12

Distributed Architecture for Intelligent Robotic Assembly Part I: Design and Multimodal Learning

Ismael Lopez-Juarez and Reyes Rios-Cabrera

1. Introduction

1.1 General Description

In this chapter we describe the general framework design for a distributed architecture to integrate multiple sensorial capabilities within an intelligent manufacturing system for assembly. We will formally define the model of the M2ARTMAP multimodal architecture. We present some simulated results of the model using a public domain multimodal data base. Initial findings have indicated the suitability to employ the M2ARTMAP model in intelligent systems when three or less modalities are involved. Taking into account these results and the M2ARTMAP's modularity it was decided to integrate this model using the CORBA (Component Object Request Broker Architecture) middleware to develop robotic assembly tasks that includes contact force sensing, an invariant object recognition system and natural language processing. This chapter introduces the overall system in the intelligent cell and the task planner (SIEM) and the object recognition system (SIRIO) are described in detail in part II and Part III.

Design and experiments of the distributed architecture using a Local Area network (LAN) and an industrial robot KUKA KR-15 to fusion different modalities in a common task are first described. The modalities are: vision, contact force sensing (tactile), and natural language processing using context free grammars. The vision modality and force sensing are implemented based on a FuzzyARTMAP neural network and a main coordinator. The testbed for the application and a general distributed environment using CORBA as a middleware is described. Later several learning simulations using M2ARTMAP, are presented (Lopez-Juarez & Ordaz-Hernandez, 2005). The main distributed objects are described in detail, the experimentation is presented and the results analyzed. Finally, future work is proposed.

Source: Manufacturing the Future, Concepts - Technologies - Visions , ISBN 3-86611-198-3, pp. 908, ARS/plV, Germany, July 2006, Edited by: Kordic, V.; Lazinica, A. & Merdan, M.

1.2 The architecture

Robots in unstructured environments have to be adaptable to carry out operation within manufacturing systems. Robots have to deal with its environment, using available sensors and their adaptability will depend on how flexible they are (Wu et al., 1999). To create that system it is necessary to integrate different techniques of artificial intelligence, on-line learning, sensorial capabilities and distributed systems.
This work stands on and improves the design of intelligent agents for assembly (Lopez-Juarez et al., 2005a) by integrating the fusion of different modalities using a distributed system based on CORBA. In the chapter it is designed a distributed architecture where different sensorial modalities, operating systems, middleware and programming languages are integrated to perform mechanical assembly by robots.
A task coordinator referred as the SICT (*Sistema Inteligente Coordinador de Tareas, in Spanish*), which plans the assembly and general coordination with the vision and system and the assembly system was designed. The SICT is described, it was built using several operating systems (Linux and windows), two ORB's (Object Request Broker) that is ORBit and omniORB and the programming languages are C and C++. The communication model includes the schema client-server in each module of the system.

1.3 Description of the Manufacturing System

The manufacturing system used for experimentation is integrated by a KUKA KR15/2 industrial robot. It also comprises a visual servo system with a ceiling mounted camera as shown in figure 1.

The main operation of the manufacturing system is about the peg-in-hole insertion where, there robot grasps a male component from a conveyor belt and performs the assembly task on a working table where the fixed female component is located. A main coordinator starts the assembly cycle using the vision system that obtains an image from the male component and calculates the object's pose estimation, later it sends information to the coordinator from two defined zones:

Zone 1 which is located on the conveyor belt. The vision system searches for the male component and determines the pose information needed by the robot.

Zone 2 is located on the working table. Once the vision system locates the female component, it sends the information to the coordinator which executes the assembly with the available information.

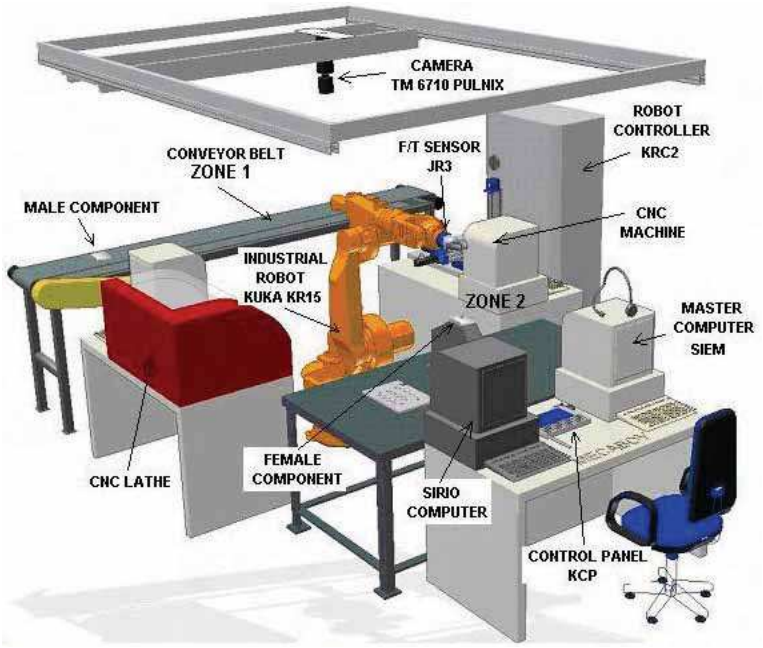

Figure 1. Manufacturing System

The NNC (Neural Network Controller) for assembly is called SIEM (*Sistema Inteligente de Ensamble Mecánico, in Spanish*) and is based on a FuzzyARTMAP neural network working in fast learning mode (Carpenter et al., 1992). The vision system, called SIRIO (*Sistema Inteligente de Reconocimiento Invariante de Objetos*), also uses the same neural network to learn and classify the assembly components. The SIRIO was implemented with a high speed camera CCD/B&W, PULNIX 6710, with 640x480 resolution. The camera movement on the X-Y plane was implemented using a 2D positioning system.

2. State of the Art

2.1 Distributed Robotic Systems

The concept of distributed systems and other technologies recently have made possible the creation of new application called "Networked Robot Systems". The reduction in cost is one of the several advantages of creating distributed systems. This is mainly for the use and creation of standard components and infrastructures (Amoretti, 2004); in addition the CORBA standard solves the heterogeneity problem which is found in the robotic systems in a network. It permits the interaction and interoperability of different systems developed with different technologies, programming languages, operation systems or hardware.

Currently, the development of robot systems base on distributed components is being developed by different researchers. In (Amoretti et al., 2003), Michael Amoretti et al., present an analysis of three techniques for data distributing of sensors through the network. The first technique is called *Callback*, where the clients call a method of a server and at the same time send an object reference to inform the server to which client has to send the answer and information. When the server finishes the asked task, it checks the object number to which it has to send the results. The second technique is based on the services or events of CORBA. Where the servers produce events and the clients receive them using an *event channel*. The event channel conducts the events of the servers to the clients, without having information about the clients and vice versa. In spite of the advantages of the event channel, when it is used, it generates negative aspects, such as the data type, since it has to be of type "any" or IDL, and it makes the communication not very secure. Clients have to convert the data

to their respective type and another problem of the event channel is that it could saturate the network, since it does not have an event filter and sends all messages to all clients. The event services does not contemplate the use of QoS (Quality of Service), related with the priority, liability and order.
The third technique is based on the Notification Service of CORBA. It is an improvement of the Service of Events. The most important improvement includes the use of QoS. In the notification service each client uses the events in which it is interested.

The implementation of the Callback technique offers a better performance than the others; however the ones based on the event channel are easily scalable. The technique used in our research is Callback since the number of clients, is not bigger of 50.
In (Amoretti, 2004) it is proposed a robotic system using CORBA as communication architecture and it is determined several new classes of telerobotic applications, such as virtual laboratories, remote maintenance, etc. which leads to the distributed computation and the increase of new developments like teleoperation of robots. They used a distributed architecture supporting a large number of clients, written in C++ and using CORBA TAO as middleware, but it is an open architecture, and it does not have intelligence, just remote execution of simple tasks.
In (Bottazzi et al., 2002), it is described a software development of a distributed robotic system, using CORBA as middleware. The system permits the development of Client-Server application with multi thread supporting concurrent actions. The system is implemented in a laboratory using a manipulator robot and two cameras, commanded by several users. It was developed in C++ and using TAO.
In (Dalton et al., 2002), several middleware are analyzed, CORMA, RMI (Remote Method Invocation) and MOM (Message Oriented Middleware). But they created their own protocol based on MOM for controlling a robot using Internet.

In (Jia et al., 2002), (Jia et al., 2003) it is proposed a distributed robotic system for telecare using CORBA as communication architecture. They implemented three servers written in C++, the first one controls a mobile robot, the second one is used to control an industrial robot and the last one to send real time video to the clients. On the other side of the communication, it is used a client based on Web technology using Java Applets to make easier the use of the system in Internet. In (Jia et al., 2003), the authors increased the number of servers

available in the system, with: a user administrator and a server for global positioning on the working area.
In (Corona-Castuera & Lopez-Juarez, 2004) it is discussed how industrial robots are limited in terms of a general language programming that allows learning and knowledge acquisition, which is probably, one of the reasons for their reduced use in the industry. The inclusion of sensorial capabilities for autonomous operation, learning and skill acquisition is recognized. The authors present an analysis of different models of Artificial Neuronal Networks (ANN) to determine their suitability for robotic assembly operations. The FuzzyART-MAP ANN presented a very fast response and incremental learning to be implemented in the robotic assembly system. The vision system requires robustness and higher speed in the image processing since it has to perceive and detect images as fast as or even faster than the human vision system. This requirement has prompted some research to develop systems similar to the morphology of the biological system of the human being, and some examples of those systems, can be found in (Peña-Cabrera & Lopez-Juarez, 2006), (Peña-Cabrera et al., 2005), where they describe a methodology for recognising objects based on the Fuzzy ARTMAP neural network.

2.2 Multimodal Neural Network

A common problem in working in multimodality for robots systems is the employment of data fusion or sensor fusion techniques (Martens, S. & Gaudiano, P., 1998 and Thorpe, J. & Mc Eliece, R., 2002). Multimodal pattern recognition is presented in (Yang, S. & Chang, K.C., 1998) using Multi-Layer Perceptron (MLP). The ART family is considered to be an adequate option, due to its superior performance found over other neural network architectures (Carpenter, G.A. et al., 1992). The adaptive resonance theory has provided ARTMAP-FTR (Carpenter, G.A. & Streilein, W.W, 1998), MART (Fernandez-Delgado, M & Barro Amereiro, S, 1998), and Fusion ARTMAP (Asfour, et al., 1993) —among others— to solve problems involving inputs from multiple channels. Nowadays, G.A. Carpenter has continued extending ART family to be employed in information fusion and data mining among other applications (Parsons, O. & Carpenter, G.A, 2003).

The Mechatronics and Intelligent Manufacturing Systems Research Group (MIMSRG) at CIATEQ performs applied research in intelligent robotics, concretely in the implementation of machine learning algorithms applied to as-

sembly tasks —using distributed systems contact forces and invariant object recognition. The group has obtained adequate results in both sensorial modalities (tactile and visual) in conjunction with voice recognition, and continues working in their integration within an intelligent manufacturing cell. In order to integrate other sensorial modalities into the assembly robotic system, an ART-Based multimodal neural architecture was created.

3. Design of the Distributed System

3.1 CORBA specification and terminology

The CORBA specification (Henning, 2002), (OMG, 2000) is developed by the OMG (Object Management Group), where it is specified a set of flexible abstractions and specific necessary services to give a solution to a problem associated to a distributed environment. The independence of CORBA for the programming language, the operating system and the network protocols, makes it suitable for the development of new application and for its integration into distributed systems already developed.

It is necessary to understand the CORBA terminology, which is listed below:

A **CORBA object**	is a virtual entity, found by an ORB (Object Request Bro ker, which is an ID string for each server) and it accepts petitions from the clients.
A **destine object**	in the context of a CORBA petition, it is the CORBA ob ject to which the petition is made.
A **client**	is an entity which makes a petition to a CORBA object.
A **server**	is an application in which one or more CORBA objects run.
A **petition**	is an operation invocation to a CORBA object, made by a client.
An **object reference**	is a program used for identification, localization and di rection assignment of a CORBA object.
A **server**	is an identity of the programming language that imple ments one or more CORBA objects.

The petitions are showed in the figure 2: it is created by the client, goes through the ORB and arrives to the server application.

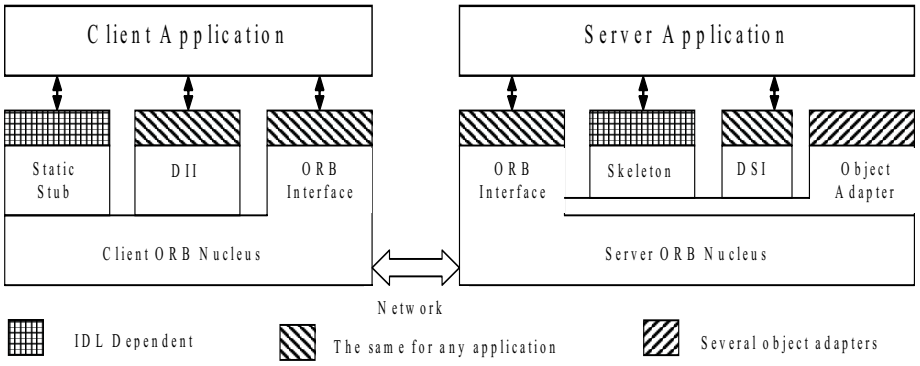

Figure 2. Common Object Request Broker Architecture (COBRA)

- The client makes the petitions using static stub or using DII (Dynamic Invocation Interface). In any case the client sends its petitions to the ORB nucleus linked with its processes.
- The ORB of the client transmits its petitions to the ORB linked with a server application.
- The ORB of the server redirect the petition to the object adapter just created, to the final object.
- The object adapter directs its petition to the server which is implemented in the final object. Both the client and the sever, can use static skeletons or the DSI (Dynamic Skeleton Interface)
- The server sends the answer to the client application.

In order to make a petition and to get an answer, it is necessary to have the next CORBA components:
Interface Definition Language (**IDL**): It defines the interfaces among the programs and is independent of the programming language.

Language Mapping: it specifies how to translate the IDL to the different pro gramming languages.

Object Adapter: it is an object that makes transparent calling to other ob jects.

Protocol Inter-ORB: it is an architecture used for the interoperability among different ORBs.

The characteristics of the petitions invocation are: transparency in localization, transparency of the server, language independence, implementation, architecture, operating system, protocol and transport protocol. (Henning, 2002).

3.1 Architecture and Tools

The aim of having a coordinator, is to generate a high level central task controller which uses its available senses (vision and tactile) to make decisions, acquiring the data on real time and distributing the tasks for the assembly task operation.

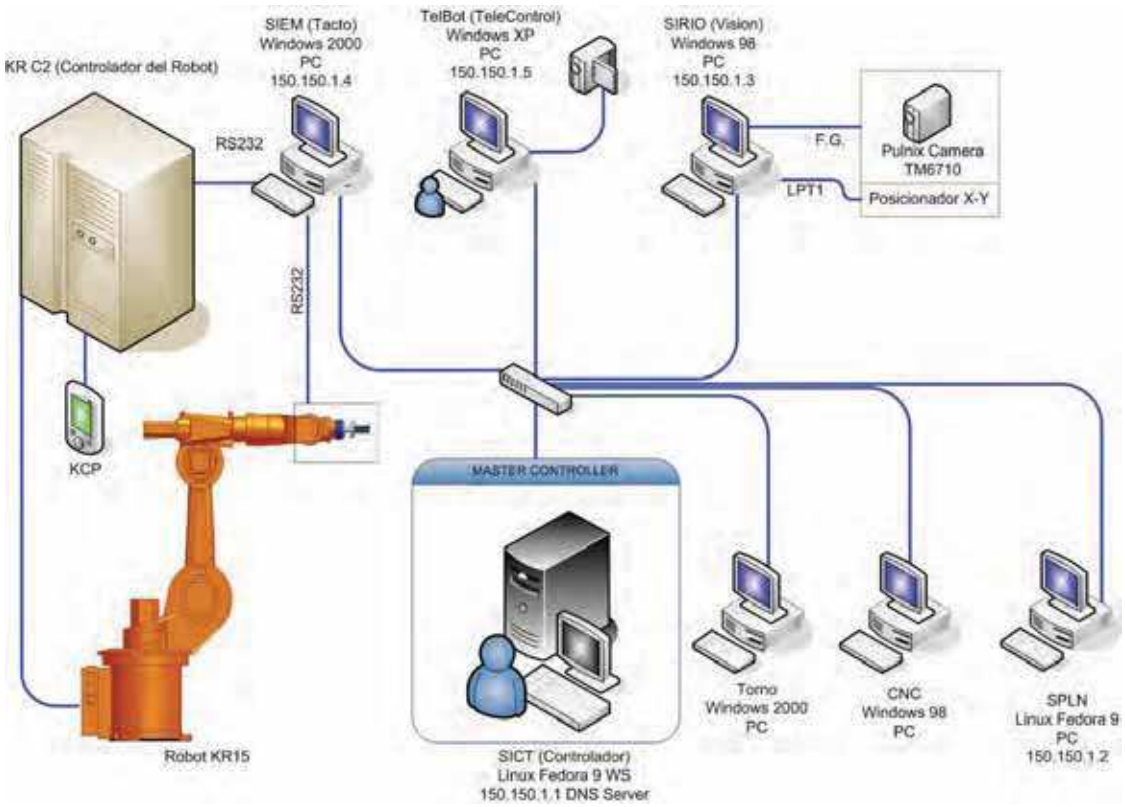

Figure 3. Distributed Manufacturing Cell

Figure 3 shows the configuration of the network and the main components of the distributed cell, however, the active ones are: SIRIO, SIEM, SICT and SPLN. The system works using a multiple technology architecture where different operating systems, middleware, programming language and graphics tools were used, as it can be seen in figure 4. It describes the main modules of the manufacturing cell SIEM, SIRIO, SICT and SPLN.

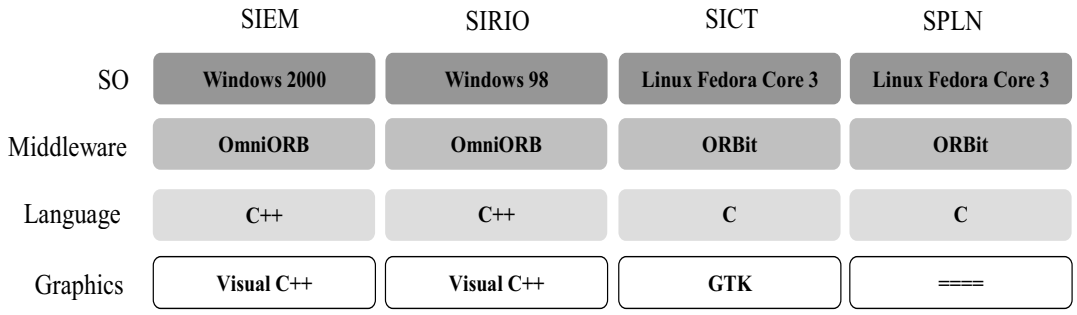

Figure 4. Different operating systems, middleware, programming languages and graphic tools.

The architecture of the distributed system uses a Client/Server in each module. Figure 5 shows the relationship client-server in SICT for each module. But with the current configuration, it is possible a relationship from any server to any client, since they share the same network. It is only necessary to know the name of the server and obtain the IOR (Interoperable Object Reference).

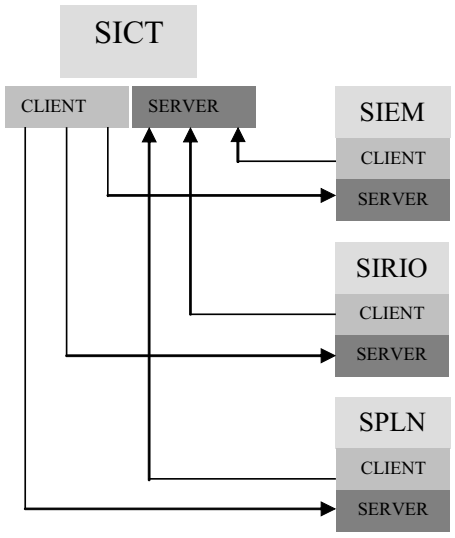

Figure 5. Client/server architecture of the distributed cell

The interfaces or IDL components needed to establish the relations among the modules SICT, SIRIO, SIEM and SPLN are described in the following section.

4. Servers Description

4.1 SICT Interface

This module coordinates the execution of task in the servers (this is the main coordinator). It is base in Linux Fedora Core 3, in a Dell Workstation and written in C language using gcc and ORBit 2.0. For the user interaction of these modules it was made a graphic interface using GTK libraries.

The figure 6 shows the most important functions of the IDL.

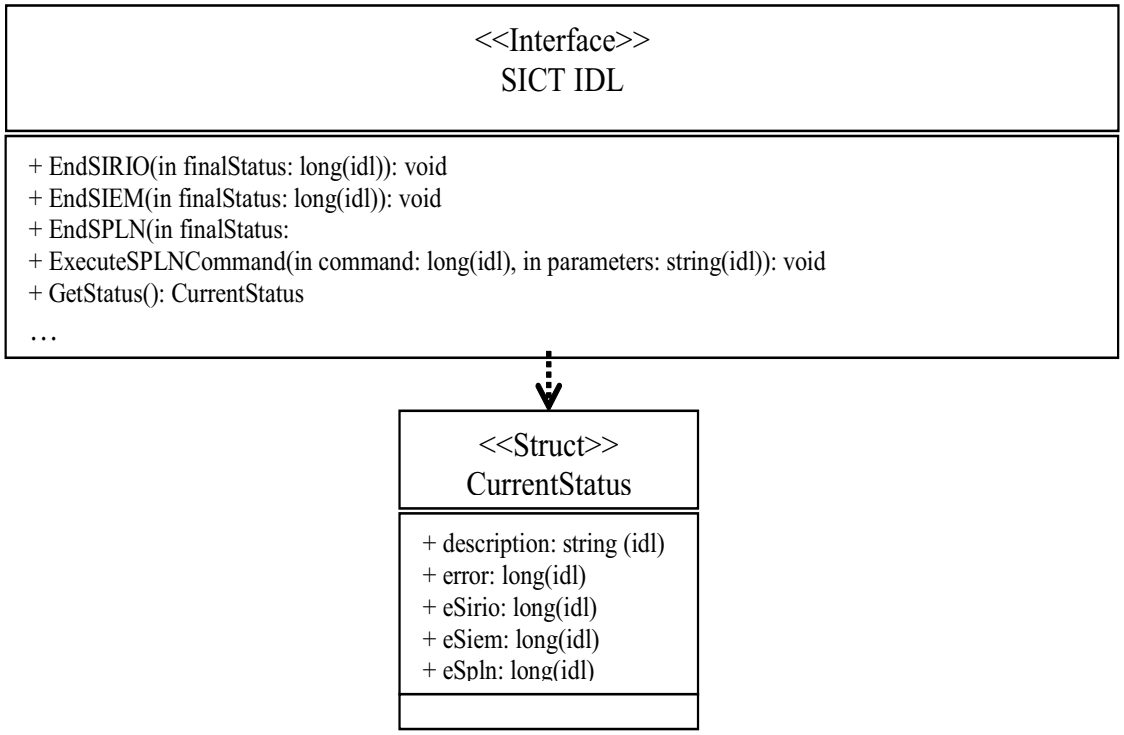

Figure 6. SICT Interface

iSICT: the functions of this interface are used for SIRIO and SIEM to indicate that they have finished a process. Each system sends to SICT a finished process acknowledgement of and the data that they obtain. SICT makes the decisions about the general process. The module SPLN uses one of the functions of SICT to ask it to do a task, sending the execution command with parameters. The figure 7 shows the main screens of the coordinator.

Figure 7. Controls of the interface SICT

4.2 SIRIO Interface

This system is the vision sense of the robot, using a camera Pulnix TM6710, which can move around the cell processing the images in real time. SIRIO carries out a process based on different marks. It calculates different parameters of the working pieces, such as orientation, shape of the piece, etc. This system uses Windows 98 and is written in Visual C++ 6.0 with OmniORB as middleware.

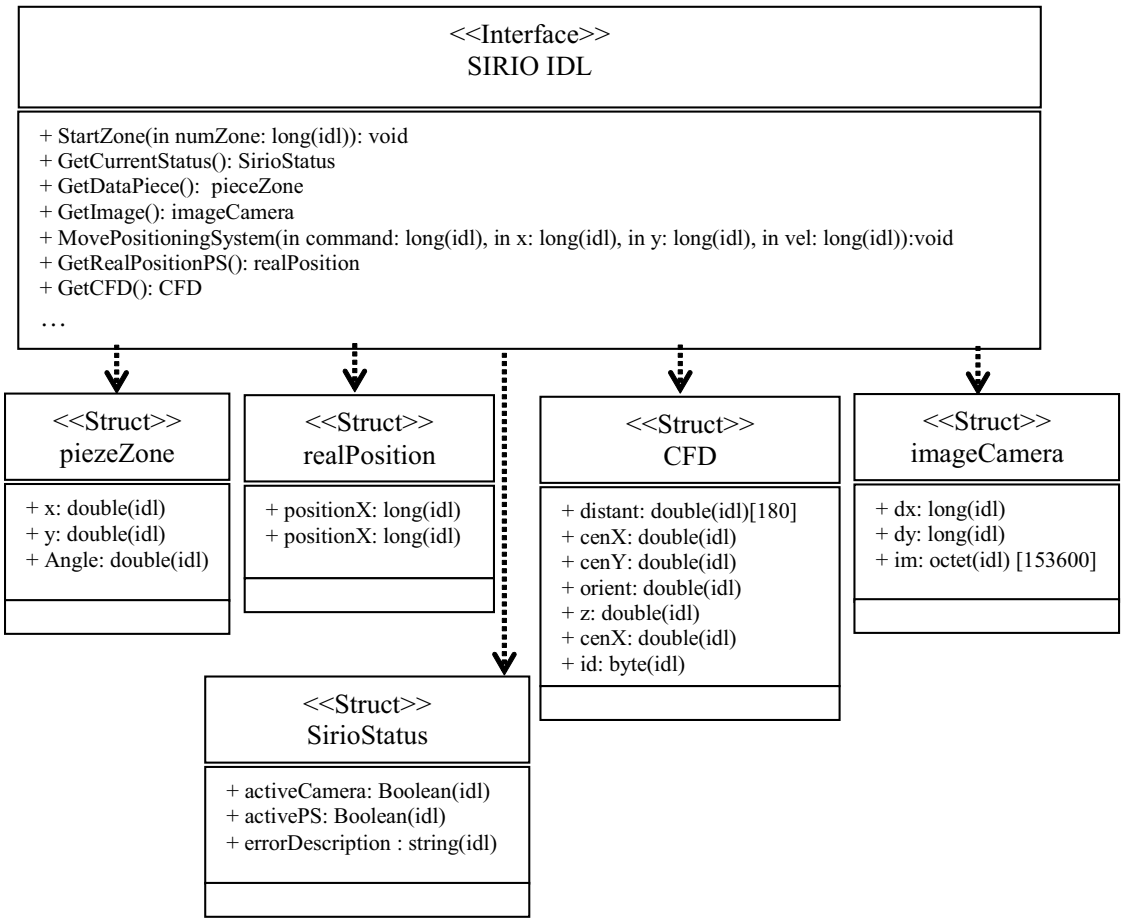

Figure 8. SIRIO Interface

iSIRIO interface contains functions used by the SICT to initialize the assembly cycle, to obtain the status of SIRIO, an image in real time or to move the camera over the manufacturing cell. The function StartZone, calls a process located in SIRIO to make the positioning system move to different zones of the cell. The function GetCurrentStatus is used to get the current status of the SIRIO

module, and it sends information about the hardware. When SIRIO finishes processing an image it sends an acknowledgement to SICT and this ask for the data using the function GetDataPiece which gives the position and orientation of the piece that the robot has to assembly.

The function GetImage gives a vector containing the current frame of the camera and its size. The function MovePositioningSystem is used by SICT to indicate to SIRIO where it has to move the camera. The movements are showed in table 1, where it executes movements using the variables given by the client that called the function.

Tabla 1.	**Command**	**Tabla 2.**	**X**	**Tabla 3.**	**Y**	**Tabla 4**	**Speed**
Tabla 5.	Start	Tabla 6.	No	Tabla 7.	No	Tabla 8.	Yes
Tabla 9.	Zone 1	Tabla 10.	No	Tabla 11.		Tabla 12.	Yes
Tabla 13	Zone 2	Tabla 14.	No	Tabla 15.	No	Tabla 16.	Yes
Tabla 17	Moves to (x,y)	Tabla 18.	Yes	Tabla 19.	Yes	Tabla 20.	Yes

Table 1. Commands for moving the positioning system.

The function GetRealPositonPS obtains the position (x, y) where the positioning system is located.

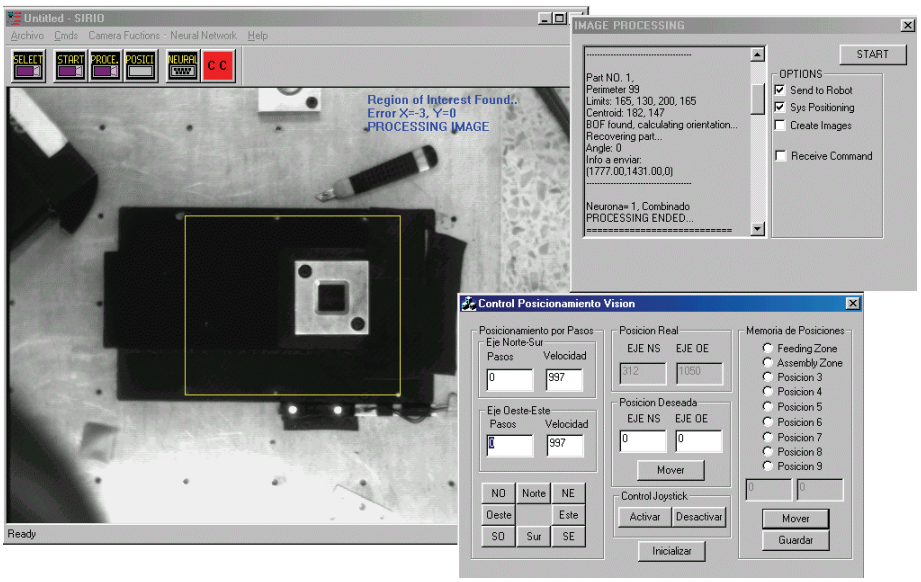

Figure 9. SIRIO main scream

The last function GetCFD(), gets the CFD (Current Frame Descriptor) of a piece. The piece is always the last the system used, or the one being used. The CFD contains the description of a piece. For more details the reader is referred to part III of this work (Peña-Cabrera, M. & Lopez-Juarez, I, 2006).

4.3 SIEM Interface

This contact force sensing system resembles the tactile sense, and uses a JR3 Force/Torque (F/M) sensor interacting with the robot and obtaining contact information from the environment. SIEM is used when the robot takes a piece from the conveyor belt or when or when an assembly is made. The robot makes the assemblies with incremental movements and in each movement, SIEM processes and classifies the contact forces around the sensor, using the neural network to obtain the next direction movement towards the assembly. SIEM is implemented in an industrial parallel computer using Windows 2000 and written in Visual C++ 6.0 and OmniORB.

Figure 10 shows the main functions of the IDL SIEM.

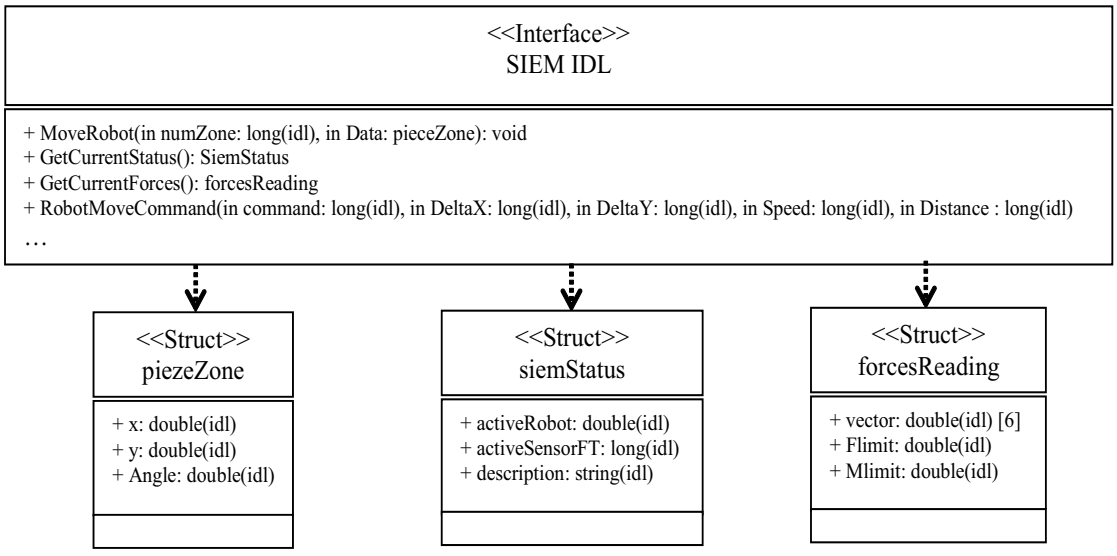

Figure 10. SIEM Interface

iSIEM: SICT moves the robot thought SIEM, obtains the components state and the reading of the current forces in the different zones of the manufacturing cell. The function GetCurrentStatus, is used to obtain the status of the hard-

ware (sensor F/T and communication) and software of the SIEM. The function MoveRobot is used when SIRIO finishes an image processing and sends information about the piece to the task coordinator.

The GetCurrentForces function helps the SICT to acquire force data from the JR3 Force/Torque (F/T) sensor at a selected sampling rate. This function returns a data vector with information about the force and torque around X, Y and Z axis.

Finally, the function RobotMoveCommand is used by the SICT to indicate appropriate motion commands to SIEM. These types of motions are shown in Table 2. Here is also shown the required information for each command (distance, speed). The windows dialog is shown in Figure 11.

Command	Distance	Speed
Do nothing [static]	No	No
Home position	No	No
Coordinates world	No	No
Tool Coordinates	No	No
Axe by Axe Coordinates	No	No
Base Coordinates	No	No
Movement X+	Yes	Yes
Movement X-	Yes	Yes
Movement Y+	Yes	Yes
Movement Y-	Yes	Yes
Movement Z+	Yes	Yes
Movement Z-	Yes	Yes
Rotation X+	Yes	Yes
Rotation X-	Yes	Yes
Rotation Y+	Yes	Yes
Rotation Y-	Yes	Yes
Rotation Z+	Yes	Yes
Rotation Z-	Yes	Yes
Diagonal X+Y+	Yes	Yes

Command	Distance	Speed
Diagonal X+Y-	Yes	Yes
Diagonal X-Y+	Yes	Yes
Diagonal X-Y-	Yes	Yes
Finish Communication	No	No
Open griper	No	No
Close griper	No	No
Rotation A1+	Yes	Yes
Rotation A1-	Yes	Yes
Rotation A2+	Yes	Yes
Rotation A2-	Yes	Yes
Rotation A3+	Yes	Yes
Rotation A3-	Yes	Yes
Rotation A4+	Yes	Yes
Rotation A4-	Yes	Yes
Rotation A5+	Yes	Yes
Rotation A5-	Yes	Yes
Rotation A6+	Yes	Yes
Rotation A6-	Yes	Yes

Table 2. Commands to move the robot

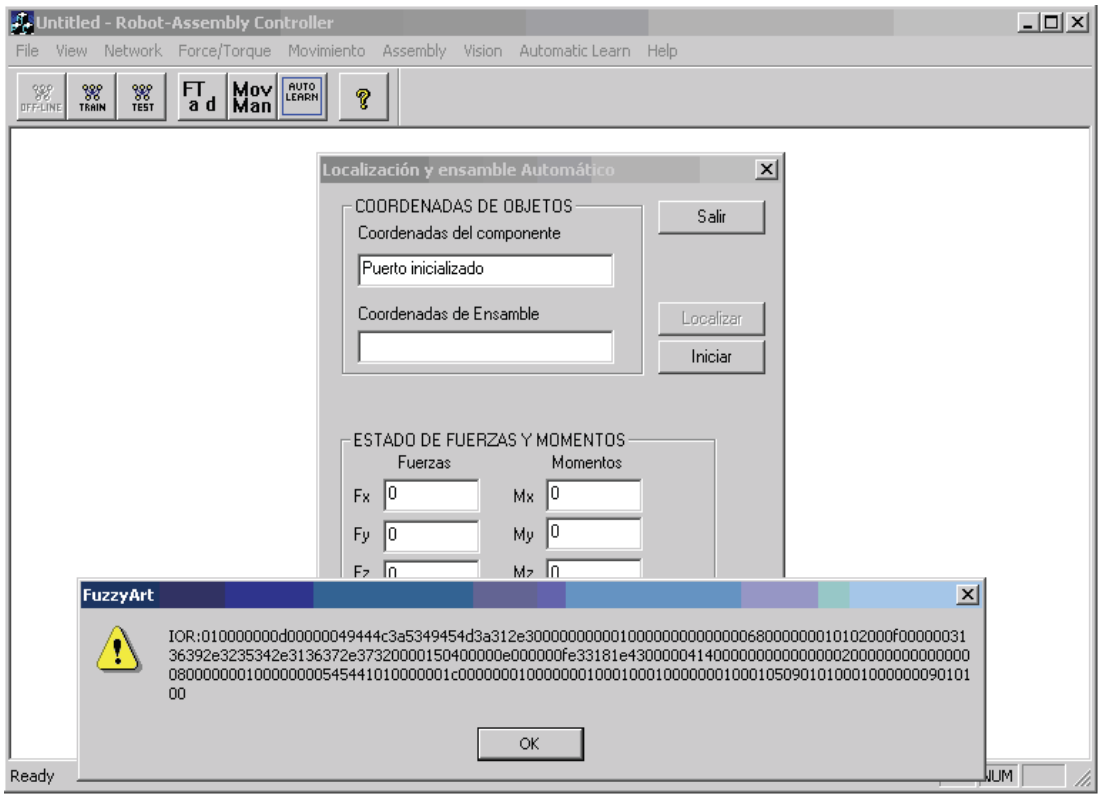

Figure 11. SIEM screen

4.4 SPLN Interface

The system provides a user interface to receive directions in natural language using natural language processing and context free grammars. After the instruction is given, a code is generated to execute the ordered sentences to the assembly system. The SPLN is based on Linux Fedora Core 3 operating system using a PC and programmed in C language and a g++, Flex, Yacc and ORBit 2.0. compiler.

iSPLN: This interface receives the command status from the SPLN, and gets the system's state as it is illustrated in Figure 12.
EndedTask is used by the SICT to indicate the end of a command to the SPLN like the assembly task. As a parameter, SICT sends to SPLN the ending of the task. GetStatus function serves to obtain the general state of the SPLN.