Web-Based Activities Around a Digital Model Railroad Platform

Pedro Sánchez, Bárbara Alvarez, Andrés Iborra, José María Fernández-Meroño, and Juan A. Pastor

***Abstract*—This paper describes a laboratory equipped for the teaching of advanced courses in computer engineering, computer science, information systems, and software engineering. Other related work areas include computer vision, real-time systems, programming languages, and computer architectures. The laboratory has been built around a digital model railroad platform controlled by a client–server system using an object-oriented language. The characteristics of this laboratory are suitable for implementing Web activities for educational purposes. The paper also includes an overview of the system in which most of these topics have been considered and a summary of the relationship with the most relevant international curricula in computing.**

***Index Terms*—Computer science, curriculum issues, Internet, laboratory, object-oriented programming.**

I. Introduction

A TRANSFORMATION is taking place in society, powered by increasing use of the Internet and the World Wide Web in all aspects of information systems. Web-based systems and applications now offer a huge array of content and functionality to end users of all kinds, and this trend will continue. New engineering disciplines are dealing with the process of developing Web-based systems [1]. This type of system is especially suited to learning in computer engineering. However, advanced aspects, such as distribution, concurrency, timing constraints, etc., also need to be considered. The Web can be particularly useful for educational purposes in the control of physical real-time systems, where the above ideas can be put into practice. Although the delays inherent in the Web make real-time responses almost impossible to accomplish, the use of an intranet in the laboratory produces good results. That accomplishment is the purpose of the system described here for monitoring and controlling a digital railroad over the Web, using Java programming language. The basic purpose of the system is to control several concurrent digital locomotives, using a Java client running in a Web browser. This environment allows the user to control each locomotive and the turnouts on the layout. The laboratory has been designed to support several kinds of advanced courses by controlling real-time systems over the Web within a distributed architecture.

This laboratory is an ideal environment in which several programming facets can be integrated. Although many undergraduate courses in computer engineering acquaint students with the fundamental topics in real-time computing, many do not provide adequate laboratory platforms to exercise the software skills necessary to build physical real-time systems. This paper presents the types of practice that can be implemented using the proposed laboratory. Section II summarizes a number of studies in the field of multipurpose laboratories and Web-based computing. Section III gives a complete description of the railroad platform. Section IV details some platform aspects related to this paper. Section V presents an architecture, a basic set of requirements that can be attached to it, and a summary of activities to be carried out. Section VI reviews the subjects that can be taught at the laboratory, based on curricular recommendations of the ACM [2] and the IEEE-CS [3]. The conclusions summarize the benefits of this paper and outline future plans.

II. Related Studies

Telecomputing is an important educational field, as demonstrated in [4], where a teleteaching environment is given: a tool provides a remote laboratory in which theoretical aspects can be put into practice by means of several Java applications. Another highly innovative educational trend is the implementation of multipurpose laboratories. Reference [5] explains how students can carry on telecommunications, computing, and software engineering activities all on the same computing platform. This paper integrates both ideas: the teleoperation of the system plus the integration of several engineering activities: *mechatronics*, in a general sense. This term refers to the combining of software, mechanisms, and electronic circuits. Mechatronics is a new engineering discipline with some peculiar nonfunctional properties (product complexity, safety, high usability, etc.). Other related research in this field includes [6], which describes a client–server system to control robots over the World Wide Web. Reference [7] describes some virtual reality tools for Internet robotics to perform robot teleoperation via the Internet. An analogous project can be found in [8], which introduces a laboratory platform on which students can construct software (using the Ada language) to control true physical systems. Another relevant project is one carried out at the State University of New York (SUNY), Plattsburgh [9], where it was realized that the computer-controlled model railroad provided a strong incentive for the students to learn concurrent programming concepts in a short time.

III. Railroad System

The proposed railroad system has been developed around a commercial digital control system by Märklin.[1] In our railroad

This work was supported in part by the Spanish Ministry of Education (with reference ACF2000-0037-IN).

The authors are with the División de Sistemas e Ingeniería Electrónica (DSIE), Universidad Politécnica de Cartagena, E-30202, Spain (e-mail: pedro.sanchez@upct.es; barbara.alvarez@upct.es; andres.iborra@upct.es; josem.fernandez@upct.es; juanangel.pastor@upct.es).

[1]http://www.marklin.com.

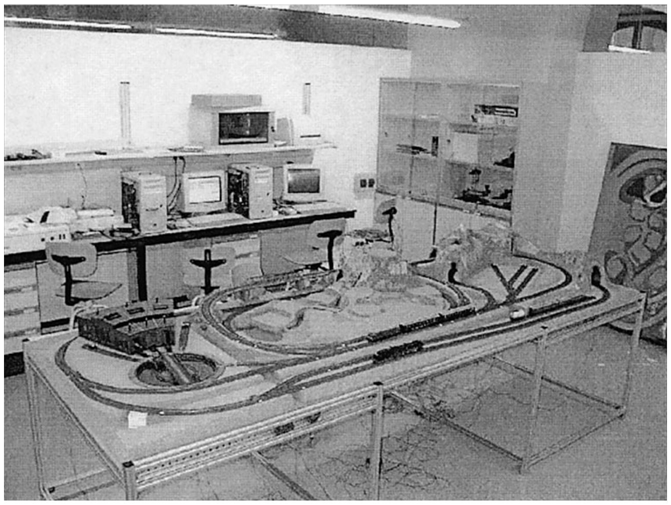

Fig. 1. The laboratory.

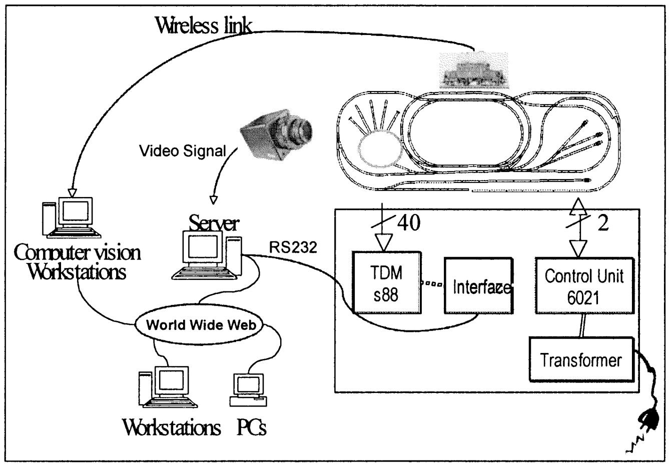

Fig. 2. Laboratory layout.

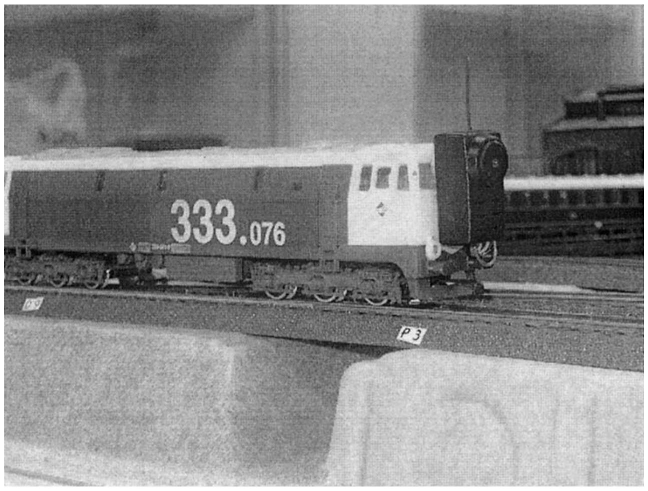

Fig. 3. Micro wireless camera mounted on a train.

layout there are five digital locomotives, sixteen digital turnout switches (where two or more tracks are joined), six semaphores, and twenty-one reed contact sensors to manage and control. The reed contacts are placed before and after turnouts around the track in order to monitor the passing of locomotives. The total length of the track plus the turnouts results in a complex circuit in which several levels of difficulty (and risks) can be simulated. Because of the digital characteristics of the Märklin locomotives and tracks, all the information needed to control the railroad is run through the tracks. Several Märklin modules are used to connect (using an RS232 serial interface) the track to the computer (the server). Several video cameras are connected to the system. One of them shows users a video stream of the actual physical train system and provides the opportunity to carry out visual supervision of the locomotives. The other cameras are placed inside the locomotives for computer vision tasks (such as stopping before a semaphore, emergency stop before an obstacle, adjustment of train speed to the traffic conditions, etc.).

The Web clients, developed in Java, allow users to stop, reverse, and change the speed of each locomotive by means of its unique address. Also, users are able to switch any of the sixteen connected turnouts on the layout. Each of these elements responds to computer commands sent to their addresses. With this platform, users can manually control all the trains from any remote Java client. The server receives commands from clients and has to decide on the feasibility of each petition in order to avoid a system crash. When one train comes too close to another train, the server decides what to do in this situation. Although the reed contact sensors do not give addressable track detection information, it is possible to monitor the trains with little processing. Fig. 1 shows a panoramic view of the laboratory.

IV. Laboratory Equipment

The laboratory (see Fig. 1) consists of the above digital model railroad, the server (a Sun UltraSparc 10 workstation), and twelve computers connected to the Web (eight Intel Pentium computers and four Unix workstations). One of the cameras is used to view the surroundings and is connected to the server. The others are mounted on the locomotives with the purpose of obtaining image sequences of the railroad track. A commercial micro-wireless camera was used with four eligible transmitting channels (2413–2470 GHz.). The size of the camera is $1.2 \times 0.7 \times 0.6$ inches. As mentioned above, the serial connection is between the MDCS (Märklin Digital Control System) and the server. The server contains all tasks required to manage the railroad. Clients are connected to the server by means of the World Wide Web. Two of the workstations interact with the server in a distributed computing framework also using the Web. Fig. 2 shows a sketch of the platform and Fig. 3 shows a detail of a locomotive where a micro-wireless camera has been mounted. The 6021 Control Unit is the core of the Märklin Digital system. This module receives the commands from the Interface Module and converts them to electric signals that are transmitted through the rails using only two wires. All the turnouts, semaphores, and locomotives carry a device to decode control commands. In this way, the number of wires is minimized to two. Each reed contact consists of two wires that indicate that some locomotive has traveled over it. Twenty reed contacts are installed to monitor the traffic on the railroad. The Märklin Track Detection Module (TDM) includes three Märklin S88 decoders. Each one provides a reading of the status of the eight reed contacts.

V. A Web-Based Laboratory

Following are some details of the design and implementation of the laboratory to give a clear idea of its possibilities:

- Distributed computing was used because every workstation simulates a real railway station with booking and querying operations. In this connection the Java RMI (Remote Method Invocation) utilities were used to allow the writing of efficient and fault-tolerant distributed applications with very little effort. Although only the server has the knowledge to monitor and control the railroad, the other workstations cooperate in the execution of the global Web application.
- Client–Server connections are implemented using Java Swing to provide a platform-independent graphical user interface. Users can then query and modify the status of the system (by querying information or by booking). A Java Database Connectivity (JDBC) interface was selected to manage the distribution of data. One of the workstations acts as an alternative server in the event the main server crashes.
- The server dedicates one concurrent task (composed of other intraconcurrent tasks) to the control of each locomotive. The computer vision workstation is used for binarized image processing, line extraction, and locomotive speed control. The image processing is performed through specific hardware for computer vision applications (Matrox Genesis board), which provide many useful capabilities for simultaneous online preprocessing of up to four video channels. Using machine-vision techniques, such as horizontal and vertical gradient, threshold and Hough Transform [10], it is possible to detect the rail tracks from the video image sequences and to determine their curvature. The speed of each locomotive is adjusted to that curvature, and the computer vision workstation sends commands to the server via the World Wide Web.

With the basic infrastructure described above, a highly comprehensive set of programming practices were developed around the Web environment. The authors have compiled a small set of requirements for this system. Some of them are briefly described below, simply to give an overall idea of the laboratory's possibilities. The system must meet at least the following set of functional requirements:

- Information available to the customer at each railway station (electronic blackboard with data of trains, etc.) or accessible via the World Wide Web;
- Booking and selling of tickets: at each train station or from travel agencies;
- Information about each train: position, direction and speed, overflow in tracks, starvation and deadlock of trains, semaphore state, turnout status, etc.;
- Route and train control: manage train motion, failure detection, deadlock detection, adaptive train control depending on exceptions (net overflow, weather conditions, etc.), trains in the same route, control of semaphores, control of security, etc.

A set of quality constraints applies when implementing control of the digital model railroad:

- Safety: train stop caused by passengers, driver, or system signals, etc.;
- Security: of the information system, of the control communications, reliability (failure detection and recovery), overflow control (dynamic load distribution), etc.;
- Modification: modify the hardware for signaling and control (sensors and actuators), new tracks, routes, or communication protocols (radio, etc.), etc.;
- Portability: new hardware and software platforms, reuse of current software;
- Interoperability: the new system must sometimes coexist with legacy systems, booking services between different communication nodes and operating systems, etc.;
- Performance: hard real-time constraints imposed by deadlines (alarms, signal interaction, emergency stopping, etc.); soft real-time constraints imposed by throughput, worst case (in information retrieval and booking services).

A. *Web-Based Laboratory Activities*

The aforementioned functional and quality requirements were considered in defining a set of feasible practices to be followed in this laboratory. Some information about the architecture is included to provide an overview of the modeled system. Three external systems deal with the relationship between the railroad system and its environment: the Märklin Interface module, a reduced set of industrial robots, and the users who interact with the railroad via the Web. The total system is broken down into differentiated subsystems which address all the desired functionalities in a distributed configuration. Following are the software engineering requirements:

- Web-based distributed computation with Java RMI. The authors opted for distribution, although there is only one server in this system, because the other workstations provide separate parts of the functionality (i.e., the booking system apart from the control system, etc.). With this middleware technology, Java objects can be distributed so that they communicate with one another on a client–server basis.
- Web-based concurrent programming with Java threads. Each locomotive is treated as a separate task that interacts with the other system tasks. Particular objectives in this respect are starvation and deadlock detection.
- Web-based remote database access using Java JDBC. Each railway station has remote access to a centralized database that contains booking and travel information.
- Real-time programming using Ada language to comply with the hard and soft real-time constraints of the system. Ada tasks must interact cooperatively with Java threads running in the server machine.
- Friendly Web-based GUIs using Java Swing. The same environment must be portable to different display devices.

UML standard notation [11], [12] was chosen for all activities, following the methodology proposed by Gomma [13] for the design of concurrent, distributed, and real time systems. Some ideas from [14] were also incorporated. The media introduced here also serve as a framework for cross-disciplinary study of *mechatronics* systems by means of activities related to this discipline [15]. These activities include machine, electronic and software design, the integration of design tools (Grasp, Autocad, 3-D Studio for robotics modeling, and VHDL for hardware simulation and synthesis), simulation of Global Positioning System (GPS) by means of a zenithal camera using vision techniques, and so on. Four educational Fischer Techniques Robots have been integrated in the railroad platform (in particular, mobile, anthropomorphic, and pneumatic robots). This option is now being carefully studied to be put into practice.

VI. Curriculum Recommendations

Both the Association for Computing Machinery (ACM) and the Computer Society of the Institute of Electrical and Electronics Engineers (IEEE-CS) have addressed the changing needs of professional and student curriculum recommendations in several areas over the last three decades. In particular, the ACM published the document "Guidelines for Associate Degree Programs to Support Computing in a Networked Environment" two years ago [2]. This document is intended to help graduates perform competently in careers that support computing in a networked environment. The report identifies nineteen content areas that two-year colleges should include in their curricula to prepare students for this type of position. Each area has an outcome goal and a list of topics. After analyzing the spectrum of activities to be carried out in the proposed laboratory, the authors identified six content areas (with their respective significant topics) of particular interest (see Table I).

At the end of 1998, the IEEE-CS and the ACM decided to review and improve the existing curricula in computing technologies [3]. Four computing fields were considered (Computer Science, Computer Engineering, Software Engineering, and Information Systems). Although the work is still in progress, some drafts have now been published. In particular, there is a final draft for Computer Science undergraduates. The body of knowledge covered by these curricula falls into fifteen areas. Table II shows the areas and topics (units) relating most closely to the purpose of this laboratory.

The laboratory has been used in courses such as *Concurrent Programming* and *Real-Time Programming*. These courses are placed in the last year of the *Industrial Electronics and Control Engineer* graduate curriculum, and they cover a good percent of the previous curriculum recommendations. Students have learned the use of the Java language for concurrent programming and the Ada language for real-time programming. The course Real-Time Programming had seven lectures which concentrated on the following topics:

- Characteristics of real-time systems and introduction to the Ada 95 language;
- Scheduling schemes (cyclic executive and priority-based models);
- Reliability, fault tolerance and low-level programming.

The laboratory experiments consisted of the following:

- Programming in Ada 95 (information hiding, separate compilation, exception handling, reuseability);
- Tasking in Ada 95 (concurrent execution, synchronization, rendezvous);

TABLE I
Correspondences With the ACM Curriculum for Computing in a Networked Environment

Content Areas	Topics
Surfing the Internet	• Protocols • Using the Internet • Application Installation and configuration
Web Authoring	• Page Design • Authoring tools • Accessibility issues • Page development
Web Multimedia	• Graphics • Sound and video • Virtual reality
Web Interactivity	• Security
Web Site Creation	• Setup • Management tools • Security
Internet Servers	• Server set up • Firewalls

TABLE II
Correspondences With ACM/IEEE-CS Computing Curricula

Content Areas	Units
Programming Fundamentals	• Event-driven programming
Operating Systems	• Concurrency • Real-time embedded systems
Real-time embedded systems	• Communications and networking • Network security • The web as an example of client/server computing • Building web applications • Wireless mobile computing
Programming languages	• Object-oriented programming
Human-Computer Interaction	• Graphical UI design
Information management	• Distributed databases
Software Engineering	• Using APIs • Software tools and environments • Software reliability

- Programming a priority-based system (rate monotonic priority assignment, deadline monotonic priority assignment, immediate ceiling priority inheritance).

The course Concurrent Programming had five lectures with the following topics:

- Introduction to concurrent programming: the mutual exclusion problem;
- Tools for managing the concurrency (critical sections, semaphores, monitors, synchronization aspects, etc.) and concurrent programming languages.

The laboratory experiments consisted of the following:

- Basic implementations of concurrent programming concepts using the Java language;
- Implementation in Java (using the Java Development Kit 1.4.) of a subset of the previous functional and quality requirements. The graphical user interface was developed using the Swing Java API. Distribution was implemented using the Java RMI facility.

Students implemented some mini-applications related to detection train collisions, and similar. They could put in practice conceptual terms, such as shared resources, mutual exclusion, task synchronization, and so on. The use of the railroad platform increased the student motivation and made possible the teamwork.

VII. Summary and Future Work

The platform described here requires students to utilize and exercise their knowledge of concurrent programming, real-time constraints, computer vision, GUI applications, client–server programming, and so on. The control software accomplishes the purpose of maintaining control of several digital locomotives while at the same time allowing users around the world to control the operations of the trains manually, using a Java client running in a Web browser. Obviously, the benefits of the laboratory do not end there. Mechatronics are applicable in any context involving electronics and robotics. Moreover, the robot and railroad platform can be teleoperated via the Internet from anywhere in the world by way of web pages. At the same time, remote users can view the current status of the real environment through the visual monitoring system, based on a webcam that shows the movements of each robot and locomotive and their behavior depending on the command sent (received) from the remote web page. This kind of laboratory has particular advantages for education, in that it addresses a significant percentage of the subjects contained in international computing curricula. In this respect, such a cross-disciplinary approach is desirable; however, it does demand more effort from students and teachers in terms of preparation and application of practical classes.

The DSIE research group at the Technical University of Cartagena has recently started up a number of activities addressing the objectives stated above. Some are theses or final study projects; others are teaching tasks aimed at last graduate courses or pre-doctoral courses for students of mechatronics engineering. An important decision involves the type of users for which the laboratory has been conceived. This physical laboratory is not for beginner students because an advanced level is mandatory and is not manageable with more than six or eight students at the same time. In the authors' opinion, the combining of physical environments with entertainment activities produces both increases in student motivation and good learning results.

References

[1] A. Ginige and S. Murugesan, "The essence of Web engineering: Managing the diversity and complexity of Web application development," *IEEE Multimedia*, vol. 8, pp. 22–25, Apr. 2001.

[2] ACM. (2000, May) ACM guidelines for associate-degree programs to support computing in a networked environment. [Online]. Available: http://www.acm.org/education/curricula.html (date last accessed July 2002).

[3] ACM and IEEE-CS. (2001, Dec.) Year 2001 model curricula for computing (Computing Curricula 2001). The joint IEEE computer society/ACM task force. [Online]. Available: http://www.acm.org/education/curricula.html (date last accessed July 2002).

[4] M. Llamas, L. Anido, and M. J. Fernández, "Simulators over the network," *IEEE Trans. Education*, vol. 44, no. 2, pp. 212–212, May 2001.

[5] V. E. DeBrunner, L. S. DeBrunner, S. Radhakrishnan, and A. Kamal Khan, "The telecomputing laboratory: A multipurpose laboratory," *IEEE Trans. Education*, vol. 44, pp. 302–310, Nov. 2001.

[6] A. Malinoski and B. Wilamowski, "Controlling robots via Internet," in *First Int. Conf. Inform. Technol. Mechatronics*, Istanbul, Turkey, Oct. 2001, pp. 101–107.

[7] R. Chellali, "Web based tools for Internet robotics," in *Proc. 1st Int. Conf. Robotics Automation*, Seoul, Korea, June 2001.

[8] M. B. Feldman, "Ada experience in the undergraduate curriculum," *Commun. ACM*, vol. 35, no. 11, pp. 53–67, Nov. 1992.

[9] J. W. McCormick, "A model railroad for Ada and software engineering," *Commun. ACM*, vol. 35, no. 11, pp. 68–70, Nov. 1992.

[10] E. R. Davies, *Machine Vision: Theory, Algorithms and Practicalities*. London, U.K.: Academic, 1997.

[11] OMG. (2001, Sept.) UML reference manual v1.4. Object management group. [Online]. Available: http://www.omg.org/technology/documents/formal/uml.htm (date last accessed July 2002).

[12] J. Rumbaugh, I. Jacobson, and G. Booch, *The Unified Modeling Language: Reference Manual*. Reading, MA: Addison-Wesley, 1999.

[13] H. Gomaa, *Designing Concurrent Distributed and Real-Time Applications With UML*. Reading, MA: Addison-Wesley, 2000.

[14] B. P. Douglas, *Real-Time UML. Developing Efficient Objects for Embedded Systems*, 2nd ed. Reading, MA: Addison-Wesley, 1999.

[15] A. Iborra, B. Alvarez, C. Fernández, P. Sánchez, J. Suardíaz, and J. M. Fernández, "Mechatronics: A new engineering discipline," in *First IEEE Int. Conf. Inform. Technol. Mechatronics*, Istanbul, Turkey, Oct. 2001, pp. 209–232.

Pedro Sánchez received the Ph.D. degree in computer science from the Technical University of Valencia, Spain, in 2000.

He joined the Technical University of Valencia in 1993 where he was a teacher of software engineering. Since 1996, was a member of several research projects that were focused on software engineering. In 2001, he joined the Systems and Electronic Engineering Division (DSIE), Technical University of Cartagena (UPCT), where he is currently an Associate Professor of computer science. His current research interests include conceptual modeling, specification languages, software architectures and object-oriented programming.

Bárbara Alvarez received the Ph.D. degree in telecommunication engineering from the Technical University of Madrid, Spain, in 1997.

Since 1995, she has participated in different projects focused in robotics applications for the industry. In 1998, she joined the Systems and Electronic Engineering Division (DSIE), Technical University of Cartagena (UPCT). She is currently an Associate Professor of computer science with the UPCT. Her current research interests include real-time systems and software architectures for teleoperation and computer vision systems.

Andrés Iborra received the M.S. degree in industrial engineering in 1989 and the Ph.D. degree in 1993 from the Technical University of Madrid, Spain.

He is the head of the Electronic Technology Department, Systems and Electronic Engineering Division (DSIE), Technical University of Cartagena, Spain, where he has been since 1988. He has worked as a research engineer in the field of robotics for nuclear power plants with ENWESA (joint venture between Westinghouse Electric Corporation and Equipos Nucleares S.A.) during 1993–1994. In this company, he was R&D Director since 1994, participating as Project Leader in different works for the industry, mostly focusing on mechatronics and robotics applications for hazardous environments. His current research interests include mechatronic systems design and analysis, computer vision, robotics, and engineering education.

José María Fernández-Meroño received the Ph.D. degree in engineering from the Technical University of Madrid, Spain, in 1989.

From 1970 to 1999, he was a research engineer working in the field of automatic control and electronic engineering at the Murcia University. Currently, he is Head of the Systems and Electronic Engineering Division (DSIE), Technical University of Cartagena, Spain. His current research interests include mechatronic systems design and analysis, computer vision, robotics, and engineering education.

Juan A. Pastor received the M.S. degree in telecomunication engineering from the Technical University of Madrid, Spain, in 1995, and the Ph.D. degree in software engineering from the Technical University of Cartagena (UPCT), Spain, in 2002.

Since 1995, he has worked as a research engineer in the field of robotics for nuclear power plants with ENWESA (joint venture between Westinghouse Electric Corporation and Equipos Nucleares S.A.) In 2001, he joined the Systems and Electronic Engineering Division (DSIE), UPCT, where he is an Assistant Professor of software engineering. His current research includes software architectures, notations, and frameworks for the development of teleoperation and computer vision systems.