Dependability Issues for a Curriculum in Real-Time Systems

Jean-Claude Geffroy, Claude Baron, Gilles Motet
LESIA, DGEI/INSA, Complexe Scientifique de Rangueil, 31077 Toulouse, France
e-mail : {geffroy, baron, motet}@dge.insa-tlse.fr

Abstract

This paper develops the dependability issues addressed by the new educational program named 'Real-Time and Systems' of the Department of Electrical Engineering and Computer Sciences at INSAT (National Institute of Applied Sciences of Toulouse). Engineers working in the field of embedded systems have to cope with process control and real-time computing. We show how this new curriculum integrates both aspects with the aim of mastering the development of dependable real-time systems.

1. Introduction

The National Institute of Applied Sciences of Toulouse (INSAT) is an engineering school that delivers several engineer graduations. Each teaching program covers 5 years of study: the first two years are common to all students and the three remaining years take place in different specialization departments (Civil Engineering, Biology, Mechanics, Physics, etc.). The Department of Electrical Engineering and Computer Sciences (DEECS) proposes two 3-year graduate curricula:

- CECS (Control, Electronics and Computer Science),
- ICSE (Industrial Computer Sciences Engineering).

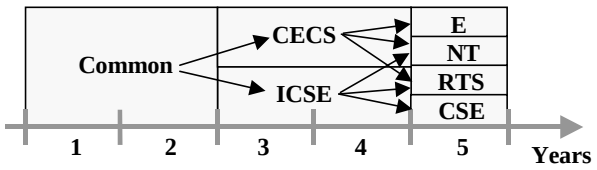

Figure 1. New teaching structure at DEECS

In 1998-99, the last year of study has been restructured with 4 specialized training programs (Figure 1):

- Electronics (E),
- Networks & Telecoms (NT),
- Real-Time & Systems (RTS),
- Computer Science Engineering (CSE).

Two of them (NT and RTS) are accessible to all students from the fourth year.

In this paper, we consider the new specialization program entitled 'Real-Time and Systems' (RTS).

Real-time systems are used in more and more applications of modern life, and they are given more and more responsibilities. This evolution is particularly important for embedded systems (avionics, spatial and transportation means in general). The responsibility increase of these systems implies special classes of issues and, consequently, special classes of methods and techniques to handle them [3]. The development, production and utilization steps in this case must provide a system satisfying the conventional performance requirements but also new dependability requirements.

The city of Toulouse is well known for critical real-time system development and production in aeronautics (Aerospatiale, Airbus Industry), space (Matra Marconi Space, Alcatel Space), control systems for vehicles (Siemens), etc. These companies employ a growing number of engineers who must have received an appropriate training. It is the aim of the new RTS curriculum to provide students with such training.

An overview of the RTS curriculum is presented in section 2. Then, we analyze dependability issues and solutions in section 3 and show in section 4 how these needs are covered by the teaching program. The last section (section 5) deals with our expectations concerning this program.

2. A new teaching curriculum: Real-Time & Systems

Automatic Control and Computer Sciences are two domains which are generally mastered by scientists having different cultures. However, most real-time industrial

applications require knowledge in both domains. This is true, for example, for aircraft flight control or automatic gas injection in car engines.

The RTS specialization program results from a synergetic process to mix these two domains in order to cope with embedded real-time systems. It comprises 420 hours of theoretical and practical training period at INSAT, and a 4-month full-time training period in industry (not covered by our paper). An analysis of the industrial expectations led us to define five scientific groups (Figure 2):

- Control (80H), which is the application domain of this curriculum,
- Real-Time (110H), heart of this curriculum,
- Laboratory (71H), which integrates the different theoretical knowledge on practical applications,
- Computer Networks (53H), as embedded systems are frequently distributed using communication means (for instance in the 'Integrated Modular Avionics' concept recently introduced [17]),
- Engineering Techniques (105H) such as project management, English training, technical complements, etc.

The fourth group brings knowledge on industrial local networks, services and protocols and client-server architectures. The last group provides several complementary technical material and general useful teaching. The three other groups, which constitute the kernel of our specialization, will be analyzed in section 4.

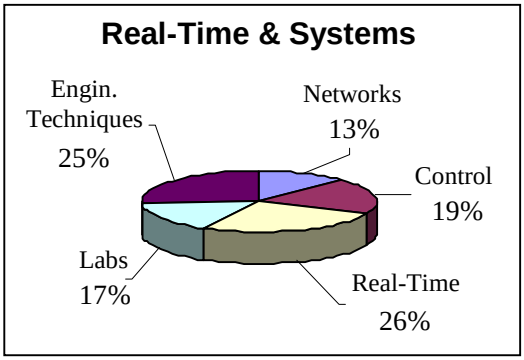

Figure 2. Organization of the RTS curriculum

This program has been defined by the DEECS with the useful help of several industries (Aerospatiale, Turbomeca, etc.), research laboratories in Automatic Control, Computer Sciences and Electronics (LAAS-CNRS and LESIA) and other academic institutions such as ENSAE (National Engineering School for Aeronautics and Space) and AIME (Inter-University Center for Micro-Electronics).

We will now introduce the dependability needs for the RTS curriculum (in section 3) before analyzing (in section 4) the elements provided by the lectures and laboratories of the main teaching groups for answering these needs.

3. Dependability issues

Figure 3 shows the general context of a control system: the controller is connected to the process in order to perform a mission (for example, avoidance of the blocking of the wheels of a car during breaking) in a given environment (characterized for instance by temperature, vibration, with the possible action of human beings).

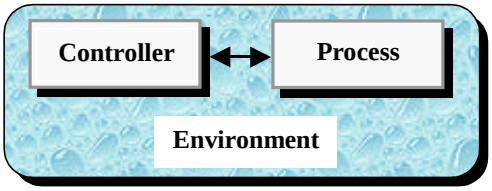

Figure 3. Real-Time Control context

The controller must follow a full development cycle before being produced and used. We consider the five main steps of a simplified life cycle (Figure 4):

Specification: it formulates a solution to the client requirements with formal and/or informal specifications.

Design: the specifications are progressively transformed into a structure made of interconnected functional modules.

Implementation: an executable system is realized using hardware/software technology.

Production: industrial realization of the controller.

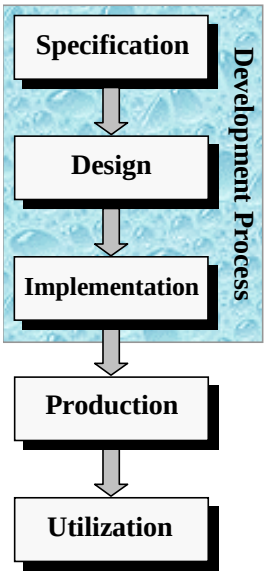

Figure 4. Life Cycle

Utilization: the controller is connected to the process in the application environment in order to perform the desired mission.

Unfortunately, practical experience shows that various problems, named *faults*, may alter the controller structure. These faults may then be activated as *errors* in the internal functioning of the controller. Finally, contamination of the errors through the structure of the controller will produce *failures* of the mission. Failures of real-time systems may have a wide range of consequences on the application (standard DO 178B [7]): minor, severe, serious and catastrophic. A recent significant example to mention is the failure of the first launch of *Ariane V*.

Faults may alter the controller, the process and/or the environment. Concerning the controller, they are introduced during any of the five steps of the life cycle: specification faults, design faults, implementation faults, production faults and use faults (Figure 5). They may be caused by humans involved in the life cycle, the development or production tools and the technology chosen for the realization of the controller. Traditionally, three kinds of faults are identified: functional faults (for example a wrong feature – transistor, gate or statement - has been employed during the design step), technological faults (e.g. a 'stuck-at' fault in an electronic circuit) and perturbation produced by the process or the environment.

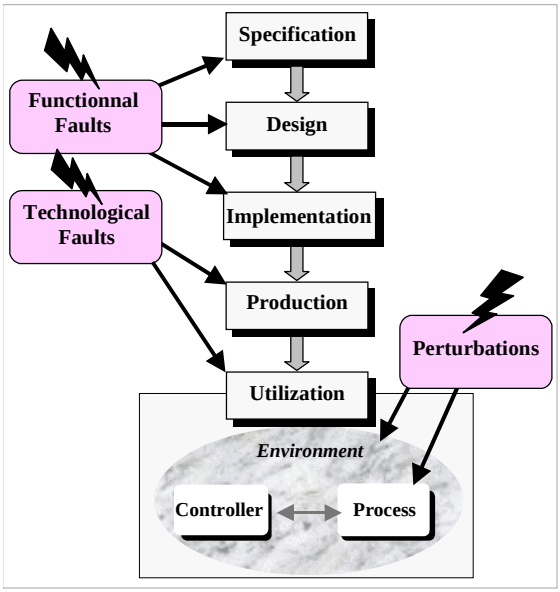

Figure 5. Fault production

The mastering of these numerous and varied problems make necessary the use of three complementary groups of methods and techniques [11, 16]: *Fault Avoidance*, *Fault Removal* and *Fault Tolerance*.

Fault Avoidance. The aim of these methods and techniques is to avoid the introduction of faults during the creation steps. It requires to choose proper methods for specification, design, implementation, production, and to correctly use them. The use of 'coding rules' is an example of such a means to avoid the creation of programming faults.

Fault Removal. The aim of this approach is to remove residual faults that could have been introduced at each step. For example production or maintenance testing methods belong to the fault removal family: an external input sequence is applied to an executable product and the outputs produced are analyzed in order to detect and/or to localize faults.

Fault Tolerance. The aim is here to tolerate the activation of faults during the utilization step. This result is generally obtained by the use of redundancy techniques. For example, the N-Version programming technique used in software design is based on 'active redundancy'.

Figure 6 illustrates these various groups of protective actions along the considered life cycle. Let us note that in the general case, a maintenance step is coupled with the utilization step. During this step, testing operations are applied to the system in order to detect faults, diagnose them and, if possible, repair the system.

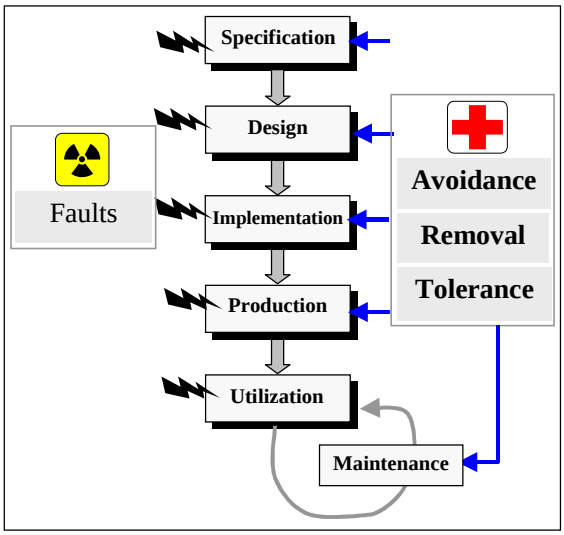

Figure 6. Protection mechanisms in the life cycle

All the related means are chosen to fulfill the dependability requirements. Some criteria are considered in order to estimate the quality of the proposed system: *reliability* (probability that the system remains good from its creation to time *t*), *availability* (probability that the system works properly at time *t*, integrating the corrective actions), *testability* and *maintainability* (ability to be

tested and modified during the maintenance) and *security* (probability to work without catastrophic failures).

So, according to our analysis, a full training of specialists in real-time embedded systems requires both:

- the understanding of the characteristics of the process to be controlled and its control methods,
- the understanding of real-time systems and their development process (know-how),
- the mastering of failure mechanisms (creation of faults, activation of errors and contamination of these errors) and protection techniques (to avoid, remove and/or tolerate faults).

4. How the RTS curriculum addresses these issues

During the two first teaching years at the DEECS (third and fourth years at INSAT), all students receive basic education in Control, Electronics and Computer Sciences.

In this section, we will successively analyze the three main groups of the RTS program: Control, Real-Time and Laboratories. For every teaching unit, we will indicate the step of the life cycle that is referred to, and show how the dependability issues are addressed.

4.1 Process control

This group deals with the process and its control. It constitutes a necessary background prerequisite to any real-time development of a controller. The problem is to model the process and to control it in order to satisfy performance and optimization requirements but also robustness and dependability criteria. Indeed, before any development and implementation of a Real-Time Control System, it is necessary to master the modeling and the control of the process. In fact, any misunderstanding may have negative consequences on the dependability criteria of the final product.

The five associated courses are illustrated and integrated to the real-time knowledge during the laboratories.

Multivariable systems. The goal of this unit is to introduce state-space description and control of multivariable systems. The first part deals with state-space and matrix-fraction descriptions of multivariable systems. In the second part are presented methods of pole and eigen-structure placement by state and output feedback, with application to robustness. The different phases of the course are illustrated with an aircraft automatic flight control system.

Optimal control. This course aims at presenting the methods and their associated algorithms to maximize (or minimize) a cost function subject (or not) to equality and inequality constraints, in a structure independent of the variable time. In a second step, these methods are applied to the optimal control of dynamic linear systems, in discrete time, with an associated quadratic cost function. The problem is here transformed into the resolution of the very well known discrete Riccati equation. At the end, the equivalent control problem, in the continuous case is also considered.

Non-Linear systems and their control. The first part of this unit deals with the main characterization of non-linear systems. Then, in a second part, the control of such systems is treated. Models in the form of recurrence relationship (or map) result either directly from intrinsically discrete systems (the information is obtained only at discrete times), or indirectly from continuous systems observed at discrete times, or from a numerical simulation. The lecture presents:

- a prospective of such models with an introduction to chaotic dynamics, and fractal sets,
- the mathematical tools derived from the differential geometric theory,
- a study of fundamental notions like non-linear change of co-ordinates by use of diffeomorphism, relative degree, input/output and input/states linearization,
- the extension to the MIMO case and an introduction to the flatness theory.

Stochastic processes and identification. The aim is to present means used to model and analyze some processes. The main part deals with the modeling problem for systems for which application of physical laws is not possible. In this case, the only way is to derive a model from an input/output experiment. Methods allowing this derivation are known as "Identification methods" and this course presents the main methods to be used when the derived model is linear.

Symbolic and connectionist control. This course presents an alternative approach to solve control problems where classical mathematical models are not available, using here soft computing tools like fuzzy logic (to take profit of an expertise formulated in terms of fuzzy rules) and neural networks (to extract knowledge from data). In a first part, the basic elements of fuzzy logic are presented and used with some extends for the design of fuzzy controllers. In a second part, the general concepts on neural networks are offered. The ways to use these networks for the identification and control of non-linear systems are explained. Finally, complementary of fuzzy systems and neural networks is underlined as arrangement for the design of neuro-fuzzy systems.

4.2 Real-Time

Dependability. This first lecture presents the general issues introduced in section 3: the basic notions and concepts associated with the destructive mechanisms (faults and their propagation as errors) and then the methods and techniques to protect the system.

Thus the various types of faults (internal/external, functional/ technological) are explained and the contamination phenomenon (faults – errors – failures) is analyzed. To handle these problems, the dependability domain proposes methods and techniques suitable to the different steps of the life cycle. As previously explained, these solutions are conventionally split into three groups: Fault Avoidance, Fault Removal and Fault Tolerance. The dependability assessment means (estimation study at the development steps or useful life experience analysis) are based on statistical models expressing the criteria of reliability, availability, testability and security. Let us note that safety is a fundamental criterion when real-time applications are concerned due to the catastrophic consequences of non-mastered failures.

The Fault Avoidance methods are illustrated on the software implementation step. We show how faults introduction in Ada programs can be reduced thanks to style rules [1]. Less conventionally, we also introduce the use of the FMEA (Failure Modes and Effects Analysis) [16] and the FTM (Fault Tree Method) [14] as Fault Avoidance means. The test methods are studied in the second part of this lecture (Fault Removal approaches). The principles of functional and structural testing with or without fault models are exposed. Some significant test methods applied to electronics technology (at gate level) and software technology (Ada, C and C++) are finally detailed.

The Fault Tolerance techniques are finally presented and illustrated by means of Ada programs [16].

Performance evaluation. The aim of this unit is to present concepts, models, methods and tools to evaluate the physical performance (space/time) of discrete event dynamic systems implying the execution of tasks using resources. These limited resources may be multiple and distributed. The presented tools are used to analyze the predictable performance of a system during its design. So they are useful to help in the optimization of its parameters, but also to guarantee the accordance between a system design and the temporal requirements. This analysis contributes to Real-Time System dependability. Markovian models, queuing theory and discrete event simulation are the basic components of this unit dedicated to tasks and resources found in computer systems.

Temporal models. Whereas the previous lecture concerns asynchronous tasks cooperating by means of resources such as buffers, the lecture dealing with temporal models focuses on synchronized tasks. Two kinds of models are presented: the temporal extensions of the composed automata and those of Petri nets. Analysis concerns qualitative viewpoints (properties on precedence relationships, etc.) and quantitative ones (measurements on temporal parameters).

Such studies are useful to master the dependability of Real-Time Systems. Indeed, discrete event system modeling must be checked from the first steps of the development. This is particularly useful at specification and design levels as a significant amount of faults is introduced during these steps [17]. When real-time systems are concerned, temporal modeling means must be available allowing analysis of the model correctness to be processed.

Object-oriented development. The object-oriented approach is useful during specification [10], design and programming steps for software applications belonging to various domains (Information Systems, etc.). Recently, this point of view has also been used to develop real-time applications [8, 9]. This approach is relatively natural because it easily takes the controlled process objects into account. Thus, its using reduces fault introduction. Moreover, verification techniques, in particular testing procedures, were recently proposed for object-oriented programs in order to detect and to remove faults. Unfortunately, they do not concern temporal faults.

Synchronous languages. This lecture concerns the implementation of reactive applications (in avionics in particular, where response time is a critical parameter). The behavior of these applications must be characterized and mastered to guaranty the correctness of their functioning. Concurrent languages are mostly asynchronous, they come from time-sharing operating system concepts [4]. Execution of system components is therefore not deterministic. Testing such a system is difficult because of its non-deterministic features and also because measuring the execution time is a tricky issue. Whenever the component execution time can be predicted or neglected with regard to the process dynamic, then a sequential program (automaton) can be automatically derived (by synchronous language compilers) from the design of parallel activities. This program is then deterministic and its properties can be formally verified in a static way.

The goal of this course is to present the main characteristics of the synchronous approach with regard to the asynchronous one [12]. The lecture is illustrated by the LUSTRE language and the graphical simulation and verification by using the LESAR environment.

Real-time development. This lecture provides an overview of the real-time software development process. As explained in section 3, the mastering of this process is necessary to avoid faults and/or to detect and remove them at the end of each step instead of performing these difficult operations during a final test step.

Its contents are scheduled according to the development cycle.

Specification: we present how Petri nets can be used to express the behavior of real-time systems. At present time, we are examining how UML could improve this part of the lecture [8]. However, we consider that we must wait for a stabilization of the standard features to take the real-time requirements into account or for new proposals concerning the use of the UML annotations to express real-time behaviors.

Design: HOOD is presented as design model [13]. This model was chosen for two reasons: it is used in spatial area, which is important in Toulouse (required by ESA – European Space Agency-) and it proposes an associated method based on design guides useful to avoid faults.

Programming: we present Ada95 language real-time features (tasks, protected objects, synchronization and communication means) [5] and the "Real-Time" annex of the language standard. This language was chosen because it is used for aeronautics and spatial embedded systems, numerous verification means exist (such as ANNA [15], SPARK [2], etc.) and students received at INSAT good basic knowledge on the conventional features of Ada language during the first years of their training.

Real-Time Executive. Real-time systems are used to control a process embedded into an asynchronous environment. They must quickly react in order to treat every event. Thus, time consumed to execute internal operations must be as short as possible. Indeed, a late answer is an error, which can alter the system safety and consistency. We stress the importance of the Real-Time Executive on the dependability of the final application.

The use of a real-time executive (VxWorks/Tornado [18] for example) will help the system designer not only because it provides high-level primitives, but also because it allows to guaranty fixed performances (if predictable scheduling methods are employed) (cf. the Task Scheduling unit). Moreover, in the case of embedded systems, execution resources are limited, so must be the functionality of executives. How VxWorks reaches this goal is exposed in the course and is developed during the laboratories.

Task Scheduling. This lecture concerns real-time software implementation. At first, it presents the internal structure of the real-time kernels and the principles of the implementation of the components (handling of real-time primitives, time management, scheduling, dispatching, including preemption). We show that this implementation makes difficult the demonstration of the determinism of the behavior of real-time applications, determinism that is however included in the dependability requirements.

In the second part of the lecture, we present predictable scheduling policies for non-periodic tasks (Jackson) and periodic tasks (Rate Monotonic Scheduling, Earliest Deadline First) and based on servers having fixed priority (Polling Server) or dynamic priority (Dynamic Priority Exchange) [6]. We conclude by explaining the use of these policies in real-time kernels considering Ada and VxWorks real-time kernels.

4.3 Laboratories

Principles. All concepts, models and methods of the real-time area, developed during the different courses, must be illustrated through several practical laboratories. We thus identified a group of laboratories in the RTS program covering the main real-time issues. Moreover, these laboratories are coupled together to show how the different concepts, models and methods, coming from different areas of real-time and control, can be integrated in order to reach global application goals (design and test issues, integration of the control and implementation parts).

Labs are organized on a base of 12 student groups supervised by one tutor (lecturer, researcher or engineer). According to the treated subject, the students work by 2 or 4 on identical or complementary tasks. They must produce a well-documented report providing their solutions but also justifying their choices. The report is made during homework.

Process Control (25 training hours). The control laboratories constitute the convergence point for all different courses of the control group in the RTS program. Students have to make a full study of an actual real-time system, the control system of a gas turbine for helicopters. They have to analyze the process, build a model for it (using the Matlab tool to identify and adjust the control law parameters), simulate this model, and finally define the system control unit and experiment on it. These laboratories are led in collaboration and with the practical assistance of a senior engineer of Turbomeca avionics industry.

Testing (9 hours). The aim of the test laboratories is to provide the students with concrete experience with industrial tools. It illustrates the main testing methods for hardware and software.

The hardware part is conducted on Sun workstations at the AIME (Inter-University Center for MicroElectronics). The students use Cadence CAD and Test tool in order to analyze the fault coverage of some functional and

structural sequences applied to various combinational and sequential circuits described at gate and VHDL levels. A Tektronix industrial tester is also used to show fault testing of ASIC circuits.

The software part is performed on PC stations at DEECS and deals with two points of view: the functional and the structural testing. This last aspect must illustrate a wide range of coverage, from simple coverage (statement coverage) to complex one (MC/DC: Modified Condition/Decision Coverage) which is required for instance by aircraft certification organisms. We use two tools for Ada, C and C++ languages: Logiscope produced by Verilog and Attol Unitest-Coverage produced by Attol Testware. At the beginning, an application specification is given to the students. They must deduce a functional testing sequence, which is then applied to a given executable implementation of the application. The existing failures must be highlighted. Then, the structural testing tool is used to assess the functional test sequence; a 'coverage rate' is deduced for various elements (statements, etc.). With these results, students are asked to complete the test sequence in order to increase the coverage.

Real-Time (37 hours)

i) Implementation with the VxWorks (Tornado) executive. These laboratories offer a project consisting in developing a communication system using a simulated parasitic telecom line. The sent messages must be coded using a detecting & correcting code to protect the transmission against parasitic aggressions. Students must design and implement the transmission system and also a supervision unit for on-line analysis of the coding efficiency. This project illustrates the basic features of VxWorks (primitives and scheduling principles), and the notion of execution on a remote target system. It also constitutes a good introduction to the final laboratories (next section). Students are invited to manage their working planning at every step (specification, design and test).

Another interesting point can be mentioned about this project: it shows how important and difficult is the design, test and debug of a real-time application. Tornado provides several useful tools for that such as a complete debugger (Xwind) and a graphical distant control tool (WindView).

ii) Design and realization of a control real-time system. This final laboratory consists in defining and implementing the control rules of a real-time system: a satellite embedded inertial wheel. Its main interest relies on the fact that it offers the simultaneous study of process control aspects and of real-time computing aspects, which are strongly correlated in most real-time industrial applications.

The identification of process parameters is made using Xmat tool. The software implementation requires the use of Tornado.

5. Conclusion

Real-time systems constitute a scientific area in which numerous researches are conducted as well as an industrial domain employing more and more engineers. The new RTS curriculum offered at INSAT aims at training specialists in this domain having knowledge in both control and real-time areas. This curriculum also considers dependability as an important requirement when real-time systems are concerned. We showed in this paper how the courses provide answers to this need. The mastering of dependability of the real-time systems is particularly important at Toulouse where critical embedded systems are developed.

The main difficulty is to provide the students with a high level of knowledge during a few numbers of lecture weeks. So, we did some choices to select the curriculum contents, taking the characteristics of the systems developed at Toulouse into account. Additionally, we tried to balance theoretical presentations (lectures) and practical exercises (laboratories). Moreover, in order to train the students to manage projects and to correlate the various courses, we created laboratories dealing at the same time with several aspects of the curriculum. The RTS training started in September 1998. We hope that it will be successful because:

- we created the curriculum considering industrial requirements and not only lecturers' proposals;
- we received advice and help from firms developing real-time systems and from laboratories conducting researches in this domain in order to implement the curriculum;
- the project was supported by lecturers, engineers and technicians working at INSAT;
- the employment in this area is increasing, in particular in Toulouse in aeronautics (Aerospatiale), in spatial (Matra Marconi Space, Alcatel Space, French Space Agency) and for other embedded real-time systems (for instance, Siemens develops systems for cars, Actia for buses, Motorola for mobile phones).

Acknowledgements

The authors would like to thank all people from the academic and industrial fields who contributed to the

definition and the implementation of the new Real-Time & System curriculum.

References

[1] C. Ausnit-Hood, K.A. Johnson, R.G. Pettit, S.B. Opdahl, " Ada 95 Quality and Style ", Lecture Notes in Computer Sciences, N° 1344, Springer, 1997.

[2] J. Barnes, " High Integrity Ada. The Spark Approach ", Addison-Wesley, 1997.

[3] C. Baron, J-C. Geffroy, G. Motet (editors), " Embedded System Applications ", Kluwer Academic Publishers, 1997.

[4] A. Burns, A. Welling, " Real-Time Systems and Programming Languages ", Addison-Wesley, second ed., 1997.

[5] A. Burns, A. Welling, " Concurrency in Ada ", Cambridge University Press, 1998.

[6] G.C. Buttazo, " Hard Real-Time Computing Systems. Predictable Scheduling Algorithms and Applications ", Kluwer Academic Publishers, 1997.

[7] DO-178B, " Software Considerations in Airborne Systems and Equipment Certification ", Issued in the USA by the "Requirements and Technical Concepts for Aviation" (document RTCA SC167/DO-178B) and in Europe by the European Organization for Civil Aviation Electronics (EUROCAE document ED-12B), December 1992

[8] B.P. Douglass, " Real-Time UML ", Addison-Wesley, 1998.

[9] J.R. Ellis, " Objectifying Real-Time Systems ", SIGS Books, New York, 1994.

[10] D.G. Firesmith, " Object-Oriented Requirements Analysis and Logical Design ", Wiley, 93.

[11] J-C. Geffroy, G. Motet, " Dependability of Computer Systems ", (in french), coll. 'Informatiques', Inter-Editions Masson-Dunod, 1998.

[12] N. Halbwachs, " Synchronous Programming of Reactive Systems ", Kluwer International, Series in Engineering and Computer Science, Vol. 215, 1994.

[13] HOOD Users'Group (J.P. Rosen editor), " HOOD. An Industrial Approach for Sofware Design ", Messages, Toulouse, 1997.

[14] N.G. Leveson, J.L. Stolzy, " Safety Analysis of Ada Programs Using Fault Trees ", IEEE Transactions on Reliability, Vol. R-32, N° 5, IEEE Publisher, December 1983.

[15] D. Luckham, " Programming with Specifications. An Introduction to ANNA, a Language for Specifying Ada Programs ", Springer-Verlag, 1990.

[16] G. Motet, A. Marpinard, J-C. Geffroy, " Design of Dependable Ada Software ", Prentice Hall Inter., January 1996.

[17] G. Motet, " Importance of Specification Means to Design Integrated Modular Avionics Systems ", in Embedded System Applications, Kluwer Academic Publishers, 1997.

[18] WindRiver Systems, VxWorks Reference Manual.