A Requirements Specification Template of a Communication Network Based on CAN Protocol to Automotive Embedded Systems

Dario Almudi Neto
Faculty of Exact and Natural Sciences
Universidade Metodista de Piracicaba (UNIMEP)
Piracicaba – São Paulo – Brazil
e-mail: netoneto@globo.com
and
Luiz Eduardo Galvão Martins
Faculty of Exact and Natural Sciences
Universidade Metodista de Piracicaba (UNIMEP)
Piracicaba – São Paulo – Brazil
e-mail: martinsleg@hotmail.com

ABSTRACT

This paper presents the results of studies that made possible to propose a particular contribution to improve the quality on developing automotive embedded systems through a requirements specification template of a communication network based on CAN protocol. The whole template structure is composed by sections that specify the controlling units, the subnetworks, the data dictionary and the general aspects of the communication network. A study case was performed to test the proposed template and a study of the requirements of an embedded automotive environment was specified. The conclusions of this study and the evaluation are presented and further studies are suggested. This template can be used by engineers and designers with industrial or scientific purposes.

Keywords: Communication Requirements, Embedded Automotive Systems, Template Specifications, CAN Net, Requirements Engineering

1. INTRODUCTION

The automotive embedded systems have a meaningful role in the industrial automotive sector and many technological laboratories around the world work on developing several resources. The result of these researches innovate the automotive sector through the application of new technologies in the design of vehicles.

The current systems of vehicular automation involve mechanical and electronical devices and computational systems, and due to their complexity, they stimulate the introduction of new design methodologies. An important issue is that the development methodologies require a high level of abstraction for design, verification and validation of the proposed system.

Some weaknesses are diagnosed in the context of automotive embedded systems in the current scenario and some of these processes can be identified in the context of automotive embedded systems. Processes that enable the organization of necessities to develop automotive embedded systems are scarce, and the specifications of the design of automotive embedded systems are often supplied by automakers which keep their planning and documentation in a confidential basis. Companies, and some research sectors, attempting the development of their automotive applications, whatever their commercial purpose or not, perform their work without a requirement specification document suitable for embedded systems.

Another important aspect to be considered in the development of automotive embedded systems scenario is the wide variety of existing communication protocols, allowing a wide field for alternatives of communication network design. The CAN (Controller Area Network) communication protocol has been used in the development of embedded automotive design, and consequently has lead the companies that strongly invest on solutions to be used on the manufacturing of a vehicle to provide CAN as a part of their products.

This wide variety of communication protocols available and their operating criteria and specifications turn the requirements specification for automotive embedded systems into a standard commercial challenge.

Based on the presented scenario, the aim of this paper is to present a template for requirements specification of a communication network based on CAN protocol for automotive embedded systems.

The methodology for the development of the template began with the study of the data communication network of automotive embedded systems, seeking to understand its operation and specifically focusing CAN as a communication protocol. The elicitation processes and the analysis were performed based on technical documents and on standards related to the researched context. Volere [1] and IEEE Std 830-1998 [2] were also studied with the purpose of acknowledging the specification documents of existing requirements and they helped clarify how a template can be structured. A template validation was performed through its usage in a study case to prove its applicability. Results and final considerations are discussed and presented.

This paper is organized as follows: section 2 presents the aspects of Requirements Engineering that work as a basis to assess the needs to structure a template. Section 3 presents an explanation on the CAN communication network, which is the basis for the development of this template. Finally section 4 presents the template and its applicability in a study case and also presents a discussion of the results obtained from the evaluation performed.

2. REQUIREMENTS ENGINEERING

Sommerville [3] defines Requirements Engineering (RE) as a process created to cover all the activities involved in discovery (production), documentation and maintenance of a set of requirements for a computer based system.

The phases during the development of software constitute its life cycle, which have several proposals and designations. Pressman's proposal [5] identifies six delimited phases according to their typical events during their different life cycles. Each phase includes a set of activities or disciplines that must be performed by the parts involved.
According to Sommerville [3] the requirements of software systems are often classified as functional, non functional or as domain requirements.
A proposal presented by Kotonya and Sommerville [6] of a spiral model of the phases of Requirements Engineering is shown in Figure 1.

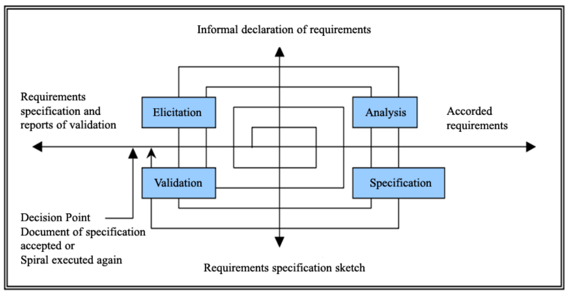

Informal declaration of requirements
Requirements specification and reports of validation
Elicitation
Analysis
Accorded requirements
Validation
Specification
Decision Point
Document of specification accepted or
Spiral executed again
Requirements specification sketch

Figure 1 – Spiral Model for the Requirements Engineering Process [6].

Figure 1 presents a spiral model for the Requirements Engineering process based on the elicitation, analysis, specification and validation of the requirements activities. Kotonya and Sommerville [6] propose a model of iterative and incremental process just like the approach in the spiral model for the software development, which includes the necessary feedback to the characterization of the requirements dynamic nature.
According to Taurion [7], to develop embedded systems it is necessary to adopt methodologies starting with a high level of abstraction as well as the use of tools that automate the most all the stages of the methodology.

Requirements Elicitation
To Belgamo and Martins [8], despite requirements elicitation is the first step in Requirements Engineering, it doesn't happen only once, for requirements elicitation is an iterative process where all the other phases may contain extractions and requirements analysis that may happen whenever the analyst thinks it is necessary.
The information gathered during the requirements elicitation must often be interpreted, analyzed, modeled and validated before the engineer starts feeling confident that the set of requirements was obtained. Therefore, the elicitation techniques are closely related to other RE activities – mostly the elicitation technique used is motivated by the choice of modeling and vice versa: many modeling involve the use of certain types of elicitation techniques [9].

Requirements Analysis
At this phase the clients and users necessities are analyzed to get the definition of the software requirements. The goal is to detect imperfection, omissions and redundancies in order to discover the necessary and desired software requirements. Inconsistency, duplicity of information, ambiguity, conflicting requirements are also examples of problems that can be found out in this phase of the process. The commitment and participation of the stakeholders is crucial in this phase.

Requirements Specification
To organize the requirements of the system, the results obtained from the elicitation and the analysis will be then documented through texts, diagrams, models or rules of prototyping. This process has a high level of difficulty and heavily relies on the engineer writing skills.
Among the benefits obtained by the generated documents, it is possible to cite [4]:

a) The specification document is the basic communication vehicle between developers and users on what has to be built;
b) The specification document register the results of the analysis of the problem (obtained through elicitation and requirements analysis);
c) The specification document defines the properties the system must have along with its restrictions imposed to the design and implementation;
d) The specification document is the basis for cost and schedule rates;
e) The specification document is the basis for developing the system test plan;
f) The specification document provides a behavior standard definition expected by the professionals involved in the system maintenance;
g) The specification document is used to register changes in the system engineering.

Validation
Requirements validation is concerned with showing that the requirements actually define what system the customer wants [3].
According to Martins [4] the main problems found during the requirements validation are:

- Not meeting the standards of quality;
- Requirements poorly described; which lead to ambiguity;
- Errors in modeling the problem or system;
- Conflicting requirements not identified during the analysis phase.

This step is also concerned in finding problems in the requirements. However the processes (analysis and validation) are different. Validation concerns are related to the development of a complete draft for the requirements document, while analysis involves working with incomplete requirements [3].

3. CAN (CONTROLLER AREA NETWORK)

The data transmission systems in the industry began in a quite simple way, using connection like serial RS-232 and RS-485 protocols. However, industries started developing more complex systems, with their own technologies, protocols, software and hardware better fitted to their necessities.
CAN is an internationally standardized serial bus system providing functionality of data link layers of OSI/ISO reference model. In 1983, Bosch Company had started its work of development of a data network automobiles, thus resulting in CAN protocol. The company has improved the use of this protocol to other industrial applications such as: medical systems, navigational instrumentation, elevators control systems, textile production, general production control systems [10].
A CAN network is usually characterized as a network that, despite having been conceived for embedded systems, has working groups in the automation area that envision the systems suitability to be used as an industrial local

network and created an organization named CiA (CAN in automation) with users and manufacturers of products bases on this protocol. Data communication in a CAN network protocol is based in messages loaded in bit frames which, in turn, consist of bit fields with specific functions in the frame. The CAN bus works on multicast oriented to the content of the message and not to address of the message as it is traditionally [11].

CAN is a synchronous serial communication protocol. The timing among the modules connected to the network is performed based on the beginning of each message posted to the bus. All the network nodes continuously monitor all the messages, discarding or recognizing each of them according their convenience. As the transmission protocol does not require a recipient physical address, it holds multiple reception as well as synchronization of distributed processes, i.e., the necessary measurement for the various controllers can be transmitted via network, making unnecessary that each controller has its own sensor [11].

4. CAMA TEMPLATE DEFINITION

CAMA Template structure, which acronym is formed by the words CAN, Martins and Almudi, referring respectively to the CAN communication protocol and to the proponents of the template's surname.

In the construction of embedded systems, the software starts being developed when the hardware is already in a very advanced stage of development. The hardware design tends to be dominant due to having a major cycle of development, being more stable and requiring logistical dependence on external partners, such as suppliers and outsourced developers. There isn't, so far, an appropriate methodology to help developers to specify the requirements to automotive embedded systems, causing a large gap between designers from hardware and software areas, especially in the early phases of structuring the design. One of the possibilities CAMA Template offers is the integration between developers through a resource that provides easy communication channels between the hardware designers and software engineers.

CAMA Template Divisions

The CAMA Template will be used to specify the relevant and special aspects of the automotive embedded communication systems network that use CAN protocol in exchanging information. The template is structured in Figure 2. Each section of the template is detailed into several features. These features are organized using specification cards.

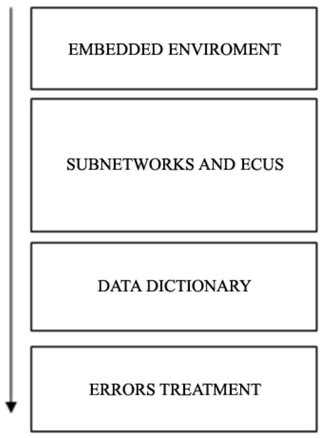


Figure 2 – CAMA Template's General Structure

a) Embedded Environment Definition

The environment of an automotive embedded system may have different purposes. It is possible to establish an environment featuring the control of certain functions, or define an environment to provide diagnosis of situations occurring inside the environment or even more common, to implement these two functionalities inside the automotive embedded system, developing a simultaneous controlling and diagnosis structure. The items that detail the environment definition are (see Figure 3):

Purpose of application: The embedded environment could be defined as control application, diagnosis or control and diagnosis.

Type of Architecture: The distributed architecture is characterized by the presence of several intelligent modules throughout the application, each one receiving only part of the data, usually those generated close to them, and sending them to the modules that require such information to their own processing.

The choice between distributed or centralized architecture will be done according to the necessity and size of the project. The centralized architecture is composed by only one controlling unit and there is not the necessity of a bus communication since it is slightly performed with only sensors and actuators. If the choice is a centralized architecture, the use of CAMA template is not necessary.

Logical Domains: Performing division into the vehicle's logical domains provides an overview on the interrelated structures that formally describe an integrated system. Examples of logical domains: Propulsion System, Vehicle's Motion, Body and Interior, Electric System, Multimedia, etc. This item requires the data of all logical domains specified in the environment.

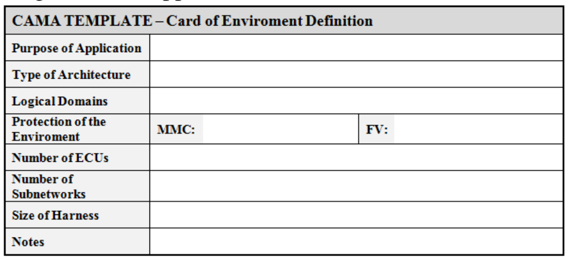
Environment Protection System: The protection systems on embedded environment are performed by fuses in charge of protecting the electric harness of electronic components attached to it. Two elements have to be defined in this item: The maximum current of the circuit (MCC) and the fuse value (FV) to be used as a protection.

Number of ECUs: The Electronic Control Unit (ECU), also known as controlling unity or controlling module is an electronic device that controls one or more electrical systems in a vehicle. Some modern vehicles have up to 80 ECUs. This item requires the data of the number of ECUs independently of the type of established attachment, even if among ECUs the communication is performed by a protocol other than CAN. This item has to specify the total number of ECUs presents in the bus.

Number of Subnetworks: Subnetwork is the segment that establishes the communication between a set of ECUs in a different way on the bus. Each subnetwork must have the detailed specifications of communication on defined files. This item specifies the amount of subnetwork presents on the total bus of the automotive embedded enviroment.

Total Size of the harness: This item specifies the total size of the data bus harness of the environment, adding the harness segments of all subnetworks.

Notes: Additional information about automotive embedded environment. Aspects of the environment that can assist the understanding or special features of the design could be approached.



CAMA TEMPLATE – Card of Enviroment Definition		
Purpose of Application		
Type of Architecture		
Logical Domains		
Protection of the Enviroment	MMC:	FV:
Number of ECUs		
Number of Subnetworks		
Size of Harness		
Notes		

Figure 3 – Card of Environment Definition

b) ECU Definition

Electronic Control Units are the main components of the automotive embedded system. Through them the points of input and output of information are set, since they are in charge of processing it. ECUs may or may not contain a CAN transceptor attached to its structure and when it doesn't happen, this resource must be installed in the ECU so that the communication with the CAN bus occurs. The items to be specified about ECUs are presented below (the specification card is presented in Figure 4).

ECU Identifier: A numeric or mnemonic label that identifies the ECU in an unique way inside the automotive embedded system environment.

Type of ECU: It identifies the electronic module according to its application. Acronyms are used to identify the electronic modules: BCM – body controlling module, BAM – back area module, TCM – transmission controlling module. These are only some examples to illustrate this feature.

Type of CAN Controller: Relating to transmission and reception buffers. BasicCAN – low cost controller with simple capacity of filtering acceptance. FullCAN – controller that handles the most complex messages through dual-port RAM, freeing CPU to manage just a few bits.

Quantity of inputs: Number of input the ECU has, determining which are digital and which are analog.

Type of Input: These can be digital or analog. Digital inputs capture information in two states "0" or "1", and can be translated by states of voltage (0 and 5 Volts or 0 or 12 Volts). They may or may not be supervised (in case of being supervised the input must be connected to an analog gate of the ECU and have a resistor connected in parallel). Analog inputs are able to capture information that varies infinitely between two values, 0 or 5 Volts and 0 or 12 Volts. This is the item to specify the voltage of the input and when they are digital, if they are supervised or not (WS – with supervision / NS – no supervision).

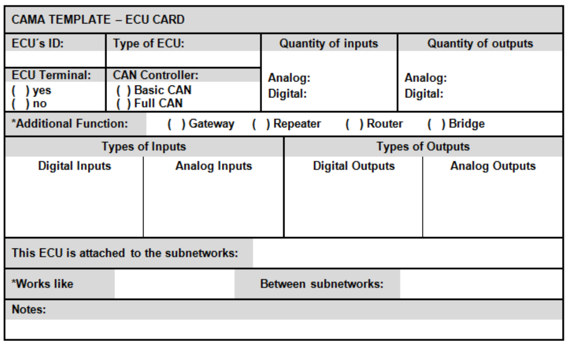
Quantity of output: the number of outputs the ECU has, determining which are digital and which are analog.

Type of Output: Outputs can be digital or analog. Digital outputs are divided into two groups: Low Side Driver (LSD) e High Side Driver (HSD). Both may or may not be protected. It specifies the voltage of the outputs and if they are protected or not (WP – with protection / NP – no protection).

ECU Terminal: Defines if this ECU will be terminal carrier (120 Ohms resistors) inside the bus.

Additional Function. Gateway: resource which primary purpose is to interconnect distinct networks in a manageable way with the possibility of separating colliding domains and interpreting different protocols. Bridge: used to interconnect networks, allowing free access between them. Repeater: it is an equipment used to interconnect identical networks, since they electrically amplify and regenerate transmitted signals in the physical environment. Routers: equipment used to create forwarding routes of data packages in different networks. See rule SAE J1939 for an understanding on each of the detailed applications.

Notes: Specification and additional information on ECU. Other technical details about the ECU can be showed. Due to the wide variety of manufacturers in the market, one suggestion is to specify the source of the ECU, in order to make easier its identification.



CAMA TEMPLATE – ECU CARD			
ECU´s ID:	Type of ECU:	Quantity of inputs	Quantity of outputs
ECU Terminal: ( ) yes ( ) no	CAN Controller: ( ) Basic CAN ( ) Full CAN	Analog: Digital:	Analog: Digital:
*Additional Function:	( ) Gateway ( ) Repeater ( ) Router ( ) Bridge		
Types of Inputs		Types of Outputs	
Digital Inputs	Analog Inputs	Digital Outputs	Analog Outputs
This ECU is attached to the subnetworks:			
*Works like		Between subnetworks:	
Notes:			

Figure 4 – ECU Card

c) Subnetwork Definition

Subnetworks are formed by part of the bus to which two or more ECUs are attached. The established characteristics inside the subnetworks are standardized to allow that the ECUs linked to this subnetwork bus to communicate.
The information on the subnetworks is transmitted through fixed format frames with different but limited lengths. When the bus is idle, any connected node can start transmitting a new frame. If two or more nodes start transmitting frames simultaneously, the conflict of accessing to the bus must be solved by arbitration through the contention identifier. The arbitration mechanism must ensure that there are not waste of information neither time. The transmitter with the highest priority frame will have access to the bus. The items that detail the subnetwork definition are (see Figure 5):

Subnetwork Identifier: A numeric or mnemonic label that identifies the subnetwork in a unique way inside the automotive embedded system environment.

Type of Subnetwork: Specifications of the subnetwork that are attached to the ECU. Example: CAN network, LIN network, SDI network, etc.

Operating voltage: Specifies which operating voltage will power the bus. The values must be determined to VCAN_H, VCAN_L, zero dominant logic level (Vdiff) and the one recessive logic level (LR). There are values to be specified to the voltage operating on the rules that standardize the CAN protocol implementation (e. g.: ISO 11898).

Transmission Speed: Data transmission speed is proportional to the length of the bus. The highest specified data transmission rate is 1Mbps considering a 40 meters bus.

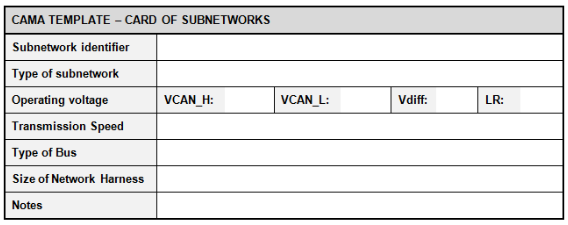
Type of Bus: Twisted pair (two or four wires) or single wire. The choice of the bus type is determined by the features of the application wanted to be developed. It can be:

- For one wire – CAN Line
- For two wires: CAN_L and CAN_H
- For four wires: CAN_L, CAN_H, VCC (input) e GND (ref.)

Size of Network Harness: Specifies the size of the harness of the subnetwork data bus. It is important to note the distances to be taken when building the bus.

- Maximum bus length for a speed of 1Mbps = 40m;
- Maximum branch length (connection between the harness and the main ECU) = 30 centimeters;
- Minimum distance to be respected between branches = 0,10m.

Notes: Specification and additional information on the subnetwork.



CAMA TEMPLATE – CARD OF SUBNETWORKS				
Subnetwork identifier				
Type of subnetwork				
Operating voltage	VCAN_H:	VCAN_L:	Vdiff:	LR:
Transmission Speed				
Type of Bus				
Size of Network Harness				
Notes				

Figure 5 – Card of Subnetwork

d) Structure for Defining the Field of Arbitration

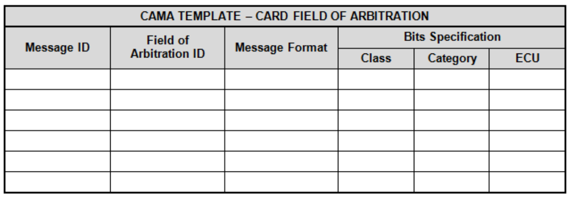
Specifications to be presented in the template are:

Message ID: A numeric or mnemonic label that identifies the message in a unique way inside the automotive embedded system environment. This ID must be specified in the file of the message.

Field of Arbitration ID: A numeric or mnemonic label that identifies the field of arbitration in a unique way inside the automotive embedded system environment. This ID has to be the same specified in the file of the message.

Message Format: It determines which will be the standard of the message: 11 bits for the standard format or 29 bits for the extended format.

Bits specification: Bits will be specified according to the option of the message format. It can adopt the suggestion of the division in classes, categories and address of the node.



CAMA TEMPLATE – CARD FIELD OF ARBITRATION					
Message ID	Field of Arbitration ID	Message Format	Bits Specification		
			Class	Category	ECU

Figure 6 – Field of Arbitration Card

e) Structure for Data Dictionary Definition

Specifications to be presented in the template are:

Message ID: A numeric or mnemonic label that identifies the message format in an unique way inside the automotive embedded system environment.

Message name: A label that helps identifying the purpose of the message.

Field of Arbitration ID: A numeric or mnemonic label identical to that found in the card of the Field of Arbitration.

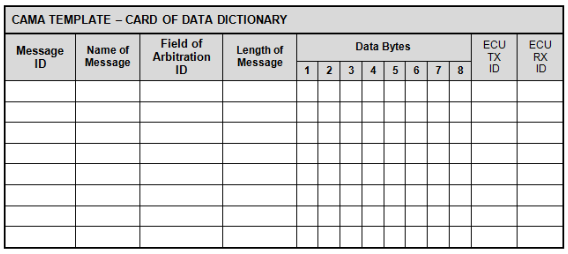
Size of Message: Size in bytes of message according to what was established in the field of controlling.

Data Bytes: Data bytes specification that must be performed in binary or hexadecimal format to compose the message.

ECU TX ID: A numeric or mnemonic label identical to that found in the ECU card. It identifies the ECU that transmits the message.

ECU RX ID: A numeric or mnemonic label identical to that found in the ECU card. Ii identifies the ECU that receives the message.

Notes: Specification and additional information about the Data Dictionary.



CAMA TEMPLATE – CARD OF DATA DICTIONARY													
Message ID	Name of Message	Field of Arbitration ID	Length of Message	Data Bytes								ECU TX ID	ECU RX ID
				1	2	3	4	5	6	7	8		

Figure 7 – Card of Data Dictionary

f) Error Checking

CAN has a very reliable error handling. The errors can be detected, whatever they are global or local. This possibility sets CAN as a high level security solution to be implemented in automotive embedded systems.
There are five ways of handling errors that CAN enables:

a) Bit Error: any transmitter continues monitoring the data bus while transmitting. If the monitored bit has a different value from the one sent, an error is flagged.
b) Coding Error: it happens when the monitored bit had the same value six times. In the sixth occurrence the error is flagged.

c) CRC error: in case the value of the transmitted CRC field is not equal to the CRC recalculated in the receiver, this error is flagged.
d) Formatting error: it occurs when a pre-defined field format (CRC - Cyclic Redundancy Check, ACK - Acknowledgement, Final of Frame) has one or more illegal bits.
e) ACK Field error: this error will be flagged in case the transmitter does not detect a dominant bit while ACK field transmission.

5. STUDY CASE

An automotive embedded system was chosen for the study case to have its requirements specified through the proposed template. The system could not be identified because of confidential commercial reasons. Issues of confidentiality are very sensitive in automotive embedded systems development and there was a commitment by the authors of this paper, so that the study case would be possible. It was asked not to reveal or identify parts of the documentation used to evaluate the template.

The study was based on a finished specification from a major international car manufacturer with a plant in Brazil, with a large insertion in automotive Brazilian market. The part of the documentation provided to use in the study case did not include the physical layout of the embedded system's data communication network and in order to have a full performance of the study, the author suggested a physical layout of the automotive embedded environment, as showed in Figure 8. The manufacturer requirements specification includes only the aspects related to de ECU and subnetwork identification, approaching the message framework in a more substantial way.

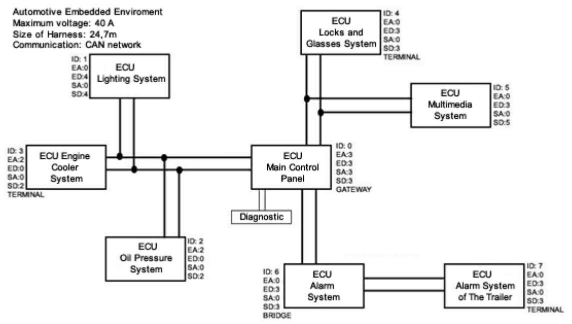

Automotive Embedded Enviroment
Maximum voltage: 40 A
Size of Harness: 24,7m
Communication: CAN network
ECU Lighting System
ID: 1 EA:0 ED:4 SA:0 SD:4
ECU Locks and Glasses System
ID: 4 EA:0 ED:3 SA:0 SD:3 TERMINAL
ECU Multimedia System
ID: 5 EA:0 ED:3 SA:0 SD:5
ECU Engine Cooler System
ID: 3 EA:2 ED:0 SA:0 SD:2 TERMINAL
ECU Main Control Panel
ID: 0 EA:3 ED:3 SA:3 SD:3 GATEWAY
Diagnostic
ECU Oil Pressure System
ID: 2 EA:2 ED:0 SA:0 SD:2
ECU Alarm System
ID: 6 EA:0 ED:3 SA:0 SD:3 BRIDGE
ECU Alarm System of The Trailer
ID: 7 EA:0 ED:3 SA:0 SD:3 TERMINAL

Figure 8 – Physical Layout of the Embedded Environment Adopted in the Study Case

The aim of this study case was to practice with the proposed template to specify the automotive embedded systems' data communication requirements providing by this experience, a template's experimental evaluation. During the study case three characteristics of the template were aimed to be analyzed:

Adequacy of coverage: checking if the technical variables reached in the *template* covered the main elements that constitute an automotive embedded systems environment;

Easiness of use: checking whether the explanations provided for filling in forms as well as their layout were easy to understand and use;

Practical use: checking how useful the template showed itself to the automotive embedded system designer, assessing whether the requirements specifications of the environment were well documented through the use of the template.

- **Positive evidence related to the use of the Template**

The template proved to be easy to use. The study case showed that all models of specification files could be used without any difficulties.

The scope of the template proved to be sufficient, since there was not a record of non-relevant item in the template in the study case.

The structure proposed for the template, dividing it into their respective specifications files, proved to be adequate because there was not a register of inconvenience or misunderstanding about the use of the proposed files.

- **Negative evidence related to the use of the Template**

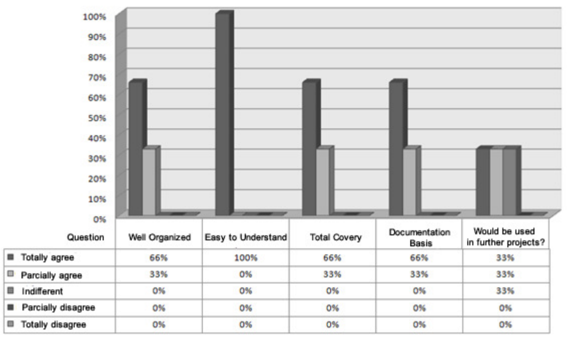
Some aspects regarding to the standardization of terminology our units of measure interfered with the interpretation of requirements. It was observed that even if the automakers follow the rules established by SAE and ISO, they also create mechanisms to keep their commercial secret.

The proposed template approached the CAN protocol more specifically due to its use be more intense in the embedded system environment. However, new technologies and commercial decisions can change this preference.

- **Appraisers' analysis**

The CAMA Template was taken to three automotive embedded system engineers working in multinational companies with more than ten years of expertise. They were asked to issue their opinion on the template based on their daily basis professional activities.

Figure 9 presents the results of the evaluation performed by the automotive engineers that participated in the analysis of the CAMA Template. The Likert Scale was used to check the level of organization, understanding, approach, documentation and use of the template.



Question	Well Organized	Easy to Understand	Total Covery	Documentation Basis	Would be used in further projects?
■ Totally agree	66%	100%	66%	66%	33%
□ Parcially agree	33%	0%	33%	33%	33%
▨ Indifferent	0%	0%	0%	0%	33%
■ Parcially disagree	0%	0%	0%	0%	0%
▨ Totally disagree	0%	0%	0%	0%	0%

Figure 9 – Results of Evaluation by Engineers

Some aspects discusses by the automotive engineers were:

a) A positive highlight is related to the template's organization, which is very important for a better

understanding and documentation of the requirements specification.
b) It is important to verify that the template can be considered a relevant element to form the basis of the development of the design when planning the automotive embedded design.
c) The requirements to be specified in the template broadly include the design developers' necessities with wide cover.
d) According to the evaluators CAMA template does not provide a complete basis to assess the project for the Automotive Embedded Systems' cost and schedules (though it partially meets this requirement), because cost assessments involve some other aspects to be considered.
e) As the template approaches only CAN protocol, it limits some features when it comes to more sophisticated designs. It was indicated to amplify the template to approach other communication protocols for automotive embedded systems.
f) Overall, the template was well evaluated by the engineers considering its application and easiness, especially because it allows an agreement basis between software and hardware engineers, one of the proposals of this study.

6. CONCLUSIONS

Although this study provides a special contribution focused on a specific aspect of the automotive embedded system, which is the data communication network through CAN protocol, it is a relevant and innovative contribution, since any embedded system needs a complete and organized documentation about the environment requirements in which the software will be implemented.
The software plays a key role in many products that incorporated technology. The software is a priority factor for the automotive industry which can present several problems, though it is crucial for competitiveness [12].
The template driven documentation produced will turn easy the automotive embedded software developers' work since they will have a requirements specification with easy access to the understanding of a data communication network for an automotive environment.
For those interested, CAMA template automotive embedded system will provide benefits that will enable:

- Establish an agreement basis between stakeholders;
- Include as an item on the basis of the project planning;
- Reduce the effort of development between hardware and software engineers;
- Provide a starting point for the project;
- Reuse of requirements for future projects;
- Form a basis to enrich the project's documentation.

The study case provided a first trial on the proposed template. The initial experience has confirmed a promising perspective of application of the template, illustrating the usefulness and relevance of the proposal. It is clear that more experiments have to be conducted to confirm the efficiency and benefits of the template, as well as to point the necessary adjustments and adaptations in this requirements specification tool.
As future works, it is intended to develop software to support the use of template and expand it to other communication protocols adopted by the automotive industry.

7. REFERENCES

[1] J. Robertson, S. Robertson, Volere – Modelo para Especificação de Requisitos. 14. ed. Versão em Português. London: [s.n.], ago. 2009.
[2] IEEE Std 830-1998. Recommended Practice for Software Requirements Specifications. IEEE - Institute of Electrical and Eletronic Engineers. Inc., 1998.
[3] I. Sommerville, Engenharia de Software. São Paulo: Addison Wesley, 2007.
[4] L. E. G. Martins, Uma Metodologia de Elicitação de Requisitos de Software baseada na Teoria da Atividade. Tese de Doutorado. Campinas, SP: UNICAMP, 2001.
[5] R. S. Pressman, Engenharia de Software. 6. ed. São Paulo: McGraw-Hill, 2006.
[6] G. Kotonya, I. Sommerville, Requirements Engineering: Processes and Techniques. Chichester: John Wiley & Sons, 1998.
[7] C. Taurion, Software Embarcado. A nova onda da Informática. Rio de Janeiro: Brasport, 2005.
[8] A. Belgamo, L. E. G Martins, Um Estudo Comparativo sobre as Técnicas de Elicitação de Requisitos do Software. In: Congresso Brasileiro da Sociedade Brasileira de Computação, 20. Concurso de Trabalhos de Iniciação Científica, 19., 2000. Curitiba: PUCPR, 2000. p. 7-20.
[9] J. Goguen, M. Jirotka, Requirements Engineering: Social and Technical Issues. Seattle: Academic Press, 1994.
[10] Bosch, CAN Specification Version 2.0. Stuttgart: Robert Bosch GmbH, 1991.
[11] A. A. GUIMARÃES, Eletrônica Embarcada Automotiva. São Paulo: Érica, 2007.
[12] M. Broy, Challenges in Automotive Software Engineering. 28th International Conference on Software Engineering (ICSE). China: Shanghai, 2006.

Received: Aug. 2010. Accepted: Sept. 2010