Architectural Decisions for HW/SW Partitioning Based on Multiple Extra-Functional Properties

Gaetana Sapienza
ABB Corporate Research
and Mälardalen University, Västerås, Sweden
gaetana.sapienza@se.abb.com

Ivica Crnkovic
Mälardalen University
Västerås, Sweden
ivica.crnkovic@mdh.se

Pasqualina Potena
Universita' degli Studi di Bergamo
Dalmine, Italy
pasqualina.potena@unibg.it

***Abstract*—Growing advances in hardware technologies are enabling significant improvements in application performance by the deployment of components to dedicated executable units. This is particularly valid for Cyber Physical Systems in which the applications are partitioned in HW and SW execution units. The growing complexity of such systems, and increasing requirements, both project- and product-related, makes the partitioning decision process complex. Although different approaches to this decision process have been proposed during recent decades, they lack the ability to provide relevant decisions based on a larger number of requirements and project/business constraints. A sound approach to this problem is taking into account all relevant requirements and constraints and their relations to the properties of the components deployed either as HW or SW units. A typical approach for managing a large number of criteria is a multi-criteria decision analysis. This, in its turn, requires uniform definitions of component properties and their realization in respect to their HW/SW deployment. The aim of this paper is twofold: a) to provide an architectural metamodel of component-based applications with specifications of their properties with respect to their partitioning, and b) to categorize component properties in relation to HW/SW deployment. The metamodel enables the transition of system requirements to system and component properties. The categorization provides support for architectural decisions. It is demonstrated through a property guideline for the partitioning of the System Automation and Control domain. The guideline is based on interviews with practitioners and researchers, the experts in this domain.**

I. Introduction

In recent years, diverse hardware technologies have enabled a significant improvement in software performance. These hardware technologies offer heterogeneous platforms consisting of different computational units on which a particular application utilizes the specific properties of the platform. In addition to the already-present multicore CPUs, other computational units such as GPUs (Graphical Process Unit) and FPGA (Field-Programmable Gate Array) are becoming available for general-purpose software applications. This capability introduces software into new domains, and enables more sophisticated applications, but it also poses new challenges for software development. Although the computational units are characterized by particular features (such as full parallelism, or fast process context switch) it is not always obvious which parts of a software application should be deployed on which unit. This is especially true for different types of embedded systems, or cyber-physical systems (CPSs), which have specific requirements of runtime properties such as performance, resource consumption, timing properties, dependability, and lifecycle properties such as productions costs. In particular, the architectural decision about HW/SW partitioning, i.e. which application components will be implemented and deployed as software executable units (e.g. the compiled C/C++ source code), and which as hardware executable units (e.g. synthesized from VHDL), is becoming increasingly challenging. The partitioning decision is a known problem and there is a considerable body of knowledge related to that (e.g., [43], [28], [24]). In these approaches, a few factors are typically taken into consideration for a trade-off partitioning decision: e.g. resource availability (power, CPU utilization) and performance. However, due to the increased complexity and demands on system and project performance efficiency, partitioning decisions are related to many requirements, not only to run-time properties, but also to project constraints (such as available expertise, or development costs), or to business goals (such as development of mass-products, or product-line, etc.). This makes the design process quite complex and inaccurate in taking ad-hoc decisions, or manually processing all requirements. While many such decisions depend on the architects expertise and gut feeling, it is not guaranteed that a good (not to say the best) decision can be taken. To be able to come to an accurate decision, we must take a systematic and, when possible, an automatic approach to provide the decision.

In our previous work [38] and [39] we proposed a partitioning decision process, MULTIPAR, for component-based CPSs based on a) transformation of the requirements and constraints to Extra-Functional Properties (EFPs) through Software Architecture, and b) Multi-Criteria Decision Analysis (MCDA) of component EFPs that depends on the component implementation and deployment (as HW or SW units). MULTIPAR enables the consideration of many component EFPs identified in the architecting process, and the discovery of a (semi)optimal deployment architecture in respect to the HW/SW deployment. This approach is appealing since it takes into consideration many requirements and many properties that reflect not only run-time aspects but also business and development project-related aspects. It does, however, introduce a more complex decision process. To make such a process feasible, two important questions must be addressed:

1) *How to specify different EFPs in a uniform way so that it is possible to use them in an deployment analysis?*
2) *Which EFPs depend, and to which extent they depend on the deployment?*

The goal of this paper is twofold: a) to provide a theoretical

978-1-4799-3412-6/14 $31.00 © 2014 IEEE
DOI 10.1109/WICSA.2014.19

model of component-based systems with HW/SW components and their properties, and b) to categorize component properties in respect to HW/SW deployment. The first part includes the extension of some existing component models. The extension is necessary because of the dual nature of component implementations, namely as HW or SW units. The second part includes an analysis of EFPs defined in several standards and quality models, in respect to HW/SW deployment, and then a discussion with researchers and industry experts in the Automation and Control domain, which results in a comprehensive survey of EPFs and the impact of the partitioning decisions on their values. The concrete contribution of this paper is a) a component and EFP model for HW/SW component-based systems, b) categorization of EFPs in respect to partitioning, and c) analysis of the impact of HW/SW partitioning on EFPs for the Control and Automation domain, provided by experts from industry and academia.

The paper is structured as follows. Section 2 describes a formal model of component-based CPSs including EFP specifications. In Section 3, the paper gives the partitioning quality framework with system architecture metamodel, together with a categorization of EFPs. Section 4 provides an analysis of EFPs in the System Automation and Control domain, based on an empirical survey. Related work is described in Section 5 and, finally, Section 6 concludes the paper.

II. Modeling HW/SW component-based systems

The main idea of MULTIPAR is to automate the HW/SW partitioning decision based on the properties of components, either implemented as SW or HW. The MULTIPAR approach uses the Model-Driven Architecture with (i) the Platform-Independent Model (PIM) stage for initially achieving technology-independent design and (ii) the Platform-Specific Model (PSM) stage for subsequently enabling HW-specific and SW-specific designs. This implies first the design of the software architecture, with the identification of components and the connection between them, and then the design of the deployment view. The focus of MULTIPAR is the partitioning decision process in the PSM architecting stage.

An equally important approach in MULTIPAR is the reuse of existing components, for which we not only have the available implementation, but also specifications of their EFPs within a particular context (e.g. execution platform, implementation, etc.). A critical part of this kind of approach is breaking the system requirements down into component requirements, and then selecting the most appropriate components, i.e. selecting the components whose properties provide the best solutions in respect to the requirements.

We extend the component-based development (CBD) technique a well-known approach in SW development but not used for development of both HW and SW - to represent HW components as well. We adapt the component-based system formalism provided in [13] in order to define (i) the system as a number of components able to represent both SW and HW components, (ii) the interconnections between the components, and (iii) the platform on which the components are deployed.

We define a system $\mathbb{S}$ that consists of a platform $\mathbb{P}$ and a set of applications $\mathbb{A}$ that includes *n* applications A_i, i.e.

$$\mathbb{S} =< \mathbb{P}, \mathbb{A} > \ where\ \mathbb{A} = \{A_i\},\ i = 1..k \tag{1}$$

Note that the platform can be distributed, i.e. it can consist of different and multiple execution units. We consider applications built from new and existing components, i.e. component-based applications. A component-based application $\mathbb{CBA}$ is formally described as a pair of the following elements:

$$\mathbb{CBA} =< \mathbb{C}, \mathbb{B} > \tag{2}$$

where $\mathbb{C}$ represents the set of components in which the application is decomposed[1]; $\mathbb{B}$ represents the set of bindings interconnecting the components, referred to as connectors.

A component is meant to be a modular, deployable, reusable and replaceable self-contained unit of one or more functionalities of the application. It can be implemented and deployed as HW or SW, i.e. it can either be synthesized into HW blocks or compiled into SW executable machine code. Some examples of CPS application components are: a PI-regulator, a Robot-axis controller, a FIR filter, etc.

Each component C (later deployed as either HW or SW) is characterized by a number of properties: functional properties, specified by its interface *I* and extra-functional properties (EFPs):

$$\mathrm{C} =< I, P > \tag{3}$$

Functional properties are expressed as provided interface. In addition a component has a required interface that specifies the functions the component is using. A simple example of the interface with respect to a PI-Controller component may be the error signal as required interface and the controlled output as provided interface. Examples of EFPs are: execution time, memory size, reliability, etc. For each component C, which is represented as a model (i.e. a specification), there can exist more implementations, i.e. component variants.

$$\mathrm{C} = \{C_i\}\ : i = 1..n \tag{4}$$

$$\mathrm{C}_i =< I, P_i >, \qquad P_i \subseteq\ P \tag{5}$$

Where C_i is the i-th instance (also called a variant) associated to the component C and *n* is the total number of variants for that component. Each variant implements the same interface, but the EFPs, defined as P_i may vary from variant to variant.

$$P_i = \{P_{ij}\}\ : j = 1..m \tag{6}$$

where *m* is a number of specified properties for C_i.

Note that this is significantly different to the specification in most component models (see in [13]), in which components comply to the same rules, i.e. to the same interface and the same EFPs. The different implementations of a component not only provide different values of the same properties, but may also have different properties. This is a direct consequence of their implementations that can be SW or HW.

[1]If not explicitly specified differently, by "components" we assume "application components"

Required and Exhibited Properties. The EFPs specified above are the properties *exhibited* by the components, i.e. they are immanent parts of the component instances. We designate these properties as P_{iE}. These are different from the *required properties* [12]. The required properties P_{iA} are the required values of the component properties which are derived from the application requirements, the project constraints, and the architectural analysis. The application requirements typically consider run-time aspects of the application, and the required properties P_{iA} derived from the application requirements define the values related to the run-time component properties. Examples of P_{iA} are component execution time, and memory size that can be allocated by the component. The project constraints are usually related to the product lifecycle and business issues, and P_{iP} represents the required properties derived from these project constraints. For instance, P_{iP} related to the efforts and costs are the time estimated for components adaptation, design, and testing, or the cost needed for purchasing Intellectual Properties (IP) components.

For a particular application, not all available component properties are of interest, but only a subset that includes those properties that are identified from the application requirements and project constraints. Additionally, a component instance may not have all properties specified that are defined as required properties. In this case the architect/expert has to provide the values of these properties, either by estimation, by measurement or from historical data. To achieve correct design partitioning, a set of all the relevant exhibited properties P_{iE} must satisfy the required properties P_{iA} and P_{iP} for each component instance C_i.

$$P_{iE} \models \{P_{iA}, P_{iP}\} \tag{7}$$

The condition (7) is required but it does not necessarily provide an optimal solution. Many configurations from different instances can satisfy this rule. To find an optimal solution, a cost function that includes the weighted sum of the properties must be defined. There exist different methods to provide this result, one of which is a Multi-Criteria Decision Analysis (MCDA).

III. The Partitioning Quality Framework

The next steps in our formalization are i) specification of a property metamodel which is able to encompass the heterogeneity derived by the intrinsically different nature of HW or SW components, and ii) categorization of EFPs with respect to their sensitivity to the HW/SW partitioning decisions.

A. Component property metamodel

The property metamodel is a part of the entire component-based system metamodel, which is based on the definitions from the previous section. Figure 1 shows a simplified overview of the component-based system; a completed metamodel and detailed description can be found in [39]. Note that here, in contrast to typical metamodels for software components, different variants of the same component can have different EFPs. A component variant type is related to the component implementation: it can be SW, HW, or ”virtual”; the

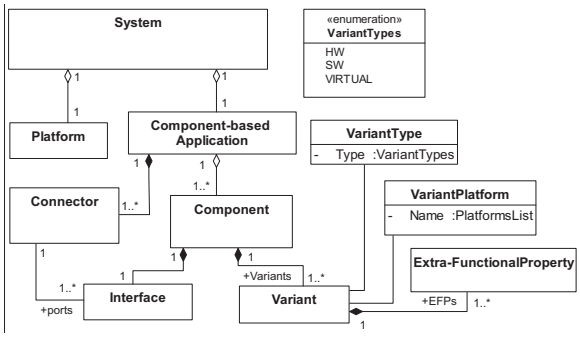

Fig. 1. Component-based System Metamodel (Simplified Overview)

”virtual” type is defined for not yet implemented components.

Figure 2 shows an overview of the diagram describing the Component Extra-Functional Property. The key entities of this metamodel are: the *Component Extra-Functional Property*, the *ComponentVariantPropertyValue* and the *ComponentPropertyCategory*.

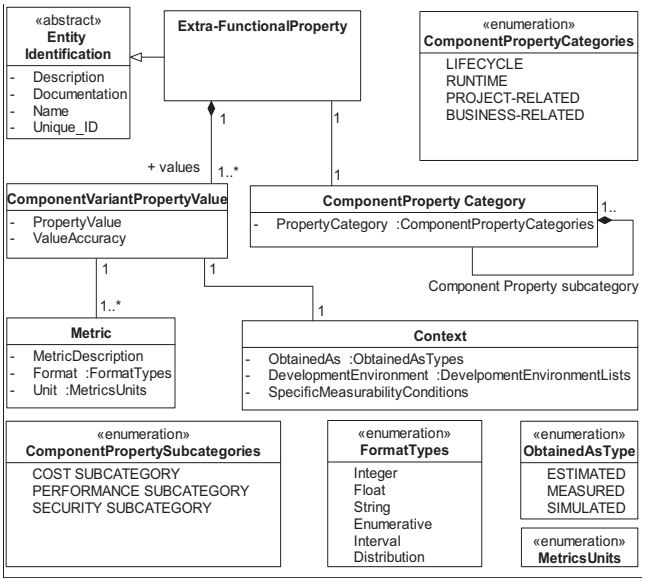

Fig. 2. Component Extra-Functional Property Metamodel

Component Extra-Functional Property. The central core is represented by the component EFP entity which abstracts a component property. It belongs to a specific category as modeled by the ComponentPropertyCategory and has an associated set of values related to a specific component variant.

ComponentPropertyCategory. There exist several different categories of properties. This categorization is used to group similar properties. The categories are listed through the *ComponentPropertyCategories* as shown in Figure 2. Each category can also be divided into subcategories. This enables categorizations that follow different standards, or it can be used for grouping complex properties that are related to a number of other (sub)properties - for example a performance, or a categorization used in a particular domain to group strongly

related properties, for instance, the COST subcategory can include costs for the design, implementation, verification and validation, and maintenance. The definition of the subcategory refers to the value associated to a specific component variant.

ComponentVariantPropertyValue. The *ComponentVariantPropertyValue* abstracts the value associated with the property. Each value has associated (i) a metric to quantitatively or qualitatively express the value (or set of), and defined through a description, a format (described by the *FormatType* entity), and (ii) a context which serves to capture the conditions under which the value is estimated, simulated or measured (as described by the *ObtainedAsType*).

B. The categorization of partitioning-related properties

The main driving question in this research is *which are the properties whose values strongly depend on the partitioning decisions, and which properties are independent of the partitioning*? To answer this question we provide (i) a list of possible EFPs and their categorization following the metamodel shown in Figure 2, and (ii) an analysis of the EFPs in respect to their dependencies on the partitioning.

To provide a list of all possible EFPs we use three types of sources a) existing quality models and standards; b) existing literature related to the partitioning decisions; and c) experience from the practice provided by experts in the field.

- *Quality models and standards*. Specifically, we exploited the following standards and quality models.
  1) ISO/IEC International Standard Software product Quality Requirements and Evaluation (SQuaRE) [20] defined for software qualities, i.e. an extension of the International Standard ISO/IEC 9126 [21], which was extensively exploited in the state of the art;
  2) Several well-known quality classifications, in particular the McCall's quality model [27], the Boehm's quality model [5], the Laprie's dependability tree (attributes) [1];
  3) The quality attribute utility tree modified for embedded hardware platforms (later referred to as HW-ATAM) [36] based on the well-known Architecture Tradeoff Analysis Method (ATAM-CYC [22]);
- *State Of The Art (SOTA)*. There is a rich body of knowledge related to HW/SW partitioning. We have extracted the EFPs addressed in these works. The details of this work are given in Section 5 (Related Work).
- *State Of The Practice (SOPA)*. We provided a survey carried out within the iFEST (industrial Framework for Embedded Systems Tools) Artemis JU research project , with a consortium of more than 20 companies and universities[2] in which discussion was held with the project members about important EFPs and their relation to the HW/SW partitioning. To this end, we conducted an empirical study that, in combination with the survey, included data collection through in-depth interviews with experts and researchers from both companies involved in the iFEST project, and companies and universities in Sweden. The details of this study are presented in Section 4.

We categorized the EFPs in three main categories: i) LifCycle EFPs; ii) RunTime EFPs; and iii) Project/Business-related EFPs. As presented in [12], the system lifecycle perspective encompasses those properties which are related to the system development process and its maintenance, while the runtime perspective interests those properties whose behavior is visible, applicable, and measurable at run-time. In summary, all identified and of-interest properties are divided into three categories, namely "LifeCycle", "RunTime" and "Project/Business-related".

Table I reports the details of our categorization. The table is structured by category i.e. it is divided into LifeCycle, RunTime and Project/Business-related. Each category is then divided into subcategories, which group properties having similar descriptions. This is done in order to improve understandability of the categorization. However, as in most of the cases in Table I, this does not preclude a subcategory from also being a property. In particular, in Table I each category is organized as follows: the first and second columns represent the subcategories and the group of related properties, respectively. For instance, in the first row of the LifeCycle category, Usability is a subcategory grouping properties such as Appropriateness, Recognizability, User error protection, User interface aesthetics, Learnability and Training. In addition to the subcategory name, the sources from which the property is taken are reported. The categorization follows the classification in the standards and quality models (SQuaRE, ISO9126, McCall, etc.) as much as possible. Using the example just mentioned above, for the Usability subcategory we have to exploit both the SQuaRE and the McCalls quality model in order to identify its related properties. For each property (i.e., related to a given subcategory), its source is also reported in parenthesis.

We can observe that LifeCycle properties are present in the largest quantity. Short descriptions (titles) of the properties can be found in [37]. Many of those properties are difficult to measure. The RunTime properties are extensively studied in research and used in practice, and it is possible to express many of them quantitatively. The Project-related properties are time- and effort-related properties, as properties of project-related activities. The Business-related properties consider types of products (like mass products, product families, and similar). A relevant observation is that some properties have subcategories in different categories. For instance, Reliability is a RunTime property, but many LifeCycle properties have direct impact on the reliability. Such properties are marked in the table with an "*".

IV. Survey of EFPs in the Automation and Control Domain

Table I presents a list of possible EFP candidates which may be important in the partitioning decision process. The influence of partitioning on specific EFP also depends on other factors, for example on particular architectural solutions, or the overall goals of the application, so it is not possible to provide

[2]iFEST - http://www.artemis-ifest.eu/

TABLE I. LIFECYCLE CATEGORY, RUNTIME CATEGORY, PROJECT/BUSINESS-RELATED CATEGORIES.

LifeCycle Category	
Subcategory	**Property**
Usability (SQuaRE, McCall)	Appropriateness recognizability (SQuaRE) User error protection (SQuaRE) User interface aesthetics (SQuaRE) Learnability (SQuaRE) Training (McCall)
Integrity* (McCall)	Access audit (McCall)
Reliability* (SQuaRE)	Maturity (SQuaRE)
Security* (SQuaRE)	Accountability (SQuaRE)
Compatibility (SQuaRE, McCall)	Compatibility (SQuaRE) Interoperability (McCall) Communication Commonality (McCall) Data Commonality (McCall)
Modifiability (SQuaRE, Boehm)	Modifiability (SQuaRE) Expandibility (McCall) Scalability (HW-ATAM, SOPA) Upgradability (SOPA) Augmentability (Boehm) Structuredness (Boehm)
Maintainability (SQuaRE, McCall)	Maintainability (SQuaRE, MaCall, Laprie) Analysability (SQuaRE) Reusability (SQuaRE) Conciseness (McCall) Modularity (SQuaRE)
Testability (McCall)	Testability (SQuaRE, HW-ATAM) Instrumentation (McCall) Simplicity (McCall) Generality (McCall) Debuggability (SOPA)
Understandability (Boehm)	Understandability (Boehm) Legibility (Boehm)
Flexibility (McCall)	Flexibility (McCall) Self-descriptiveness (Boehm)
Human Engineering (Boehm)	Human Engineering (Boehm) Communicativeness (Boehm)

LifeCycle Category	
Subcategory	**Property**
Correctness (McCall)	Correctness (McCall) Completeness (McCall) Consistency (McCall) Traceability (McCall)
Portability (SQuaRE)	Portability (SQuaRE) Adaptability (SQuaRE) Installability (SQuaRE) Replaceability (SQuaRE) Platform Independency (McCall) Integrability (SOPA)

Project/Business related Category	
Subcategory	**Property**
Maintainability*	Integration with legacy system (SOPA) Product support (SOPA) Volatile requirements (SOPA)
Production type	Mass production (SOPA) Single/Small production (SOPA) Product variability (SOPA) Product lifetime (SOPA)
Cost	Design Cost (SOPA) Implementation Cost (SOPA) Integration Cost (SOPA) Testing Cost (SOPA) Production Cost (SOPA) Maintenance Cost (SOPA) Development Environment Cost (SOPA) Maintenance Environment Cost (SOPA)
Lead Time	Design Lead Time (SOPA) Implementation Lead Time (SOPA) Integration Lead Time (SOPA) Testing Lead Time (SOPA) Production Lead Time (SOPA) Maintenance Lead Time (SOPA) Time-to-market (HW-ATAM, SOPA)
Project Team	Distributed project environment (SOPA) Available expertise (SOPA)

RunTime Category	
Subcategory	**Property**
Usability (SQuaRE)	Operability (SQuaRE) Accessibility (SQuaRE)
Integrity (McCall)	Access control (McCall)
Reliability (SQuaRE, McCall)	Reliability (SQuaRE) Accuracy (McCall) Error Tolerance (McCall) Availability (SQuaRE) Fault tolerance (SQuaRE) Recoverability (SQuaRE) Self-containedness (Boehm)
Security (SQuaRE)	Security (SQuaRE) Authenticity (SQuaRE) Confidentiality (SQuaRE, Laprie) Integrity (SQuaRE, Laprie) Non-repudiation (SQuaRE)
Modifiability* (SOTA)	Configurability (HW-ATAM)
Functional Suitability (SQuaRE)	Functional Suitability (SQuaRE) Functional Appropriateness (SQuaRE) Functional Completness (SQuaRE) Functional Correctness (SQuaRE)
Dependability (Laprie)	Dependability (Laprie) Safety (Laprie) Robusteness (HW-ATAM) Schedulability (SOTA, SOPA)
Performance (SQuaRE)	Performance Efficiency (SQuaRE) Resource Utilization (SQuaRE) Time Behaviour (SQuaRE) Capacity (SQuaRE) Power Consumption (HW-ATAM, SOTA, SOPA)
Compatibility (SQuaRE)	Co-existence (SQuaRE) Interoperability (SQuaRE)

a general list valid for any type of application. However, in specific domains in which there are similar architectural solutions, and similar non-functional requirements, it is more feasible to provide such a list. We aim to identify which EFPs are of interest for partitioning in the Automation and Control domain, which covers a vast range of business domains, such as energy generation and transportation, industrial plants (chemical, pulp and paper, food industry), medical instruments, etc.

In order to achieve this goal we conducted an interview-survey involving several companies working in the Automation and Control domain. We specifically targeted companies whose products vary from low voltage systems (e.g. motor controllers, industrial robots) up to systems for controlling electric power transmission lines. The interview-survey research followed a method specified in [3]. An overview of data collected is presented here. Complete documentation of the results can be found in [37].

A. Objective

To identify the key partitioning properties for the aforementioned domain, for each EFP from Table I we pose the following questions:

RQ1 *Does the partitioning choice (HW or SW) have a direct impact on the property?*

RQ1 *To ensure or achieve this property would you prefer to have a solution in HW or SW or it Does Not Matter?*

With these two research questions, we aim to (i) identify the EFPs which have the most impact on the partitioning decisions (related to RQ1), and (ii) provide guidance for architects (related to RQ1 and RQ2) by highlighting a default preference (HW or SW) for the realization of each property.

B. Survey Design and Process

The interview-survey was carried out among 15 experts. The participants were selected based on their affiliation and expertise. The participants were selected from industry (in total five companies and an industrial research center) and from academia (two universities). All participants had more than five years of experience as system architects (i.e., experience with both HW and SW design) and nine of them had been working for more than 15 years in the field. In addition, we selected the participants according to their prevailing work experience, with the participants pool balanced as follows: five specialists with prevalent work experience in HW design, five specialists in SW design, and five specialists in both HW and SW design. Table II summarizes the distribution of the participants with respect to their work experience (i.e. work specialization, affiliation and years of experience in the field).

The interaction with the participants was organized in two phases: 1) an introduction to the interview-survey, which was arranged in the form of individual face-to-face meetings. The purpose of the meetings was to get the participant familiar with the overall goal of the research, presenting the EFP categorizations, running through the EFP lists, and finally explaining

TABLE II. DISTRIBUTION AMONG THE PARTICIPANTS

No	Profession	Affiliation	Expert	Experience
1	System Architect	Industry	HW	15 years
2	System Architect	Industry	HW	15 years
3	Senior Developer	Industry	HW	5 years
4	Principal Industrial Scientist	Industry/University	HW	15 years
5	Principal Industrial Scientist	Industry/University	HW	15 years
6	Senior System Architect	Industry	SW	15 years
7	Industrial Scientist	Industry/University	SW	15 years
8	Researcher	Industry/University	SW	5 years
9	Professor	Industry/University	SW	15 years
10	Researcher	University	SW	5 years
11	Senior Product Manager	Industry	HW/SW	15 years
12	Senior Developer	Industry	HW/SW	5 years
13	Industrial Scientist	Industry/University	HW/SW	5 years
14	Senior Researcher	University	HW/SW	5 years
16	Professor	University	HW/SW	5 years

TABLE III. EXAMPLE OF QUESTIONNAIRE RECORDS FOR THE LIFECYCLE CATEGORY

RQ1. Does the partition choice (SW or HW) have a direct impact on the property?
RQ2. To ensure/achieve this property would you prefer to have a solution in HW or SW or it does not matter?

Property Name	Property Description	RQ1	RQ2
Access Audit	The ease with which the component itself and data can be checked for compliance with standards or other requirements (McCall).	YES	HW
Accountability	The degree to which the actions of an entity can be traced uniquely to the entity (SQuaRE).	YES	HW
Structuredness	The degree to which a component possesses a definite pattern of organization of its inter-dependent parts (Boehm)	YES	HW

how to fill out the questionnaire; and 2) the completion of the questionnaire by the participants.

The questionnaire consists of three main parts, related to the three main EFP categories. For each EFP specified in Table I the participants answered RQ1 and RQ2, as shown in Table III. The complete questionnaire is presented in [37].

C. Results

The collected data is presented in Figures 3, 4, 5, which summarize the results related to each category respectively, i.e. LifeCycle (Figure 3), RunTime (Figure 4) and Project/Business-related (Figure 5 EFPs). Each figure contains two stacked bar graphs, the one on left showing the RQ1 results and the one on the right reporting the RQ2 results. The properties in the graphs are sorted in descending order, according to the impact on a given property (provided by RQ1), as judged by the participants. For each graph the vertical axis label reports the properties per category. There are 44 properties for the LifeCycle category, 31 properties for the RunTime category and 24 for the Project/Business-related category.

The graphs show percentage distributions of the answers (YES or NO) for the Partitioning Impact graph, and HW, SW or DNM (Does Not Matter) for Preference (HW-SW) graph.

D. Analysis

To start analyzing the results, we introduce two indices:

The **Impact Partitioning Factor** (IPF), which gives a statistical measure of the impact that a given property has on the partitioning decisions for the specific domain. This index is used to analyze the results with respect to RQ1. It is calculated by the ratio in percentage between the total number of positive answers and the number of participants. We interpret the IPF according to the Fleiss Kappa agreement interpretation[3], (60-100 % - a substantial or almost perfect agreement).

TABLE IV. IPF INTERPRETATION

Impact Partitioning Factor	Interpretation
0-40% YES, (60-100% NO)	No Impact on Partitioning
40-60% YES, (30-70% NO)	Unclear Impact
60-100% YES, (0-40% NO)	Clear Impact on Partitioning

The **Preferable Deployment Choice** (PDC), which gives a statistical measure of the deployment preference for each possibility, i.e. HW, SW or DNM ("Does Not Matter") with respect to the given property for this specific domain. This index is used to analyze the results with respect to RQ2. In this case we also use the Fleiss Kappa interpretation.

LifeCycle Category. *Adaptability*, *Flexibility*, and *Mantainability* result in an "almost perfect IPF" as it can be observed in Figure 3, (above 80%), which we interpret as a clear impact of the partitioning on these properties. The properties such as *Installability*, *Analyzability*, *Expandibility*, *Platform Independency*, *Portability*, *Access control*, *Augmentability*, *Integrability*, *Modifiability*, and *Upgradability* have a "substantial IPF" (above 61%), so a substantial agreement that these properties are clearly affected by the partitioning decision. Conversely, the properties *Accountability*, *Communication Commonality*, *Consistency*, *User Error Protection*, *Completeness*, *Data Commonality*, *Self-descriptiveness*, *Simplicity*, *Training*, *Understandability*, *Access Audit*, *Maturity*, *Learnability*, *Legibility*, *Structuredness*, *Traceability*, *Appropriateness*, *Appropriateness Recognizability*, *Usability*, *Correctness*, *Human Engineering*, and *Communicativeness* belong to a set of EFPs that are not sensitive to the partitioning decision. With regards to RQ2 (i.e. a preferable solution) we can point out that the preferable solutions for the EFPs with a clear partitioning impact are software implementations.

RunTime Category. In this category the *Power Consumption* property is the only one that shows a clear impact of the partitioning on these type of properties. Nevertheless, properties like *Co-existence*, *Performance*, *Efficiency*, *Recoverability*, *Time Behavior*, *Reliability*, and *Security* show a substantial agreement in the understanding of a clear impact of the partitioning on these properties. In contrast to the LifeCycle properties, here the hardware solutions prevail to achieve or ensure these properties in an advantageous way. The properties *Functional Correctness*, *Accessibility*, *Authenticity*, *Functional Appropriateness*, *Functional Completeness*, *Operability*, *Functional Suitability*, and *Non-repudiation* are not directly affected by partitioning.

Project/Business-related Category. By looking at the results in Figure 5 we can conclude that the partitioning decisions in the vast majority have an impact on the Business- and Project-related EPFs (that are directly related to the project constraints and business goals). In the majority, the software solutions

[3]Fleiss Kappa - http://en.wikipedia.org/wiki/Fleiss'_kappa

prevail. There is large number of properties here which have either an ”almost perfect IPF” (6 EPFs of 24 properties in total, compared to 3 of 44 in the LifeCycle category and 1 of 31 the RunTime category) or a ”substantial IPF” (10 EFPs of 24). With respect to the”almost perfect IPF”, it is interesting to note that the *Maintenance-related* properties (i.e. cost and lead time) have a clear impact similar to the *Maintainability* property in the LifeCycle category, as well as a preference for a SW implementation solution.

General remarks. General remarks in relation to the current state of the art are as follows: (i) with respect to the Run-Time EFPs, the results show a confirmation of the common properties used in literature, and some new properties (such as Recoverability); and (ii) with respect to LifeCycle and Project/Business-related EFPs, it is observed that many new properties are considered important for the partitioning decision process. In particular, the outcomes confirm our hypothesis that project and business constraints should be considered in partitioning decisions, which is in general missing in the state of the art and practice today.

E. Validity

In this section we address the validity of our empirical study. To this end, we discuss the following main types of validity threats proposed in [47].

Construct Validity. As common practice, in order to assure a high construct validity (i.e., a high relation between the theory behind our study and its observation), we used indirect measures (such as the reported working hours for the questionnaire review taken as a measure of effort) to conduct all steps of our work. The steps span from questionnaire preparation through recruitment of respondents to evaluation of results. Measures have been used in order to, for example: (1) avoid mono-operation bias by selecting participants with different backgrounds and work experience (see Section IV-B); and (2) assure rigorous planning of the study with a solid protocol for data collection and analysis. Additional threats to construct validity are represented by, for example: (1) the questionnaire structure that facilitates the data collection; (2) the individual face-to-face meetings, based on a presentation illustrating the key concepts leading the research, to get the participant familiar with the overall goal of the study; and (3) the anonymity and confidentiality guaranteed in the processing of the results to avoid evaluation apprehension.

Internal Validity. As done in [46], we addressed the confounding variables representing a major potential source of bias in empirical studies ”by exclusion and randomization”. Exclusion concerns the fact that we did not select participants who were not sufficiently experienced in HW or SW design. On the one hand, we carefully balanced our pool of participants (see Section IV-B) according to their prevailing work experience and affiliation. On the other hand, we have used a random sample of the population in order to avoid the well-know problem of selection bias [48]. Moreover, we addressed the issue of ambiguously and poorly-worded questions [46] as described in the following points. (1) Before being released to the participants the questionnaire was iteratively reviewed by university and industry experts. Moreover, our questionnaire was based on well-assessed standards and quality models

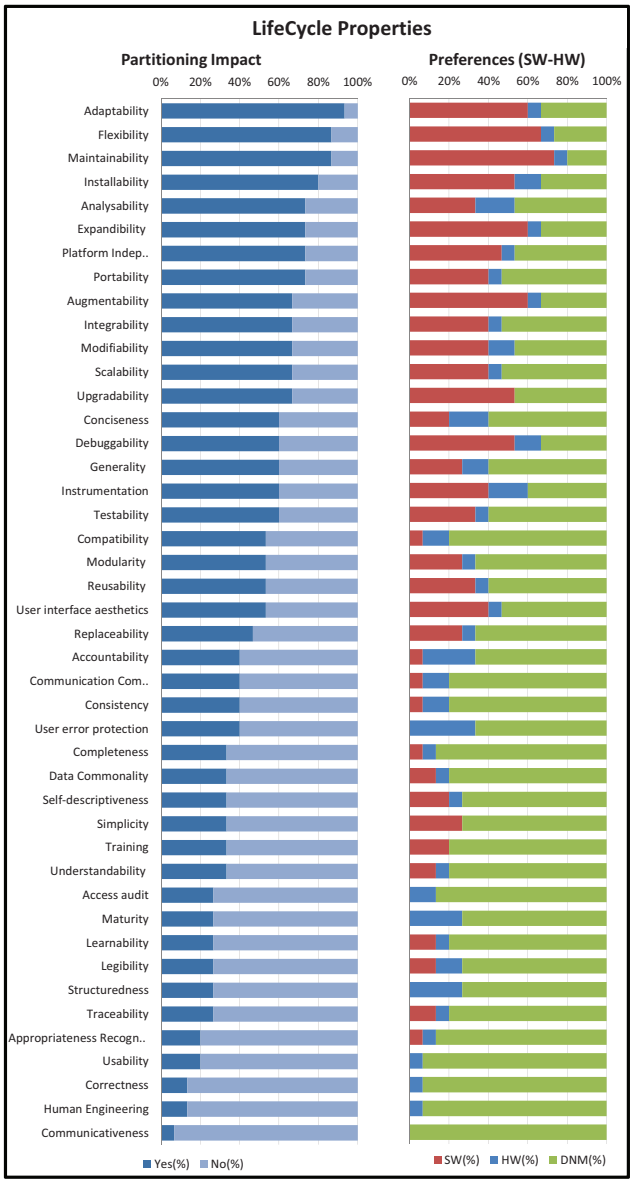

Fig. 3. LifeCycle Property Graphs. Partitioning Impact (left). HW/SW Deployment Preference (right).

(see Section III-B). (2) To let the participants understand the background and objective of this work, we performed individual face-to-face meetings. In addition, we piloted the respondents' work in different interactions.

External Validity. This validity deals with the generalization of the results outside the scope of the study. Based on discussions with the experts, our assumption is that the results are applicable to the Automation and Control domain, i.e. to a population having our adopted sampling profile. It was not the intention of this paper to deal with different application domains, but we intend to work on this aspect by analyzing possible changes in the results when domain features are adjusted.

Conclusion Validity. This validity is focused on how sure

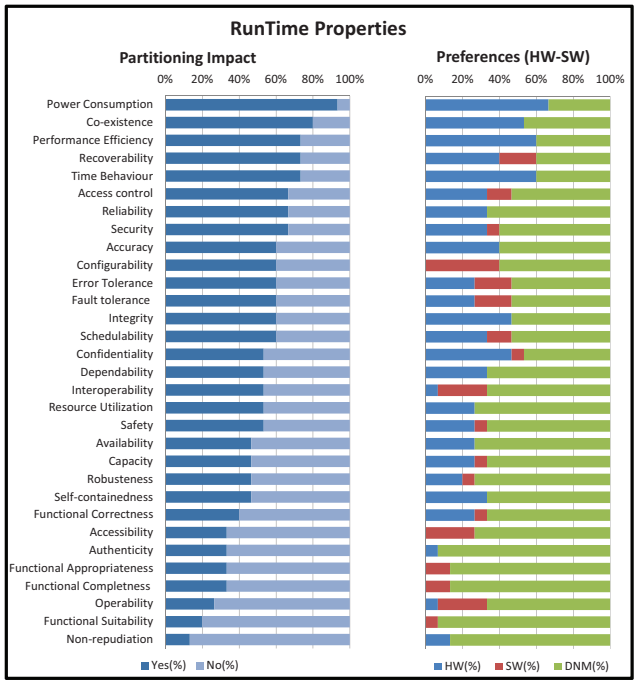

Fig. 4. RunTime Property Graphs. Partitioning Impact (left). HW/SW Deployment Preference (right)

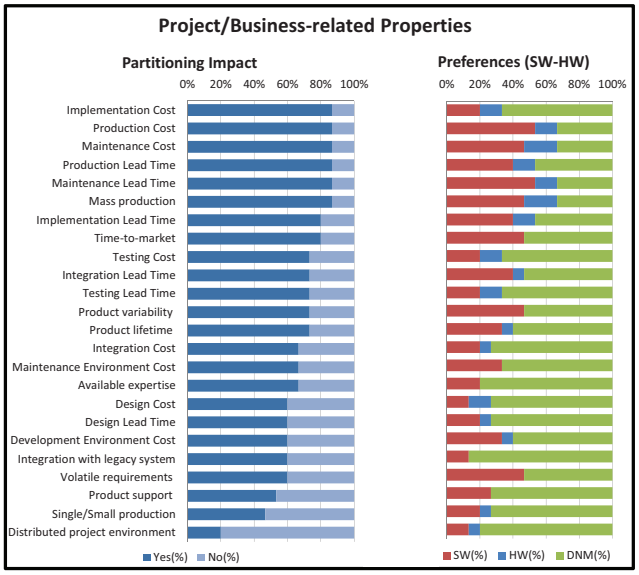

Fig. 5. Project/Business-related Property Graphs. Partitioning Impact (left). HW/SW Deployment Preference (right)

we can be of drawing correct conclusions about the relation between the treatment and the actual outcome we observed. We achieved reliable results by piloting the questionnaire in multiple interactions, and by providing the participants with key concepts on the study. We also assured a reliable treatment implementation by using the same treatment/procedure with all participants (e.g., we distributed the same questionnaire). We found the right heterogeneity among the respondents (i.e., we carefully defined our pool of participants by not using a very diverse group of respondents) by selecting the participants from a population general enough to not reduce the external validity.

V. Related Work

The work related to our research can be divided into three categories: (i) *partitioning and HW/SW co-design*; (ii) *analysis of EFPs for embedded systems*; and (iii) *design space exploration*.

Partitioning and HW/SW co-design. During the last few decades, several approaches have been proposed to application architecture partitioning and procedures oriented towards solutions which satisfy performance requirements, e.g. [9], [24]. A wide range of approaches have been proposed in order to automate/support the hardware/software partitioning activity using different strategies such as dynamic programming [49], heuristic algorithm based on the tabu search techniques [50], integer programming [30], and genetic algorithms [32]. A list that categorizes these approaches as well as other approaches focused on implementation issues, can be found in [25] and [43]. However, all these approaches basically provide guidelines to deciding which parts of the specification should be implemented in software and which in hardware. They do this by considering platform-related indicators, such as potential speedups, area, communication overheads, locality and regularity of computations [17]. Usually, they deal with a few EFPs and are focused only on technical issues. However, some partitioning approaches do exist which optimize the combination of extra-functional requirements (e.g. design costs, energy consumptions, performance, etc.) [44],[14], but are still limited in their total number of properties. In order to provide an answer to (i) the increase in complexity of the CPSs and (ii) the advances in underlying hardware technologies when it comes to the architectural partitioning decision, in contrast to existing approaches to a component-based application we propose a general and systematic methodology capable of accounting many properties and based on MCDA techniques which are, to the best of our knowledge, non-existent today.

Analysis of EFPs for embedded systems. In the last few years, several research efforts have been devoted to the definition of methods and tools able to predict and evaluate the quality of embedded systems (e.g., [11], [23], and [33]). In particular, different techniques have been introduced to support the specific features of an embedded system (e.g., the analysis of timing properties that are typically computational and time consuming [7], or the management, preservation and analysis reuse of EFPs that are also critical tasks [35], [10]). Research efforts have been made to support the development and adaptation of embedded systems. For example, component models have been introduced to support the development of embedded systems (e.g., [18], [41], and [29]), and approaches to the adaptation of embedded systems under EFP constraints have been proposed (see, e.g., [52], [53]). Several approaches have also been introduced to estimate EFPs for software and hardware implementations (see, for example [19], [40], [6] for power and energy estimation). The run time of a hardware implementation on a FPGA, for example in [4] is estimated by exploiting the simulation and a performance model of the

FPGA. In contrast, a statistical approach is proposed in [16] to estimate the execution time of embedded software.

The topic of the definition of metamodels to specify EFPs has been also studied intensely. For example, in [54] fault tolerance aspects were covered by using a metamodel, while in [2] a service-oriented metamodel for distributed embedded real-time systems covers real-time properties of services, like response time, duration, and deadline.

Other challenges related to quality analysis are seen in the strict design constraints (typical, for example, of mission-critical systems [51]) that affect the interaction between hardware and software components. Several papers have focused on these hardware/software co-verification[4] issues (see for example [42] and [15]). In the recent decades, digital and software designs methodologies have become more alike [28] and, as already foreseen in [45] they require designers to have a unified view of software and hardware, which converge with the concurrent design of hardware/software [28] (see, for example, the HWSWCO project[5] and the co-design framework in [8]).

Design space exploration: As recalled in [34], design space exploration is an umbrella activity defined by a set of tasks addressing aspects of representation, estimation, and exploration algorithms. An overview of design space exploration techniques, such as those used for System-on-a-Chip architectures can be found in [17]. Several frameworks and tools have been built for the efficient multi-objective exploration (e.g., energy/delay tradeoff [26], occupation of the FPGA, load of the processor, and performance of the application tradeoff [31]) of the candidate architectures in order to, for example: (i) enable run-time resource management in the context of multiple applications [26], or; (ii) simulate the target system and dynamically profile the target applications by exploiting heuristic algorithms for the reduction of the Pareto set exploration time [31].

VI. Conclusion and future work

In this paper, we have (i) presented a theoretical component-based approach to support the deployment of CPSs (as HW or SW components) into heterogeneous platforms based on many EFPs, and (ii) categorized EFPs in relation to the HW or SW deployment, and analyzed the HW/SW component deployment impact on these EFPs in the Automation and Control domain.

Specifically, with respect to the state of the art, the novelty of our contributions can be summarized in the following points. To the best of our knowledge, this is the first paper that (i) proposes a metamodel that *enables the management of both application requirements and project/business constraints for the CPS application deployment*, and (ii) provides a categorization for *HW/SW deployment* of EFPs, derived by different quality models, standards, literature and the current state of practice.

The analysis of the categorization has led to some interesting findings: i) Although the values of EFPs depend on the application architecture and the runtime context, there are EFPs that are strongly influenced by the partitioning decision. In general HW solutions provide better runtime EFPs related to system performance, while SW solutions provide better characteristics of lifecycle EFPs, such as modifiability, evolvability, and variability. ii) The project constraints play important roles in the partitioning decision process which in general lacks support in existing partitioning approaches.

Our intention with the categorization and partitioning impact analysis is to provide support to system and software architects in taking architectural deployment decisions. Additional work may be required in the specialization of the EFP list, and in the partitioning impact on the EFPs. Furthermore, the findings will be integrated into our MultiPar framework in the form of semiautomatic support.

[4]It is also called co-simulation in the hardware industry [8].

[5]http://hwswcodesign.gmv.com/

Acknowledgment

This research is supported by the Knowledge Foundation through ITS-EASY, an Industrial Research School in Embedded Software and Systems, and by the Swedish Foundation for Strategic Research through the RALF3 project.

References

[1] A. Avizienis, J.-C. Laprie, and B. Randell. Dependability and its threats - A taxonomy. In R. Jacquart, editor, *IFIP Congress Topical Sessions*, pages 91–120. Kluwer, 2004.

[2] M. Aziz, R. Mohamad, D. Jawawi, and R. Mamat. Service based metamodel for the development of distributed embedded real-time systems. *Real-Time Systems*, 49(5):563–579, 2013.

[3] E. Babbie. *Survey Research Method, 2nd edition*. Wadsworth Publishing Company, 1990.

[4] P. Bjureus, M. Millberg, and A. Jantsch. FPGA resource and timing estimation from Matlab execution traces. In *Hardware/Software Codesign, 2002. CODES 2002. Proceedings of the Tenth International Symposium on*, pages 31–36, 2002.

[5] B. W. Boehm, J. R. Brown, and M. Lipow. Quantitative evaluation of software quality. In *Proceedings of the 2nd international conference on Software engineering*, ICSE '76, pages 592–605, Los Alamitos, CA, USA, 1976. IEEE Computer Society Press.

[6] C. Brandolese, W. Fornaciari, F. Salice, and D. Sciuto. The Impact of Source Code Transformations on Software Power and Energy Consumption. *Journal of Circuits, Systems, and Computers*, 11(5):477–502, 2002.

[7] J. Carlson. Timing analysis of component-based embedded systems. In *Proceedings of the 15th ACM SIGSOFT symposium on Component Based Software Engineering*, CBSE '12, pages 151–156, New York, NY, USA, 2012. ACM.

[8] A. Cau, R. Hale, J. Dimitrov, H. Zedan, B. C. Moszkowski, M. Manjunathaiah, and M. Spivey. A Compositional Framework for Hardware/Software Co-Design. *Design Autom. for Emb. Sys.*, 6(4):367–399, 2002.

[9] K. B. Chehida and M. Auguin. HW/SW partitioning approach for reconfigurable system design. In *Proceedings of the 2002 international conference on Compilers, architecture, and synthesis for embedded systems*, CASES '02, pages 247–251, New York, NY, USA, 2002. ACM.

[10] A. Cicchetti, F. Ciccozzi, T. Leveque, and S. Sentilles. Evolution management of extra-functional properties in component-based embedded systems. In *CBSE*, pages 93–102, 2011.

[11] P. Costa, G. Coulson, C. Mascolo, L. Mottola, G. P. Picco, and S. Zachariadis. Reconfigurable Component-based Middleware for Networked Embedded Systems. *IJWIN*, 14(2):149–162, 2007.

[12] I. Crnkovic, M. Larsson, and O. Preiss. Concerning Predictability in Dependable Component-Based Systems: Classification of Quality Attributes. *Architecting Dependable Systems III*, pages 257–278, 2005.

[13] I. Crnkovic, S. Sentilles, A. Vulgarakis, and M. R. V. Chaudron. A Classification Framework for Software Component Models. *Software Engineering, IEEE Transactions on*, 37(5):593–615, 2011.

[14] R. P. Dick and N. K. Jha. MOCSYN: multiobjective core-based single-chip system synthesis. In *Proceedings of the conference on Design, automation and test in Europe*, DATE '99, New York, NY, USA, 1999. ACM.

[15] M. El Shobaki. Verification of embedded real-time systems using hardware/software co-simulation. In *Euromicro Conference, 1998. Proceedings. 24th*, volume 1, pages 46–50 vol.1, 1998.

[16] P. Giusto, G. Martin, and E. Harcourt. Reliable estimation of execution time of embedded software. In *Proceedings of the conference on Design, automation and test in Europe*, DATE '01, pages 580–589, 2001.

[17] M. Gries. Methods for evaluating and covering the design space during early design development. *Integration*, 38(2):131–183, 2004.

[18] H. Hansson, M. Akerholm, I. Crnkovic, and M. Torngren. SaveCCM - a component model for safety-critical real-time systems. In *Euromicro Conference, 2004. Proceedings. 30th*, pages 627–635, 2004.

[19] J. Henkel and Y. Li. Avalanche: an environment for design space exploration and optimization of low-power embedded systems. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 10(4):454–468, 2002.

[20] ISO/IEC 25000. Software product Quality Requirements and Evaluation (SQuaRE), Guide to SQuaRE. In *International Standard Organization*, June.

[21] ISO/IEC 9126. Information Technology - Product Quality - Part1: Quality Model. In *International Standard ISO/IEC 9126, International Standard Organization*, June.

[22] R. Kazman, M. H. Klein, M. Barbacci, T. A. Longstaff, H. F. Lipson, and S. J. Carrière. The Architecture Tradeoff Analysis Method. In *ICECCS*, pages 68–78. IEEE Computer Society, 1998.

[23] D. Lohmann, W. Schroder-Preikschat, and O. Spinczyk. Functional and non-functional properties in a family of embedded operating systems. In *Object-Oriented Real-Time Dependable Systems, 2005. WORDS 2005. 10th IEEE International Workshop on*, pages 413–420, 2005.

[24] M. López-Vallejo and J. C. López. On the hardware-software partitioning problem: System modeling and partitioning techniques. *ACM Trans. Des. Autom. Electron. Syst.*, 8(3):269–297, July 2003.

[25] M. López-Vallejo and J. C. López. On the hardware-software partitioning problem: System modeling and partitioning techniques. *ACM Trans. Des. Autom. Electron. Syst.*, 8(3):269–297, July 2003.

[26] G. Mariani, V. Sima, G. Palermo, V. Zaccaria, C. Silvano, and K. Bertels. Using multi-objective design space exploration to enable run-time resource management for reconfigurable architectures. In *Design, Automation Test in Europe Conference Exhibition (DATE), 2012*, pages 1379–1384, 2012.

[27] J. McCall, P. Richards, and G. Walters. *Factors in Software Quality. Volume I: Concepts and Definitions of Software Quality*. AD A049. General Electric, 1977.

[28] G. D. Micheli and R. K. Gupta. Hardware/Software Co-Design. *IEEE MICRO*, 85:349–365, 1997.

[29] J. F. Navas, J.-P. Babau, and J. Pulou. A component-based run-time evolution infrastructure for resource-constrained embedded systems. In *Proceedings of the ninth international conference on Generative programming and component engineering*, GPCE '10, pages 73–82, New York, NY, USA, 2010. ACM.

[30] R. Niemann and P. Marwedel. Hardware/software partitioning using integer programming. In *European Design and Test Conference, 1996. ED TC 96. Proceedings*, pages 473–479, 1996.

[31] G. Palermo, C. Silvano, and V. Zaccaria. Multi-objective design space exploration of embedded systems. *J. Embedded Computing*, 1(3):305–316, 2005.

[32] M. Purnaprajna, M. Reformat, and W. Pedrycz. Genetic algorithms for hardware-software partitioning and optimal resource allocation. *Journal of Systems Architecture*, 53(7):339 – 354, 2007.

[33] U. B. K. Ramesh, S. Sentilles, and I. Crnkovic. Energy management in embedded systems: Towards a taxonomy. In R. Kazman, P. Lago, N. Meyer, M. Morisio, H. A. Müller, F. Paulisch, G. Scanniello, and O. Zimmermann, editors, *GREENS*, pages 41–44. IEEE, 2012.

[34] J. T. Russell and M. F. Jacome. Embedded architect: a tool for early performance evaluation of embedded software. In *Proceedings of the 25th International Conference on Software Engineering*, ICSE '03, pages 824–825. IEEE Computer Society, 2003.

[35] M. Saadatmand and M. Sjödin. Towards Accurate Monitoring of Extra-Functional Properties in Real-Time Embedded Systems. In K. R. P. H. Leung and P. Muenchaisri, editors, *APSEC*, pages 338–341. IEEE, 2012.

[36] F. Salewski and A. Taylor. Systematic considerations for the application of FPGAs in industrial applications. In *Industrial Electronics, 2008. ISIE 2008. IEEE International Symposium on*, pages 2009–2015, 2008.

[37] G. Sapienza, I. Crnkovic, and P. Potena. Technical Report: Architectural Decisions for HW/SW Partitioning based on multiple Extra-Functional Properties. Technical report, School of Innovation, Design and Engineering, Mälardalen University, http://www.es.mdh.se/pdf_publications/3132.pdf.

[38] G. Sapienza, I. Crnkovic, and T. Seceleanu. Partitioning Decision Process for Embedded Hardware and Software Deployment. In *In Proceeding of International Workshop on Industrial Experience in Embedded Systems Design, COMPSAC 2013*. IEEE, Jul 2013.

[39] G. Sapienza, T. Seceleanu, and I. Crnkovic. Modelling for Hardware and Software Partitioning based on Multiple Properties. In *39th Euromicro Conference Series on Software Engineering and Advanced Applications*. IEEE Computer Society, Sep 2013.

[40] L. Senn, E. Senn, and C. Samoyeau. Modelling the Power and Energy Consumption of NIOS II Softcores on FPGA. In *Cluster Computing Workshops (CLUSTER WORKSHOPS), 2012 IEEE International Conference on*, pages 179–183, 2012.

[41] S. Sentilles, A. Vulgarakis, T. Bures, J. Carlson, and I. Crnkovic. A Component Model for Control-Intensive Distributed Embedded Systems. In *Component-Based Software Engineering*, volume 5282 of *Lecture Notes in Computer Science*, pages 310–317. Springer Berlin Heidelberg, 2008.

[42] J.-P. Soininen, T. Huttunen, K. Tiensyrjä, and H. Heusala. Cosimulation of real-time control systems. In *EURO-DAC*, pages 170–175. IEEE Computer Society, 1995.

[43] J. Teich. Hardware/Software Codesign: The Past, the Present, and Predicting the Future. *Proceedings of the IEEE*, 100(Centennial-Issue):1411–1430, 2012.

[44] J. Teich, T. Blickle, and L. Thiele. An evolutionary approach to system-level synthesis. In *Proceedings of the 5th International Workshop on Hardware/Software Co-Design*, CODES '97, pages 167–, Washington, DC, USA, 1997. IEEE Computer Society.

[45] F. Vahid and T. Givargis. *Embedded system design - a unified hardware/software introduction*. Wiley-VCH, 2002.

[46] U. van Heesch and P. Avgeriou. Mature Architecting - A Survey about the Reasoning Process of Professional Architects. In *WICSA*, pages 260–269. IEEE Computer Society, 2011.

[47] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén. *Experimentation in software engineering: an introduction*. Kluwer Academic Publishers, Norwell, MA, USA, 2000.

[48] H. K. Wright, M. Kim, and D. E. Perry. Validity concerns in software engineering research. In *Proceedings of the FSE/SDP workshop on Future of software engineering research*, FoSER '10, pages 411–414, New York, NY, USA, 2010. ACM.

[49] J. Wu and T. Srikanthan. Low-complex dynamic programming algorithm for hardware/software partitioning. *Information Processing Letters*, 98(2):41 – 46, 2006.

[50] J. Wu, T. Srikanthan, and T. Lei. Efficient heuristic algorithms for path-based hardware/software partitioning. *Mathematical and Computer Modelling*, 51(78):974 – 984, 2010.

[51] F. Xie and H. Liu. Unified Property Specification for Hardware/Software Co-Verification. In *COMPSAC (1)*, pages 483–490. IEEE Computer Society, 2007.

[52] M. Zeller and C. Prehofer. A Multi-Layered Control Approach for Self-Adaptation in Automotive Embedded Systems. *Adv. Software Engineering*, 2012, 2012.

[53] M. Zeller and C. Prehofer. Modeling and efficient solving of extra-functional properties for adaptation in networked embedded real-time systems. *Journal of Systems Architecture*, (0):–, 2012.

[54] A. Ziani, B. Hamid, and J. Bruel. A Model-Driven Engineering Framework for Fault Tolerance in Dependable Embedded Systems Design. In *Software Engineering and Advanced Applications (SEAA), 2012 38th EUROMICRO Conference on*, pages 166–169, 2012.