



## COMMUNICATION AMONG HARDWARE AND SOFTWARE ENGINEERS BY TECHNICAL NOTATIONS

<sup>1</sup>ALI MOHAMMADALIZADEH GHAZIJAHANI, <sup>2</sup>RODZIAH ATAN, <sup>3</sup>NOR FAZLIDA M. SANI

<sup>1</sup>Faculty of computer science and IT, University Putra Malaysia, Kuala Lumpur, Malaysia

<sup>2</sup> Assoc. Prof., Faculty of computer science and IT, University Putra Malaysia, Kuala Lumpur, Malaysia

<sup>3</sup>Senior Lecturer, Faculty of computer science and IT, University Putra Malaysia, Kuala Lumpur, Malaysia

E-mail: <sup>1</sup>[a.alizadehgh@gmail.com](mailto:a.alizadehgh@gmail.com), <sup>2</sup>[rodziah@fsktm.upm.edu.my](mailto:rodziah@fsktm.upm.edu.my), <sup>3</sup>[fazlida@fsktm.upm.edu.my](mailto:fazlida@fsktm.upm.edu.my)

### ABSTRACT

Hardware and software engineers need to cooperate with each other in developing and building hardware/software systems. Any cooperation among engineers requires a convenient and efficient communication. There is a gap between hardware and software engineers that makes the communication difficult among them. They have difficulty in understanding each other's language because they are different in the field of study and profession. If good communication is not established between these two groups, then hardware/software systems will encounter tremendous number of serious problems and defects which may increase the expenses of system in terms of time and resources. Hardware and software engineers need communication mostly for developing hardware-software interfaces and for clarification of type and format of the data that will be transferred among hardware and software components. In this paper, three different notations are proposed to help software and hardware engineers communicate with each other. By using these notations which are understandable by hardware and software engineers, the requirements relating to the data types and data formats will be depicted in uniform, detailed and accurate forms of documents.

**Keywords:** *Engineers Communication, Communication Notation, Requirements, HW/SW Systems, Hardware/Software Engineers*

### 1. INTRODUCTION

Communication among software and hardware engineers has consistently been one of the main challenges in development of HW/SW (Hardware/Software) systems. If engineers cannot communicate with each other efficiently then the HW/SW systems will experience serious problems after development that may force HW/SW systems to be changed. In addition, the defects rate of them will be increased and the system's reliability, consistency and even functionality will be breached. In this situation our software may experience failure during its operational period [2]. Any changes of software or hardware components very often entail changes in the hardware-software interface. Such a modification may be a redesign of a component, changing its implementation from software to hardware or vice versa [12]. In all these conditions, the expenses of system development will increase in terms of time and resources. It is widely recognized that communication problems are a major factor in the delay and failure of

software projects [6]. Considering this situation, using some unified and standard common notations that could be understood by both software and hardware engineers will help them to overcome some of these difficulties.

HW/SW systems are those systems that contain some software and some application specific hardware components. These components work together to fulfill a specific responsibility [12]. In these systems, information (data or control information) are continuously exchanged between hardware and software components. Determination of type, format and specification of the exchanged data comprises an important part of the technical communication between hardware and software engineers.

Hardware and software engineers communicate each other mostly during the requirements phase of developing HW/SW systems. This phase is considered the most important phase for the communication between software and hardware engineers because concerns and challenges of these

two groups are shared during this phase. Furthermore, the data types, data formats, input/outputs, constraints, Read/Write commands and even control commands are argued and specified in detail during the requirements phase.

Software engineers have access to a great number of methods and methodologies (like UML) to model their development process and to understand each other's language [1]. Hardware engineers also use models and diagrams to communicate each other in their projects, but much less formal. A big question now arises that if hardware engineers want to communicate software engineers or vice versa, what are the methods and methodologies that they can use them to make their communication formal and efficient?

This research has been initialized by observing the actual communication weaknesses and problems in a specific workplace. These problems are rarely considered to be researched and is proved by little literature discussions regarding the notations.

To be more specific, the problem statements are divided as follows:

- There is an absence of notations by which hardware and software engineers communicate and share their constraints and concerns in developing SW/HW systems in a uniform manner.
- Sometimes miscommunications happen between hardware and software engineers during requirements phase of HW/SW systems.
- Understanding and clarification of some parts of the requirements like common interfaces, interaction data types and data formats by hardware and software engineers take considerable time of system development.

Therefore, three objectives are set for this research:

- To propose and define notations that will help hardware and software engineers to communicate each other in order to clarify and specify their interaction data types and data formats.
- To develop a tool that applies the proposed notations on a communication platform.
- To analyze the acceptance attributes (easiness and accuracy) of the proposed notations and the implemented tool using experts' validation.

## 2. RELATED WORK

There are plenty of works in requirements notations in different areas and the use of them in capturing more accurate information in requirements phase. International Telecommunication Union (ITU) proposed User Requirements Notation (URN) as a standard for the representation of requirements in telecommunication systems and services and in software systems in general [4]. It is the first and currently only standardization effort that combines goal and scenario models in one language [7]. The URN consists of two components, GRL and UCM. The Goal-oriented Requirement Language (GRL) is used to describe business goals, non-functional requirements, alternatives, and rationales, whereas Use Case Map (UCM) enables the description of functional requirements as causal scenarios [5]. The point with this approach is that it does not cover HW/SW common concerns and the ways that they technically communicate each other, rather, connects requirements to the business objectives.

There are also many studies and works concentrated on the co-design of HW/SW systems. These studies which have been accomplished mostly by non-software engineers, have concentrated on how hardware and software components communicate each other in embedded systems. Co-design is a concurrent and cooperative design approach that considers both hardware and software options and includes as a fundamental component the capability to explore hardware/software trade-offs. This capability leads to more efficient implementations and improves overall system performance, reliability and cost effectiveness. In this approach problems can be detected and changes can be applied earlier in the design process [8].

Since co-design concept has been presented, many methodologies have been proposed for different systems especially for embedded systems. Lecomte, etc. (2010) proposed a co-design methodology based on model driven architecture for real time embedded systems [9]. Also, Grabbe, etc. (2005) introduced an interface and communication based design of embedded systems and showed how the communication between hardware and software components should be established to make the design and implementation better [12]. Hardware-software co-design is a recent research area growing mostly from hardware synthesis and mainly focused to facilitate the design of small embedded systems. Co-design

methodologies are intended to give relief to designers struggling with provisional divisions of hardware and software components, and the attendant integration problems.

None of these studies are concentrated on communication among hardware and software engineers and also no attempt is taken in this regard to cover the communication among humans in building HW/SW systems so far.

*Currently used methods for communication*

Each HW/SW development environment has its own specific method for communication among hardware and software engineers. Among these methods the most reasonable and practical method is the use of PRDs (Project Requirements Document). PRD document is prepared by cooperation of both software and hardware engineers and, as its name suggests, includes the detailed requirements of project. In this document all software and hardware engineer’s detailed requirements including interfaces and interaction data types and formats can be found. This document can be updated at any stage of the project life cycle and any change to it, is done in presence of both hardware and software engineers. Here is a sample part of a PRD document:

*“Format of last row of data in memory to be exchanged between hardware and software:*

Byte 15-14	Byte 13-12	Byte 11	Byte 10-9	Byte 8-4	Byte 3-0
Signal In Frequency	Flags	Extra Flags	Encountered Error	Start Time	Duration

**Extra Flags (Byte 11):**

Bit4-0: Current State  
Bit7-5: start time

**Flags (Byte 13-12):**

Bit0 = 0  
Bit1 = 1: this means this is last row of Idle Sequence  
Bit2: Idle Direction

0: HW to SW  
1: SW to HW

Bit4-3: Don't care  
Bit5: Symbols Format

0: Decimal format  
1: Hex format

Bit7-6: Device speed  
00: 8.5 Gbps  
01: 10.0 Gbps  
10: 12.0 Gbps  
11: Reserved

Bit8: Reserved  
Bit9: Removed/Inserted Number (bit 3)  
Bit10: Connection Closed  
Bit13-11: Removed  
Bit14: Speed negotiation flag.  
Bit15: '1' means the Idle Sequence belongs to a multiplexed link”

**3. PROPOSED NOTATIONS**

Specifying the format, order, length and type of the data which are exchanged between hardware and software components comprises the most important and time consuming part of the communication between hardware and software

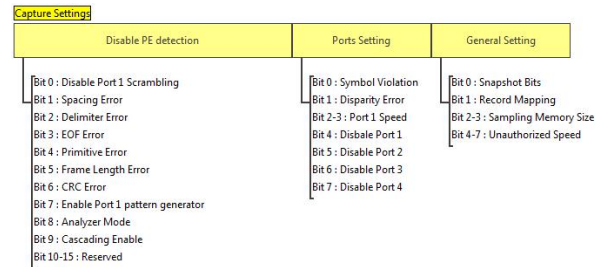
engineers. This is mostly done for hardware programming where a specific kind of data is expected to be written into a segment or memory of the hardware. This clarification is also needed in reading the data from hardware and it is essential to know where and how the data should be read from the hardware. The exchanged data can be a part of sampling memory, raw data, control bits, frames or settings. To clarify these kinds of data, three important notations have been proposed to be used by software and hardware engineers. The first notation is Small Data Format (less than a DWORD); the second one is Big Data Format (bigger than a DWORD) and the third one is Memory Format notation. Below is the description and attributes of these notations:

**3.1 Notation 1(Small Data Format)**

This notation is used to show the data which is equal or less than a DWORD (4 Bytes). Here is the attributes of this notation: (Figure 1)

*Name:* Small Data Format

*Shape:* Generally looks like Figure 1, however it depends on the number of columns (maximum number of columns is 4 and minimum 1). The details and descriptions of each column (BYTE or WORD) will be showed by the use of arrows and braces.



Disable PE detection	Ports Setting	General Setting
Bit 0 : Disable Port 1 Scrambling Bit 1 : Spacing Error Bit 2 : Delimiter Error Bit 3 : EOF Error Bit 4 : Primitive Error Bit 5 : Frame Length Error Bit 6 : CRC Error Bit 7 : Enable Port 1 pattern generator Bit 8 : Analyzer Mode Bit 9 : Cascading Enable Bit 10-15 : Reserved	Bit 0 : Symbol Violation Bit 1 : Disparity Error Bit 2-3 : Port 1 Speed Bit 4 : Disable Port 1 Bit 5 : Disable Port 2 Bit 6 : Disable Port 3 Bit 7 : Disable Port 4	Bit 0 : Snapshot Bits Bit 1 : Record Mapping Bit 2-3 : Sampling Memory Size Bit 4-7 : Unauthorized Speed

Figure 1. Small Data Format

*Number of columns:* This attribute shows how many columns will be in the notation. The maximum number of columns is 4 because we are able to show only 4 bytes or less by this notation. Note that, a column size could be 1, 2, 3 or 4 bytes.

*Column attributes:*

*Column Length:* Columns length is between 8 and 32 bits. (1 and 4 bytes)

*Bits range:* This attribute specifies the range of bits for a specific description.

*Bits Description:* Bits description will be showed by left braces. They describe what the exact use of specified bits is.

**3.2 Notation 2 (Big Data Format)**

This notation is used to describe the data which are bigger than DWORD size (4 bytes). Here is the list of its attributes (Figure 2):

Environmental Data	BYTE 0	BYTE 1	BYTE 2	BYTE 3
DWORD 0	SOF	Symbol Err	Duration	Type of symbols
DWORD 1	Extended flags	Running disparity value		
DWORD 2	APU current state			
DWORD 3	Start time (Low)			
DWORD 4	Start time (high)	Extra Info (low)		
DWORD 5	Extra Info (high)	Control	Destination ID (Low)	
DWORD 6	Destination ID (high)	Source ID		
DWORD 7	Priority	Frame control		
DWORD 8	Sequence ID		Sequence count	
DWORD 9	Parameter	Data (0-2)		
DWORD 10	Data (3-6)			
DWORD 11	Data (7-10)			
...	...			
...	...			
...	...			
DWORD 124	Data (459)	End Address		Ordered Set Type
DWORD 125	Reserved	flags	Number of payload bytes	
DWORD 126	Number of excluded DWORDs			
DWORD 127	Protocol error			EOF

Figure 2. Big Data Format

*Name:* Big Data Format

*Shape:* The shape of Big Data Format notation looks like the shape in Fig. 3. It is too similar to an ordinary table whose rows are DWORDS and columns are the parts of that specific DWORD.

*Number of DWORDS (Rows):* This attribute specifies the number of DWORDS that makes the data and it can be from 1 to any number (depending on the size of data). In fact rows number could be calculated by dividing the size of data to DWORD size (4).

*Row attributes:*

*Number of columns:* This attribute specifies the number of columns for each row. Each row can have maximum 4 and minimum 1 column(s).

*Row Name:* Row names have DWORD string plus the number of the Row by default. But it could be changed when software or hardware engineer draw this notation.

*Column attributes:*

*Column Size:* Specify the number of bytes for each column

*Column Name:* This attribute specifies the name of each column.

*Byte Order:* This attribute determines the order of data for the notation. (Left and Right)

Notation 2 (Big Data Format) is capable of describing any kind of data at any size. This notation is not able to show the columns information in bits and even unable to show the bits descriptions. However, engineers can use the benefits of notation 1 (Small data format) in depicting their detailed information. This is possible by using notation 1 to show one, two or even more columns of notation 2. In this way, the columns that need more clarification and specification in notation 2 will be separately described by the use of notation 1.

**3.3 Notation 3 (Memory Format)**

This notation is used to show the format and the contents of the hardware memory. The length of memory column is DWORD. (4 Bytes)

*Name:* Memory Format

*Shape:* Generally it looks like Figure 3 in next page. But the memory addresses and memory data names will be different.

*Memory Address line:* shows the real addresses of the memory that should be filled by Hardware/Software engineers.

*Memory Size:* The most important attribute of this notation is considered memory size. By specifying memory size, memory address ranges and its minimum and maximum values will be considered.

*Row attributes:*

*Row Ranges:* This attribute specifies the memory range for a specific memory data. Based on this attribute the memory address line will be updated and will be showed. (For example 0000H-FFFH)

*Row name:* This attribute is the name that should be specified for a memory data. This attribute is the identifier for each row of memory column.

Control Commands programming	
Address	Memory Data
00000-00087	Reserved
00088	No. of Ctrl Cmds
00089	Cmd 1 Code
0009A	Cmd 1 Mode
0009B	Cmd 1 Param Len.
0009C	Cmd 1 Param 1
0009D	Cmd 1 Param 2
0009E	Cmd 1 Param 3
0009F	Cmd 2 Code
000A0	Cmd 2 Mode
000A1	Cmd 2 Param Len.
000A2	Cmd 2 Param 1
...	...
F0076	Cmd n Code
F0077	Cmd n Mode
F0078	Cmd n Param Len.
F0079	Cmd n Param 1
...	...
F0A11	Cmd n Param 998
FFFFF	Reserved

Figure 3: Memory Format

#### 4. THE EXPERIMENTAL STUDY

To validate the proposed notations and to find out whether they are understandable by hardware and software engineers, an experimental study was conducted. A chain of related tasks were taken to conduct the experimental study. First of all, a framework was designed and implemented to prepare the proposed notations and to make them accessible to draw in models. This framework is called Communication Notation Designer (CND). Second, a set of case studies were prepared by defining a sample system and describing all available ways of communication (including the new proposed notations) between hardware and software engineers. Third, a questionnaire was designed and prepared to cover all questions regarding the new way of communication and also the CND software. And finally, the questionnaires were filled by experts. Here is a description of all these tasks:

##### 4.1 Tool development (CND Software)

To develop CND (Communication Notation Designer) software, different frameworks and environments were studied and finally Eclipse environment and Java language were selected. GEF (Graphical Editing Framework) framework was

also used to design and make the notations accessible by users.

CND is developed to give hardware and software engineers the ability of drawing their desired notations and changing the notation's attributes. Drawing of notations is simply possible by choosing the desired notation icon from left side bar and putting it in the main page. Figure 4 shows the left side bar of the CND software. By putting the icon in the main page, the notation will be drawn by its default values.

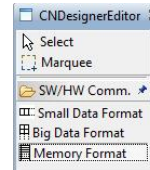


Figure 4: The notations selection bar

Changing of attributes is possible via property dialogs which are specific for each notation. In these dialogs all necessary tables, buttons and edit boxes are considered to satisfy the user the best. Figure 5 shows the property dialog of a sample Big Data Format notation. The other two notation's property dialogs are almost similar to this dialog with some different tables and arrangements.

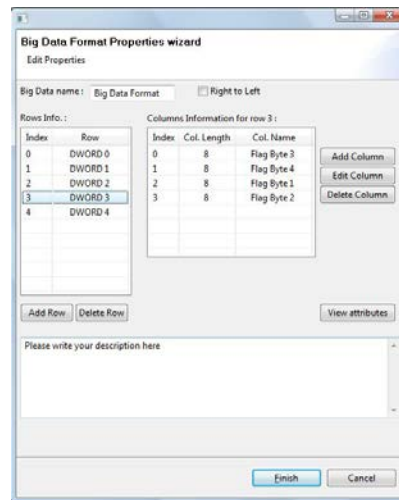


Figure 5: The property dialog of Big Data Format notation

The CND software also gives the user the ability of saving the notations by loading them into a binary file or printing them on the paper. In addition, it has lots of different facilities to help engineers have a better view to the drawn notations, retrieve saved notations and edit them. These entire things have been accessible through the menus and toolbars of the CND software.



## 4.2 Preparing case studies and the questionnaire

This research needs case studies to show how the proposed notations are prepared in real world, how they are used for communication in developing real HW/SW systems and how they are different from the other (available) ways which are used in communication between hardware and software engineers. These case studies attempt to understand the validity of data format notations and CND software through the participants' interpretation of their context. There are three case studies (each for one notation) which have been defined on a sample HW/SW system. Each case study shows and describes two ways of communication (communication by PRD and notation based communication) between hardware and software engineers. These case studies are prepared in the context and level that are easily understood by the participants. The nature of this research dictates that the experiment should be conducted in the environment that software and hardware engineers are involved in common projects and they communicate each other continuously during their daily jobs. Therefore, these engineers were completely familiar with the nature of the sample HW/SW system in the case studies.

Apart from case studies, questionnaires are required to measure the understandability of the proposed notations and the acceptance of the tool from the viewpoint of the people who are dealing with HW/SW systems. In these questionnaires all questions regarding the notations and CND software are covered and the participants are allowed to give their extra opinions and recommendations.

## 4.3 Do The Experiment

To do the experiment, there were two groups of respondents, software engineers and hardware engineers. As mentioned earlier, they worked in HW/SW systems development domain and communicated their software/hardware counterparts daily. For this experiment 10 persons from each group were asked to participate in the survey. Among these 20 engineers, there were 4 system analysts and 9 designers and the rest were developers. All hardware engineers and 4 of software engineers had completed their masters in science and the rest had their bachelors' degree. 5 and 11 engineers had at least 10 years and 5-10 years of experience respectively and the rest had 5 years of experience or less.

All participants were given enough time to get familiar with CND software as well as the notations

and their definitions. Then any of case studies were described to all participants and the old and new ways of communication were depicted to them. Then they were asked to draw the mentioned notations in case studies in CND software and make them ready as had been described in case studies. This was done to make sure that they had learned how to work with CND software and they had gained enough information about the differences between the old way of communication and the notation based technical communication. After this small session of teaching the questionnaires were given to them and they were asked to fill the questionnaires.

## 5. RESULTS AND ANALYSIS

The experiment was conducted to validate the proposed notations and to see whether the CND is applicable to the communication between hardware and software engineers or not. The results generally showed that participants are happy using notation-based technical communication and they are satisfied using the CND software and its facilities. There were two groups of questions. The questions which should be answered by choosing a number between 1 and 10 and the questions which should be answered by stating agree, strongly agree, disagree, strongly disagree or neutral.

Results showed that about 70% of participants (14 engineers) strongly agreed or agreed that the proposed notations are easier and faster than PRD documents to specify system interaction data formats and data types. 8 of them were software and the rest were hardware engineers. Only 5% of participants disagreed with the statement and the rest were neutral.

Almost all participants agreed that doing changes on CND diagrams is more suitable and easier than changing the PRD documents. Only one engineer was neutral. They also agreed that CND diagrams need less reworking than PRD documents.

40% and 20% of participants agreed and strongly agreed that the number of misunderstandings happen in communication with the proposed notations is less than PRD documents. 30% of participants were neutral and 10% of them disagreed with the statement.

Almost all participants strongly agreed or agreed that the proposed notations are more accurate than PRD documents in specifying interaction data formats and data types. Only two

software engineers were neutral about this statement.

Figure 6 shows the level of understandability of the proposed notations for hardware and software engineers. As you can see in the chart all engineers have understandability equal or higher than 5. The average level of understandability for our respondents is 7.2 (from 10). This average shows that engineers have acceptable and reasonable level of understanding of the proposed notations.

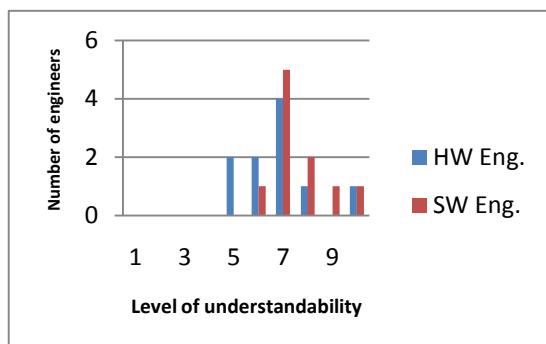


Figure 6. Level of understandability of the proposed notations

The respondents were also asked about the easiness of preparing desired notations by CND, changing/editing previously saved notation's properties and saving/loading the diagrams (by giving a number between 1 and 10). The average rates of responses for these questions were 7.5, 8 and 6.9 respectively.

## 6. CONCLUSION AND FUTURE WORK

In this study we proposed and presented three different notations which can be understood by both hardware and software engineers. These notations will help both engineer groups to communicate each other in order to clarify and specify their interaction data constraints including data types and data formats. In addition, the interaction data constraints will be able to be presented in written way in PRDs by the use of these notations rather than being mentioned oral or by different formats. Furthermore, by using these notations the clarification of interaction data will be specified in detail, during requirements stage rather than postponing them to the implementation stage of building software and hardware.

Since the notations have been designed and presented for HW/SW systems domains, these results and conclusions are only applicable to the HW/SW systems environments and to the software and hardware engineers who are communicating each other in developing these kinds of systems.

However, the nature of the proposed notations allows the engineers to use them at any environment to model and depict any kind of data at any size.

For future work we would like to suggest other perspectives of the communication among hardware and software engineers to be studied and researched. In this research we only focused on interaction data formats and data types notations. These notations can be increased in number and type by looking at other communication parameters and items in or order to ease it as much as possible.

## REFERENCES:

- [1] M. Bjorkander and C. Kobryn, "Architecting system with UML 2.0", IEEE Computer Society, Vol. 20, No. 4, pp. 57-61, 2003
- [2] K. Saleh and A. Al-Zarouni, "Capturing non functional software requirements using the user requirements notation", In proceedings of 2004 international research conference on innovation in information technology, pp.222-230, 2004
- [3] B.D. Theelen, O. Florescu, M.C.W. Geilen, J.Huang, P.H.A van der Putten and J.P.M Voeten, "Software/Hardware engineering with the parallel object-oriented specification language", In proceedings of the 5th IEEE/ACM International Conference on Formal Methods and Models for Codesign. Washington DC. USA, pp. 139-148, 2007
- [4] International Telecommunication Union, "User requirements notation (URN) – Language requirements and framework", Geneva, Switzerland, 2003.
- [5] D. Amyot, "Introduction to the user requirements notation: learning by example", Computer networks: The International Journal of Computer and Telecommunications Networking, Vol. 42, No. 3, pp. 285-301, Ottawa, Canada, 2003
- [6] A. Al-Rawas and S. Easterbrook, "Communication problems in requirements engineering: A field study", In proceedings of First Westminster Conference on Professional Awareness in Software Engineering, Royal Society, London, Feb. 1996



- [7] G. Mussbacher, "Aspect-Oriented user requirements notation: aspect in goal and scenario models", Lecture Notes in Computer Science, Vol. 5002/2008, Berlin Heidelberg, pp.305-316, 2008
- [8] S. Kumar, J. H. Aylor, B.W. Johnson and W. A. Wulf, "A framework for hardware/software co-design", IEEE computer Society, Vol. 26, No. 12, pp. 39-45, Dec 1993
- [9] S. Lecomte, S. Guillouard, C. Moy, P. Leray and P. Soulard, "A co-design methodology based on model driven architecture for real time embedded systems", Mathematical and Computer Modelling, Vol. 53, No. 3-4, pp. 471-484, Feb 2011
- [10] P.H. Chou, R.B. Ortega and G. Borriello, "The Chinook hardware/software co-synthesis System", In proceeding of the 8<sup>th</sup> international symposium on system synthesis. New York, USA, 1995
- [11] A. D. Andrade, "Interpretive research aiming at teory building: adopting and adopting the case study desing", The Qualitative Report, Vol. 14, No. 1, pp. 42-60, Auckland, New Zealand, March 2009
- [12] C. Grabbe, F. Appenheimer and T. Schubert, "Requirements on hardware /software communication design based on abstract communication models", unpublished