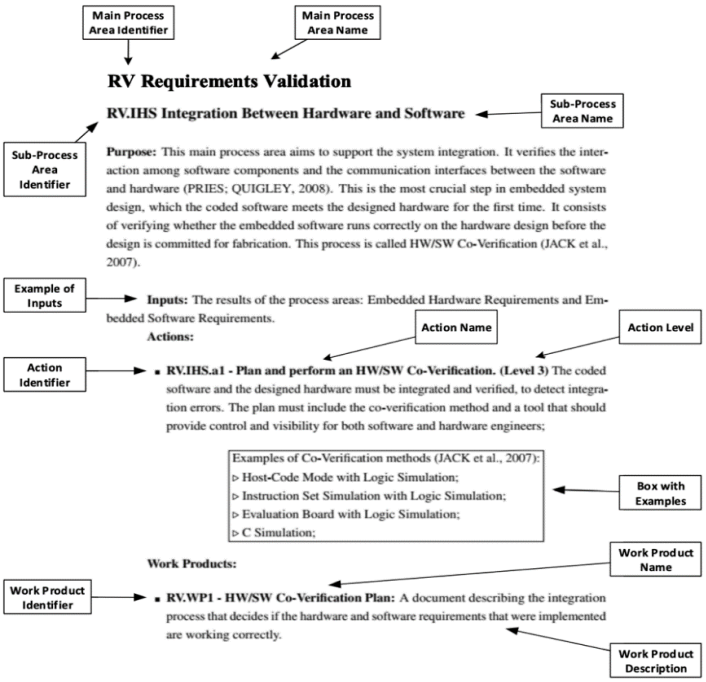

Fig. 2. A snapshot of an Uni-REPM ES action

Another example of a sub-process is the EL.SHC Software and Hardware Constraints that was inspired by some concepts present in the METAMODEL (Software, Hardware, Software Requirements, and Hardware Requirements), as well as the STATE-OF-THE-ART and some standards.

Studies in the STATE-OF-THE-ART category highlight the constraints on static requirements and over the system behavior. Additionally, authors in [3] stated that embedded systems are not related only to functional constraints but also to real-time constraints [3,23]. When it comes to hardware, constraints are associated with physical components such as circuit safety constraint [5,6].

In the REQUIREMENTS-STD category, the ISO/IEC 12207 [12] requires the transformation of implementation constraints into actions to satisfies the system requirements. Moreover, ISO/IEC 15289 [15] and 29148 [14] covers the description of constraints that affect the system under development. In terms of interface constraints, ISO/IEC 15288 [13] aims the definition of mechanical, electrical, mass, and data constraints.

4 Validation

We are planning to perform a static validation of the module applying survey and interviewing domain experts from different embedded systems areas. Additionally, we also intend to perform a dynamic validation to evaluate the embedded system module in case studies.

The goal of the static validation is to collect expert's opinion to assess the proposed module regarding its coverage, correctness, usefulness and applicability. Thus, we can get early feedback that helps to identify potential problems and improve the module for a dynamic validation [8].

The interviews will be conducted face-to-face and the survey will rely on self-administered questionnaires sent by email. Our aim is that the subjects provide their opinion about the SPAs, actions, and their maturity levels to validate its accuracy and adequacy. Our process for static validation is been inspired in the work of [17] and [8], and consists of the following steps:

(1) Problem Statement. The background and problem need to be clearly articulated and defined;
(2) Selection of Experts. A fair number of experts need to be identified based on a set of criteria. The set of criteria should include the credibility, knowledgeability, and dependability of experts;
(3) Elicitation of Opinions. This step poses the right question and ensures conditions for an elicitation process;
(4) Aggregation of Opinion. The basic idea here is to reach an aggregated opinion or a consensus based on which a decision can be made; and
(5) Decision Making. This step makes the decision based on the aggregated opinion.

After the static validation, we will perform a dynamic validation. At this stage, the goal is to conduct case studies to investigate the application of Uni-REPM ES in a real-life context, evaluating the maturity level of embedded system processes in RE. Thus, it will be possible to get feedback about the ES module regarding its coverage, usefulness, and applicability [8].

We intend to select embedded systems companies from different domains to participate in dynamic validation. Each company should designate one person responsible by the validation. Thus, it will be possible to evaluate the maturity level of different ES projects using the Uni-REPM ES module. The main objective will be: "Analyze the requirements engineering processes for embedded systems of companies that develop embedded systems. With the purpose of investigate the maturity level of their projects. In the point of view of Requirements Engineer, Software Engineer, Hardware Engineer, Project Manager and Product Manager. In the context of a view of embedded systems development."

The selected subject responsible by the embedded system project will be asked to answer an assessment instrument which has questions related to the 89 RE actions for ES. When answering the actions, one of the following situations may be found: - the action was deemed vital but was performed partially or not all in this RE process. It should be marked as "Incomplete" (IC); - the action was completed in this RE process. It should be marked as "Complete" (C); and - the action was not necessary or possible to be executed in this process. It should be marked as "Inapplicable" (IC). After the maturity level evaluation, the subjects will be asked to answer questions regarding their feedback about the ES module. Then, we will perform data analysis to check whether the dynamic validation objective was attended.

We emphasize that the actions presented in our module were based on an extensive systematic literature review (SLR), international requirements engineering standards (ISO/IEC, IEEE Std), metamodel concepts, and adopted embedded systems books and reports. However, they will be probably revised after the static and dynamic validation, to take in consideration the new information acquired from practitioners and case studies from different embedded systems domains.

5 Conclusions and Future Work

This paper describes a requirements engineering maturity module for embedded systems which is an extension to Uni-REPM [23]. Our goal is to provide a guide for companies improve the maturity of their processes in terms of an evolutionary path from chaotic and eventual processes towards mature and disciplined software processes.

The main contributions of this paper are the following:

- Benefits to academia: The embedded system module is based on an extensive systematic literature review and industry requirements engineering standards. Thus, the module is a summary of the state of the art by providing the identification and systematization of embedded systems practices.
- Benefits to industry: The module provides a process evaluation model for requirements engineering practices in the development of embedded systems. Additionally, it can guide requirements engineers to develop better embedded systems by assessing their maturity and improving the completeness and correctness of the embedded system elicitation and specification.

- Embedded systems issues: Through the development of the Metamodel for Embedded Systems (MM4ES) [20], it was possible to identify the main embedded systems concepts. These concepts contributed to the identification of relevant requirements engineering actions to be included in the module. Thus, the module provides a way to evaluate sub-process areas (SPAs) related to requirements engineering for embedded systems.
- Identification of weak areas for improvements: The embedded system module has a measurement instrument to support engineers and managers to evaluate the current status of an organization. Hence, it helps to identify weak areas for improvements, allowing a better decision-making process. Besides, the evaluation can serve as a basis for the creation of a roadmap for improving the organizational processes.
- The Module can be used in different embedded system domains: The studies from our systematic literature review we used to create the module are from different areas such as avionics, ground transportation, automation, medical, and energy technology. Hence, some practices maybe not appropriate for a specific organization and need never be implemented. In these cases, they are considered "Inapplicable". Thus, the module is suitable for embedded systems companies of different types and sizes.
- Module does not require you to use a specific technique or process: The module focuses on what can be done to achieve the maturity of processes, without depicting how to perform the activities. Hence, the module is independent of any technology such as a tool, RE modeling languages, software and hardware development methodology, and programming languages.

We are currently planning a static validation to assess the proposed module regarding its coverage, correctness, and usefulness and applicability. We also intend to perform a dynamic validation to evaluate the embedded system module in case studies.

This work has generated some research directions that we intend to address in future works:

(1) What is the effect of applying Uni-REPM ES when it is instantiated in different embedded systems domains?
(2) How to evaluate whether the module has a sufficient complete coverage of actions?
(3) How to integrate the Uni-REPM ES with the existing standards/practices in organizations?
(4) How to integrate the Uni-REPM ES with the existing software standards?
(5) What is the level of acceptance of Uni-REPM ES by practitioners?
(6) What are the features of a software tool to support the module adoption?
(7) How to develop a CASE tool to support the module?
(8) How the viewpoint of evaluators could be considered when determining the overall maturity level?
(9) How to integrate safety concerns in the present module?

(10) What is the module impact considering before/after assessment of RE practices in organizations?

6 Acknowledgments

We would like to acknowledge that this work was partially supported by Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), Fundação de Amparo à Ciência e Tecnologia do Estado de Pernambuco (FACEPE), and Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES).

References

1. Bate, R., Kuhn, D., Wells, C., Armitage, J., Clark, G.: A systems engineering capability maturity model, Version 1.1. Technical Report, DTIC Document (1995).
2. Bechaam, S., Hall, T., Rainer, A.: Defining a requirements process improvement model. Software Quality Journal, v. 13, pp. 247–279 (2005).
3. Brink, K., van Katwijk, J., Spelberg, R. L., & Toetenel, W. J.: Experiences with analysis of formal specifications in Astral. In: Proceedings of the Fourth International Workshop on Real-Time Computing Systems and Applications, pp. 143-150. IEEE (1997).
4. Broy, M., Stauner, T.: Requirements engineering for embedded systems. It – Information Technology, v. 41, pp.7-11 (1999).
5. De Lemos, R., Saeed, A., Anderson, T.: A train set as a case-study for the requirements analysis of Safety-Critical Systems. Computer Journal v. 35, pp. 30-40 (1992).
6. Doering, D., Pereira, C.E., Denes, P., Joseph, J.: A model driven engineering approach based on aspects for high speed scientific X-rays cameras. In: IEEE 16th International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing (ISORC), pp. 1-8. IEEE (2013).
7. Gibson, D. L., Goldenson, D. R., Kost, K.: Performance results of CMMI-based process improvement. Technical Report, Carnegie-Mellon Univ Pittsburgh Pa Software Engineering Inst (2006).
8. Gorschek, T., Garre, P., Larsson, S., Wohlin, C.: A model for technology transfer in practice. IEEE software v. 23, no. 6 pp. 88–95, (2006).
9. Gorschek, T., Gomes, A., Pettersson, A., Torkar R.: Introduction of a process maturity model for market-driven product management and requirements engineering. Journal of software: Evolution and Process v.24, no.1 pp. 83-113 (2012).
10. IEEE Std 1233: Guide for Developing System Requirements Specifications (1998).
11. IEEE Std 830: Recommended Practice for Software Requirements Specifications. (1998). https://doi.org/10.1109/IEEESTD.1998.88286.
12. ISO/IEC Std 12207: Standard for Systems and Software Engineering - Software Life Cycle Processes. (2008).
13. ISO/IEC/IEEE Std 15288-2008: Systems and Software Engineering - System Life Cycle Processes. (2008), c1–84. https://doi.org/10.1109/ IEEESTD.2008.4475828.
14. ISO/IEC/IEEE 29148:2011(E): Systems and software engineering – Life cycle processes – Requirements engineering. (2011), 1–94. https://doi.org/10.1109/IEEESTD.2011. 6146379.

15. ISO/IEC/IEEE 15289:2015(E): International Standard Systems and software engineering – Content of life-cycle information items (documentation). (2015), 1–95. https://doi. org/10.1109/ IEEESTD.2015.7109795.
16. Kotonya, G., Sommerville, I.: Requirements engineering: processes and techniques. Wiley Publishing (1998).
17. Li, M., Smidts, C. S.: A ranking of software engineering measures based on expert opinion. In: IEEE Transactions on Software engineering v. 29, no.9 pp. 811-824 (2003).
18. Nasr, E.: Challenges in Requirements Engineering for Embedded Systems. In Requirements Engineering for Sociotechnical Systems. IGI Global, 21–36 (2005).
19. Pereira, T., Albuquerque, D., Sousa, A., Alencar, F. M., Castro, J.: Retrospective and Trends in Requirements Engineering for Embedded Systems: A Systematic Literature Review. In CIbSE pp. 427-440 (2017).
20. Pereira, T., Sousa, A., Silva, R., Albuquerque, D., Alencar, F., Castro, J.: A metamodel to guide a requirements elicitation process for embedded systems. In 2018 11th International Conference on the Quality of Information and Communications Technology (QUATIC), pp. 101-109. IEEE (2018).
21. SEI/CMU. CMMI for Development, Version 1.3, Improving processes for developing better products and services. Software Engineering Institute (2010).
22. Sikora, E., Tenbergen, B., Pohl, K.: Industry needs and research directions in requirements engineering for embedded systems. Requirements Engineering v. 17 no.1, pp. 57-78 (2012).
23. Svahnberg, M., Gorschek, T., Nguyen, T. T. L., Nguyen, M.: Uni-REPM: a framework for requirements engineering process assessment. Requirements Engineering v. 20, no. 1 pp. 91–118 (2013).
24. Vilela, J., Castro, J., Martins, L. E. G., Gorschek, T.: Safety Practices in Requirements Engineering: The UniREPM Safety Module. IEEE Transactions on Software Engineering, v. 43, n. 3, 222-250 (2020).
25. Wendler, R.: The maturity of maturity model research: A systematic mapping study. Information and software technology v.54, n.12 pp. 1317-1339 (2012)