SMC'2006 - IEEE International Conference on Systems, Man and Cybernetics
Taipei, Taiwan, 8-11 October 2006

An Empirical Study of SysML in the Modeling of Embedded Systems

Alexandre José da Silva, Marcos Vinicius Linhares, Rafael Padilha, Nestor Roqueiro, Rômulo Silva de Oliveira

Abstract - **The complexity increase and variety of equipments controlled by embedded computers generate the need of a multidisciplinary approach for the process of development of those equipments, involving the areas of software, mechanics, electric and electronic engineering. System engineering uses different techniques to concurrently integrate these different views. In this sense, it is being specified by OMG a modeling language, denominated SysML (System Modeling Language). It intends to include in a single specification an integrated view of the system, one that includes hardware, software and electro-mechanics parts. The objective of this paper is to do an empirical evaluation of the modeling language SysML, in the sense of identifying its capacities and its limitations for the modeling of embedded systems. For that purpose, an electronic fuel injection system developed by Department of Automation and Systems at our university will be used.**

I. Introduction

Due to the technological progress of the semiconductor industry, and consequently the reduction of processor costs, embedded systems won space in practically all segments of industrial and residential products. Those improved products have embedded software that is responsible for their operation and control. With the increase of the complexity and variety of embedded systems applications, the need for multidisciplinary collaboration appears in the development process, including experts in the areas of software, mechanical, electric and electronic engineering. The integration of these areas will result in the embedded system as a product. However, each area uses different development methodologies and tools. System engineering uses different techniques to integrate the many different engineering areas.

The first stage of the development of an embedded system is to specify its functional and non-functional requirements. Several conceptual models can be used in order to understand and to organize the requirements in a systematic way. A development process or methodology is often seen as a sequence of steps used to transform an abstract specification in a detailed specification that can be manufactured. The input of a methodology is the specification of the system requirements, and the output is its implementation.

There are several ways of describing the behavior of a system. When the behavior is described in a natural language, there may be ambiguities, what will hinder the product development. For that reason it is essential to insert formal or semi-formal models that are capable of specifying the requirements, making possible the employment of automated methods of computer systems design, besides providing a more abstract view of the system [2].

According to [3], many companies are using software engineering technologies in order to obtain the best opportunities, productivity and quality in the development of embedded software. The biggest problem is that most of these available technologies of software development do not consider the specific needs of the development of embedded systems, such as timing constraints, memory limitations, power consumption limitations, specific hardware, etc.

Due to the lack of a tool for embedded software engineering, it is common in embedded software projects the employment of free format diagrams together with block diagrams that remind UML (Unified Modeling Language), data flow diagrams (DFD) and flowcharts [3]. A general purpose drawing tool is used to make the diagrams for the software project. The lack of a standardized syntax and semantics often provides for the misinterpretation of these diagrams. This is more evident when the project is multidisciplinary, where the members have different backgrounds.

A specific modeling language for embedded systems could avoid the ambiguities among the different areas involved in the development. This would be achieved through a standardized syntax and semantic for the description of all system functionalities. Motivated by this gap, the OMG (Object Management Group) proposed the creation of a language to assist system engineering activities. This new modeling language, named SysML (System Modeling Language), is a simplification of UML 2.0 that facilitates the inclusion in a single specification of a complete view of the system. This complete view should include hardware, software, electric and mechanic parts.

The objective of this article is to make an empirical evaluation of the modeling language SysML, in the sense of identifying its capacities and its limitations for the modeling of embedded systems. For that matter, the object used for this case study is an electronic fuel injection system developed in

Alexandre José da Silva is with the System and Automation Department, UFSC Federal University of Santa Catarina, Florianópolis, Brazil (phone: 55 47 3372 4765; e-mail: alexjos@das.ufsc.br).

Marcos Vinicios Linhares is with the System and Automation Department, UFSC Federal University of Santa Catarina, Florianópolis, Brazil (e-mail: marcos@das.ufsc.br).

Rafael Padilha is with the System and Automation Department, UFSC Federal University of Santa Catarina, Florianópolis, Brazil (e-mail: padilha@das.ufsc.br).

Nestor Roqueiro is with the System and Automation Department, UFSC Federal University of Santa Catarina, Florianópolis, Brazil (e-mail: nestor@das.ufsc.br).

Rômulo Silva de Oliveira is with the System and Automation Department, UFSC Federal University of Santa Catarina, Florianópolis, Brazil (e-mail: romulo@das.ufsc.br).

the Department of Automation and Systems of the Federal University of Santa Catarina, Brazil. In section 2 the language SysML is briefly described. In section 3 the electronic fuel injection system is described in natural language. In section 4 the diagrams of the SysML modeling are presented. Section 5 contains comments about the experience of using SysML for the modeling of a real system. Finally, section 6 brings the conclusions.

II. System Modeling Language (SysML)

SysML is a general purpose modeling language for applications of system engineering. It supplies a description pattern for a great variety of complex systems. Those systems can include hardware, software, data, methods, personal and instruments. Besides supplying a modeling pattern, SysML looks for the improvement of the quality of the system, assuring the exchange of information and decreasing the semantic distance among system engineering, software engineering and other areas.

Just like UML unified the object-oriented modeling languages used in the software industry, SysML intends to unify several modeling languages used by system engineers.

The SysML language is being created by a group of high-technology industries and institutes. In November 2005 there were two teams working to evolve the original SysML draft specification, called version 0.9. They submitted two new draft propositions to the OMG, they were called versions 0.98 and 1.0 alpha. In February 2006, those two teams announced their fusion in a new single team, the SysML Merge Team, and they submitted a single merged SysML draft specification, that was called version 0.99 this time [5].

SysML is being defined as a language that reuses a subset of UML 2. This subset is called UML4SysML. System engineers that do modeling with SysML and software engineers that do modeling with UML 2, will be capable of collaborating in their models of the software and of the system. That will improve the communication among the several designers that participate in the development of the system, promoting interoperability among modeling tools [6].

The language SysML should be compatible with two interoperability norms, ISO AP-233 (data interchange standard for system engineering tools) and OMG XMI 2.0 (model interchange standard for UML 2.0 modeling tools).

SysML reutilizes and extends some of the packages of UML 2. A package is a general purpose mechanism used for the organization of elements in groups. UML meta-model defines packages for the structural parts, behavioral parts and auxiliary constructions, such as the profiles for language customization. The package of the structural part includes Classes, Composed Structures, Components and Strategies. The package of the behavioral part includes Actions, Activities, Interactions, State Machines and Use Cases.

Since SysML reuses a subset of UML 2 and it creates extensions to assist the specific requirements of system engineering, many packages of the UML meta-model were not used. UML packages not considered important for system engineering applications were not included in SysML. The following packages of the UML meta-model are included in SysML without modifications: State Machines, Interactions and Use Cases. Other packages such as Activities, Classes and Auxiliary Constructs are reused with the addition of some extensions. New packages such as Requirements, Parametric and Allocation were added to support new constructions [6].

The task of modeling a complex system is extensive. It is necessary to describe several different aspects, including the functional (static structure and dynamic interactions), the non-functional (processing time, reliability, production) and the organizational (organization of the work, mapping and coding). Through modeling a simplification of the reality is created to better understand the system being developed. A model is an abstraction semantically closed of a system, what means that it represents a complete simplification of the reality, created with the purpose of allowing a better understanding of the system [6].

Just like UML, SysML supplies multiple views of the system to be modeled, analyzing it under several aspects, using different diagrams. Each diagram describes the system, or a part of it, under a certain perspective. It is as if the system was modeled in layers, and some diagrams shows the system in a more general way, while others offer a more technical view of the system or of a specific characteristic of it.

A diagram is the graphic presentation of a group of elements, generally represented as graphs of vertexes (items) and arcs (relationships). They are drawn to allow the visualization of a system under different perspectives, that is to say, a diagram constitutes a projection of a certain system. In all systems, except for the most trivial, a diagram represents a partial view of the elements that compose the system. The same element may appear in all the diagrams, in only some of them (the most common case) or in no one (the most infrequent case) [4].

There are some diagrams of UML that are not identified as a single diagram type of SysML. An example is the deployment relationship that represents the deployment of the software into the hardware. It is integrated with the Assembly Diagram of SysML. Two new types of diagrams were created, Requirements and Parametric, for the initial version of SysML and other diagrams are planned for the future versions [6]. The ten diagrams of SysML can be divided according to their use in three parts: structural, behavioral and crosscutting.

III. Electronic Fuel Injection in Combustion Engines

The electronic fuel injection system for internal combustion engine, described in this article, is being developed in the Department of Automation and Systems at the Federal University of Santa Catarina. This electronic injection system will be used in an engine of one cylinder with capacity of 200cc and maximum rotation of 12000 rpm.

The parts of an internal combustion engine depend on the type of the engine. For a four-stroke engine, the main parts

include the axis crank, axes eccentric and valves (intake and exhaust).

The engine accomplishes work burning a mixture of vapor of fuel and air inside of a cylinder. For this reason, it is also called an internal combustion engine. When the mixture of air with fuel burns, it is formed hot gases. These expand quickly and they push the piston of the engine, moving the camshaft. This movement can turn wheels, helixes and to operate machines. The power of this kind of engine is generally expressed in horse-power or watts.

Due to the fast evolution of the engines of automobiles, when the carburetor didn't supply the needs of the new vehicles concerning pollution, fuel economy, power, fast accelerations and etc, it is developed systems for electronic injection of fuel. Its objective is to provide to the engine better accelerations with more economy, in all operation modes. To guarantee such objective, the system of electronic fuel injection supplies to the engine the perfect mix of air/fuel in all the rotation range. The more appropriate it is this mixture, better the efficiency and economy, and smaller the emission of pollutant gases [1].

For the case study described in this paper, the person responsible for the modeling in SysML conducted several meetings with the control engineer responsible for the development of the electronic fuel injection system, defining the characteristics, restrictions and operation of the system.

Sensors are components installed in several points of the engine and they purpose is to send information to the electronic control unit. The following sensors are used:

- engine temperature sensor;
- intake temperature sensor;
- exhaust temperature sensor;
- engine speed sensor;
- throttle position sensor;
- λ probe (responsible for the measurement of the fuel concentration in the air/fuel mixture).

The actuators are components that receive information from electronic control unit and they act in the feeding system, varying the volume of fuel that the engine receives. It is used the following actuators: fuel injector, spark plug and throttle.

The electronic control unit is responsible for the measurement of the sensors and the calculation of the action times for each actuator, respecting timing restrictions. The block diagram of the fuel injection system is in Figure 1.

The timing constraints of the system are imposed by the characteristics of the internal combustion engine to be controlled. It is defined that a turn of the engine (360°) is completed once every 5us for 12000rpm. Another definition is the period of the pulse that activates the fuel injector, which should have duration of 25us for 200cc. The actuator of the throttle valve considers position 0° as a pulse of 1ms and 90° as a pulse of 2ms, within a period of 25ms. Considering these timing constraints, the reading of the sensors and the calculation of the acting times for the actuators should be processed in at most 15ms.

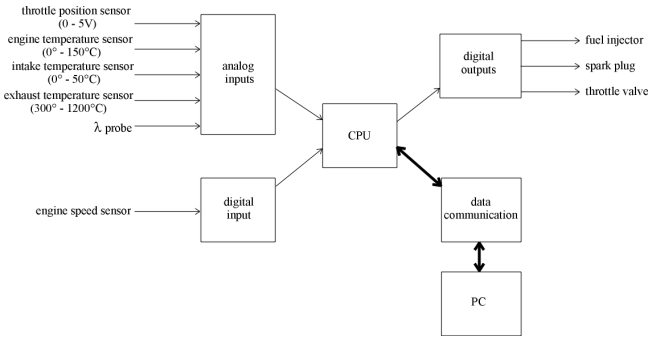

Fig. 1. Block diagram of electronic fuel injection system.

IV. Description of the System using SysML

The next stage is the specification of the system, transforming its informal description into a specification based on diagrams. The electronic injection system was described in SysML. In this paper it was used the SysML draft specification 0.9, because that was the main draft at the time. The other specifications are still evolving and changing very quickly. It should be notice that there are some differences between these newer versions and version 0.9 used in this paper. In the modeling of the embedded system for electronic injection, the following SysML diagrams were used: Use Case Diagram, Class Diagram, Requirement Diagram, Assembly Diagram, Parametric Diagram, Activity Diagram and Timing Diagram.

Use Case Diagram:

The Use Case Diagram, illustrated by Figure 2, is used to describe the external behavior of the system, presenting the functionalities and services of the fuel injection system from the user's point of view.

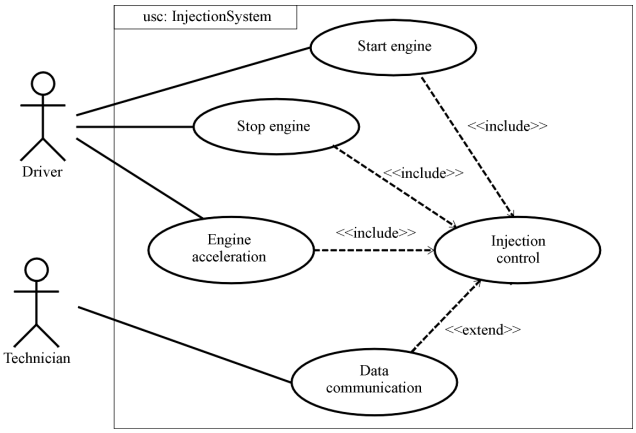

Fig. 2. Use Case Diagram of electronic fuel injection system.

The fuel injection system is formed by five services: Start engine, Stop engine, Engine acceleration, Injection control and Data communication. The services Start engine, Stop engine and Engine acceleration include the service Injection control. The service Injection control is responsible for the measurement of the sensors and calculation of the acting

times for each atuador. An optional service exists to the Injection Control that is named Data communication. It allows the on-line monitoring of the engine operation. The service Engine acceleration supplies to the service Injection control information about current need for the composition of the air/fuel mixture.

The fuel injection system has two types of users, the Driver and the Technician. The Driver interacts with the system through the services for starting/stopping the engine and Engine acceleration. The Technician uses the optional service of Data communication to monitor and to make adjustments.

Class Diagram:

Figure 3 shows the Class Diagram of the fuel injection system. That diagram presents a static view about how the elements are organized. A class is a group of objects that share the same attributes, methods and relationships. Through this diagram it is possible to observe all the parts that compose the system and the relationships among them.

Requirements Diagram:

The Requirement Diagram is showed in Figure 4. It models all the requirements that were identified in the natural language description of the system. It can be modeled all the requirements in a textual way, by referencing technical norms, timing aspects and other diagrams of the system.

The module of electronic fuel injection should satisfy the requirements of the emition standards, and also to execute the whole processing in less than 15ms. The injection module contains other three specific requirements: engine speed sensor, fuel injector and electronic accelerator. Each one of these three requirements is modeled and its details described. In order to better describe the requirements of the electronic accelerator and of the fuel injector, the Timing Diagram is used to show the action of the throttle valve and to show the action of the fuel injector.

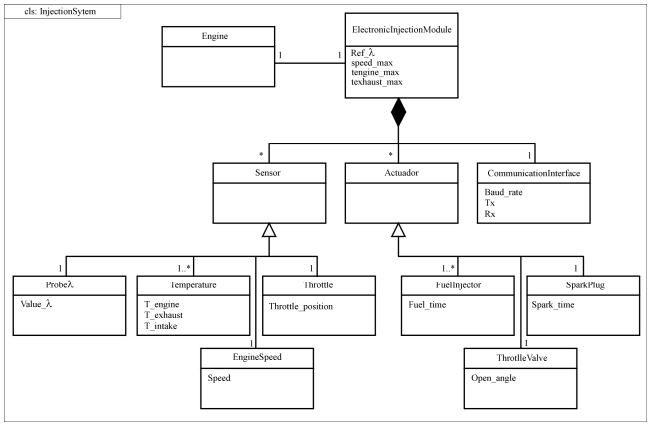

Fig. 3. Class Diagram of electronic fuel injection system.

Assembly Diagram:

The Assembly Diagram is used for modeling of the hardware parts of the electronic fuel injection system. The modeling of the hardware parts begins with a high abstraction level. By the time the model becomes more detailed, the abstraction level is decreased, by improving the specification of the hardware parts.

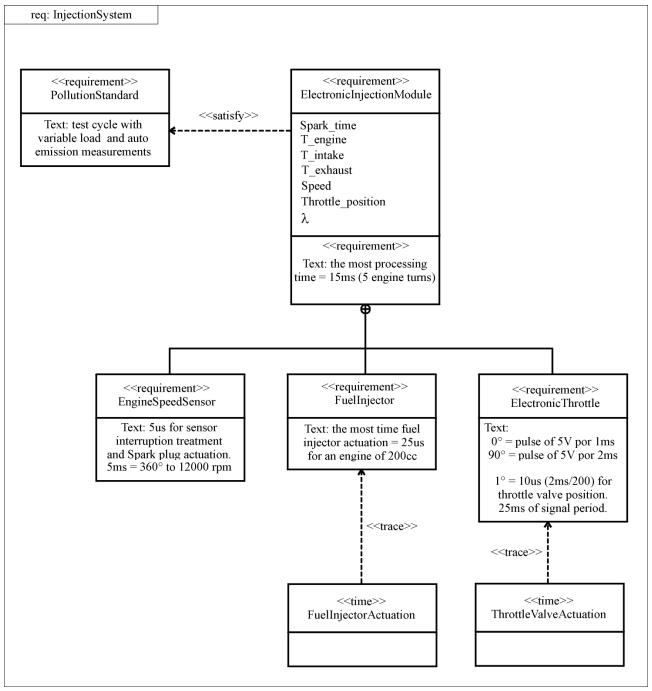

Fig. 4. Requirements Diagram of electronic fuel injection system.

Through the Assembly Diagram it is possible to model a group of parts with specific functions inside of a larger part of the system, specifying the connections and interfaces among them. Figure 5 illustrates the Assembly Diagram of the electronic fuel injection system as a whole, modeling the interface of the fuel injection module with the internal combustion engine.

In the Assembly Diagram of Figure 6, the fuel injection module is showed with details, highlighting its main parts, connections and electric signals. The processor is represented as a hardware component, facilitating the identification of some electric and physical constraints, such as feeding tension, pin numbers and the need of specific peripherals.

However, to better model the specific characteristics of the processor, it is developed an Assembly Diagram of the processor that models with details its peripherals, as it is illustrated in Figure 7.

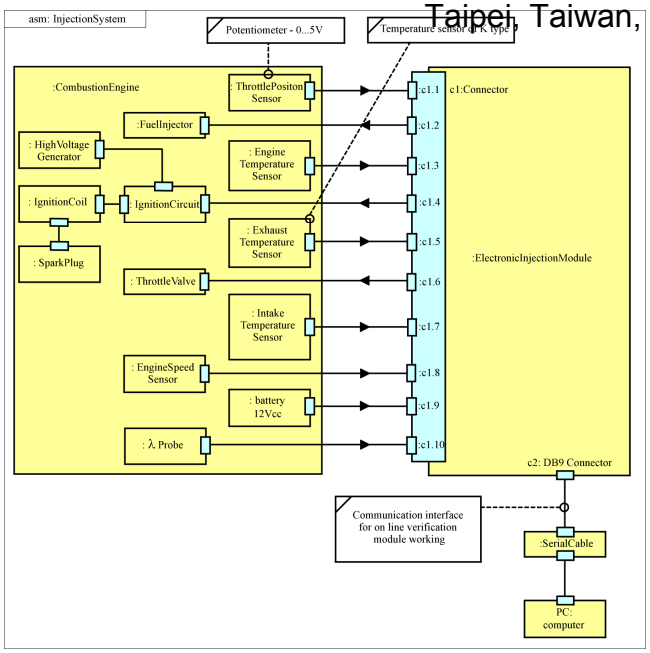

Fig. 5. Assembly Diagram of electronic fuel injection system.

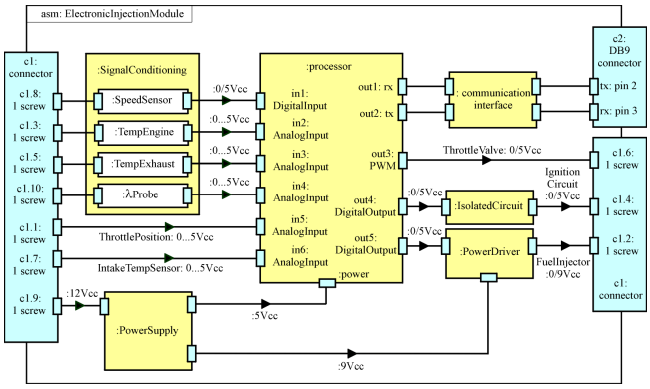

Fig. 6. Assembly Diagram of the electronic module.

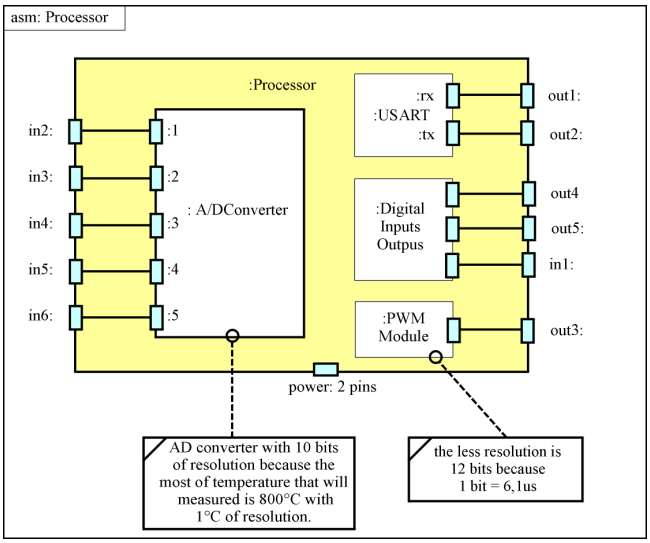

Fig. 7. Assembly Diagram of processor of electronic injection module.

Parametric Diagram:

The Parametric Diagram describes all the equations involved in the development of the embedded system. Figure 8 shows the Parametric Diagram of the electronic fuel injection system. It highlights the calculation of the air/fuel mixture for the internal combustion engine. As it can be observed, the Parametric Diagram describes the calculations that should be implemented on the embedded processor.

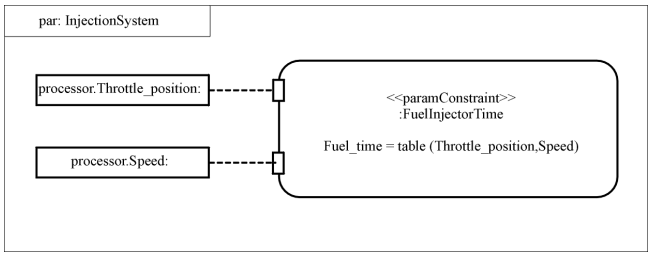

Fig.8. Parametric Diagram of fuel electronic injection system.

Activity Diagram:

The embedded software modeling for the electronic fuel injection system appears in the Activity Diagram.

The Activity Diagram of Figure 9 illustrates the class Engine Control, which models the main program of the fuel injection module. It is possible to identify the routines of Acceleration, Maximum Torque and the interruption of the Rotation.

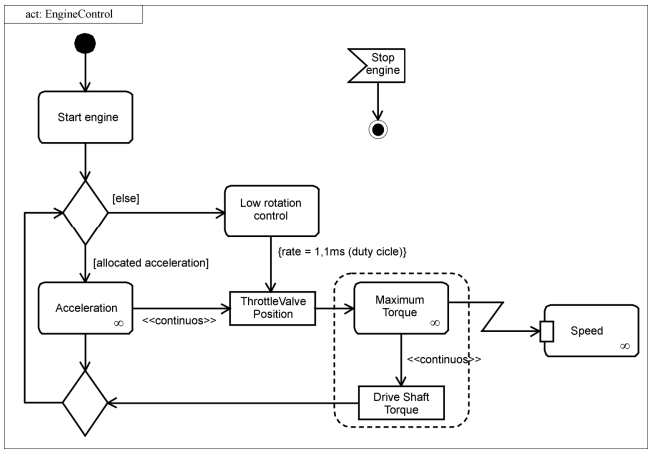

Fig.9. Activity Diagram of electronic fuel injection system.

Timing Diagram:

The Timing Diagram models constraints in a bi-dimensional graphic that is very intuitive. This diagram aids in the understanding of certain requirements and it is very useful for the system implementation.

Figure 10 illustrates the Timing Diagram for the control signal applied to the throttle valve. The Timing Diagram of Figure 11 specifies the control signal for the fuel injector.

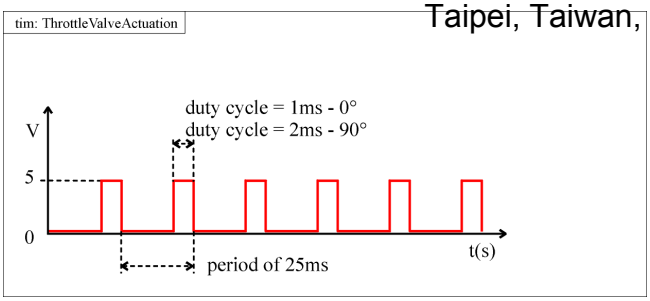

Fig. 10. Timing Diagram of throttle valve actuation.

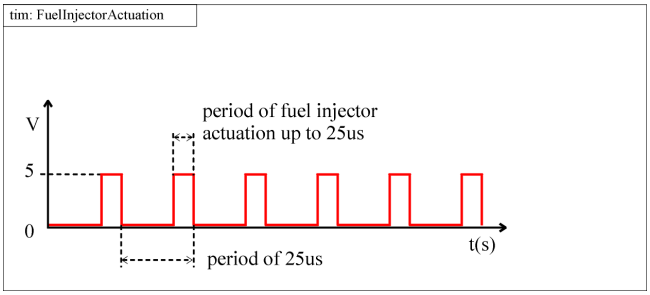

Fig. 11. Timing Diagram of fuel injector actuation.

V. Comments on the SysML Modelling Case Study

The modeling of this case study presented two very different stages. The first was the requirement analysis and the study of the system operation. The Second stage was the study and use of the language SysML.

Since SysML is a simplification of UML 2 and some diagrams are used without modifications, it was necessary the use of support material on UML, and object-oriented modeling during the modeling of the case study.

The Requirements Diagram and the Parametric Diagram were developed for SysML and they have an important role in the system modeling. Initially, the Requirements Diagram was perceived as a great improvement in relation to UML, for the modeling of embedded systems, even so it presents a high degree of informality. The requirements are described through text in natural language, inserting the possibility of inconsistencies in the extraction of the information of the diagram. However, the requirements can be grouped in a clear way, documenting its origin. The Timing Diagram was used together with the Requirements Diagram to model the electric signals that should be applied to control some actuators. It was used to model the control of the actuator.

The Parametric Diagram is the diagram used for the modeling of the mathematical functions associated with the development of the embedded system. Since they are systems of specific application, they present calculations for control and processing of signals, that once modeled they aid in the definition of routines with timing constraints. They affect the selection of the processor architecture, as well as they help to estimate the size of the program.

In spite of the Assembly Diagram to be based on the Structure Composed Diagram of UML 2, it was introduced the NestedConnectorEnd that is a stereotype of ConnectorEnd of UML. It facilitates the modeling of any connector included in different levels of a part or port of the system. Auxiliary constructions such as the itemFlow facilitate the modeling of the electric signals and data inside the Assembly Diagram.

The Use Case Diagram and the Class Diagram are similar to those in UML. Just like in software system projects, these diagrams aid the modeling of embedded systems during the initial stages, when the level of abstraction is high.

VI. Conclusion

The language SysML produced a good specification through diagrams for the embedded system. That is especially true when we compare it to a specification in natural language. This modeling based on diagrams includes the mechanical, electric, hardware, software requirements, technical norms and mathematical functions.

However, SysML modeling requires from the design team of the embedded system, which is generally formed by many engineers, to have knowledge in object-oriented modeling and in UML. This requirement can be considered a barrier to the use of the language SysML.

By using SysML as a modeling language in embedded system projects it is possible to reduce the documenting and communication problem among different teams.

Since the development team of an embedded system is multidisciplinary, involving professionals from different areas, SysML is a very interesting proposal to improve the communication among those many areas. An example to illustrate such statement would be a meeting on an embedded system, to which an electronic engineer comes with the electronic diagrams, a control engineer comes with the control diagrams and a programmer with the software diagrams. The interaction among team members is limited because one does not have knowledge about the modeling language used in the diagrams of the others. However, if SysML was used, the modeling language would be the same for all the different area experts, providing maximum interaction among the team.

References

[1] BOSCH. Sistemas de Injeção Eletrônica. São Paulo: 2005. Available: http://www.bosch.com.br

[2] Edwards, S.; Lavagno, L.; Lee, E.; and Sangiovanni-Vincentelli, A.; Design of embedded systems: formal models, validation and synthesis. Proceedings of the IEEE, March 1997.

[3] Graff, Bas; Lormans, Marco; Toetenel, Hans; Embedded software enginnering: the state of the pratice. IEEE Computer Society, 2003.

[4] Guedes, Gielleanes T. A ; UML 2 – Guia de Consulta Rápida ; NovaTec Editora, São Paulo, 2005.

[5] Object Management Group (OMG). Available: http://www.sisemg.omg.org

[6] System Modeling Language (SysML) Specification. Version 0.9, Jan 2005. Available: http://www.sysml.org