

The Partitioning Methodology in Hardware/Software Co-design Using Extreme Programming: Evaluation through the Lego Robot Project

Heeseo Chae, Dong-hyun Lee, Jiyong Park and Hoh Peter In
Department of Computer Science & Engineering, Korea University,
{royalhs, tellmehenry, jayyp, hoh_in}@korea.ac.kr

Abstract

This paper argues about the partitioning in hardware/software co-design and suggests the methodology applying extreme programming to complement the co-design. This approach, contrary to complex method in the existing development environment of embedded systems, arises from need of agile methodology in latest development environment which requires time to market. Through adding a guide line of extreme programming to the advantage of co-design, the synergy effect of general process is expected. The risk management such as the cost management and the time management is suggested on the basis of PAMUX (Partitioning Methodology Using eXtreme Programming) and also the action plan such as the group organization and process establishment is suggested on the basis of the methodology which increases the reliability of partitioning. Finally, traits and case study of suggested methodology are evaluated by Lego robot project and it allows the PAMUX to add reliability.

1. Introduction

Designing hardware of embedded system is more difficult affair than attaching the proper processor to some peripheral devices. It is core of making embedded system what kind of division approaches are applied in order to implement the function of hardware and software. This part is not clearly decided by any research method, so continuous research is needed. Undoubtedly, there are no hardware designers who select the processor or architect the some hardware without consideration. However, in fact, there are frequent cases in many R&D laboratories that choosing the processor, designing the hardware and transferring it to software group without communication or interaction. This buck passing approach brings many risks in many ways such as cost

and time, so it makes the flexible progress of project difficult [1].

HW/SW co-design is approach which emphasizes the sameness of development process of hardware and software through the concept of collaborative design and mutual verification. This method comes into the spotlight according to the technological development of embedded designing process and it also emphasizes the advantage of incipient HW/SW integration.

Likewise, remarkable approach among latest development methodology, eXtreme Programming (XP) is one of the agile software developments, which can delivery the best value to customer as soon as possible on the basis of principles such as simplicity, mutual understanding and feedback [2]. The XP methodology is a suitable approach in latest development environment which emphasizes time to market and it can bring the positive change to the process of embedded development.

Through the XP methodology is applied to HW/SW partitioning stage which is the most important stage in the life cycle of HW/SW co-design, it can help the decision of division stage (partitioning stage) in order to manage the cost, development time and risk. Also, through the XP methodology suggests the guide line to co-design process which is noticed but has no definite action plan, it can support the introduction of detailed HW/SW design stage and change the current view of partitioning in order to success the embedded project.

2. Related Work

This chapter describes the general architecting method and life cycle of embedded system which can be applied by HW/SW co-design. XP methodology which can give the guide line to embedded designing method is also described.

2.1 Life Cycle of Embedded System Design

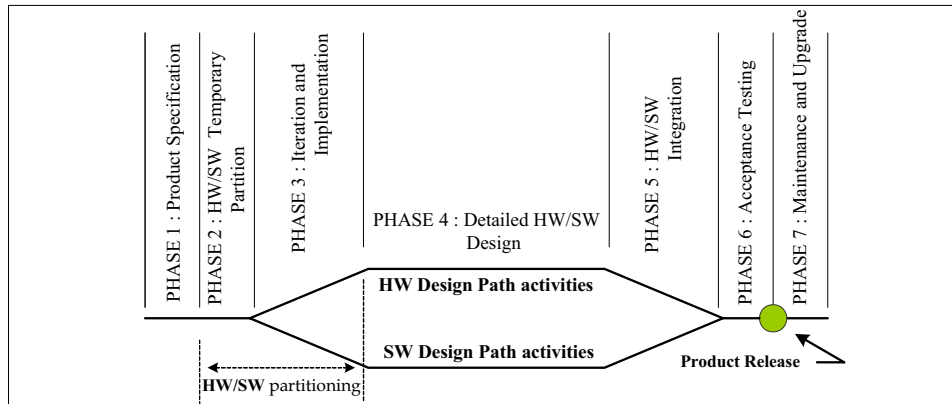


Figure1. Life Cycle of Embedded System Design (revised [1])

Contrary to designing applications in the standard platform, architecting software and hardware is progressed at once in the embedded system [1].

Fig.1 is typical life cycle of embedded system design. The passage of time is left to right. This is the simplest figure and it can be more detailed by iteration and optimization.

Life cycle of embedded system design is composed by seven phases such as figure1. The first phase is specifying products, next phase is temporary partitioning the hardware and software, the third is iterating and implementing phase and the fourth phase is called detailed HW/SW designing. Also, the fifth and the sixth are HW/SW integration phase and acceptance testing phase. Finally Maintenance and upgrade phase is the seventh phase.

The important part of the life cycle is iterating and implementing phase which is previous phase of firm HW/SW decision. The phase composes the part of HW/SW partitioning with HW/SW temporary partitioning phase, and locates itself on the previous phase of detailed HW/SW designing. It is known to optimizing part through iterative implementation and testing before the detailed design and development of each part in complex embedded system. For example which can appear the importance of this phase, if the critical error is detected in testing process of product, the algorithm will be rewritten or hardware on demand such as ASIC (Application-Specific Integrated Circuit) will be designed again. Also, processor may be changed by faster one or new one [1].

2.2 The Feature of eXtreme Programming

The eXtreme programming (XP) is a agile methodology which prefers immediate actions and instant process. One important feature of XP is pair

programming which can review all of the code and test by mating a developer. At this time, in case of need, customer can participate in the test. Through the refactoring, designing systems are operated in association with all the developers who are not take charge of the particular affair, and it can make steady progress of iteration and test without disturbing. Also, with the concept of planning game, the iterative plan is established in short term which are not cycle of years or cycle of months but cycle of minutes or cycle of hours, and it can check the degree of satisfaction and prospect the period of development. Namely, XP seems to spiral model which has a small radius, and integrating of system which is made up short interval is preparation for the instant testing and the forward movement to next phase. Finally, Metaphor, co-ownership image of the whole system concept, suggests the approach of considering and developing projects with developers, customers and managers.

The feature of XP is described as follows:

Table 1 The Feature of XP [2]

Commonsense	XP extreme	XP implementation on practice
Code review	Always Code Review	Pair programming
Testing	Always Test	Unit Testing Functional testing
Design	Fixed portion in a day is allotted system design	Refactoring
Architecture	All the participant refine architecture	Metaphor
Integration testing	Many Integration and test in a day	Continuous integration
Short	Short Cycle of	Planning game

iterations	iteration - hours minutes, second
------------	--------------------------------------

3. PAMUX

Our PAMUX adopts the advantages of the two methodologies in the previous section in order to improve co-design processes and overcome critical points. It offers an optimized method to embedded systems by complementing extreme programming and the traditional HW/SW co-design methodology.

3.1 Organization of HW/SW Group

PAMUX is an improved co-design methodology which can reduce costs and development time through adapting XP into the iteration and implementation phase (phase 3 in Fig. 1). Applying PAMUX when developing an embedded system, it is need to organizing hardware and software groups enough to apply extreme programming as shown in Fig. 2.

Legacy development methodology does not provide a clear guidance of hardware-software partitioning, but PAMUX offers a distinct guideline imported from extreme programming, so this can remove uncertainty of system specification and find errors effectively. It can also make possible testing real development codes not a virtual hardware at some development times. Eventually, it is possible to improve hardware-software co-design and co-verification by organizing hardware and software groups

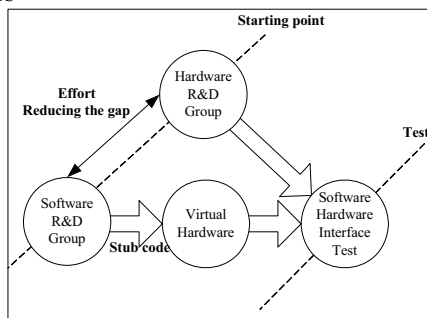


Figure2. Organization of HW/SW Group

In Fig.2, software developers use stub codes for virtual hardware testing without a real hardware test-bed, and hardware-software interface tests can start when the real hardware test-bed have been constructed.

In the legacy co-design methodology, the more hardware/software error detection in testing process is delayed, the more time to correct errors spends.

So, in the development cycle, development costs of a project can be reduced by providing verification and of hardware and software and co-work environments of hardware and software team.

3.2 PAMUX Process

PAMUX provides a process in which partitioning process is iterated before implementing details of whole system, under the environment that hardware and software groups can co-operate effectively, as mentioned in section 3.1.

This process is called PAMUX process, which can confirm hardware-software partitioning, establish the direction of a project, and reduce delay of the project by detecting and correcting errors at the later stage in development cycle.

The PAMUX process is shown in Fig. 3, which describes characteristics of extreme programming, first, development cycles per small, compact and complete units, second, frequent tests. These characteristics help to determine partitioning time and range with high reliability.

The PAMUX Process has following steps:

- **Architecting:** establish system metaphor and start to discuss architectures
- **Planning Game:** A step for the range of hardware and software roles. Small units of tasks are ordered and prioritized in development or integration sequences.
- **Iteration:** obtaining reliable partition result by iterating hardware-software integration and test
- **Release:** delivering partition results
- **Acceptance Decision:** starting to design details after partitioning range and confirmed role are delivered

Though architecting and planning game are in phase 2 and release and acceptance decision are in phase 4, these steps can be considered as steps of PAMUX process because these steps are in the series of hardware-software partitioning processes.

Iteration step is composed in forward engineering, reverse engineering and integration, as shown in Fig. 3. Forward engineering is a sub-process in which hardware and software group develop their projects and test with small units and intervals, while reverse engineering is a step in which feedbacks of a system or a unit constructed in the previous step and communications of related parts are. In integration sub-process, ideas of each group are tuned and integration environment constructed up to the current step is tested.

The three sub-processes are tested iteratively along with the goal of phase 3, integrated again, and results of these refactoring are delivered to phase 4, detailed HW/SW design.

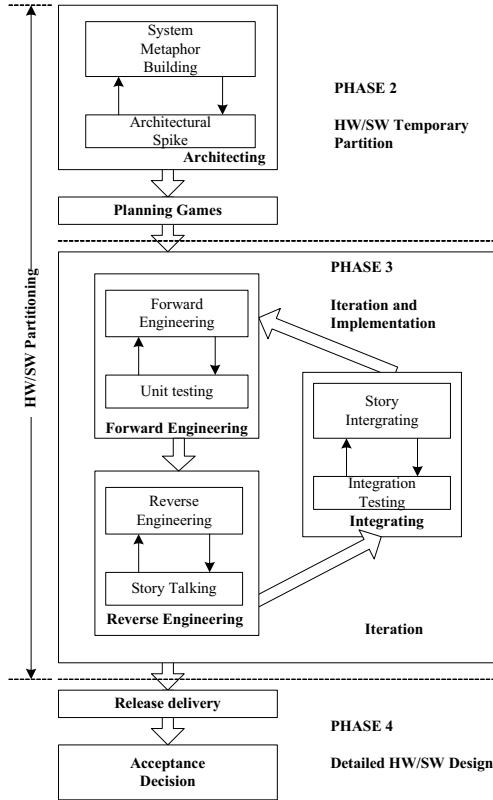


Figure.3 PAMUX Process (revised [3])

3.3 PAMUX Practice

As practice items of extreme programming, PAMUX also needs the fitness training of various practice items which meets the goal and character of project and group. In other word, practical procedures and guide lines are suggested to co-design theory which has no guiding principle. Through the practical procedures of each allotted process definition, HW/SW group can continuously train practical procedures which are fitted each role. Those matters can be realized by executing the PAMUX practical procedures which are simplified as figure4.

The life cycle which arrays practical procedures of XP and their practice method in flow of time offers concrete guide line which can solve the fundamental problem of HW/SW co-design. Fig.4 fitness training of PAMUX reduces the time which is consumed in embedded system designing and the cost which is spent in correcting the defects and they also become criterions of project accomplishment. Each practical procedure in life cycle is described as follows:

- **Exploration:** It is the first fact of practical

procedures; project is started by concept of metaphor.

- **Planning:** It is similar fact to planning game in XP. The scope of group is set and requirements of each group are assorted. Then, the next fact is prepared.
- **Iterations to Small Release:** The development of short interval is progressed in each group and the result of development which is smallest credit but complete unit (the testing is possible) is produced.
- **Producing:** the fact which includes the important practical procedures of XP also produces the development results through the mutual verification among groups.
- **Maintenance:** this fact maintains and mends not only development procedures but also process. The product can be already produced and it assists the progress of code.

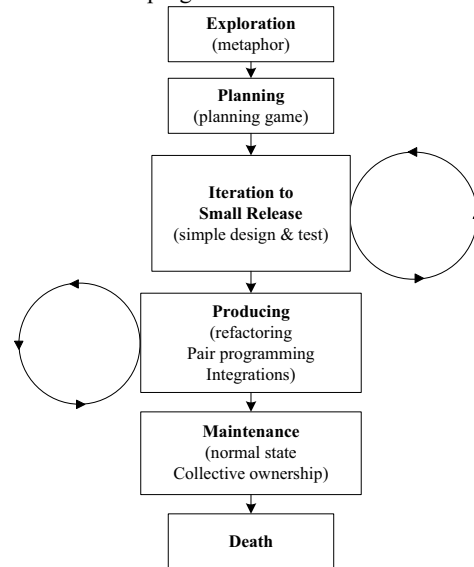


Figure.4 Life Cycle of PAMUX

The practical facts include each detailed procedures. These practical procedures can help to advance the project and, in the case of PAMUX execution, it is possible to learn the concrete methodology [7].

Table 3 Practical Procedures of PAMUX

Practices of PAMUX	Practical Group	Procedure and Character
System Metaphor	HW/SW	Offering to rough sketch about system for various components, showing the architecture how to interaction between components..

Planning Game	HW	Remained story card of system requirement which have to do are selected when planning game.
Small release	HW/SW	Short and frequently Release. Communication and feed back between HW/SW groups are realized. It reduces the error.
Testing	HW/SW	Always verify the HW/SW. Through the approachable test among each group, show the requirement each other and correct the state of requirement.
Refactoring	HW/SW	There is no change of exterior function but design of internal code and structure are progressed Refactoring is operated continuously in development process to improve design gradually and reduce the entropy of code.
Pair programming	HW/SW	Two persons develop the project in one development environment It does not mean simple co-work and it also shows synergy effect.

4. Experiment

The chapter 4 is compared with the PAMUX and a convention co-design using the Lego robot project. Also, usefulness of the PAMUX is validated.

4.1 Experimental Setting

The experiment is accompanied by four teams in Korea university students (2005year CSCE316 class). It is a part of the robot project of artificial intelligence in sensor network. The goal of the project is that Lego robots find a path efficiently through ad-hoc communication each other.

The four team part hardware and software components and develop in three Lego robots based the co-design methodology. Then, the two teams focus partitioning using extreme programming and others develop the project using the conventional co-design.

The main parts of Lego robot in the experiment is divided the drive part, network part, and control part (table 4).

Table 4 Lego Robot Organization

HW	Drive part	Network part		Control part
		Communication interface in robots	Sensed information collector	
SW	MIT Yellow Brick Logo (MIT-YBL) application			

The drive part load a drive gear of operating a Lego robot and the control part is executes the finding path

algorithm and manages the robot. And also the network part composites a communication interface in robots each other and a sensed information collector in environment. The experiment goes along with the two project groups divided the hardware part organized three parts and the software part developed in MIT-YBL language.

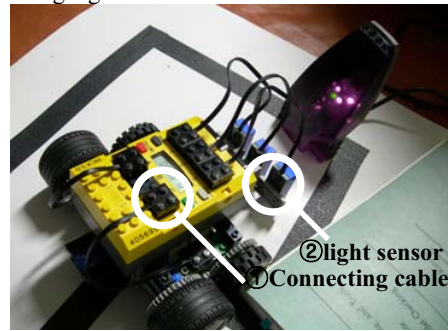


Figure 6 The Path Finder

The co-design part obtain efficiency in the possible part of changing hardware to software and XP partitioning methodology focus communication with hardware and software groups in subdivision of partitioning process. So, the specificity of HW/SW co-design in the experiment differed from memory bus cycle in previous chapter is listed:

- **Control part partitioning:** The control part can be covered by software components through changing a number of connecting cable or block arrangement. Development cost and error rate is reduced by communicating about these changing issues between hardware and software development groups (figure 6-①).
- **Communication part partitioning:** The communication part can confirm the importance of opinion decision between two groups by changing a number of light sensors or allocating a part of information collector to one robot among three Lego robots and expanding communication application between robots instead of installing a information collector in each robots (figure 6-②).

The PAMUX helps communication of partitioning and offers a guide line made possible test based development process.

4.2 Experimental Results

The result of the experiment is derived from four student teams in the finding path project. Figure 7 is a part of task cards helped systematic communication in the project.

DATE:	2005-4-29		
STORY NUMBER:	1.2.4-3		
SOFTWARE ENGINEER:	Park, S. Y. J.	TASK ESTIMATE:	good
TASK DESCRIPTION	Escape Maze by Intelligent Lego Robot		
SOFTWARE ENGINEERS NOTES	There are many changes to be done in source codes But robot sensor is very sensitive to amount of lights, so it is appropriate to change to touch sensor.		
TASKTRACKING			
Date	Done	To Do	Comments
2005. 4. 16	main-gene program	robot programming	
2005. 4. 20	robot program.	Sensor	
2005. 4. 29	Sensor	Test	

Figure.7 A Task Card

We compare the PAMUX concept project and the conventional project using the result of the experiment.

Table 5 Comparison of projects

process feature	PAMUX Concept Project		Conventional project	
	TEAM1	TEAM2	TEAM3	TEAM4
Productivity	143 LOC/day	152 LOC/day	102 LOC/day	100 LOC/day
Error rate (LOC)	0.04 KLOC	0.08 KLOC	0.4 KLOC	0.3 KLOC
Sum of term / person	12 week	13 week	23 week	20 week
Project term	3 week	3.5 week	6 week over	5 week over

Documents of productivity, error rate, sum of development term per person, and project development term are grasped in milestones of project teams. Though the project term is only task per week of five days unit, a development term is a short period project differed from the field.

The table 5 shows that productivity is improved and error rate become less in the PAMUX concept project than the conventional project. The reason is that the PAMUX makes up for insufficient communication with developers and lack of error test, and add XP constituents; communication of hardware, software teams and pair programming, etc.

5. Conclusion

This paper presented the discussion issue of partitioning of HW/SW co-design using XP

methodology and PAMUX. Using XP techniques in iterative design life cycles and implementation phase, our proposed methodology raises the credibility of partitioning decision very critical especially in the perspectives of managing cost, schedule, and risk. Besides, group organization, process establishment, and execution strategy are presented to show an application and credibility of the proposed methodology through the LEGO project.

As future work, more case studies applying XP methodology to various HW/SW integrating co-designs are required to refine and complement the process of XP partitioning decision methodology. Additionally, continuous researches for an assurance methodology are necessary to increase the success rate of a project and decrease the risk factors of it with the base on quantitative evaluation skills and efficient integration processes. H/W verification is needed to complement the weak point XP cannot cover so as to establish the credible process.

6. Acknowledge

We appreciate software engineering (CSCE316, 2005 Korea Univ.) class students who took part in the experiment.

References

- [1] Arnold S. Berger : "Embedded System Design", CMP Books, 2002
- [2] Mark C. Paulk, "Extreme Programming from a CMM Perspective", Software, IEEE Volume 18, Issue 6, Nov.-Dec. 2001 Page(s):19 - 26
- [3] Bin XU, "Extreme Programming for Distributed Legacy System Reengineering", Volume 2, 26-28 July 2005 Page(s):160 - 165 Vol. 1 COMPSAC
- [4] Bin XU, "Extreme programming in reducing the rework of requirement change", Electrical and Computer Engineering, 2004. Canadian Conference on, Volume 3, 2-5 May 2004 Page(s):1567 - 1570
- [5] Joel Spolsky "Joel on Software" Apress 2005
- [6] Kent Beck, "Embracing Change with Extreme Programming", Computer Volume 32, Issue 10, Oct. 1999 Page(s):70 - 77
- [7] Sang-Kil Park, "Overview of eXtreme Programming", <http://www.likejazz.com/29116.html>
- [8] Fabrizio Ferrandi, Pier Luca Lanzi, Donatella Sciuto, "Mining Interesting Patterns from Hardware-Software Codesign Data with the Learning Classifier System XCS", Evolutionary Computation, Volume 2, 8-12 Dec. 2003 Page(s):1486 - 1492
- [9] Kent Beck, "eXtreme Programming explained : Embrace Change", Addison Wesley 2000