The 3rd Portuguese Conference on Automatic Control
CONTROLO’98, Coimbra, Portugal, 9-11 September 1998

LOCAL ORGANIZING COMMITTEE

A. Dourado (Chairman) (Dep. Computer Eng.)
F. Silva Leite (Dep. Mathematics)
T. Pedroso de Lima (Faculty of Economics)
F. Cardoso (Dep. Physics)
J. Dias (Dep. Electrical Eng.)
N. Oliveira (Dep. Chemical Eng.)
B. Ribeiro (Dep. Computer Eng.)
J. Afonso (Dep. Mechanical Eng.)

NATIONAL ORGANIZING COMMITTEE

Carvalhal, F. (LNEC, Lisbon)
Couto, C. (Univ. Minho, Braga, P)
Duarte Cunha, A. (INETI, Lisboa, P)
Duarte-Ramos, H. (Univ. Nova Lisboa, P)
Felipe de Souza (Univ. Beira Interior, P)
Feyo de Azevedo, S. (FEUP, Porto, P)
Lemos, J.M. (IST, Lisboa, P)
Mano, A. (EDP, Lisboa, P)
Martins de Carvalho, J. (FEUP, Porto, P)
Mota, A. (Univ. Aveiro, P)
Pinho, M.R. (FEUP, Porto, P)
Rocha, P. (Univ. Aveiro, P)
Ruano, A. (Univ. Algarve, P)
Sá da Costa, J. (IST, Lisboa, P)
Sentieiro, J. (IST, Lisboa, P)

PROGRAM COMMITTEE

Albertos, P. (Univ. Polytechnica Valencia, E)
Araújo, H. (Univ. Coimbra, P)
Biegler, L.T. (Carnegie Mellon Univ., USA)
Bergholm, F. (Royal Inst. Tech., Stockholm, S)
Camacho, E. (Univ. Sevilla, E)
Cardoso, A. B. (Univ. Coimbra, P)
Carvalhal, F. (LNEC, Lisboa, P)
Castro, J. A. (Univ. Coimbra, P)
Couto, C. (Univ. Minho, P)
Crouch, P. (Arizona State Univ., USA)
Daoutidis, P. (Univ. Minnesota, USA)
Duarte, B. (Celulose do Caima, P)
Duarte Cunha, A. (INETI, Lisboa, P)
Duarte-Ramos, H. (Univ. Nova Lisboa, P)
Felipe de Souza, A. (Univ. Beira Interior, P)
Feyo de Azevedo, S. (Univ. Porto, P)
Fleming, P. (Univ. Sheffield, UK)
Frank, P. (Univ. Duisburg, D)
GeromeL J. C. (Univ. Campinas, BR)
Gilgic, V. (Univ. Beira Interior, P)
Häggglund, T. (Tech. Univ. Lund, S)
Hernández, V. (Univ. Polytechnic Valencia, E)
Leivškā, K. (Univ. Oulu, SF)
Lemos, J. M. (IST, Lisboa, P)
Lobo Pereira, F. (Univ. Porto, P)
Magalhães dos Santos, D. (Univ. Aveiro, P)
Mano, A. (EDP, Lisboa, P)
Martins de Carvalho, J. (FEUP, Porto, P)
Morgado, R. (EFACEC, P)
Mosca, E. (Univ. Florence, I)
Nijmeijer H. (Univ. Twente, NL)
Pascoal, A. (IST, Lisboa, P)
Pearson, D. (EERIE, Nimes, F)
Ribeiro, M.I. (IST, Lisboa, P)
Rocha, V. (TECNOTRON, Lisboa, P)
Ruano, A. (Univ. Algarve, P)
Sá da Costa, J. (IST, Lisboa, P)
Sarychev, A. (Univ. Aveiro, P)
Sentieiro, J. (IST, Lisboa, P)
Titli, A. (LAAS, Toulouse, F)
Traça de Almeida, A. (Univ. Coimbra, P)
Vitória, J. (Univ. Coimbra, P)
SPONSORS

Caixa Geral de Depósitos
Reitoria da Universidade de Coimbra
Fundação Oriente
Fundação da Ciência e Tecnologia
Fundação Calouste Gulbenkian
Fundação Luso-Americana
Embaixada de França
Mota e Teixeira, S.A.
Siemens Portugal, S.A.
Proelectrónica, Lda
Fisher-Rosemount, Lda
Lusaico - Indústria Cerâmica, Lda

SUPPORTERS

Departamento de Matemática
Faculdade de Ciências e Tecnologia da Universidade de Coimbra
Região de Turismo do Centro
Câmara Municipal de Coimbra
EXPO’98
The 3rd Portuguese Conference on Automatic Control
CONTROLO'98, Coimbra, Portugal, 9-11 September 1998

LIST OF REFEREES

Afonso, J.
Albertos, P.
Almeida, R.
Araújo, H.
Barbosa, M.
Biegler, L. T.
Bergholm, F.
Camacho, E.
Cardoso, A. B.
Cardoso, A.
Carvalhal, F.
Couto, C.
Crouch, P.
Daoutidis, P.
Dias, J.
Dourado, A.
Duarte, B.
Duarte-Ramos, H.
Ferreira, E.
Feyo de Azevedo, S.

Fleming, P.
Frank, P.
Geromel, J. C.
Gligic, V.
Hägglund, T.
Hernández, V.
Leite, F.
Leivitská, K.
Lemos, J. M.
Lima, P.
Lobo Pereira, F.
Magalhães dos Santos, D.
Mano, A.
Martins, A. G.
Milheiro, P.
Moreira, P.
Morgado, R.
Mosca, E.
Nijmeijer H.
Nunes, U.

Oliveira, P.
Pascoal, A.
Pearson, D.
Pimenta, P.
Pinto Ferreira, C.
Pires, N.
Ramos, J. I.
Ribeiro, M.I.
Rocha, P.
Ruano, A.
Sá da Costa, J.
Sarychev, A.
Tenreiro Machado, J.
Trave-Massuyes, L.
Traça de Almeida, A.
Vitória, J.

SPECIAL SESSION ON ROBOTICS AND AUTOMATION

ORGANIZING COMMITTEE

J. Norberto Pires, University of Coimbra (Portugal).
Jorge Dias, University of Coimbra (Portugal).
Pedro Lima, Technical University of Lisbon - IST (Portugal).
Carlos Couto, University of Minho (Portugal).
Klas Nilsson, Technical University of Lund (Sweden).
Lourdes Peñalver Herrero, University of Valencia (Spain).
Angel Valera, University of Valencia (Spain).
F. Lobo Pereira, University of Porto (Portugal)

LOCAL REPRESENTATIVES

Abílio Azenha, U. Porto (P)
Antonio Visioli, U. Brescia (IT)
Akio Gofuku, Okayama University (JP)
Fabrizio Caccavale, U. Naples (IT)
Fernando Matia, U. P. Madrid (SP)
Johan Eker, U. Lund (SE)
Juan Carlos, U. Asturias (SP)

Klas Nilsson, U. Lund (SE)
Luis Batista, IST (P)
Patrick Mattis, UK Robotics (UK)
Ramon Costa, U.P. Catalunya (SP)
Rui Aratijo, ISR Coimbra (P)
Sandor Kopacsi, MTA-SZTAKI (HU)
CLIENT/SERVER SOFTWARE ARCHITECTURE FOR A ROBOTIC SYSTEM*

Pedro Gil, Francisco Martinho Lopes, João Pereira
Instituto de Engenharia de Sistemas e de Computadores, R. Alves Redol, 9-6E, sala 620, Apartado 13069,
1000 Lisboa, Portugal
Email: {Pedro.Gil,Francisco.Martinho,jap}@inesc.pt

Abstract: The RESOLV project requires a core for its software architecture, providing several services to the client modules involved in a 3D reconstruction process supported by an autonomous mobile robot. This vehicle must be remotely operated via a human-computer interface, being the WWW the chosen media for its implementation. This paper describes a server architecture developed for the project, but usable by any control system with several clients for different tasks. A fully integrated multipurpose web-based interface is also introduced.

Keywords: Software Architecture, Client/Server Architecture, Web-Based Control, Distributed Applications, Mobile Robots.

1. INTRODUCTION

The architecture herewith described is currently implemented in the first prototypes of the RESOLV project, two mobile robotic platforms: one autonomous (AEST) and the other manually operated (EST). The project aims at the development of a system to enable the acquisition and reconstruction of realistic 3D models of large and complex indoor environments, based on range and video data.

The acquisition head includes a Laser Range Finder, a Video Camera and a Pan & Tilt unit. The head is mounted on top of the two mobile platforms, as mentioned before. The reconstruction of an interior environment is achieved by placing the acquisition head in different positions, and acquiring the range and video data. Each position provides a separate model of the world, as seen from the respective point. Each new model is then combined with the previous ones in order to generate a larger and more complete model of the scene.

This paper describes the software architecture for RESOLV, with special incidence in the implemented server as a scalable, multipurpose adaptable system for any client-based control architecture. A web-based interface was also developed, targeting a close integration with the server and its objectives.

2. SOFTWARE ARCHITECTURE

The software architecture developed for RESOLV is the result of nearly two years of work by all partners of the project. The initial architecture [SEQUEIRA96] has been refined and redefined to the one presented here.

The software architecture of the project is peculiar in terms of its organization. Although the functionality is thoroughly distributed in different executable clients with well-defined tasks, there was the need to centralize all the shared information in a server. This server should also provide multiple functions to support its clients needs, as well as the overall control of the process and interactions.

In Figure 1 it is possible to perceive three distinct areas, each corresponding to a different processor.

* This work was supported by the RESOLV-Reconstruction using Scanned Laser and Video project of the ACTS Programme, EU.
At the top, running on a Notebook, the human-computer interface for RESOLV provides a World-Wide Web based interface to monitor and control the acquisition process.

In the middle, running on the Host PC, it is easy to identify the different software modules developed for RESOLV. They are placed around a central entity, the Host Server, which constitutes the core of the software package developed for the project and is described in the next section. It is the connecting link between all the other processes, algorithms and applications that compose the RESOLV software, allowing them to converge into one application.

The several client modules are executed once per iteration in a predefined order. A scan session, during which an environment is reconstructed, is composed of several of these iterations.

The 3D Acquisition, 3D Single Reconstruction, and 3D Composite Reconstruction modules are responsible for, respectively, acquiring the range data, reconstructing one model, and composing two existing models [SEQUEIRA96a] [SEQUEIRA96b]. The Video Acquisition module captures the video images needed for later texture mapping each of the 3D models by the Texture Mapping module. The PT Controller and Tower Control manage the acquisition head’s pan and tilt unit and the electric tower vertical movement. Registration module determines the rigid 3D transform, based on the range image data from different acquisition points. The Perception Planning module determines the next acquisition point based on information retrieved from the previously reconstructed model, and determines if the model is good enough, to stop acquiring. The Path Planning module calculates the best path from the current position to the one given by the Perception Planning. The Navigation Manager monitors the actual navigation process (platform movement) [SANTOS98]. The Mobile Robot Communications Interface (MRCI) module establishes the link between the Host Server and the autonomous processes running on the mobile platform computer. The Localisation module provides the system with the platform location in the world being acquired, [MOTA98].

At the bottom of Figure 1 the Mobile Robot Computer is depicted. This processor runs, over the Albatros operating system, other software modules that control the autonomous locomotion of the mobile platform [SANTOS98]. It also provides proper actions to the readings measured by hardware sensors (e.g. ultra sound sensors) placed at the bottom of the Autonomous EST.

3. HOST SERVER

The Host Server (HS) has two major functions. Firstly, it provides every module with a communications framework which allows them to exchange messages between each other. Secondly, using the created framework, it is responsible for synchronizing and coordinating all the client modules, allowing them to converge into an application that reconstructs the environment surrounding the acquisition platform. It is the HS that starts and terminates all modules, whilst defining the order in which they start and finish their jobs.

As well as these two major functions, the Host Server is also responsible for the manipulation of files before, during and after, a scan session, and for storing the sets of data needed for a scan, it being configuration data, or run time variable data.

The Host Server’s internal architecture was designed in order to group and structure all the above functions, as depicted in Figure 2.

The architecture is composed of five modules. The Kernel is the most important. The others are the Task Manager, the File Manager, the Data Manager and the System Config Manager.

A close observation of Figure 2 will show the same structure in all the modules composing the Host Server. This structure, Client Module, common to the HS modules as well as all the client modules, incorporates in every one of them the ability to communicate.
The Client Module structure is itself composed of two other structures, the Socket Manager and the Client Interface. The first occupies itself of the communications framework. The second provides every module with a simple graphical user interface, especially useful for debug purposes.

This library's prime characteristics are modularity and ease of integration. Designed from scratch with modularity in mind, the Client Module provides any software module with the ability to communicate through a socket interface (the Socket Manager), and with a simple dialog based user interface (Client Interface). The two possible ways of integration give this library the flexibility necessary for the integration with the different types of software modules developed (Figure 3).

The need for two ways of integration arose from existence of two different types of client modules. Some were native C++ Win32 applications that could be easily integrated with the C++ class hierarchy implemented by the Client Module library (Figure 3a), others were plain C code, constituting a separate application, that had no C++ classes at all and could not be integrated so easily (Figure 3b).

While the first type of integration allows the full benefit of the library, the second one limits the use of the Client Module library to the use of the Socket Manager (SM) substructure, only providing the communications framework. Not that the Client Interface (CI) cannot be used, but since in the second case the actual module is a separate application, it can not use the run time debug facility that the CI provides.

Kernel

Being the core of the Host Server the Kernel is also the core of the whole system. Using the Client Module library as the base of its internal structure, the Kernel is constituted by three entities that can be identified in Figure 2. They are the Server Manager, the Operations Manager and the Query Manager, and allow the logical grouping of Kernel functions.

The Server Manager is responsible for managing the other four different modules that compose the Host Server. It launches them and is responsible for their termination, while monitoring its state during run time. These modules are the Task Manager, the File Manager, the Data Manager and the System Config Manager. It is also the Server Manager that deals with the routing of all the messages exchange between different modules.

The Operations Manager is responsible for defining and managing the several working modes of the system, and dealing with emergency situations that affect more than one module. The full set of working modes as well as its implications and the dealing with emergency situations are yet unspecified and unimplemented.

The Query Manager is responsible for managing the distinct queries, needed at certain times, to ensure the correctness of certain groups of data. As an example of such critical data is the Odometry data gathered from the sensors, crucial for the automatic control of platform motion.

Task Manager

The Task Manager is the Host Server entity that establishes the order according to which all the client modules are activated, for any operation of the platform. Despite being designed to implement the sequence for a scan session, its modularity allows new operations to be defined, such as video surveillance or just a simple movement between two points.

The sequence of client activation, for a scan session, is presented in Figure 4.
define a set of default values and configuration parameters, log files, to track down the operation of the whole system, or maybe even data files, containing an x-ray of the data structures available in the Host Server at a given instant (for instance, before an emergency shut-down).

It is also responsible for the validation of file names, such as the Session Name, used to create a directory to store the data generated in each session. The creation of a directory structure for each session, as well as the placement of the selected output files in the WWW publishing directory are also among the responsibilities of the File Manager.

**Data Manager**

The Data Manager implements a data structure used to keep all run-time variable data relevant to a session. It is one of the passive modules, since it doesn’t take any initiative by itself, limiting its intervention to the response of data requests coming from the other modules. These requests will tell the Data Manager to store a given value, or to return any previously stored value.

**System Config Manager**

As the Data Manager, the System Config Manager (SCM) is also a passive module. Its function is to store the default values and configuration parameters necessary to start a scan session. The data it stores comes mostly from the reading of the configuration file through the File Manager. Whenever requested, the SCM returns such data. This data, once read from the configuration file is read only during the instance of the application.

4. HUMAN-COMPUTER INTERFACE

The HCI is a user interface, which allows the remote control, over the Internet, of an autonomous mobile platform. It acts as an indispensable part of the task distributed architecture implemented in the RESOLV project, connecting directly with the Host Server. Though it is physically located in the host PC, within its Web Server structure, the HCI can be accessed through any computer respecting the requirements for its operation. This interface was specifically designed to accommodate easy additions or changes to its functionality, thus managing to be easily adapted to any other type of web-based control application.

The Human-Computer Interface for RESOLV is structured to have only one active instance. This assumption assures that only one user can access the
interface at a given time, as most of its functions issue direct imperative commands to the robot. Another interface, based in the same design methods, will allow other users on the Internet to receive the output data of the current activities of the platform. The main asset to be accounted for this interface is, apart from its scalability and flexibility, the “feel and look” of a regular application program, although it is run from a regular web browser and through the World Wide Web.

- **Engineering Console** – provides an extensive and extendable panel with functions to be used for expert use of the platform, such as teleoperation, ultrasound readings, platform localisation, parameter settings, etc…
- **Last Iteration** – transmits the command relative to the last iteration of the current acquisition;
- **Stop Robot** – forces an emergency robot stop;
- **Close Application** – closes the application.

![Figure 5 - The HCI](image)

The HCI works by interacting with the user, assembling his requests and sending them over to the Host Server (HS), where they will be processed. Then, its actions will include the display of the results or requested data in a clear and understandable fashion. Some functions that may not require some kind of intervention from the HS are processed within the HCI’s internal architecture. The communication to and from the HS has been developed using asynchronous sockets, thus enabling the messages to be processed through the Socket Manager, just as if the HCI was a regular client.

The HCI’s design was based on the task-centred approach recommended by Clayton Lewis [LEWIS94]. This kind of task orientation resulted in a design centred mainly in a set of primary commands:

- **New Model** – begins a new acquisition session, in order to create a new VRML model;
- **Open Model** – opens a previously saved VRML model;
- **Save Model** – saves the current VRML model;
- **View Model** – enlarges the current VRML model onto the workpad;
- **Extend Model** – enhances the model resolution in specific user-defined areas by processing a new reconstruction in the selected region;
- **Engineering Console** – provides an extensive and extendable panel with functions to be used for expert use of the platform, such as teleoperation, ultrasound readings, platform localisation, parameter settings, etc…
- **Last Iteration** – transmits the command relative to the last iteration of the current acquisition;
- **Stop Robot** – forces an emergency robot stop;
- **Close Application** – closes the application.

![Figure 6 - The functional areas of the HCI](image)

Although the commands have set the perspective of the HCI’s use, the prime directive for its development was to build it in a way to simplify any changes that may become necessary. Minor changes, such as the inclusion of a new command, must be previewed through a flexible design, able to reflect them in any of the elements that compose the interface. To uphold this objective, a modular structure was defined, through the definition of functional areas within the HCI (Figure 6).

All the commands were positioned in the main interactive area of the interface, the toolbar, in which each button represents a direct order to the application. The workpad is the result of the necessity to create an area with multiple functions and, thus, maximum flexibility. It can display diversified contents, such as the engineering console or the current VRML model. The preview area provides the dynamic contents of the HCI, relatively to what is going on with the acquisition. Two windows, one with a snapshot of the environment being captured, and another with the model as it is at a particular moment. These windows also allow the visualisation of a previously saved VRML model. A message box upholds the necessary feedback to users’ requests. Movement in the RESOLV logo implies that the application is processing information, while a static picture corresponds to an idle condition.
In terms of technical approach to the problem, various solutions were adopted. Options taken to ensure optimal results include the use of the Netscape Frame Specification, ISAPI (Internet Server Application Programming Interface) and JavaScript.

The relationship between the HCI and the RESOLV architecture is depicted in Figure 7.

5. CONCLUSIONS AND FUTURE WORK

The architecture presented in this article, as well as all its constituent modules is currently implemented in the software supporting the RESOLV project. Its principal characteristics are modularity and ease of integration. System robustness is also a prime feature, given the tests that have been made so far.

Despite being developed and implemented in the RESOLV context, this architecture can be adjusted to any control project, comprising more than one software module. The Client Module structure, if integrated in all the software modules, provides them with a communications framework through the Host Server whose supporting modules (the Task Manager, File Manager, System Config Manager and Data Manager) can and should be moulded to the characteristics of every specific application.

One of the aspects to be improved in the near future is to provide the Host Server with the ability to communicate with remote client modules, allowing real distribution of the processing load. Also, the way new client activation sequences (systems tasks) are defined and configured will be improved. Regarding the HCI, the next objective is to increase its responsiveness and refresh rate.

REFERENCES


[LOPES96] Francisco Lopes, Pedro Gil, João Pereira: Host Communications Server – Architecture Proposal; Setembro 1996. RESOLV Documentation


