Towards a Methodology for Hardware and Software Design Separation in Embedded Systems

Gaetana Sapienza, Tiberiu Seceleanu
ABB Corporate Research
and Mälardalen University
School of Innovation, Design and Engineering
Västerås, Sweden
{gaetana.sapienza,tiberiu.seceleanu}@se.abb.com

Ivica Crnkovic
Mälardalen University
School of Innovation, Design and Engineering
Västerås, Sweden
ivica.crnkovic@mdh.se

***Abstract*—Development of embedded systems in automation industry often includes development of both software and hardware, which requires both software and hardware expertise. In the current practice these expertise are not often completely combined in synergic ways. Traditionally, design gets separated into hardware design and software design at very early stage which negatively impacts the overall application development process due to design flow interruption and redesign. In order to overcome to the aforementioned problems, this paper presents a new design methodology that provides platform independent design first, and pushes hardware- and software-dependent design to a later stage. This enables "software-independent" hardware and "hardware-independent" software development after the separation stage, which collectively improve the overall development process.**

Keywords: Development Process; Design Methodology; Partitioning; Multi Criteria Decision Analisys (MCDA).

I. INTRODUCTION

The continuous increase in complexity of embedded industrial applications constantly demands improvements of the overall development process. Ideally, the development process has to be able to simultaneously satisfy two main driving requests imposed by the today's market trends: (i) significantly decreasing time-to-market, and (ii) significantly decreasing development and product costs, while preserving quality and launching high-competitive products. In addition to the above, the technology advancements in semiconductor and electronics fields in a combination with the growing demands of providing more sophisticated software functionalities constantly challenge the design methodologies in order to improve the overall development process [1]. Due to the intrinsic nature of embedded systems i.e., the tight coupling between hardware and software, the development process is extremely affected by the efficiency of the design phase which has to rely on methodologies that are able to integrate key paradigms of hardware design and software design in an effective manner.

Traditionally, the system design starts with a separation of software and hardware design [2] at an early stage of the development process. The common practice of the separation into hardware and software is an iterative process, approached in a manually controlled "trial and error" mode, which is not supported by suitable and effective tools or systematic decision process. The hardware-software separation is typically done by invoking individual back-end tools several times in order to later decide which architectural solution appears to be the most suitable one. This approach, unfortunately, is prone to negatively affect the overall application development process due to e.g., issues such as flow interruptions and redesigns.

In this paper, we present a new systematic design methodology which enables hardware and software design separation as late as possible after the overall specification and design activities and a well-structured decision process. The approach is inspired by Model-Driven Architecture with Platform-Independent Model (PIM) and Platform-Specific Model (PSM) stages [3]. PIM identifies software functions independent of the underlying technology, while PSM defines technology-specific solutions. Our approach focuses on the specification and design part of the system valid for both software and hardware (the PIM part), and the design specifically for software and hardware (the PSM part). By doing this, when designing software and hardware specific parts, it is possible to minimize the dependencies between hardware and software after the design separation. Specifically, the proposed methodology will be applied in embedded applications targeting the automation domain. The concepts highlighted in this paper, are supported by years of experience in industry with design methodologies and embedded systems development. The remainder of the paper is organized as follows: the next section discusses the current state of practice for embedded application design in the automation industry domain. Section 3 describes the new proposed design methodology. Section 4 describes a case study. Section 5 concludes the paper and future work is outlined.

II. THE CURRENT STATE OF PRACTICE

A typical software-hardware industrial development process can be described as a number of sequential phases [2][4]: requirements management and system specification, design, implementation, verification and validation, as shown by the diagram A in Figure 1. The development process starts with the specification phase in which requirements are supposed to be identified and analysed. After the specification phase, the design phase usually branches into two separated design flows, for hardware and software, respectively. These flows evolve separately and get into their own implementation. When both hardware implementation and software implementation are completed,

 ISBN: 978-1-61208-230-1

the integration takes place. Subsequently the verification and validation phase get in progress. The diagram A depicted by Figure 1 represents a rather simplified development process flow. In reality, it is more complex: phases get interleaved and each of them might require be iterated and/or optimized several times over the entire development process. Consequently, this process meets several serious problems and drawbacks. We describe them briefly for each phase.

A. *Early start of the design phase*

During the specification phase, and before starting with the design phase, the requirements are expected to be fully finalized in order to efficiently support the design phase. However, in practice due to time and resources constrains the design phase is enforced to start before the requirements have reached a reasonable mature and stable stage. The incompleteness of the specification negatively affects the quality and fluidity of the design phase, and also contributes to originate the issues subsequently described.

B. *Early separation into hardware and software*

Despite the fact that hardware and software for embedded applications are tightly connected, typically the design phase splits very early into the two design flows. After the separation, hardware and software are considered as two separated activities which are seldom integrated until the integration and verification phases. In principle, (i) hardware does not take into account the computational power required by the software and the capability that the software might offer for enabling hardware optimization and (ii) software does not impact the hardware design specifications, and does not fully exploit the available hardware resources. The too early design start corresponding to the too early flow (hardware and software) separation, does not allow to properly focus on the most important and core part of the design phase which is referred in this paper as *partitioning decision process*. This process is supposed to determine which parts of the application will be designed in hardware and which parts of the application will be designed in software. Problem statement on the partitioning problem can be found in [5].

The impact of the initial decisions is critical since it will condition the remaining development process and the entire application's lifecycle; as a consequence, any decision change afterwards is arduous and costly. Starting a design phase relying on an apparently appropriate set of partitioning decisions potentially poses higher risks for the successful accomplishment of the application development process.

Although the modern design tools (e.g., The MathWorks Simulink®, IBM® Rational® Rhapsody® (UML (Unified Modelling Language)-based tool)) support well the "trial-and-error" approach, in practice the problems remain since a systematic decision process with the appropriate support is missing. Due to the aforementioned aspects related to the early start of the design, it can be highlighted that the development process (as represented by the diagram A in Figure 1) is negatively impacted in terms of quality, costs and time by the following emerging problems: (i) hardware or software flow interruptions and (ii) hardware or software redesigns.

C. *Hardware or software design and implementation interruptions*

Hardware or software design flow interruptions are observed as a break in the continuity of the design flow, due to the (partial) lack of specifications that have impact on the partitioning. The diagram A in Figure 1 shows a representation of the flow interruptions for both hardware and software. They are undesired since causing an increase in the complexity in the design flow, while affecting the overall quality. The first interruption occurs in the hardware design flow the second interruption occurs during the software implementation.

D. *Hardware or software redesign*

The need of performing redesign (either hardware or software) is usually dictated by reasons of different nature, e.g., new requirement/s, requirement/s changing, non-feasibility of requirement/s, lack of application-specific knowledge, etc. In literature, research work discussing redesign issues for embedded systems can be found in [1][6]. The hardware and software redesign process is illustrated on the diagram A in Figure 1. It may happen during the initial design, or it can be required after the implementation. It represents one of the most typical scenarios of redesigns encountered in practice: "redesign after implementation" caused by a very late integration of hardware and software. In diagram A in Figure 1, the hardware redesign is caused by the non-feasibility of the requirement A (Req_A) which leads to the necessity of the software redesign due to the non-fulfilment of the requirement (Req_B).

III. THE NEW APPROACH PROPOSAL

Given the current state of practices in automation industry, we present a new systematic design methodology able of minimizing or even overcoming the issues described above. Our proposal is a process which is mainly characterized by the following key features: (i) providing support/feedback to the specification phase, and (ii) starting with a model-based design common for both software and hardware and continuing with its separation to software-specific and hardware-specific design process when collecting all artefacts that enable software-independent hardware design and hardware-independent software design separation, as depicted in Figure 1 by the diagram B. The explanation of the key features is subsequently done through the description of the proposed approach and the overall overview presented in Figure 2.

The approach is divided into three essential stages: *Identification*, *Decomposition* and *Partitioning*. The Identification stage provides inputs to the Decomposition, while the Decomposition provides inputs to the Partitioning stage.

 ISBN: 978-1-61208-230-1

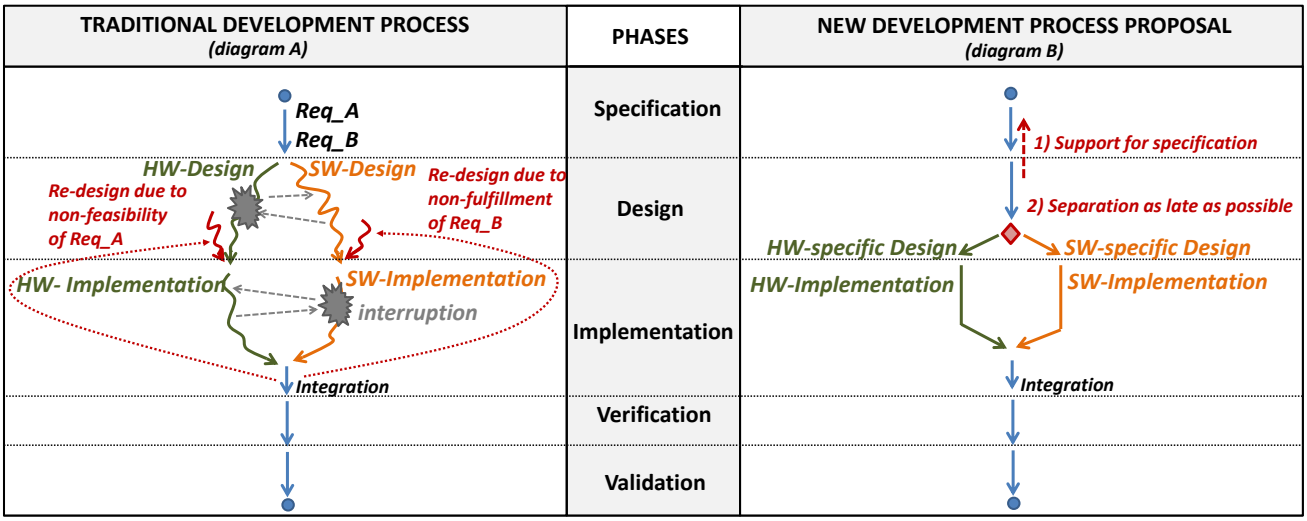

Figure 1. Traditional application development process (A) and the development process proposal

A. *Identification of key design criteria*

In our experience, a design methodology tailoring industrial automation applications has to able of meeting and efficiently trading-off a number of design boundary conditions deriving from: stakeholder concerns, technology and feasibility studies, functional and non-functional requirements, human factors (e.g., expertise, knowledge, etc.), constraints (e.g. legacy, reuse of existing platform, tool chains, manufacturing platforms and cost, etc.), technology advances in semiconductors and software, domain-specific features. As a consequence, it is crucial that all of these boundary-conditions are identified and carefully evaluated before starting the separation into hardware or software. By our approach they get identified and mapped into a set of *key design criteria*. Later they serve as inputs for supporting the subsequent stage of application decomposing and allowing the application to go through a decision process, as shown in Figure 2.

In order to define the set of key criteria, an accurate analysis of several design processes related to the application domain from different perspectives (performance, timing, overall quality, costs, etc.) will be performed. In details, it will be performed by the following steps:

1) Extrapolation of the mentioned boundary-conditions highlighting the relation with the design decisions in order to identify patterns like:
   a. the most high- impact decision choices,
   b. the most frequently adopted decision choices.
2) Classification of the above extracted design boundary conditions in relation to their hardware or software features. It is important to highlight what the cause-effect relations are in the entire design process.
3) Study to assess if and how well the design matches the required specifications, referred as design-specification matching for brevity. Interest will be also focus on biased decisions, to get a systematic interpretation of their impact on the overall design.
4) Identification of the criteria driving the strategic choices in the design.

In addition to establishing the motivation for decisions in the application development process, the key identified criteria will further provide guidelines for the refinement of the specifications. The analysis targets to gather a number of information related to the entire application life-cycle process (modelled by the extended V-Model in Figure 2) which in combination with the key identified criteria serve to complement and provide a systematic feedback to the specification phase.

B. *Application Decomposition*

Assuming that from a high abstraction level the application is modelled as a number of components, we propose an application decomposition process that extracts the elementary functionalities of the application and further refine the selection to the point in which the hardware or software implementation features of each component will be fully defined.

The proposed approach is based on both the analysis of the application specifications as well as the key design criteria identified in the previous stage. We propose a 2-step analysis:

1) Identification of the functionalities that directly matches the application specifications;
2) A decomposition of the identified functionalities in components strictly characterized by the key design criteria, and ready for the partitioning phase.

The above discussed decomposition strategy is supported by the diagram depicted in Figure 3. In practice, the two identified steps will be implemented using the following methodologies: (i) analysis of the application requirements and generation of specific functional components constituting different hypotheses of coarse-grained components suitable to be represented through well-known existing model-driven based tools like: The MathWorks

 ISBN: 978-1-61208-230-1

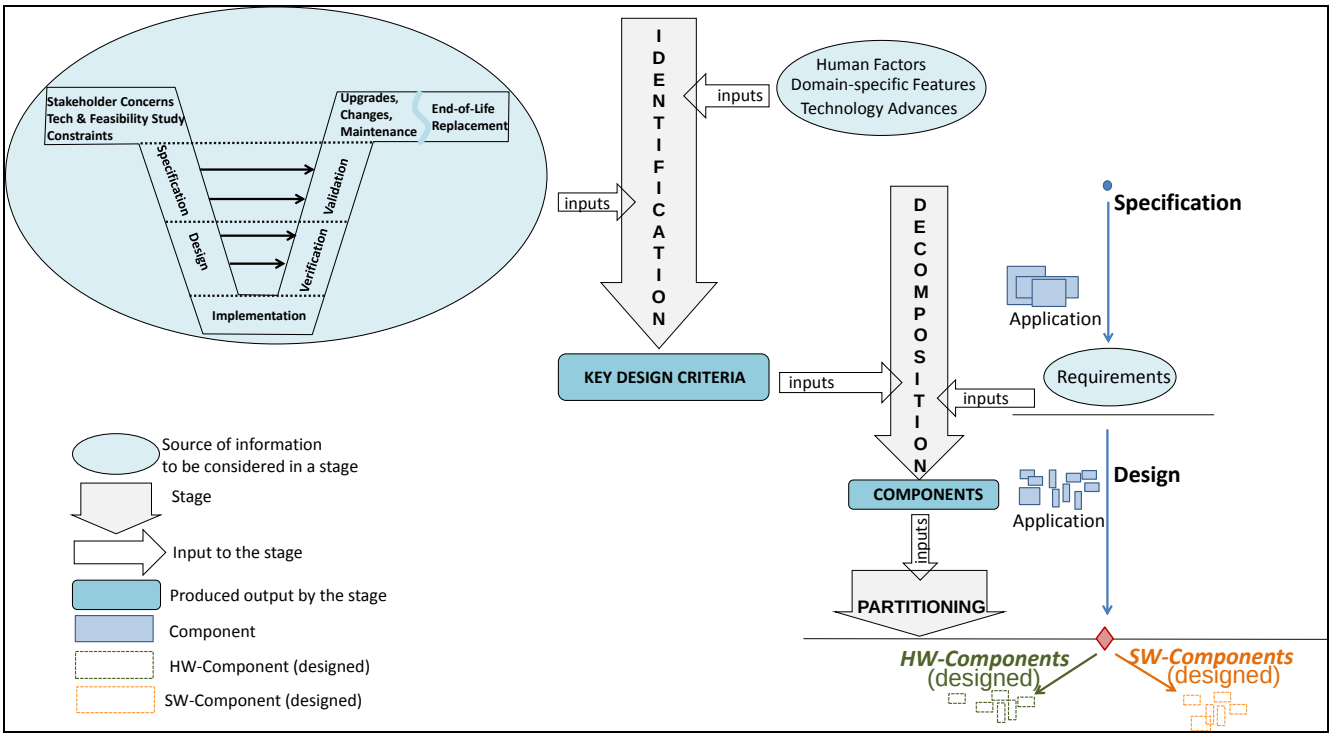

Figure 1. Relation between the proposed approach and V-Model and the Identification stage. Identification, Decomposition, Partitioning Flow

Simulink®, IBM® Rational® Rhapsody® (UML-based tool) etc., and (ii) the initial selection of coarse-grained components will be further decomposed through the key design criteria characterization to bias the generated components towards implementation issues, to create a well-posed problem as inputs to the subsequent decision process. Hence, all generated subcomponents will be strongly characterized by the key design criteria involved.

C. *Partitioning Decision Process*

Despite classical partitioning schemes that have been proposed in the past [7][8] which treat the problem as a nondeterministic polynomial problem to be optimized, we propose to face it as Multi-Criteria Decision Analysis (MCDA) problem. Unlike the approach proposed by [9] we do not intend to use MCDA for ranking the choice, but for targeting the design partitioning decisions in an efficient way. The choice of using such approach is driven by the variety and quantity of design decision criteria that require to be taken into account and their strong inter-dependencies. In addition to the above it is also motivated by the need of having a full traceability of the decision process. An intuitive and transparent procedure for generating the decisions is of crucial importance for studying the sensitivity of the design criteria in the overall decision process.

Additionally, in case of issues such as redesign or interruptions, caused by incompleteness or misleading of the specifications, it is possible to back-propagate the error and identify the major source of the unexpected behaviour in order to effective adapt the design strategy to further re-iterations. Further, the design feedback provided to the specification, enables of the hardware-specific and software-specific design separation as late as possible through the combined effects produced by the Identification as well as the Decomposition stage. Using the key design criteria for guiding the stepwise component discretization, implicitly allows the possibility of accumulating the required energy (i.e. in form of key components information) to start, after separation, design hardware and design flow where the dependencies are minimized. Furthermore, by performing the partitioning, after that the decomposition stage is completed, the set of components have been fully analysed and characterized, which consequently decreases the probability of assigning components to hardware or software based on wrong poses assumptions.

IV. TOWARDS TO AN INDUSTRIAL APPLICATION CASE STUDY

The status of this case study is referred to the context of the two first phases of the extended V-Model (i.e., Specification and Design) as well as the Decomposition stage discussed in Section III.B.

In order to verify and validate the proposed methodology, we started working on the specification and design of a wind turbine application that is supposed to be deployed in an industrial prototype within the integration framework specified and developed by the Artemisia iFEST (industrial Framework for Embedded Systems Tools) project [10]. The main purpose of the application is to convert the rotational mechanical energy of the rotor

 ISBN: 978-1-61208-230-1

blades caused by the wind into electrical energy to be redistributed via a power network.

A core component of the application is represented by the wind turbine controller, which has to be able of providing the dynamic regulation of the rotor blades at different wind profiles while maximizing the production of electrical energy. In parallel it has to be able of supervising the entire transformation process such as to guarantee the proper overall functioning of the wind turbine and minimizing any risk of damage to the physical wind turbine system.

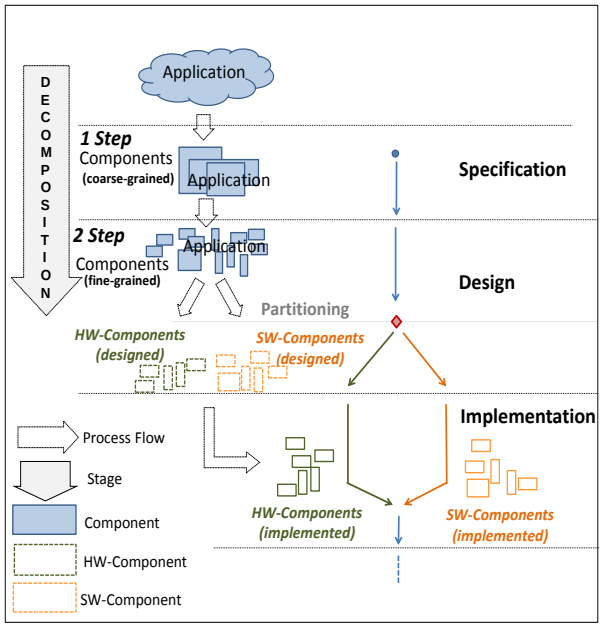

Figure 2. The Application Decomposition Process, 2-step analysis.

We intend to implement the design on several platforms, providing both for software (single and dual-core processors) and hardware (FPGA - Field-Programmable Gate Array) solutions. As tools used in the process we have chosen:

- HP ALM (Hewlett-Packard Application Lifecycle Management): for the specification and analysis phase.
- The MathWorks Simulink®: mostly for the design phase but also for Verification and Validation (simulation), and for the implementation (translation of design into C and VHDL).

According to the development process flow as well as the part of application decomposition process described above, we started with the specification phase. In order to go through the first step of the application decomposition process described in Section III.B, we took into account all of the info depicted in Figure 2, for instance the domain-specific features, the constraints, the stakeholder concerns, etc. Few examples follow:

- Domain-specific features: the application has to provide control functions allowing pitch regulation; the application has to be standard-compliant (i.e., IEC-61400); power network disturbances, etc.
- Constraints: the application has to be implemented into hardware and software; the implementation has to integrate legacy C-language code parts; the application has to allow the firmware to be field-upgradeable
- Requirements: time constraints for operations; reaction time at system failure, ambient temperature and relative humidity values; normal and extreme electrical conditions; safety procedures, etc.
- Stakeholder concerns: the project has to be able of delivering a high quality product with short time-to-market to pay-back the development cost and have a large margin profit.

After this first step, the identified key functionalities were mapped into components: (i) the pitch regulation, and (ii) the supervision.

In addition to this, we also identified the need for diagnostic and filtering functionalities. The components were modelled by using Simulink. The outcome of the mapping of the specification into the design is presented in Figure 4. It shows a two-level decomposition of the wind turbine application into components, which is achieved by the analysis of the application requirements. Level A models the Wind Turbine Plant and the Wind Turbine Controller. Level B shows a further decomposition of the Wind Turbine Controller component into four components: the Pitch Regulator, the Supervision, the Filtering and the Diagnostic.

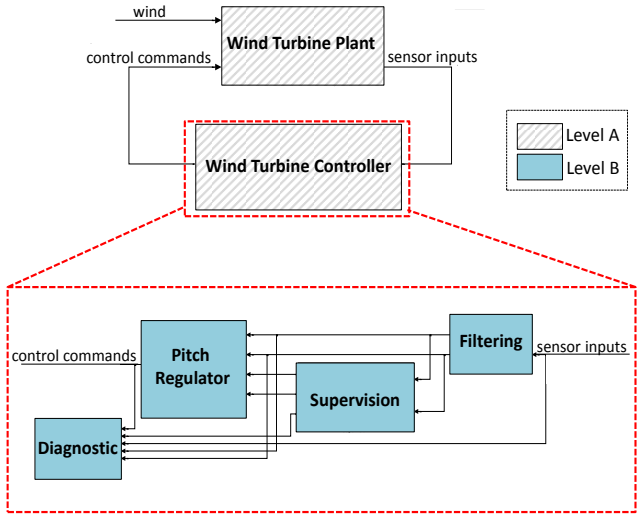

Figure 1. Wind Turbine Model (Plant and Controller). Decomposition of the Wind Turbine Controller (2-level).

What we have presented above is the first step of the application decomposition process. The next step will be to achieve a more detailed design decomposition of the application, as described in Section III.B. After that, the application will be applied for a multi-criteria decision

 ISBN: 978-1-61208-230-1

process in order to decide about which components will be implemented in hardware and in software. Subsequently the application will be deployed into several platforms in order to evaluate the proposed new approach.

V. CONCLUSION

The paper presented a proposal of a new systematic design methodology that is able to improve the overall process from the design perspective as well as from the application lifecycle perspective. It consists of three main stages: Identification, Decomposition and Partitioning, that collectively drive through the definition of the main methodology characteristics such as the support towards the specification phase and the enabling of software-independent hardware design and hardware-independent software design separation.

The next step of our research work is defining which requirements the MCDA approach has to fulfil in order to support the aforementioned partitioning process. Subsequently, we will analyse if any already available MCDA method (or a combination of more MCDA methods) is able of meeting the identified requirements and can be applied for the application partitioning. After that, we will focus on the identification and formalization of the key design criteria to use as the set of inputs (i) for guiding the fine-grained application decomposition and (ii) for supporting the partitioning process into designed hardware and software components as described by Figure 2. As final step, the proposed methodology will be evaluated on the above presented industrial application case study.

ACKNOWLEDGMENT

This research is supported by the Knowledge Foundation (KKS) through ITS-EASY, an industrial research school in Embedded Software and Systems, affiliated with the School of Innovation, Design and Engineering (IDT) at Mälardalen University (MDH), Sweden.

REFERENCES

[1] P. Koopman, “Embedded System Design Issues (the rest of the story)”, Proceedings of IEEE International Conference on VLSI in Computers and Processors, Oct. 1996.

[2] A.S. Berger, "Embedded Systems Design: An Introduction to Processes, Tools and Techniques", CMP Books; 1 edition, Dec. 15, 2001.

[3] A. G.Kleppe, Jos Warmer, Wim Bast, “MDA Explained: The Model Driven Architecture: Practice and Promise”, Addison-Wesley Professional, 1 edition , May 1 2003.

[4] H.Van Vliet,"Software Engineering: Principles and Practice", Wiley, 3 edition, Jun 27, 2008.

[5] G.De Micheli, R.Gupta, “Hardware/Software Co-Design,” Proc. of the IEEE, vol. 85, No.3, 1997, pp.349-365.

[6] C.Coelho, C.Yang,V. Mooney, G.De Micheli, “Redesigning hardware–software systems,” in Proc. 3rd Int. Workshop on H/S Codesign, Grenoble, France, Sep. 1994.

[7] Y.Fan, T.Lee, "Grey Relational Hardware-Software Partitioning for Embedded Multiprocessor FPGA Systems", AISS: Advances in Information Sciences and Service Sciences,vol. 3, No. 3, 2011, pp. 32 - 39.

[8] M.L.Vallejo, J.C.Lopez, ”On the hardware-software partitioning problem: System Modeling and partitioning techniques”, ACM Transactions on Design Automation of Electronic Systems (TODAES) vol. 8, Issue 3, July 2003, pp. 269 – 297.

[9] P.Garg, A. Gupta, J.W.Rozenblit, "Performance analysis of embedded systems in the virtual component co-design environment", Proceeding of the 11th IEEE International Conference and Workshop on the Engineering of Computer-Based Systems, May 2004, pp. 61-68.

[10] iFEST. iFEST - industrial Framework for Embedded Systems Tools. ARTEMIS JU project #100203. Retrieved September 26, 2012, from http://www.artemis-ifest.eu/

 ISBN: 978-1-61208-230-1