Object-oriented approach for Automatic Train Operation control systems

F. M. Rachel & P. S. Cugnasca
Safety Analysis Group, São Paulo University, Brazil

Abstract

In this paper, an object-oriented approach for Automatic Train Operation (ATO) is discussed in order to compare it to the traditional event-oriented software developed and now used on most of these kinds of systems.

In the last decades, the object-oriented approach has been used in a growing number of applications because it supplies flexibility and understanding easiness for the applications. The UML (Unified Modelling Language) supplies many tools for project analysis and documentation. The various UML diagrams allow many points of viewing the project: the static dimension (class diagrams), the dynamic dimension (sequence and state diagrams) and the method dimension (data flow diagrams). The ATO systems, however, had their development aligned to microprocessors development. So, the ATO software has born on assembly form, because ATO systems were concerned with microcontroller applications. With the rise of the C language, many ATO functions were developed in the C language and added to libraries. This was a first step on building pattern routines for ATO functions treatment. The C language allows firmware commands to be handled in a high-level way and this ability led its use for electronics systems controllers.

Nowadays, on São Paulo's subway system, about 77 trains have ATOs with microprocessors. Among these trains, about 19 trains have ATO software developed in the C language and no train has ATO developed by the object-oriented method. So, this paper analyses the object-oriented approach feasibility for ATO systems. For future applications, we are studying the use of fuzzy logic for train movement control and also studying the development of an ATO fuzzy control software based on the object-oriented approach.

Keywords: Automatic Train Operation, control systems, train control, object-oriented project, object-oriented analysis, object-oriented approach, event-driven approach.

Computers in Railways IX, J. Allan, C. A. Brebbia, R. J. Hill, G. Sciutto & S. Sone (Editors)

1 Traditional event-driven control system

1.1 History

Historically, Automatic Train Control (ATC) systems have the same type of development as the microcomputer systems. This happened because ATC systems were always based on the railway technologies and these technologies were tied to microprocessors development.

Since the second half of 19th century, the railway systems have improved using several technologies based on the available hardware. Table 1 shows the historical development of these technologies and the devices used [1].

Table 1: Abstract of technologies used on railway systems [1].

Year	Technology/Devices used
1850	Mechanical devices
1925	Glass tube devices
1955	Transistor devices
1970	Microprocessor devices

Software developments were limited to the hardware available. Until 1970, the ATC systems functions were realized by electromechanical devices. Relays, glass tubes or electronic circuits (with transistor, diodes, resistors and capacitors) were used to perform the ATC functions and the systems were developed by electrical engineers [1].

With the microprocessors arising, software languages were introduced to the ATC history. Initially, the software language used was the microprocessor assembly language. The peripheral electronic devices were connected as input/output (I/O) to/from the microprocessor. Some electrical and electronic engineers and technicians became specialized on control software development and the first control programs were developed in assembly language. As the time passed by, some pattern control routines were grouped and the first libraries with standard functions were made available by the microprocessor manufacturers.

The arising of C language was a mark on the control software field, because the C language has many facilities for direct firmware handling and has the high-level programming language characteristics. These characteristics made the C language the control software developers main option until nowadays. Many graphical and mathematical functions were grouped and special libraries were made available for control systems use.

The advent of object-oriented techniques has brought many changes to the C language resulting on the C++ language used for object-oriented systems development. Many libraries and standard objects were made available, but traditionally the control systems are developed on event-driven approach using C or assembly language. This approach is preferred to object-oriented one [2].

In this paper, an object-oriented control system is proposed to be compared to a traditional event-driven control.

Computers in Railways IX, J. Allan, C. A. Brebbia, R. J. Hill, G. Sciutto & S. Sone (Editors)

1.2 Traditional event-driven control system characteristics

As written in the previous item, the event-driven control software was previously tied to microprocessors development. So, the event-driven control software has a procedural aspect. The microprocessor firmware is controlled by a sequential command list.

The event-driven control systems are often developed using structured analysis. This approach is based on data flow and contents mapping, where the processes are represented by circles and the data flows are represented by arrows connecting the processes. The software artefacts of this approach are data and control flow diagrams. Using these diagrams, a behaviour model is created and all technical specifications give additional details for the software elaboration [2].

After the technical specifications were made, the next step is to put the plans into practice. This is made by means of the architectural project, where the processes and data flows are integrated. Then, the control structure is translated on a command sequence called procedural project [2].

The program modularisation is used to implement the various software functions. This modularity is very useful on the software testing phase, because the errors can be located, isolated and handled. This characteristic gives flexibility to the control software.

Another important event-driven control system characteristic is the real-time application. Real-time software is highly connected to the real world and must give responses on its time scale. Because of this quick response requirement, the real-time control systems are often dedicated systems and have a customized project [2].

Real-time control systems require time response and lead to an interruption treatment with special care. A most important task must be attended on a specified time period regardless of other events. The normal processing must be interrupted and the corresponding interruption must be handled. In real-time applications, the interrupt treatment is so important that often an additional hardware to implement this interrupt control is required.

In order to keep the normal processing, the return address and the registers contents must be saved before the interrupt treatment. This task is so important that it is implemented on the microprocessor firmware to make it automatic when an interrupt is generated.

The reliability real-time requirements are also important in the cases of restarting and after fault recovering. Critical systems where the control loss is unacceptable often have additional hardware (redundancy) for fault identification and recovering.

The automatic train control system must have a fail-safe philosophy – in the case of fault, the system must be led to a safe condition [2]. This fail-safe philosophy is the base of all São Paulo's subway company systems.

All of these real-time requirements must be either attended by object-oriented control systems. These requirements are control systems characteristics and must be attended independent of the used project approach.

Computers in Railways IX, J. Allan, C. A. Brebbia, R. J. Hill, G. Sciutto & S. Sone (Editors)

2 The proposed object-oriented control system

Now, we present a proposed object-oriented control system. It is only a possible solution among many others and it starts since it's beginning on the object point of view based on the Unified Modelling Language (UML). The UML represents a well defined and standardized methodology developed for object-oriented projects creation and it uses several diagrams for static, dynamic and method aspects project characterization [3]. Figure 1 shows these aspects and the respective diagrams.

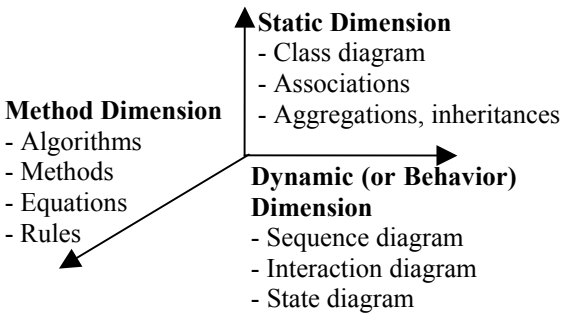

Figure 1: Static, Dynamic and Method aspects of a project.

The system project and construction process have many phases, each one focusing a system aspect. The final result of each phase is a software artefact for system project and construction guide. The model that determines these phases is called waterfall model and is shown on figure 2 [3].

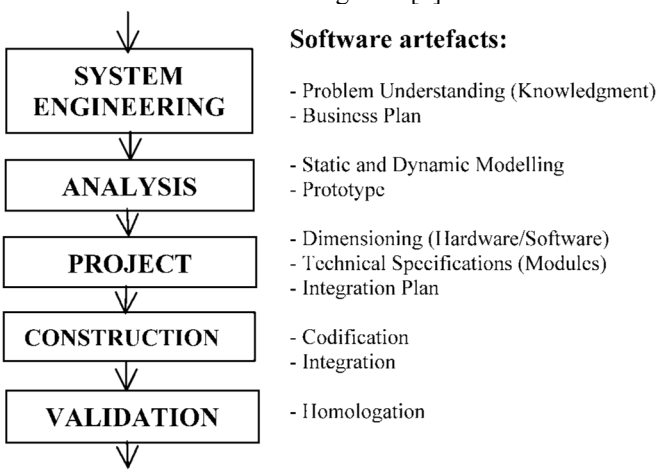

Figure 2: Waterfall model and the software artefacts related to each phase.

Computers in Railways IX, J. Allan, C. A. Brebbia, R. J. Hill, G. Sciutto & S. Sone (Editors)

2.1 System engineering

In this phase, the problem understanding is essential to the project success. The system to be controlled is the Automatic Train Control (ATC). The ATC has two main functions - Automatic Train Operation (ATO) and Automatic Train Protection (ATP). The whole system is composed by the ATC, track signal antenna, programmed stop antenna, tachometers and the Logic Control Box. The Logic Control Box main functions are: propulsion type identification (traction or brake) and motors/brakes control actuation in order to accelerate/brake the train (Figure 3).

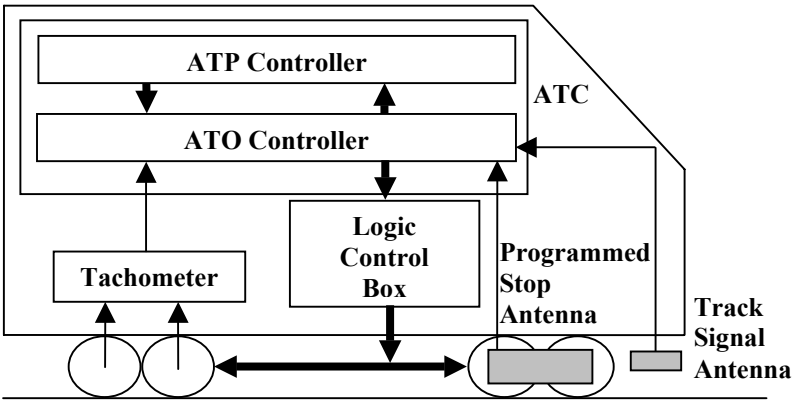

Figure 3: Schematic diagram of the ATC system.

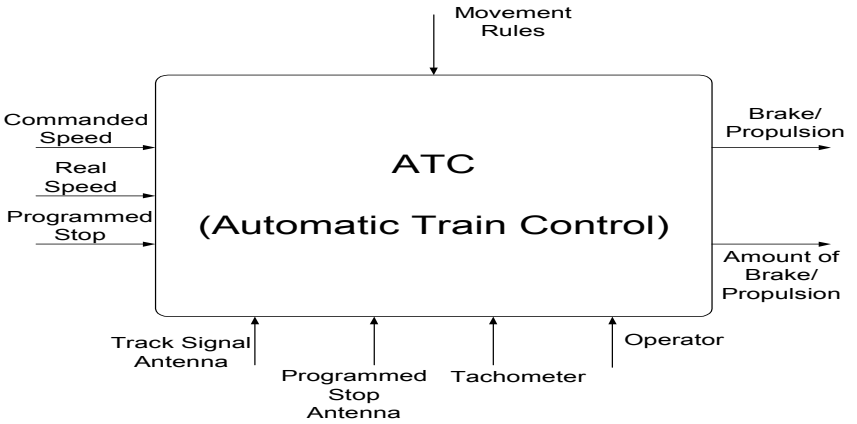

Figure 4: IDEF0 Level 0 (Context Diagram) for ATC modelling.

Computers in Railways IX, J. Allan, C. A. Brebbia, R. J. Hill, G. Sciutto & S. Sone (Editors)

The business plan is a software artefact to be produced to demonstrate the problem understanding. It must be produced in a visual representative form and not only in a text form. One artefact used to represent the business plan is the IDEF0 (Integration Definition for Function).

The IDEF0 has a pattern graphical notation to represent the information flow and the processes used. The IDEF0 has a hierarchical top-down architecture, where processes can be expanded and detailed in hierarchical diagrams. The main process is represented in the IDEF0 top diagram called IDEF0 level 0 or context diagrams [4]. Figures 4 and 5 show the IDEF0 – levels 0 and 1 for ATC modelling.

With the IDEF0 diagrams, the analyst can characterize the system to be implemented, its contents, functions, boundaries and context. The system engineering phase must either detail time and costs needed to implement the solution [3].

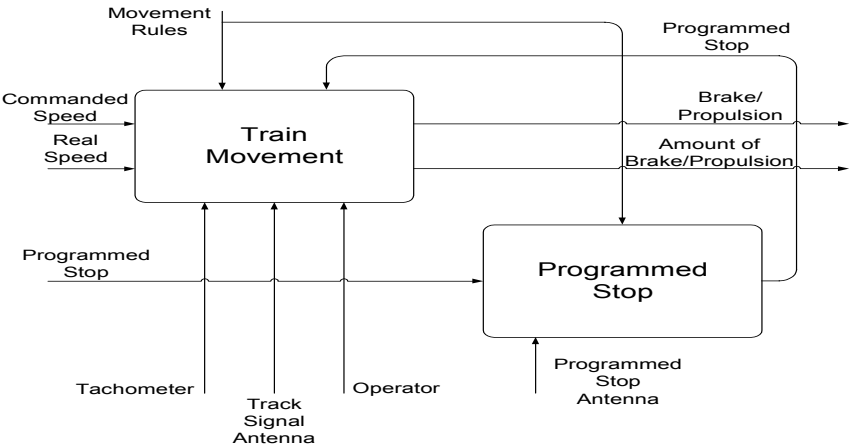

Figure 5: IDEF0 Level 1 for ATC modelling.

2.2 Analysis

After an extensive problem comprehension, the next step is to draw a solution for the problem. This solution must consider the static and dynamic aspects of the system to be implemented and is the base of the system architecture [3].

The UML has some diagrams to represent the system static model as class, objects or use case diagrams. Each diagram must be used in order to attend the system representation needs. The class diagram shows the classes, its attributes and functions besides the relations of interaction, aggregation and inheritance between classes [4]. Figure 6 shows the class diagram for ATC modelling.

The UML language has some diagrams to represent the system dynamic model, as collaboration, state and sequence diagrams. State diagrams show possible system configurations (states) and the events that lead the system from one state to another. Sequence diagrams show the system behaviour during the

Computers in Railways IX, J. Allan, C. A. Brebbia, R. J. Hill, G. Sciutto & S. Sone (Editors)

time under circumstances given by certain scenarios [3,5]. Figure 7 shows the state diagram and figure 8 shows the sequence diagrams for three different scenarios on ATC modelling.

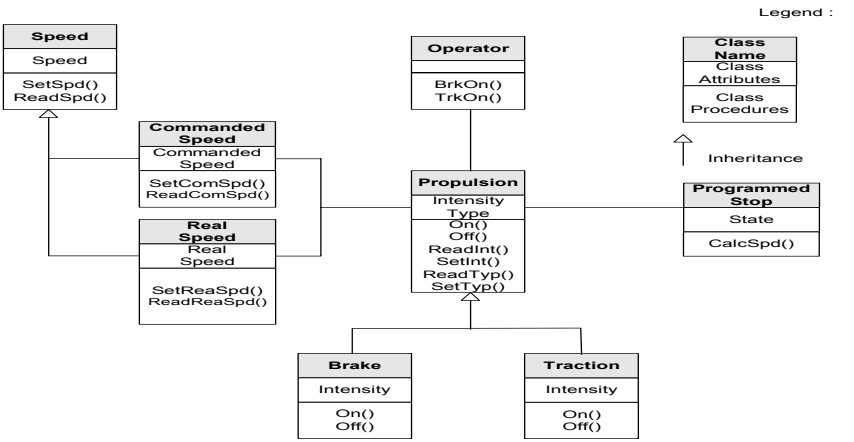

Figure 6: Class diagram for ATC modelling.

The analysis phase must either present a system prototype in order to produce a high-quality system project. In the ATC case, it is difficult to produce a prototype, because the ATC system does not have screens or user interfaces. So, a simple system functions simulation must be provided [3].

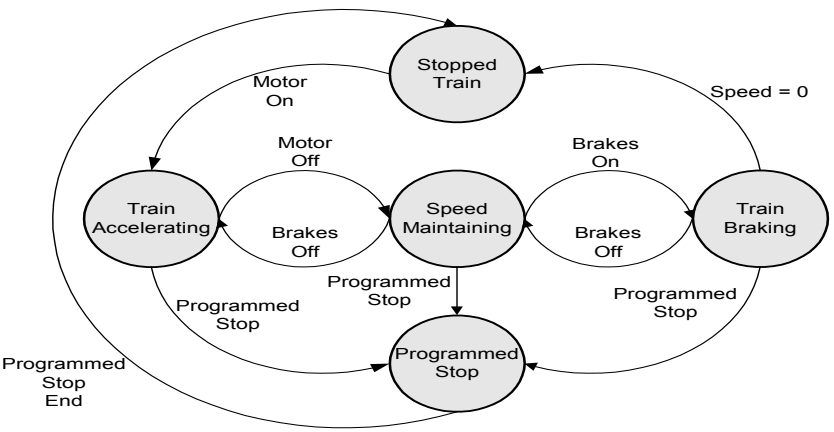

Figure 7: State diagram for ATC modelling.

Computers in Railways IX, J. Allan, C. A. Brebbia, R. J. Hill, G. Sciutto & S. Sone (Editors)

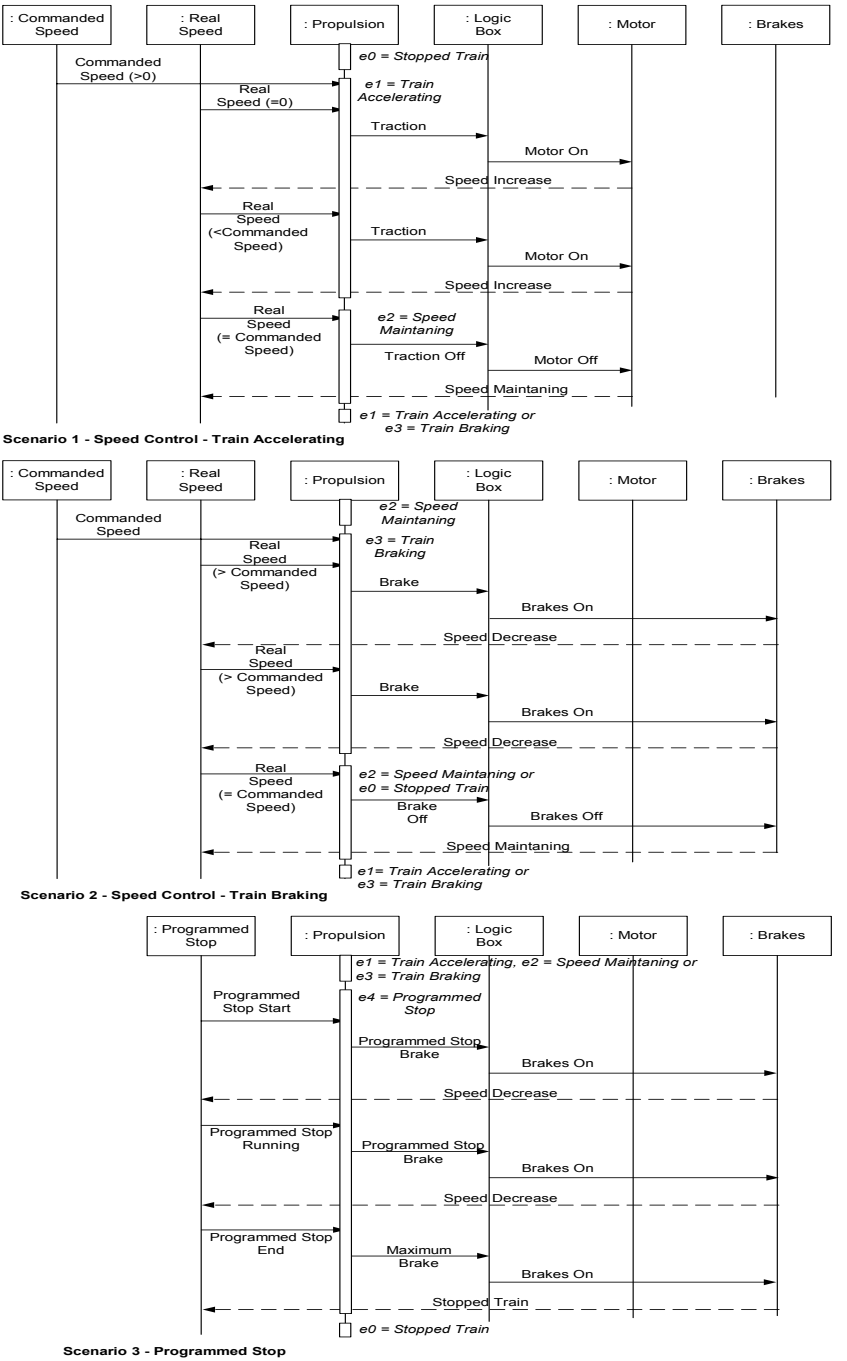

Figure 8: Sequence diagrams for ATC modelling.

Computers in Railways IX, J. Allan, C. A. Brebbia, R. J. Hill, G. Sciutto & S. Sone (Editors)

2.3 Project, construction and validation

After the whole system modelling, the next step is to put the plans into practice. All the functions set on the previous phases must be implemented by means of hardware and/or software. The software must be split into modules (or components) to facilitate the implementation. In the project phase all the modules and components must be technically specified and dimensioned [3,5].

The UML has a component diagram used to show software modules to be implemented. In the case of the ATC, the component diagram is just the class diagram divided on components, each one with a particular functionality. Figure 9 shows the UML components diagram for the ATC modelling.

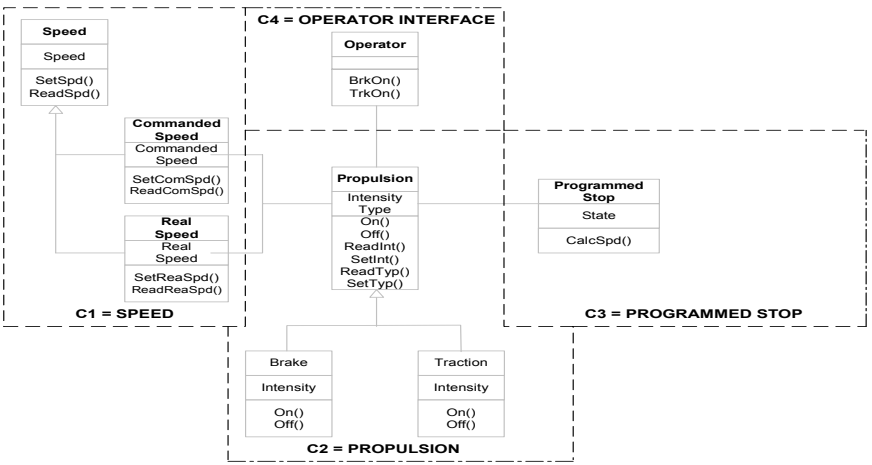

Figure 9: Component diagram for ATC modelling.

The following phases are the construction and validation. In the construction phase, the software modules are coded on a specified programming language and in the validation phase the system is tested and put into operation.

3 Comparisons and conclusions

The object-oriented project developed on the previous section showed that an object-oriented approach is possible for automatic train control systems development. The real-time operation limits present on the object-oriented project are also present on the event-driven control system.

The traditional control system is composed of a main program, many subroutines and a special interrupt routine where all the interruptions caused by external devices (like antennas and tachometers) are handled. Additional hardware for this interruptions treatment is often required, in order to synchronise, organise and give priority to the interrupt requests. The main program is run several times and stopped when an interrupt is generated.

Computers in Railways IX, J. Allan, C. A. Brebbia, R. J. Hill, G. Sciutto & S. Sone (Editors)

The object-oriented approach, on the other hand, is composed by many objects whose internal processes are simultaneously running. When an event occurs on an external disposal, the respective object generates an exception (a message) to the propulsion object that handles the event. The communication between objects is a critical point for real-time applications like the ATC. So, additional control hardware for this communication is also required. Each object runs its processes in an independent form and stops when a relevant event occurs [3,5].

The traditional control system is centred on the hardware control, putting all external devices to work together with the microprocessor while the object-oriented control system is centred on software functions needed to perform the system functions. There are no better or worse method, but because of historical reasons and as control systems are narrow related to electronics engineering, there is a paradigm to prefer event-driven to object-oriented methods. There are many object-oriented control systems, but they are minority on this field.

Nowadays, on São Paulo's subway system, about 77 trains have ATOs with microprocessors. Among these trains, about 19 trains have ATO software developed on C language and no train has ATO developed on object-oriented method. For future applications, studies on using predictive fuzzy logic are being developed by São Paulo's Subway Company with the Safety Analysis Group from São Paulo's University. The predictive fuzzy logic has been proved to be perfectly suitable for train movement control and it could be added on a new ATC project. These studies intend to implement this new ATC project using an object-oriented approach.

References

[1] United States Congress – Office of Technology Assessment, Automatic Train Control in Rail Rapid Transit (Appendix E), Chronology of Train Control Development, http://www.wws.princeton.edu/cgi-bin/byteserv.prl/~ota/disk3/1976/7614/761414.pdf.

[2] Pressman, R. S., Engenharia de Software, A Análise Estruturada e Suas Extensões, Análise Orientada a Objeto e Modelagem de Dados e Projeto de Tempo Real, Makron Books, 1995.

[3] Rumbaugh, J. et al., Modelagem e Projeto Baseado em Objetos, Ed. Campus, 1994.

[4] FIPS – Federal Information Processing Standards Publication (FIPS Std 183), Integration Definition for Function Modelling (IDEF0), 1993.

[5] De Champeaux, D., Lea, D. & Faure, P., Object-Oriented System Development, Object Dynamics and Object Interaction, Addison Wesley, 1993.

Computers in Railways IX, J. Allan, C. A. Brebbia, R. J. Hill, G. Sciutto & S. Sone (Editors)