15th Triennial World Congress, Barcelona, Spain

FRAMEWORK FOR DEVELOPING REAL-TIME MOBILE ROBOTIC APPLICATIONS BASED ON BEHAVIOURAL MODELS[1]

Houcine Hassan, Rafael Martínez, José Simó, Alfons Crespo

Dept. of Computer Engineering (DISCA), Universidad Politécnica de Valencia.
Camino de Vera s/n. 46071 Valencia. SPAIN.
e-mail:{husein ,rmartin, jsimo, acrespo}@disca.upv.es

Abstract: In complex real-time control systems such as autonomous mobile robotic systems, the use of development tools for the design, analysis and validation of robotic applications is highly desirable, specially to improve the robot performances and to avoid early software and hardware design faults. This paper presents a framework for developing real-time mobile robotic applications. The main feature of the environment is that it permits the integrated analysis and validation of the functional behaviour of the robot (i.e. robot carrying out its plans) and the guarantee of the temporal constraints of the system processes (i.e. reactive and deliberative process execution). The verification of the correctness of the functional requirements is supported by means of a robot simulation tool that reflects the progress of the system at the application level. Likewise, a temporal analysis tool, based on scheduling analytical techniques, guarantees the schedulability of the system load at the execution level. The usefulness of the development tool is shown through design examples applied to mobile robotic applications. *Copyright © 2002 IFAC*

Keywords: Autonomous mobile robots, real-time systems, behavioural models, formal specification, simulation tools.

1 INTRODUCTION

In complex real-time control systems such as autonomous mobile robotic systems, the use of development tools for the design, analysis and validation of robotic applications is highly desirable, specially to improve the robot performances and to avoid early software and hardware design faults (Capucho, 2001; Storch 1997). Graphical tools should present the progress and the state of the system at different levels (i.e. planning, execution). These tools will greatly increase the reliability and safety of deployed autonomous systems and hence the efficiency of autonomous system designers (Kortenkamp, 2000).

Generally, in most of the traditional robot control architectures, real-time constraints have been enforced by well engineered implementations largely based on resource underutilisation and a limited set of guaranteed computation, often a single real-time loop (Beccari, 1999). However, the correctness of real-time mobile robotic systems not only depends on its correct functional behaviour, but also depends on its correct temporal behaviour. That is, the system should meet besides its functional specification, the timing requirements of its processes, even in the worst case (Shuhua, 2000; Nilsson, 1998).

In this paper, an integrated environment for specifying, validating and developing real-time mobile robotic application taking in consideration the above-commented requirements is proposed. The environment is composed of different analysis tools for validating the functional behaviour of the robot (i.e. robot carrying out its plans) and for assuring the guarantee of the temporal constraints of the system processes (i.e. reactive and deliberative process execution). The verification of the correctness of the functional requirements is supported by means of a robot simulation tool that reflects the progress of the system at the application level. Likewise, a temporal analysis tool, based on scheduling analytical techniques, guarantees the schedulability of the system load at the execution level.

At the application level, the functional requirements of the robotic application are specified by means of behavioural-based models (Gat, 1998). Behavioural models have been widely used to represent advanced robotic systems operating in uncertain dynamic environments, combining information from several

[1] This work has been partially funded by the Spanish Government grant CICYT TAP99-1226-C02

sensory sources and with different vehicle dynamics. Prior knowledge of the domain may be incomplete, and in a dynamic environment, reasoning must be deliberative and fast enough to respond to unexpected events. Consequently, the planning systems must be reactive and take into account information about the current state at regular time intervals. Therefore, mobile robots must combine deliberative goal-oriented planning with reactive sensor-driven operations.
The proposed framework includes a specification language that allows the functional description of the behaviours of the application by means of a graphical tool. A simulator of robotic applications permits the verification of the correctness of their functional requirements. The robot quality of service (QoS) can also be analysed in terms of promptness (increasing the speed will increase the promptness) in reaching the objectives and in terms of the ability of the system in selecting the more appropriate activity to be executed (depending on the available time).

At the execution level, the real-time model, according to the hardness of the timing constraints of the behavioural components, will be composed of critical and optional tasks. The requirements of the critical tasks are related to low-level reactions (e.g. detecting and avoiding an obstacle) and must be strictly guaranteed. The critical level can also handle behavioural-based processes operating at variable frequencies (Stankovic, 1999; Caccamo, 2000). Consequently, fixed periodic tasks (Audsley, 1995) are used to represent reactive processes that don't depend on the application parameters, as data acquisition processes or controllers. However, for dealing with reactive processes depending on application parameters, as local map process or obstacle avoidance behaviour, variable periodic tasks have been incorporated to the task model (Hassan, 2001). The high-level deliberative processes of the behavioural model do not require strict guarantees (e.g. Global map building) and can be modelled by optional tasks (Audsley, 1991; Liu, 1991).
An automatic temporal translator integrated in the development tool generates, from the robotic application description, the temporal characteristics of the real-time task set. The environment embodies also an analysis tool for the verification of the fulfilment of the task temporal requirements. The analysis proofs are supported by pre-emptive fixed priority-based techniques (Audsley 1995) and permit to establish whether the system is schedulable or not, even when the system is in a transient state (Caccamo, 2000; Hassan, 2001). The graphical interfaces facilitate the analysis by allowing the visualisation of the task set execution, the system speed, the utilisation and the workload during the execution of the simulation.

Following the introduction, the paper presents the behavioural model for the description of mobile robotic systems in the section 2. In section 3, the real-time execution model for guaranteeing the schedulability of the system is detailed. The integrated development framework is exposed in section 4. Section 5 discusses the design phases taken in the development of mobile robotic applications. In section 6, conclusions and future work are summarised.

2 BEHAVIOURAL MODEL

Mobile robotic applications will be specified by means of a behavioural-based hybrid model (Hassan, 2001), which is composed of a set of distributed behaviour entities communicating through a blackboard memory. The behaviour entities define the task-specific knowledge for a very narrow portion of a vehicle control. The behaviours run asynchronously, using specific sensor data and control algorithms, and providing contributions to the arbiters. A control action and a motivation parameter compose a contribution. The motivation parameter reflects the behaviour's importance in the current mission. Based on the motivation values, the arbiters apply a co-operative protocol for composing contributions. The emergent control action is sent to the corresponding robot actuator. The components of a robotic application can pertain to one of the following subsystems of the behavioural-based architecture:
- Blackboard memory: represents the relevant information needed by the robot to accomplish its objectives (i.e. vehicle internal state, environment representation, robot goals, contributions, etc.). This memory is the communication interface between the different processes of the architecture.
- Sensory system: contains sensor processes representing real-time tasks that manage the hardware sensory subsystems. These processes deal with raw data like ultrasonic echoes, infrared intensity, and battery charge.
- Reactive system: includes reactive behaviours and fusion processes. This system is necessary so that the vehicle can navigate safely and can take actions in real-time. The reactive behaviours process the information coming from the sensors in order to apply primitive reactions (i.e. obstacle avoidance behaviour). The reactive fusion processes generate elaborated perceptual information by fusing basic information obtained by different sensory processes.
- Deliberative system: contains deliberative behaviours and fusion processes. This system is responsible of high level planning. While the deliberative behaviours support the long-term planning of robot objectives (i.e. trajectory planning behaviour), the deliberative fusion processes generate global maps of the environment.
- Arbitration system: solve conflicts among behaviours that try to access simultaneously the same actuator. Arbiters receive behaviour contributions, and, based on a composition protocol fuse them and issued emergent control action to the corresponding controller.

2.1 Specification of behavioural entities

The description of the components of the applications is performed by using a specific-purpose language developed with the *flex and bison* tools (Donnelly 1995). A graphical interface integrated in the framework facilitates the definition of the processes

composing the applications. Sensory processes and reactive fusion algorithms are described by the attributes shown in the Figure 1.

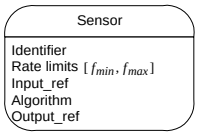

Fig. 1. Sensor attributes.

The components of a sensor process are the sensor identifier, its minimum and maximum execution frequency, a reference to an object in the blackboard memory for gathering the device register data (i.e. encoder tics), the algorithm code implemented by the sensor and a blackboard output reference for storing the sensory information results (i.e. location). It can be pointed out that the execution time of some sensor processes can be variable because it can depend on the robot environment (i.e. the computational cost of building maps will depend on the number of objects detected). The reactive behaviours use the same attributes but additionally, they will require a motivation parameter.

The deliberative fusion processes and behaviours are specified based on different function values that relates the number of objects on the environment and the computational requirements of these algorithms. Fusion deliberative processes such as *global map* act as any-time algorithms —the quality of the answer obtained (Maps), depends on the processing time dedicated to their execution. If the environment is overloaded and the time available for executing the *Global_map* process is insufficient, the quality of the map generated will be low and vice-versa. The components required for describing deliberative fusion processes are an identifier, an input blackboard object reference to read data, the algorithm that processes the information, the time provided for the execution of the process and a blackboard output reference to write the process results. Deliberative behaviours (i.e. short-term planning) can generate trajectories of varying qualities (i.e. smoothness, minimum path) depending on the algorithm executed. These behaviours will require, besides the previous components, a set of algorithms and a motivation parameter.

3 REAL-TIME EXECUTION MODEL

The temporal requirements of the robotic application processes are extracted from the behavioural model specification and are analysed in order to decide whether the application timing requirements are guaranteed or not. To allow the representation of the behaviour's temporal requirements, a real-time task model has been defined.

3.1 Task model

Reactive behaviours are mapped to either fixed or variable periodic tasks. The fixed periodic task set, T^f, represents the periodic processes of the behavioural model that don't depend on any application parameter. For example, closed-loop control processes, such as the servomotor process. The temporal parameters of a fixed periodic task T_i^f are shown in equation 1.

$$T_i^f = \{C_i, T_i, D_i, P_i, \mathrm{Im}_i, \phi_i\} \qquad (1)$$

Where, C_i is the worst case execution time, T_i is the period, D_i is the deadline, P_i is the priority, $\mathrm{Im_i}$ is the importance and ϕ_i is the offset of the *i* task.

The variable periodic task set, T^v, supports the execution of behavioural processes that are dependent on the environment and/or on the vehicle speed. For example, the obstacle avoidance behaviour has to, proportionally adapt its execution frequency to the actual vehicle speed. Hence, this process is modelled as a variable periodic task with variable temporal requirements (period and deadline). A variable periodic task, T_i^v, is characterised by the temporal parameters of the equation 2.

$$T_i^v = \{C_i(k_1), T_i(k_2), D_i(k_2), P_i, \mathrm{Im}_i, \phi_i\} \qquad (2)$$

In this case, C_i parameter is variable and depends on the parameter k_1 that can be the number of objects in the environment. T_i and D_i attributes are also variables and depend on the parameter k_2 that can be the robot speed. These variable temporal parameters should accomplish the conditions of the equation 3.

$$\begin{aligned} C_i^{\min} \le C_i(k_1) \le C_i^{\max} \\ T_i^{\min} \le T_i(k_2) \le T_i^{\max} \\ D_i^{\min} \le D_i(k_2) \le D_i^{\max} \end{aligned} \qquad (3)$$

$C_i^{\min}$ is the minimum computational time corresponding to an under-loaded environment. $C_i^{\max}$ is the maximum computational time corresponding to an overloaded environment. $T_i^{\min}$ and $D_i^{\min}$ are the minimum task periods and deadlines, and are applied for the maximum vehicle speed. When the speed is minimum, $T_i^{\max}$ and $D_i^{\max}$ are applied. The relationship between speed and periods is stated in equation 4.

$$T_i = \alpha_i \cdot \frac{1}{v_c} \qquad (4)$$

Where T_i is the *i* task period, α_i is a distance constant and v_c is the current robot speed. The vehicle speed range is [80 *mm/s*, 2 *m/s*].

The deliberative processes are generally characterised by computational times higher than those associated with reactive processes. The response of the former improves the results of the latter. For example, the deliberative *local_map*

process improves the quality of the map obtained by the reactive *local_min_map* process. Depending on the environment load, it is dedicated a corresponding computational time to the *local_map* process. The higher the computational time dedicated to a *local_map* process is, the better are the maps it generates. Deliberative processes are modelled with optional soft aperiodic tasks. Each optional task, T_i^o, is composed of different deepening levels (Liu, 1991; Audsley, 1991). The first level generates a minimum quality response of the task. By increasing the level, the quality of the task result is improved. The characterisation of an optional task, T_i^o, is given in the equation 5.

$$T_i^o = (L_{i,j}, \mathrm{Im_i}) \qquad (5)$$

Where $L_{i,j}$ is the estimated computational time of the j^{th} level of the i^{th} task. Im_i is the importance of the optional task.

3.2 Schedulability analysis

To guarantee the execution of the different real-time tasks, two schedulers have been designed. A critical level scheduler guarantees the execution of the fixed and the variable periodic tasks while an optional server guarantees the execution of the optional tasks without jeopardising the execution of the former load. Before run-time, fixed priorities are assigned to the critical level tasks. The priorities of the mandatory and the action parts of periodic tasks must be the same and are kept fixed during the system execution. To guarantee the execution of the critical tasks, a fixed priority-based pre-emptive scheduler is used and schedulability analysis for such systems is applied (Audsley, 1995). Although variable periodic tasks introduce changes in the run-time workload, this should not exceed the maximum system utilisation that is guaranteed before run-time, and consequently, the variable run-time workload is always guaranteed (Hassan, 2001).
At the optional level, the optional server, by means of the dynamic approximate slack stealing algorithm (Davis, 1993; Hassan, 2001), obtains the available slack time that is invested in the execution of optional tasks. These tasks are executed at the priority of the optional server, which is in turn executed at the priority of the periodic task that previously invoked it. The optional tasks don't interfere in the system schedulability guarantee because they are scheduled in the slack time provided by the optional server.

4 INTEGRATED DEVELOPMENT FRAMEWORK

The implementation of the integrated environment presented in this paper follows the general design references for developing real-time systems established in (Gutierrez 2000, Crespo 1990). These systems cope with the specification, schedulability analysis and automatic *Ada 95* code generation of distributed and centralised real-time systems respectively. The framework proposed here focuses in the design of real-time mobile robotic applications where specific features related to robotic systems have to be considered (i.e. behavioural based modelling, world modelling, timing requirements guarantee, application and execution level interrelation, robot QoS). This framework permits the analysis and validation of both the functional behaviour and the temporal requirements of the system, by the incorporation of simulation tools at both the application and the execution level. Once validated the application design, its corresponding executable code can be generated for the real-time kernel of *linux* operating system, *rt-linux*.

4.1 General overview of the integrated toolset

The general structure of the framework is shown in the Figure 2. The graphical tools have been developed with *Qt* multi-platform C++ GUI toolkit available for Windows, Linux, Solaris, HP, etc. (Ward, 2001). The modules composing the proposed framework are described below:
- Mobile robotic model: serves for the specification of the robotic application based on the behavioural model described in section 2. The *flex* and *bison* tools are used to create a special-purpose language that supports the specification of robotic applications. Alternatively, a graphical interface, to define the process characteristics is provided. An example showing how is performed the specification of an application can be seen in the Figure 4. The description of the world model (i.e. targets, obstacles, robot recharge base) is performed with the graphical tool of the Figure 3.

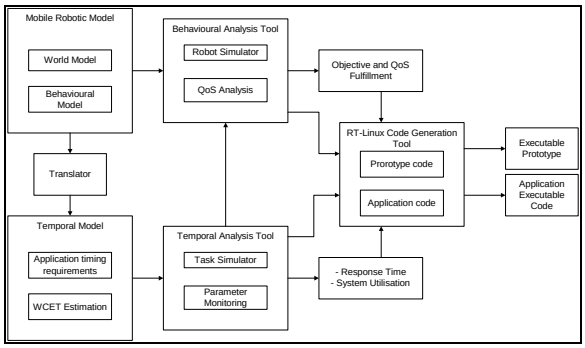

Fig. 2. Framework for developing mobile robotic applications.

- Temporal model: the timing characteristics of the application are obtained from the mobile robotic application definition by means of a temporal translator. The temporal requirements of the processes of the application are those supported by the system model stated in section 3.1. A *rt-linux* specific monitor permits the estimation of the process computational times.
- Behavioural analysis tool: includes a robot simulator that permits the validation of the functional robot behaviour. The simulator takes robot state information (i.e. robot speed, trajectory) from the blackboard that is updated by the real-time tasks during the execution of the system and simulates the behaviour of the robot. The QoS analyser monitors

the robot performances (i.e. trajectory smoothness) and can know whether the performance requirements are fulfilled or not.
- Temporal analysis tool: permits the analysis of the temporal requirements of the tasks, based on the analytical techniques presented in section 3.2. Different scheduling algorithms and optional task control strategies can be applied during the evaluation. A task simulator tool allows the visualisation of the execution of the task sets and the analysis of statistic indexes of the system execution. Critical and optional load, system speed, variable utilisation and workload can be monitored in real-time.
- The *rt-linux* code generation tool is in phase of construction and it will permit the creation of the actual executable code of the application, according to the application specification and to the temporal requirements of its processes.

5 APPLICATION DESIGN STUDY

In order to show the design phases taken for developing mobile robotic applications, a simulation case study based on the *Yair* robot (Benet 1998) is analysed. The specification of applications is performed with the tool shown in the Figure 3.

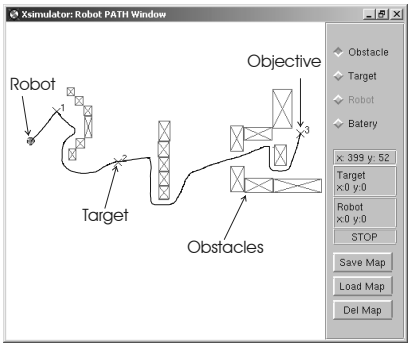

Fig. 3. Robotic simulation tool.

This tool allows the definition of the obstacles of the environment, the initial robot location, the different targets of the robot and the battery recharge points. In the application example, the robot starts from its initial position and has to reach the objectives, X_i. Along the path, the environment becomes progressively overloaded. To adequately recognise the world and take correctly its decisions, the robot regulates its speed (i.e. if the number of objects increases, the robot speed is reduced). By reducing the robot speed, the system utilisation will decrease and the spare *CPU* capacity enabled can be used for long-term planning and map fusion processing. This simulation tool permits to assess whether the robot fulfils its functional requirements or not.

Five reactive processes (Odometry, Local_min_map, Go_objective, Obstacle_avoidance, Trajectory arbiter) and two deliberative processes (Local_map and Global_map) compose the application. All of them are defined with the specification tool of the Figure 4. As an example, the attributes of the reactive *obstacle_avoidance* behaviour can be appreciated.

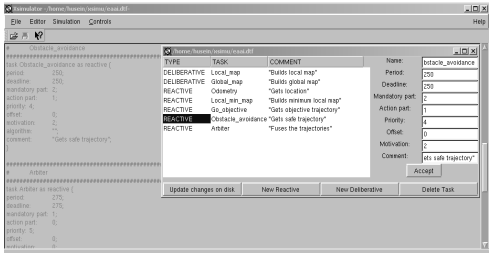

Fig. 4. Processes specification tool.

The temporal characteristics of these processes are translated to the real-time task model in order to proceed to the analysis of the temporal requirements of the application processes. The generated task sets are executed based on the scheduling algorithms selected using the tool shown in the Figure 5. The Rate Monotonic or Deadline Monotonic algorithm (Audsley, 1995) can schedule critical tasks while different heuristic strategies (boss strategy, balance strategy, etc.) schedule optional tasks (Hassan, 2001).

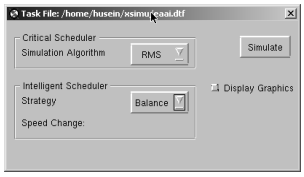

Fig. 5. Scheduler selection interface.

The task temporal requirements are analysed with the set of tools of the Figure 6.

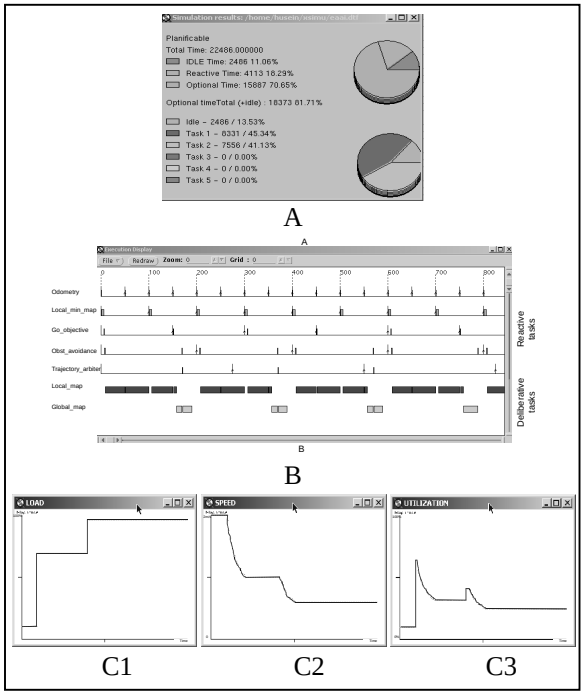

Fig. 6. Temporal analysis tools.

The Figure 6 (A) shows that the application task set is schedulable. It presents also, the computational time used by the reactive, deliberative and the idle task. The Figure 6 (B) represents a time chronogram of the different tasks cooperating to satisfy the robot objectives. This tool permits a detailed analysis of the task behaviour along the time, such as, computational time variations and temporal requirement changes.

The interfaces that allow the real-time monitoring of the workload, the system speed and the utilisation during the application execution can be appreciated in the Figure 6 (C). During the execution of the application example, the robot sonar detects an increase of the number of obstacles in two zones. Approaching *X1*, it detects few obstacles and near *X2*, it recognises a big number of obstacles. Thanks to the model of system proposed, the exact percentage of computational time consumed by the tasks can be monitored in real-time. This information is plotted in the Figure 6 (C1). It can be seen that the first increase of the workload correspond to the *X1* environment. There, the robot reacts by reducing its speed that will cause a reduction of the system speed (2m/s to 1m/s) as can be seen in the Figure 6 (C2). This situation will produce a reduction of the utilisation as is monitored by the Figure 6 (C3). In the vicinity of the *X2* point, the same effect is reproduced. The robot reduces its speed (1m/s to 0.6m/s), the system speed decreases and the utilisation is reduced. The information of these system state variables permits to take actions in order to improve the robot QoS. For example, when the system is underloaded (low utilisation), such as in *X2* zone, more computational time can be dedicated for processing long-term planning and map building tasks to improve the maps and trajectory qualities.

Finally, after checking the accomplishment of the functional behaviour of the application with the robotic simulation tool and assuring the guarantee of the temporal characteristics of the processes with the temporal analysis tools, the generation of the *rt-linux* application code can be launched.

6 CONCLUSIONS AND FUTURE WORK

An integrated framework for the specification, analysis and validation of real-time mobile robotic applications is presented. The main features of the environment is that it allows the integrated analysis and validation of both the functional and the temporal behaviour of the system. The specification of robotic applications has been performed based on behavioural models and the validation of the robotic functional behaviour has been achieved based on simulation tools.

The real-time system for modelling the timing requirements of the robotic application processes and the schedulability analysis techniques applied have been detailed. At this level, different graphical tools have been implemented for monitoring and analysing the execution of the system load.

A case study showing the phases taken in the design of *Yair* robotic applications has been described.

Future work will focus on the completion of the framework by implementing the real-time linux code generation tool. The analysis of the robot QoS (i.e. trajectory quality) is intended. The embedding of the generated code in the *Yair* robot platform and the applications evaluation are also planned.

REFERENCES

Audsley, N.C., Burns, A., Richardson M.F. and Wellings, A.J. (1991). Incorporating Unbounded Algorithms into Predictable Real-Time Systems. *Report Nº RTRG/91/102.* Real-Time Systems Group. Department of Computer Science. University of York, UK.

Audsley, N.C., Burns, A., Davis, R., Tindell, K. and Wellings J. (1995). Fixed Priority Pre-emptive Scheduling: An Historical Perspective. *The Journal of Real-Time Systems*, ***Vol. 8***. pp. 173-198.

Beccari, G., Caselli, S., Reggiani, M. and Zanicheli, F. (1999). Rate Modulation of Soft Real-time Tasks in Autonomous Robot Control Systems. *11th Euromicro Conference on Real-Time Systems*, York. UK.

Benet, G. Blanes, F. Martínez, M. And Simó, J. (1998). A Multisensor Robot Distributed Architecture. *9th IFAC Symposium on Information Control in Manufacturing.* Metz-Nancy.France.

Caccamo, M., Buttazzo, G. and Sha L. (2000). Elastic Feedback Control. *12th Euromicro Conference on Real-Time Systems.* Stockholm, Sweden.

Capucho, J., Almeida, L. and Buttazzo G. (2001). Using a Real-Time Kernel to Simulate the Micro-Rato Robotics Contest. *Proceedings of ROBOTICA 2001*. Guimarães, Portugal.

Crespo, A., De la Puente, J.A., Espinosa A. and Garcia A. (1990). Quisap: Environment for Rapid Prototyping of Real-Time Systems. *IEEE Conference on Software Engineering*. Tel-Aviv. Pp. 502-508.

Davis, R.I., Tindell, K.W. and Burns A. (1993). Scheduling Slack Time in Fixed Priority Preemptive Systems. *IEEE Real-Time System Symposium*. Dec 1993. pp 160-172.

Donnelly, C. and Stallman, R. (1995) *Bison: The Yacc-Compatible Parser Generator*. Free Software Foundation, Cambridge, Massachusetts.

Gat, E. (1998). Three Layer Architectures. Artificial Intelligence and Mobile Robots. *The MIT Press*. 292 Main Street, Cambridge, MA.

Gutierrez, JJ. and Gonzalez, M. (2000). A Framework for Developing Distributed Hard Real-Time Applications. *25th IFAC Workshop on Real-Time Programming*. Palma de Mallorca. Spain.

Hassan, H. (2001) Improving the Flexibility and the QoS of Real-time Control Systems. *Ph. D. Thesis*. Dept. of Computer Engineering. Polytechnical University of Valencia. Spain.

Kortenkamp, D. (2000) Designing visualization tools for a Distributed Control Architecture. *Intelligent Autonomous Systems Conference*. Venice, Italy.

Liu, J.W.S., Lin, K.J.L., Shih, W.K., Yu, A.C., Chung, J.Y., and Zhao, W. (1991) Algorithms for Scheduling Imprecise Computations. *IEEE Computer,* **Vol. 24, No. 5**, pp. 58-68.

Nilsson, U., Streiffert S. and Törne A. (1998) Detailed Design of Avionics Control Software. *IEEE Real-Time System Symposium*. Madrid. Spain.

Shuhua W. (2000). Verification of both functional and timing requirements of real-time systems. *25th IFAC Workshop on Real-time Programming*. Palma de Mallorca. Spain.

Stankovic, J. A., Lu, C., Son, S. H. and Tao G. (1999). The Case for Feedback Control Real-Time Scheduling. *11th EuroMicro Conference on Real-Time Systems*. York. U.K.

Storch, M.F. (1997). A Framework for the Simulation of Complex Real-Time Systems. *PhD Thesis.* Dept. Computer Science, Univ. of Illinois.

Ward, P. (2001). *Qt Programming for Linux & Windows 2000*. Prentice Hall PTR. Upper Saddle River, New Jersey.