A process framework for embedded systems engineering

Sofia Charalampidou, Apostolos Ampatzoglou, Paris Avgeriou
Department of Mathematics and Computer Science, University of Groningen
Groningen, The Netherlands
s.charalampidou@rug.nl, a.ampatzoglou@rug.nl, paris@cs.rug.nl

***Abstract*— The engineering of embedded systems is considered highly complex, due to the need for integrating multi-disciplinary, multi-lifecycle, multi-site, and multi-organization approaches. However, to the best of our knowledge, such domain specific challenges have not been collectively addressed until now in existing engineering processes. This study proposes a process framework, i.e. a set of elements and usage guidelines for process instantiation, in the domain of embedded systems. The proposed framework has been validated through two focus groups with experts on embedded system processes.**

Keywords—process; process framework; embedded systems; focus group

I. INTRODUCTION

Embedded systems engineering deals with a number of highly complex challenges, which, until now, have not been sufficiently addressed by process support. Specifically, there is a need for integrating multi-disciplinary, multi-lifecycle, multi-site, and multi-organization approaches, due to domain-specific characteristics.

First, the integration of different disciplines in a single system (e.g. development of software, hardware, mechanical and/or electronic components) hinders the applicability of traditional models, like waterfall and agile processes. Furthermore, embedded systems are usually partitioned across disciplines and also within each discipline into several components, which can be engineered separately. This in turn results in a different lifecycle for each component. However, multi-lifecycle engineering is not sufficiently handled by any known process framework. In addition to multiple disciplines and lifecycles, engineering of embedded systems often takes place in several sites and different organizations. Potential challenges that are introduced due to this type of engineering are the use of different processes, technologies and terminology among the different sites and organizations. These aspects need to be taken into account in the overall engineering process.

In this paper we propose and validate a process framework for the domain of embedded systems, by taking into account the abovementioned domain-specific challenges. A process framework is a generic model-based mechanism, for creating project-specific process instances. The proposed process framework, from now on referenced as *PROMES Process Framework*, has been developed in the context of the PROMES project[1]. According to [1], the definition of a process framework consists of three parts: (a) a meta-model defining the fundamental types of process elements, and how they inter-relate, (b) usage guidelines on how to select, integrate, and customize process element instances from the repository to produce project-specific processes, and (c) a repository of process element instances (actual descriptions of each instance). The definition of the PROMES Process Framework will follow these three parts. However, in this paper we will only cover the first two, due to size limitations. We note that the introduction of this framework is not a holistic solution to the aforementioned challenges, but it provides a starting point for handling them in the domain of the embedded systems.

In section II we will present related work, and discuss why existing process frameworks do not fulfil the abovementioned challenges. Next we will present the PROMES Process Framework (section III) and an illustrative example (section IV). The proposed framework has been validated through two focus groups, whose results will be presented in section V. Finally, in section VI we will sum up our contribution and present plans for future work.

II. RELATED WORK

To the best of our knowledge, in the literature there are no process frameworks in the domain of embedded systems. However, as related work we have considered the IEEE International Standard [4] that provides guidelines on defining system level lifecycle processes, and a specialization of the Rational Unified Process (RUP) [7] for Systems Engineering (RUP-SE)[10].

First, the 15288:2008 IEEE Standard [4] defines a set of default process element instances (actual descriptions of each instance) that can be used in the engineering of hardware / software systems. More specifically, the standard defines four process groups, i.e. Agreement Processes, Organizational Project-Enabling Processes, Project Processes and Technical Processes, in which specific processes are classified. Then, for each process the Standard defines the purpose, the outcomes, and the activities and tasks. In addition to that, the Standard states that new processes can be added to the process groups in order to extend the proposed system development lifecycle.

[1] The PROMES project is an ITEA2 project that aims at delivering process models for engineering embedded systems. The PROMES consortium consists of various research and industrial partners from Finland and the Netherlands, specialized in embedded systems. http://promes-itea2.eu/

However, the proposed default activities are quite generic and do not provide guidelines in terms of how they can be applied in specific process instantiation. In addition to that, the Standard does not explicitly provide a configuration mechanism for removing or adding an activity from the predefined set of processes. Furthermore, the sequence of the performed activities, i.e. iterative or waterfall, is not described. Thus, the Standard only provides the third part of process frameworks definition [1], i.e. a repository of process element instances and omits the first two parts (meta-model and usage guidelines), since they are considered out of its scope and intend.

Second, the RUP-SE, is a derivative of the Rational Unified Process, which makes use of the RUP Process Framework. RUP-SE instantiates new artifacts and applies modifications of default RUP disciplines and roles, so as to support the creation of such artifacts in system level [10]. The process is organized in four phases (inception, elaboration, construction and transition), which are split in multiple iterations. In each phase, multiple activities are performed. The activities are organized in workflows, and each workflow is assigned to a different effort during each phase, represented in the form of a hump. Both RUP and RUP-SE are controlled iterative processes that reflect all the benefits of iterative processes, like the possibility to achieve a robust architecture by applying iterative corrections, refining the engineering process and mitigating early any potential risks [7, 10].

The main point of advancement of the PROMES Process Framework, with respect to RUP-SE and 15288:2008 IEEE Standard, is its explicit support for addressing the aforementioned challenges of embedded systems, namely multi-disciplined and multi-lifecycle engineering in a multi-organizational and multi-site context.

III. PROMES PROCESS FRAMEWORK

In this section, we present the PROMES Process Framework. In section III.A we describe the used meta-model, i.e. available types of process elements and their relationships. In section III.B, we present guidelines on using the PROMES Process Framework.

A. *Meta-Model*

In Fig. 1, we present the meta-model of the *PROMES Process Framework*, to which all instances of the process elements should conform. In the Figure, *AbstractLifecycle, ControllableElement, Control* and *DevelopmentActivity* are denoted as abstract elements. We note that we do not consider the meta-model as a variability dimension of the PROMES Process Framework: the PROMES Process Framework users are not expected to add, remove or modify any elements of the meta-model itself. However, in cases that extensions on the meta-model are required, we suggest the creation of profiles [8]. The use of profiles is supposed to allow refinement of the meta-model elements, while preventing potential contradictions to the standard model. Thus, generally it is possible to add, remove or modify only instances of the elements, i.e. process components, unless a profile that extends the meta-model is created.

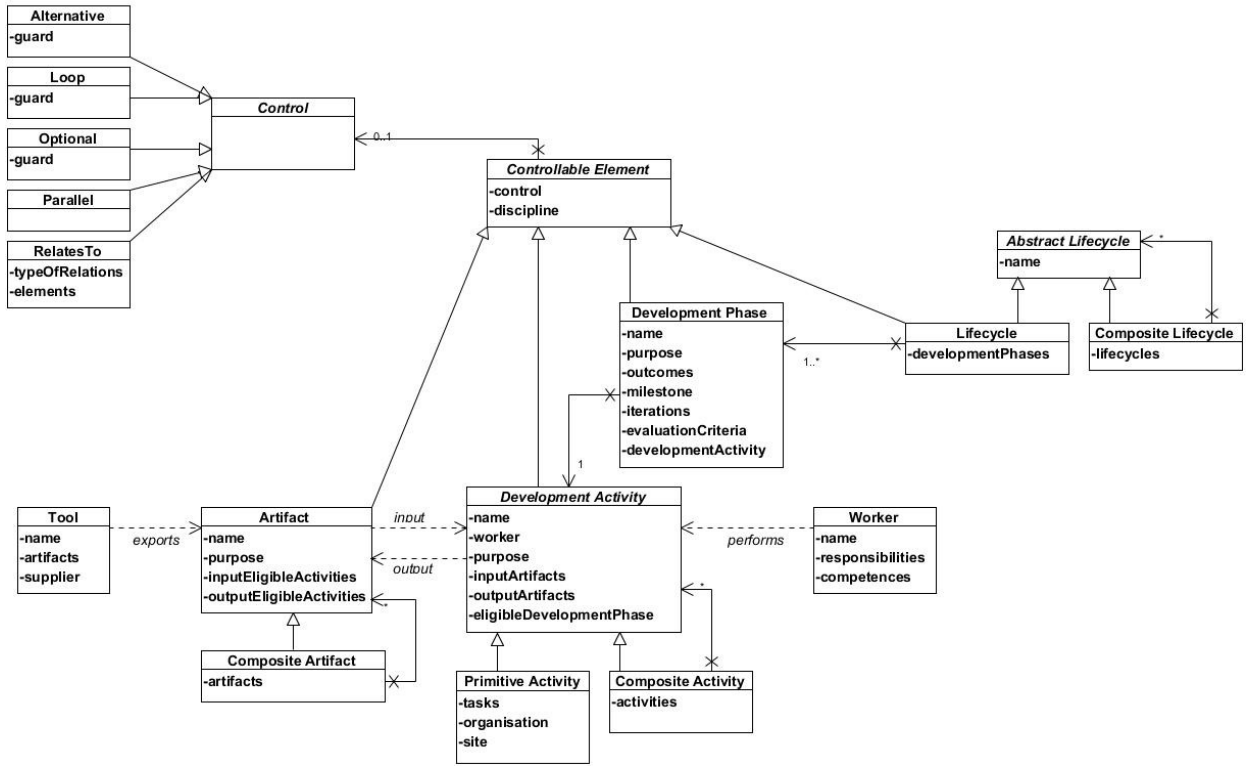

Fig. 1. PROMES Framework Meta-model

The elements presented in the meta-model are described below:

- **Abstract Lifecycle**: Corresponds to the life cycle of a project, which starts with the beginning of a project, continues while the set of activities defined in the process framework are under progress and ends when the project has come to an end [10, 4].
- **Lifecycle**: The lifecycle of this type is "simple", since it does not include other lifecycles. As an example we could consider the waterfall lifecycle, which consists of five *Development Phases*, i.e. requirements, design, implementation, verification and maintenance.
- **Composite Lifecycle**: Corresponds to the lifecycle that includes the *Lifecycles* of sub-projects. As example we could consider a project lifecycle, which encapsulates a software and a hardware lifecycle.
- **Development Phase**: It is a subdivision of the *Lifecycle*, during which a unique major goal has to be fulfilled and a number of *milestones* are set to ensure its achievement [7]. An example of a development phase, borrowed by RUP, is the "inception phase".
- **Development Activity**: A development activity, either primitive or composite is related to a *Development Phase*. In order to complete the activity, some *input artifacts* are required and some *output artifacts* are generated [7]. A development activity is an abstract element, which cannot be instantiated in any process, but only acts as an interface for *Primitive* and *Composite Activities*.
- **Composite Activity**: A set of *Primitive* or *Composite Development Activities*. An example of a composite activity is software architecting, that according to [2] consists of architectural analysis, synthesis and evaluation.
- **Primitive Activity**: A list of *tasks* that should be accomplished, in order to achieve the completion of the goal [9]. Based on the abovementioned example, a primitive activity is "architectural analysis".
- **Controllable Element**: An abstract element that acts as an interface for all elements that can contain *Controls*.
- **Control:** An abstract element that corresponds to any kind of control flow [8] that specifies the sequence of executing *Lifecycles, Development Phases, Development Activities,* and *Artifacts,* or to any kind of relationship among elements of the same type.
- **Alternative / Loop / Optional:** *Control* elements for specifying a set of alternative / iterative / optional *Lifecycles, Development Phases, Development Activities, or Artifacts.*
- **Parallel**: A *Control* element for specifying a set of *Lifecycles, Development Phases,* or *Development Activities* that are executed in parallel.
- **Related To**: A control element for creating links among a set of related *Lifecycles, Development Phases, Development Activities, or Artifacts.*
- **Worker**: The responsibilities that an individual or a team has, that are highly related to a set of *Development Activities* and the way that they should be performed [9]. As an example of a worker we can consider a Software Architect or a Tester.
- **Artifact**: A piece of information that is produced, modified, or used by a process, while targeting the final product [9]. The artifacts are used/produced by a *Development Activity/task* as input/output data. They are usually diagrams. An example of an artifact is the class diagram.
- **Composite Artifact**: Corresponds to artifacts that consist of a collection of *Artifacts*. The artifacts are used/produced by an *Development Activity/task* as input/output data. They are usually documents. An example of a composite artifact is the Software Design Document (SDD), which contains many diagrams (among which a class diagram, which was used as an example above).
- **Tool**: The set of tools that are expected to be used for supporting the *Development Activities* of the process framework. [9]. An example of a tool that could be used for modeling UML diagrams is Rational Rose.

The proposed process framework builds on the meta-model of the Rational Unified Process (RUP), i.e. a well-known process framework [7]. The basic points of deviation between the RUP meta-model and the PROMES meta-model are:

1) the use of a *Composite Lif*ecycle element. It aims at providing the ability to tackle the multi-lifecycle challenge. The idea is that there are several cases, when a project lifecycle is a complex entity consisting of different *Lifecycles*. Such cases are introduced, when a project consists of multi-disciplinary parts, each requiring a different lifecycle.
2) the use of an *Abstract Lifecycle* element. The use of this element is necessary, since a *Composite Lifecycle* can include either *Lifecycle* or *Composite Lifecycle* elements. Thus, in order to keep one list for both items, we created an additional process element type, which acts as an interface between them.
3) the merge of the *Activity* and *Workflow* elements into the *Development Activity* element. This merging aims at providing our model the flexibility to describe tasks, which can be decomposed to more than three levels. In RUP, the workflows, are decomposed to activities, which are further decomposed to tasks. In this meta-model, we use *Composite Activities, Primitive Activities,* and *tasks*, so as to create workflow hierarchies of any possible level of decomposition.

4) the use of a *Composite Artifact* element. RUP proposes three different levels of jobs that can be done by *Workers*, i.e. workflows, activities and tasks. According to their type and granularity, these jobs can produce artifacts that can be categorized as primitive (in this meta-model simply named *Artifacts*), i.e. code, diagrams, charts, tables, etc. and as *Composite Artifacts* (i.e. artifacts that can contain others), such as documents, software items, etc. Having modeled both types of artifacts with the same element, either this information would be lost (no self-reference in the model) or would lead to an inconsistent representation (self-reference in the artifact element in the model), in the sense that artifacts do not contain other artifacts.

5) the inclusion of a *Control* element. UML Specification [8] proposes 8 combined fragments for controlling the flow of occurring actions, e.g., alternatives, iterations, etc. In this framework we have defined five controls which are responsible for handling alternative, optional, parallel and iterative controllable elements. Additionally, we have included the *Related To* element, which is responsible for creating links among elements, by defining their relationship (e.g. refinement, generalization).

6) the use of a *Controllable Element*. An interface for the types of elements that can be controlled by *Control* elements (i.e. *Lifecycles, Development Phases, Development Activities,* and *Artifacts*).

Based on the proposed meta-model, the embedded system specific challenges, i.e. multi-lifecycle, multi-discipline, multi-site and multi-organization engineering, are handled in the meta-model as presented in Table I.

TABLE I. EMBEDDED SYSTEMS CHALLENGES IN METAMODEL

Challenge	Element/Attribute	Description
Multi-lifecycle	Composite Lifecycle	Different lifecycles are nested inside composite lifecycles.
Multi-discipline	Controllable Element::discipline	All lifecycles, development phases, or development activities are assigned to a discipline.
Multi-Site	Primitive Activity::site	The development activities are linked to the site where they are performed.
Multi-Organization	Primitive Activity::organization	The development activities are linked to the organization where they are performed.

B. PROMES Process Framework usage Guidelines

In this section we provide guidelines on producing project-specific process instances by selecting and integrating different instances of process elements, guidelines for customizing process element instances, and guidelines for checking the validity of process instances.

1) Producing Project-Specific Process Instances

In this section, we describe the basic steps that should be followed for creating project-specific process instances, using the PROMES Process Framework (see Fig. 2).

In order to create a new process instance first of all the project-specific process components should be specified. Then a search in the PROMES process framework repository will indicate if these components already exist and can be reused. In case that any of the components do not exist, they should be created and added to the repository for future use. The new components can be either created from scratch or after customizing similar existing components. The different process components can be integrated in a *Lifecycle*, i.e. a set of *Development Phases*, then a set of *Development Activities* within each development phase, etc. Concepts like the parallelization of development activities, development phases and lifecycles, and the handling of alternative scenarios, loops etc. are configured by the use of *Control* elements. As next step, it is important to guarantee the consistency of interfaces between successive development phases, successive development activities and tasks. Also, the composition and consistency between semantically related process components (development phases and development activities, and development activities and artifacts) should be checked. Then it is important to ensure that different lifecycles, development phases and development activities are synchronized. If at any point any of the aforementioned checks fail, the process creation is cancelled.

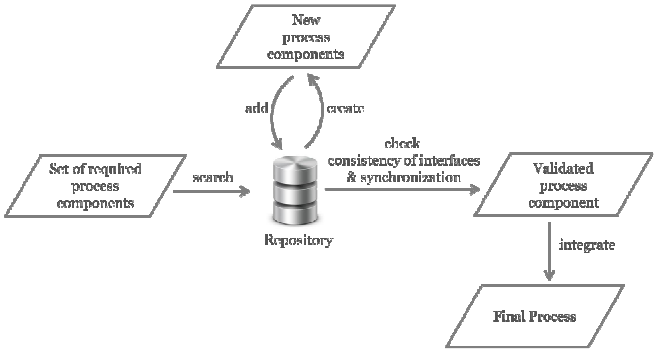

Fig. 2. Basic flow for creating project-specific processes from scratch.

2) Customizing the Process Element Instances

The process element instances within the PROMES Process Framework can be customized through three variability mechanisms:

Replacing process element instances of the same type. This customization dimension enables process engineers to replace parts of process element instances with others of the same type, e.g. a different process can be created, by replacing the artifact "Communication Diagram", with the artifact "Sequence Diagram" as the *output artifact* of the *Development Activity* "Represent the dynamic behavior view of a system". This customization dimension is applicable for all *Controllable Elements*. If a process engineer wants to provide the ability to a specific set of process components to be mutually exclusive when executed, the *Alternative Control* element can be used. However, this approach can parameterize the system only with a specified number of alternatives.

Adding or removing process elements instances. This customization mechanism allows the process engineers to add or remove parts of process element instances. The most common process element types that are expected to be taken into account in this customization dimension are *Lifecycles, Development Phases*, and *Development Activities*. We note, that in the case of removing a process element, other process elements will cascade on delete, e.g. if a process engineer decides to delete the "Detailed Design" *Composite Development Activity*, then the *Primary Activity* "Represent the dynamic behavior view of a system" will be automatically deleted. If a process engineer wants to provide the ability to a specific process component to either be executed in runtime or not, the *Optional Control* element can be used.

Lean vs. rigorous. This variability mechanism deals with how strict the methods and techniques used in all development phases are. This variability mechanism is expected to be more suitable for customizing *Development Phases, Activities,* and *tasks*. For example, in the context of *Artifacts*, this variability dimension could deal with the amount of documentation artifacts produced in the complete software lifecycle. Thus, a rigorous documentation-based approach for software quality assurance, e.g. based on 730:2002 IEEE Standards for Software Quality Assurance Plans [3], would lead to the creation of larger documentation artifacts rather than those that would be produced by a more lean / agile software engineering paradigm. If a process engineer wants to provide the ability to a specific set of process components to be mutually exclusive when executed, the *Alternative Control* element can be used (one alternative for lean activities and another for more rigorous activities).

3) Checking the Validity of Process Instances

The processes that can be created by using the PROMES Process Framework should be checked for validity during their creation or modification. During the validation of a process instance the process engineer should perform three checks: (a) check the composition of process elements (see Section III.B.3.a), (b) check the synchronization of process elements (see Section III.B.3.b), and (c) check the consistency between successive process elements (see Section III.B.3.c). In order for the inspected process instance to be valid, all checks should be successfully completed.

a) Composition Validation

The composition validation mechanism of process elements, within the PROMES Process Framework, can be achieved by using several attributes. The mechanism is a two-step process that: (a) checks the validity of aggregating process element instances inside a container element (e.g., a *Development Activity* inside a *Development Phase*), and (b) checks if the set of process element instances that have been aggregated inside the container collaborate sufficiently in order to capture the goal of the container element.

Concerning (a), the attributes *inputEligibleActivities*, *outputEligibleActivities*, and *eligibleDevelopmentPhase*, are used in order to achieve consistency, while composing different types of components: for example, between a *Development Phase* and the *Development Activities* that it consists of, or between the *Development Activities* and the *Artifacts* that they use as input or produce as output. In the case of a development phase and its development activities, we use the attribute *eligibleDevelopmentPhase* to specify in which development phase each development activity is expected to belong, while in the case of the development activities and their artifacts, we use the attributes *inputEligibleActivities* and *outputEligibleActivities* to specify in which activities each artifact is expected to be used as input or output respectively.

Concerning (b), we suggest to perform a number of additional checks, for ensuring the proper collaboration of elements contained into another element, so as to achieve a common goal. First of all, regarding the relationship between a *Development Phase* associated with a *Development Activity*, we have to ensure that all *output artifacts* of the development activity contribute to meet the *outcomes* of the development phase. In addition, these output artifacts should be able to be used for evaluating the successful completion of the development phase. Secondly, with respect to the completeness of a *Development Activity*, we should ensure that the existing *tasks* of a development activity cover sufficiently the whole set of *output artifacts* and that the created artifacts fulfill the *purpose* of the activity. A third set of checks should concern the existence of available and suitable tooling that can generate all the needed artifacts. Likewise, the compatibility of the *Tools* that we plan to use should be checked, so that *input artifacts* to be compatible with the tools that will use them as input. Regarding the use of the *discipline* attribute, a check is required for ensuring that each element is assigned to a discipline only once. Thus, if we have a case that all development activities of a development phase concern the same discipline, the activities should inherit the attribute *discipline* from the development phase element instead of getting a value that is assigned independently. Finally, the *Worker* element should be validated in order to ensure that a worker covers adequately his/her responsibilities without surpassing them.

b) Synchronization

Different *Lifecycles* in projects with *Composite Lifecycles* should be synchronized. In order to ensure their smooth collaboration, *Development Phases* that are included in the same lifecycle should also be synchronized. This step includes checks on the synchronization on multi-site and multi-organization *Development Activities* and *tasks*. The synchronization of development activities and tasks is related to the availability of *Artifacts* that they use as inputs and outputs. For this purpose we suggest to ensure the availability of *input artifacts* in terms of site location, because in cases of multi-site engineering, the creation of an artifact may not guarantee its availability.

c) Consistent Interfacing

In order to validate the consistent interfacing between sequentially executed elements, the process engineer needs to verify the consistency of collaborating components. First it is necessary to ensure that among sequential *Development Phases*, the *outcomes* of each development phase are useful for an upcoming development phase. Second, with respect to

sequential *Development Activities*, it is important to check that the *input artifacts* of a later development activity are *output artifacts* of a former development activity.

In addition to the abovementioned checks, the process engineer has to perform some inspections for every *Controllable Element* that is associated with an instance of a *Control* element. More specifically, the previous checks have to be made, for three types of controls (*Alternative*, *Optional*, and *Parallel*). For the optional control both possible execution scenarios (the optional controllable element is executed and the optional controllable element is omitted), should be validated. Moreover, while checking the parallelization of *Lifecycles, Development Phases* or *Development Activities*, all *Controllable Elements* that are executed in parallel should be considered as a single controllable element. Attention should be paid to the fact that in the case of *Parallel Development Phases*, the joint *outcome* is the union of the outcomes of each development phase, while in the case of *Parallel Development Activities*, the joint *input* and *output artifacts*, is the union of the input and output artifacts of each development activity.

IV. Illustrative Example

In order to illustrate the applicability of the proposed process framework, we demonstrate how it could be used for modeling a process pattern from the outsourcing domain [5]. This pattern has been selected, because it is simple, it demonstrates the instantiation of several process elements and it supports multi-site engineering (distributes the development activities into different sites).

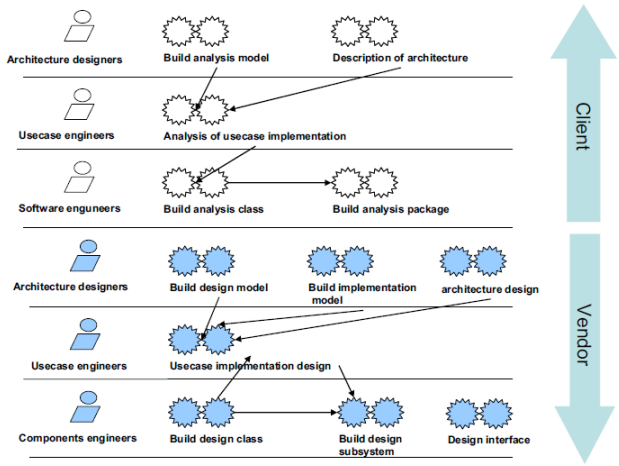

Fig. 3. Technical design outsourcing process pattern [5]

The Technical design outsourcing process pattern, presented in Fig. 3, considers that a client takes responsibility for the analysis of the problem assigning the software solution to the vendor. In the rest of this section, PROMES Process Framework elements are in italics, while the pattern elements are in quotes. In order to model this pattern by using the PROMES Process Framework, we would create a *Lifecycle* named "Technical Design Outsourcing", which would include two *Development Phases*, namely: "Analysis Phase" and "Design Phase". The "Analysis Phase" in turn, would include three *Activities*. The first one, called "Architecting Activity" that is a *Composite Activity*, which consists of two *Primary Activities*, the "Build Analysis Model" and the "Description of Architecture". Both these activities are handled by the "Architecture Designer" *Worker*, in the *site* of the "Client". The "Use Case Analysis Activity" and the "Software Engineering Activity" are modeled in a similar way. Concerning the "Design Phase", all *Primary Activities* are executed at the "Vendor" *site*. Yet again, the *Phase* includes three *Development Activities*: "Architecting Activity" (*Composite Activity*), "Use Case Activity" (*Primitive Activity*), and "Component Engineering Activity" (*Composite Activity*).

V. Focus Groups

In order to validate the requirements of the proposed process framework and its industrial applicability, we conducted two focus groups (*analysis focus group* and *validation focus group*) with experts from the embedded systems domain. Conducting a focus group was considered the most appropriate research method, because we were aiming at investigating / validating some new concepts, getting some additional ideas from experts, and prioritizing the next steps of developing the PROMES process framework [6]. We note that the process framework presented in section III is the resulting framework, after addressing the feedback that we received from both focus groups.

The two focus groups have been designed and reported according to the guidelines defined by Kontio et al. [6]. More specifically, according to [6] conducting a focus group should follow the next steps:

- Plan the focus group
- Design the focus group
- Conduct the focus group
- Analyze the data and report the results

A. Analysis Focus Group

The *analysis focus group* was performed early in the PROMES process framework development, after having designed the initial meta-model of the PROMES process framework, and after the high-level requirements were set. The participants of this focus group were Dutch industrial and research partners.

1) Plan the focus group

While planning the focus group our main goals were: (G1) to establish the framework's terminology, (G2) to identify additional requirements of the embedded systems domain, (G3) to verify the meta-model, and (G4) to brainstorm on the development of a process instance validation mechanism.

2) Design the focus group

Generally, participants in focus groups are either domain experts, or have no experience on the domain. In the case of

the *analysis focus group*, we selected to perform it with experts for three reasons. First, the domain experts have knowledge of the existing domain-specific terminology as well as the requirements of the embedded systems domain. Second, they understand how existing, or even their own, processes work, so as to check if they could map to the meta-model of the proposed process framework. Finally, their experience in the domain could be useful for providing feedback on the process instance validation mechanism. The participants of the first focus group[2] included 4 researchers (2 of them also had significant experience in the industry) and 3 practitioners. All participants had significant experience in the domain of embedded systems. The focus group included three separate groups. On the completion of the separate focus groups, the participants gathered in plenary and discussed their findings. Due to the existence of the plenary session, we performed no segmentation.

3) Conduct the focus group

In total the duration of the focus group was 2 hours. The schedule in the focus group was organized as follows: (a) presentation of the current version of PROMES process framework, (b) parallel discussion in three focus groups (first part), (c) parallel discussion in three focus groups (second part), and (d) summarizing in plenary session. The topics that were discussed in parallel and the mapping to the goals of the focus group (see Section V.A.1) were:

First Part:

- FG-1: Terminology/Focus of the proposed process framework (G1)
- FG-2: Meta-model (G3)
- FG-3: Use cases for the proposed process framework (G2)

Second Part:

- FG-1: Framework requirements (G2)
- FG-2: Process component interfacing (G4)
- FG-3: Process component synchronization (G4)

4) Analyze the data and report the results

The findings of the *analysis focus group* were used to update the PROMES Process Framework and provide guidance for further enhancements. Main changes can be summarized as follows:

- the term **process element** was explicitly defined as *any possible constituent of a process framework*, and the term **process component**, as *any potential instance of a process element, derived from the process framework, when instantiating a specific process*. (G1)
- the focus of the PROMES Process Framework changed from software to system lifecycle. (G1)

In addition, the following needs have been identified and used as directions for future work (implemented in the current version of the framework), as follows:

- the need for adding elements that concern the **control flow** of sequential *Lifecycles*, *Development Phases* and *Development Activities*. (G3)
- the need for providing a number of **variability mechanisms** that could be used for applying process instance modifications, when using the PROMES process framework. (G2)
- the need for **validating a process instance** by: *(a) checking the composition of process components, (b) checking the synchronization of process components, and (c) checking the existence of consistent interfaces among process components.* (G4)

B. Validation Focus Group

The *validation focus group* was performed in December 2013, after having completed the first version of the PROMES process framework, in which all the feedback received from the *analysis focus group* was incorporated. In this focus group the participants were industrial and research partners from Finland and The Netherlands.

1) Plan the focus group

The research problem that we wanted to investigate through this focus group was the applicability of the proposed process framework to an industrial context, and more specifically its ability to handle the embedded systems domain-specific challenges, i.e., multi-lifecycle, multi-discipline, multi-site and multi-organizational engineering.

2) Design the focus group

The participants of the focus group were selected to be experts in the domain of embedded systems. The group consisted of 12 practitioners and researchers, representing two research institutes[3] and six industrial partners[4]. In this focus group no segmentation was done.

3) Conduct the focus group

In total the duration of the focus group was 2 hours. The schedule in the focus group was organized as follows: (a) presentation of the PROMES process framework, and (b) plenary session, during which the industrial partners provided examples of industrial cases, which were later on mapped to the PROMES process framework.

4) Analyze the data and report the results

The received feedback can be summarized in two major categories: (a) feedback on the ability of the framework to model the special challenges of embedded systems (handled by the meta-model as presented in Table I, in section III.A), and (b) general feedback on the process framework.

Feedback on domain-specific challenges:

- The handling of multi-lifecycle engineering by providing a *Composite Lifecycle* element is

[2] Researchers were from Embedded Systems Institute. Practitioners were from Océ, KE-Works, and Vector Fabrics.

[3] The research Institutes were: Embedded Systems Institute (The Netherlands), and VTT Technical Research Centre (Finland).

[4] The industrial partners were: Océ, KE-Works, and Vector Fabrics (The Netherlands), and Nokia Solutions and Networks, Metso, and Haloila (Finland).

acknowledged as an efficient and effective way for modeling cases where multiple lifecycles exist.

- Placing the attributes *site* and *organization* (incorporating multi-site and multi-organization engineering) in the *Development Activity* element, was considered a simple, yet effective solution by the industrial participants. The participants also acknowledged that these two attributes help from a validation point of view too, since they clearly shows which site or organization is responsible for an activity, and thus owns its created (*output*) *artifacts*. This information is important in cases where a site/organization needs to use an *input artifact* that is not currently available, because it was created by another site/organization. However, there was a discussion whether such concepts deserved an element of their own, which on the other hand would make the meta-model more complex. This will be considered in future engineering of the framework.

- The *discipline* attribute, which at that point was placed in the *Development Activity* element (so as to handle the multi-disciplinary nature of embedded systems), was considered useful, although in some cases it was found that redundant information was stored. For example, in the case that a complete development phase is handling a certain discipline, then all its constituent development activities had to be assigned with the same value in the discipline field. For this reason, we updated the framework based on this feedback, and we moved the attribute *disciplines* at the *ControllableElement* element, since different disciplines can be identified based on *Lifecycles, Development Phases* and *Development Activities*. Thus, if *discipline* is left blank, it inherits the discipline from its container element.

General Feedback:

- The process framework is easy to use and understand.

- Although not systematically examined, the process framework seems to map to the processes of all industrial partners.

- The use of a self-reference for providing multiple levels of activities is a useful feature that resolves the restriction to have up to three levels of decomposition, which is proposed in RUP [7].

VI. CONCLUSIONS

This paper proposes a process framework that can assist process engineers in the domain of embedded system, to instantiate project specific processes. In order to design the proposed PROMES Process Framework, the challenges concerning the process engineering in the embedded systems industry (multi-lifecycle, multi-discipline, multi-site, and multi-organization) were identified. This resulted in including elements and attributes in its meta-model that can be used for modeling the aforementioned engineering aspects. At this point the framework does not provide a complete mechanism for handling the aforementioned challenges – we consider that future work. However, it offers a theoretical basis for introducing more elaborate structures, methods or tools to sufficiently address them (e.g. a method for instantiating a development phase, considering the issues induced by a potential time difference between two sites, which work on successive development activities).

The usefulness and applicability of the PROMES Process Framework has been successfully validated by two focus groups with Dutch and Finish industrial and research experts on the domain of embedded systems. During the validation, apart from the positive feedback on the applicability of the framework, we received valuable comments that will guide our future research on the subject.

As future work, we plan to systematically evaluate the usability, the effectiveness and the accuracy of the framework, with industrial partners of the PROMES consortium, through a case study. Additionally, we plan to further investigate and improve the way we model the multi-discipline aspect. Finally, after establishing the theoretical background, we plan to implement a process element repository for supporting the practical use of the proposed process framework, and further support handling the embedded systems domain challenges.

ACKNOWLEDGMENT

This research has been partially funded by the ITEA2 project 11013 PROMES. Additionally, we would like to thank the participants of the focus groups for their valuable feedback on the Process Framework.

REFERENCES

[1] D. G. Firesmith and B. Henderson-Sellers, “The OPEN process framework: An introduction”, Addison-Wesley Pearson Education, London, UK, 2002

[2] C. Hofmeister, R. Nord, and D. Soni, “Applied software architecture”, Addison-Wesley Longman Publishing, Boston, MA, USA, 2002.

[3] IEEE 730:2002 Standard for software quality assurance plans, IEEE Computer Society,23 September 2002.

[4] ISO/IEC/IEEE 15288:2008 Standard for systems and software engineering - system life cycle processes, IEEE Computer Society, 31 January 2008.

[5] Y. Jiang, X. Zhou, Y. Liu, Q. Zeng, J. Zhong, “A RUP-based process pattern for software development outsourcing”, International Conference on Service Sciences, pp. 363-369, 13-14 May 2010.

[6] J. Kontio, J. Bragge, and L. Lehtola, “The focus group method as an empirical tool in software engineering”, Chapter 4 in F. Shull, J. Singer, and D. I. K. Sjøberg (editors), “Guide to Advanced Empirical Software Engineering”, Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2007.

[7] P. Kruchten, “The Rational Unified Process: An introduction”, Addison-Wesley Longman Publishing, 2nd Edition, Boston, MA, USA. 2000.

[8] Object Management Group, “UML specification”, retrieved 12/2013 http://www.omg.org/spec/UML/2.4.1/

[9] Rational Corporation, “Rational unified process - best practices for software development teams”, IBM, online white paper, 1998.

[10] Rational Software, “Rational unified process for systems engineering – RUP SE1.1”, A Rational Software White Paper, TP 165A, 5/02, 2001.