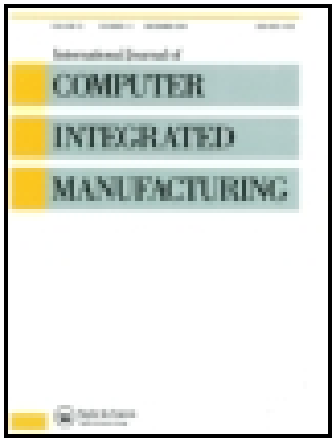

International Journal of Computer Integrated Manufacturing
Publication details, including instructions for authors and subscription information:
http://www.tandfonline.com/loi/tcim20

A software engineering method for the design of discrete manufacturing cell control

J. M. Lopez , J. I. Llorente , A. Burgos & I. Martija
Published online: 08 Nov 2010.

To cite this article: J. M. Lopez , J. I. Llorente , A. Burgos & I. Martija (1997) A software engineering method for the design of discrete manufacturing cell control, International Journal of Computer Integrated Manufacturing, 10:1-4, 126-141, DOI: 10.1080/095119297131246

To link to this article: http://dx.doi.org/10.1080/095119297131246

INT. J. COMPUTER INTEGRATED MANUFACTURING, 1997, VOL. 10, NO. 1–4, 126–141

A software engineering method for the design of discrete manufacturing cell control

J. M. LÓPEZ, J. I. LLORENTE, A. BURGOS, I. MARTIJA

Abstract. The objective of this work has been to develop a cell control system which could be used in manufacturing cells, truly adapted to real industrial environment needs and suited for different physical cell configurations. The main goals achieved within the project have been the development of a user-oriented methodology and software tools, a reference architecture for cell control applications and a common language for the analysis, design, simulation and implementation of the cell control logic. The proposed methodology is characterized by modular, integrable, and distributed developments and is described in an Implementation Guide, which gives a complete project view and covers different issues related to cell control development projects, and also compiles and organizes the previous knowledge and experience existing in cell control. In order to validate, evaluate and prove the methodology, an industrial pilot application was developed using a Petri-net-based graphical tool. With the proposed approach most of the problems appearing in the software development for manufacturing control systems have been reduced, giving a complete project support, and covering all the development stages. The use of Petri nets also offers the possibility to deal with synchronous and concurrent events as it is required for manufacturing control systems.

1. Introduction

1.1. *The problem*

This work aims to solve some of the problems which are most commonly found when dealing with software for manufacturing applications. Several of those problems, the most outstanding ones, are listed below.

(1) Cell controllers are generally not very flexible systems; that is, they lack the capacity to adjust to changes easily.
(2) Cell controllers usually are systems which have been designed off-line at pre-installation stages, which makes it very difficult to modify them. This fact is extremely relevant, since the design parameters of the manufacturing cell may vary all throughout the cell life-cycle (generally estimated at about five to six years, depending on the kind of sector where the production plant is placed).
(3) In general, we find that the work carried out previously in similar projects is not commonly re-used.
(4) The level of integration achieved at the different manufacturing applications is usually low. Very frequently, however, custom-made interfaces are created among the different software applications, which are specific to every plant.
(5) Carrying out a project of this kind demands a great effort.
(6) Many cell controllers installed in factories are too complex and not adequate for the real needs of their users.

Due to the inherent complexity and variety of manufacturing cells, there is a lack of standard discrete cell control applications capable of coping with all of the above problems. In addition to this, it is nearly impossible for a unique solution or software package to satisfy all the requirements inherent to each specific manufacturing cell environment. The major outstanding issues in existing cell controllers are modularity, reliability, autonomy and distributed management, scalability and interoperability among all the modules which constitute the global system, aiming at implementing an open cell control architecture.

In order to solve these problems, this paper presents a methodology for the design and development of cell control software systems. The proposed

Authors: J. M. López, ROBOTIKER, Technology Transfer Centre, Parque Tecnológico, Edificio 202, 48170 Zamudio (Bizkaia), SPAIN. J. I. Llorente, A. Burgos and I. Martija, Department of Mechanical Engineering, University of the Basque Country, Faculty of Engineering, Alameda de Urquijo, s/n, 48013 Bilbao, SPAIN.

0951-192X/97 $12·00

methodology enables the generation of cell control application with the features listed below.

(1) Modularity in order to accommodate several plant configurations. The developments must be modular, in order to make the integration of different suppliers in the flexible manufacturing system (FMS) easier, and in order to allow for the development of the system after tuning. Modularity, then, should affect both the physical elements of the plant and the software.
(2) Integrability with other applications, especially, connection with higher control systems.
(3) Distributed FMS control, which should be able to go from automatic to manual easily, without acting on plant devices.
(4) Distributed databases which operate in real time: also plant variables state, devices, transport system, etc., which may be configured or modified by the user after implanting them.
(5) Reliable communication system, in charge of all FMS messages (with different degrees of complexity). It should be standards compliant, so that the connections among the different elements which make up the system can be carried out quickly and reliably.
(6) Client–server architecture: interfacing between modules throughout the whole software system is carried out by means of service requests, and continuous attention to event arrivals.
(7) Use of both graphical tools and a prototype construction environment for the development of the Basic Control Dispatcher module (BCD).
(8) Object–oriented plant model, divided into several levels in order to describe the manufacturing plant and the peculiarities of the control system software.
(9) The use of a methodology to implement the manufacturing system software reduces the likelihood of error occurrence from the very early development stages. By means of this we also aim at improving and increasing the efficiency of the implementation task, as well as reducing the amount of time which is necessary to take us from system conception to implementation.

This methodology is described in an Implementation Guide, which gives a complete project view and covers different issues related to cell control development projects, and also compiles and organizes the previous knowledge and experience existing in cell control. Also, a reference architecture for cell control application is defined and a common graphical language for the analysis, design, simulation and implementation of the cell control logic is used.

1.2. *Present situation of discrete manufacturing systems*

Flexible manufacturing cells and systems are one of the most relevant moves towards the concept which is generally known as 'the factory of the future'. The benefits which those companies incorporating these production systems may obtain should be highlighted, and so should the great impact that they have in achieving an improvement in manufacturing quality. Some of the most significant benefits that user companies may derive could be pointed out, for example, competitivity improvements and a rapid response to market demands. People selling systems of this kind, on the other hand, should maintain their competitive position in the market.

The main reasons for the advantages mentioned above can be found in their high automation level (high productivity level), together with high flexibility level (that is, their component elements are highly integrated). These capacities regarding both productivity and flexibility are materialized and harmonized by means of very diverse software systems; the range includes low-level systems, such as numerical control (NC), programmable logic control (PLC) or robot control (RC) programs, and the most sophisticated cell control applications.

Flexible manufacturing cells are particularly important. They are very closely linked to a part family. In this way, the complexity of the control software, as well as that of other areas, such as direct numerical control (DNC) and tool management, is considerably reduced. Moreover, when the most relevant manufacturing task being performed is part machining, these systems are usually known as flexible machining cells. This paper focuses on this field.

A basic feature of the production systems that we are dealing with is that their behaviour cannot be described by means of mathematical models with differential equations, according to classical control theory. Rather, they generate a series of events that the control system must know how to co-ordinate adequately. They are generally known as discrete event dynamic systems.

In order to describe and model their behaviour, several different methods are used, such as discrete event simulation, Petri nets, several queuing network theories, automaton theories, and finite state machines (Sauter and Judd 1990). Up to now there has been no unique method to model a production system, and several of the techniques mentioned above are used

jointly in practice. Among others, we should highlight some graphical systems of modelling, such as GRAI networks (based on the mathematical theory provided by Petri nets) (Meyer 1991), and tools such as GRAFCET (which is commonly used to help in PLC programming) (AEW 1992, IEC 1988).

We talk about computer aided control engineering (CACE) (Lydon 1992) when computer help is available. Its main goal is to make several tasks easier and more agile, including, for example, such stages as the design, programming, simulation, debugging, downloading to the shop-floor, monitoring and improvement of the production systems control applications. In this way, we are able to fill the gap between the problems of the system to be controlled and the organization environment.

The behaviour of a FMS can be, to a certain extent, made similar to a materials and information processor. That is, on the one hand, materials are processed, and events are produced. These events are treated by the control system (information processor) which, in turn, generates the corresponding commands. According to this view, one of the most effective techniques used nowadays in order to model the behaviour of production systems (as well as to support making manufacturing decisions) are those systems based on knowledge (Meyer 1991, Sacket and Fan 1988, Fan and Sackett 1988, Sackett and Fan 1989).

Cells and FMS systems are widely used, and thus how to carry out their control is a topic that has been approached from different points of view. On the other hand, there are not very many examples of true implementation of this kind of cell control (that is, a standard system) in factories.

However, there do exist many implementations of automatic (or even super-automated) systems. The controllers which rule them, however, have been devised as the need has arisen, so that we cannot freely talk of cell controllers, but rather of a set of controllers which are coordinated with different degrees of success. The rules that govern the controller are usually a consequence of off-line systems analysis (Gross and Ravi Kumar 1988), and thus it is generally difficult to carry out any modification during the normal functioning of the system.

In the market, however, there are products offered either by complex manufacturing process companies (such as the aeronautical or automation industry), or by manufacturers of integrated systems of the FMS type (turnkey projects). The former are complex and difficult to adapt, and the latter are not standard. Providers of control equipment have, in turn, evolved towards cell control levels. On the one hand, we find those who specialize in high-level computers (IBM, DEC) (IBM_CIM) (Stanislawski 1990), who are always trying to make a better offer, in order to cover the requirements of plant information systems. On the other hand, we find those who specialize in shop-floor equipment (Siemens, Allen Bradley) (SICOMP) (Baumgartner *et al.* 1991), (Henderson 1987), who offer more and more solutions at the production activity supervision level.

Machine tool manufacturers, in turn, generally have a wide supply of flexible systems available. The range includes the simplest FMS integrable machines, equipped with automatic pallet chargers and small cells with robots or manipulators for part load and unload, as well as the most sophisticated FMS systems, with scalar dimension, functionality and services (these systems are fitted with a control system which, in general terms, is very complex, but which is designed considering the machine manufacturer's demands very closely; on the other hand, they are rarely open systems) (Baumgartner *et al.* 1991, Khermouch 1989).

2. Developing software for manufacturing systems

2.1. *Techniques which are generally used*

The development of a control system for a general FMS, valid for systems with an average degree of complexity, and which might be implemented in different environments, involves the participation of a number of people: the customer and final user, the system analysts, programmers, mechanical designers, the staff in charge of tuning it, etc. According to the work of researchers such as Boucher and Jafari (1992), and Jafari and Boucher (1994), it is necessary to co-ordinate the design of automated manufacturing systems and the design of their controller. These activities are usually carried out by people who belong to different engineering fields, so that a common result is that the design of the physical system and the design of the control system software are evaluated and analysed separately and sequentially. This fact may lead to design mistakes which are not solved until the integration stage is reached, or until the initial system prototypes are tested. This obviously makes system development or system installation and commissioning an unnecessarily long process.

All this makes it necessary to use a high-level specification language which should be understood by everybody, so that communication is easier; at the same time, this kind of a language would provide great accuracy and formalism, expressed by means of simple graphical diagrams, in order to model the system internal state, transitions, task priority, triggering

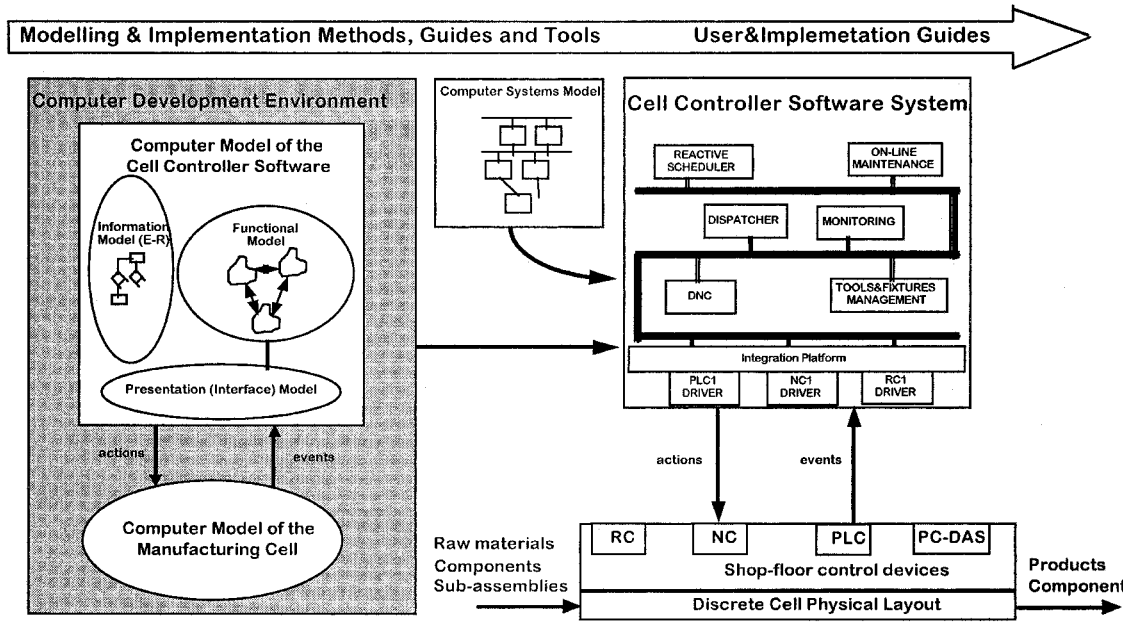

Figure 1. Working environment both with the model and with the real system.

conditions, timeouts, delays, parallel activities, and its co-ordination. Figure 1 shows the recommended work outline at the different stages of the manufacturing system (analysis, design, simulation and operation). It would be very interesting if we could use tools to exchange effort among all stages, so that models, experiences, knowledge, etc., could be reused.

Other approaches have been proposed (Joshi *et al.* 1995) which suggest solutions to automatically generate the cell control software, which comprises both decision-making modules (to carry out the planning and scheduling functions) and execution modules (in charge of interacting with the physical production system). These decision-making modules are implemented by means of discrete event simulation techniques, using in particular the SIMAN simulation language. The performance modules, in turn, use the part state degrees, which are based upon the message exchange, and thus they constitute a kind of formal description model based upon finite state automata. However, these modelling techniques concerning the interaction between the control system and the manufacturing plant by means of finite state automata are quite difficult to interpret for non-experts. What is more, sending several messages in parallel among different system elements is not considered, and this is an important difficulty when trying to build the system model, specially if we bear in mind the asynchronous and concurrent nature of manufacturing systems.

A high-level formal language may, nevertheless, be used both in order to describe a manufacturing system design, and to specify the control strategy that should be implemented in the controller's software. This high-level specification may be used by the designers of the manufacturing system; their knowledge of control engineering and control software design does not need to be very deep for the whole life-cycle of the project (from the design stage to those of integration, documentation, tests, maintenance, etc.) (Boucher and Jafari 1992, Thomas and McLean 1988). This specification might be used later on to be directly executed within a simulation and development environment, or to implement the controller in a conventional programming language, provided the indicated guidelines are followed.

There are some additional advantages to using high-level languages to describe a manufacturing system controller.

- The time that the engineers invest in manufacturing system design and evaluation processes is considerably reduced.
- They provide a reference framework for fluent exchange of ideas. They also allow one to quickly obtain feedback regarding evaluation of the system performance for the design alternatives that are proposed.
- They offer a basis for a closer interaction among the different engineering disciplines that are involved in the system design process, so that the need to work in teams, and the need for concurrency within the engineering practice are approached.

- Simplicity is brought into different areas: code modularization, task specification, communication among parallel processes, detecting control flow problems, code reusability, introducing design alternatives, document generation, and in general the good practice of software engineering.
- Interactive execution of the controller design is also made possible; control flow may be monitored by graphical evolution of the different stages on the screen. This allows the system designer to contrast his/her ideas interactively with those of the customer or end user, and also to tune certain control sequence at the very first system design stages. The system dynamic testing and debugging is also made easier. At the end of the design stage, the controller will be sufficiently advanced to be integrated with other plant software, and to work with real shop-floor data.

A practical implementation of these ideas is that suggested by Jafari and Boucher (1994), proposing the use of formal models such as GRAFCET. This method has been widely accepted in Europe, especially in France, as well as in Canada, in its francophone areas (Boucher and Jafari 1992), where the GRAFCET representation has become quite popular, to the point of becoming the standard for sequential control in manufacturing systems. Following the above mentioned researcher, other techniques, such as contact diagrams, continue to be used for these tasks, but none of them offers the analytical possibilities or the maintainability and flexibility of the programs that the GRAFCET language offers. In fact, there are some tools on the market which implement the GRAFCET language, such as CIMPICS by Reflex (DTI 1990, DTI/SERC 1991, Franks *et al.* 1990) or Flexis ToolSet by Savoir (Wilczynski 1988, Thomas and McLean 1988), and which are oriented towards obtaining automatically an application coded in the C language.

2.2. *Reference architecture for cell control software*

According to Adiga (1993), a reference architecture for a system serves basically to help build it, following an integrated view of the whole system. In turn, the architecture of an information system is generally made up of two elements: a structure and an organization. The structure describes the components and their inter-connection, whereas the organization represents the dynamic aspects of the components, and their management sides, according to the selected operational principles.

Those hierarchical architectures that are traditionally proposed process actions and events at different control levels (Figure 2).

Taking a more global view, the software system integrated within the cell level may be represented just as Figure 3 shows; this figure contains the architecture of the different modules which make up the

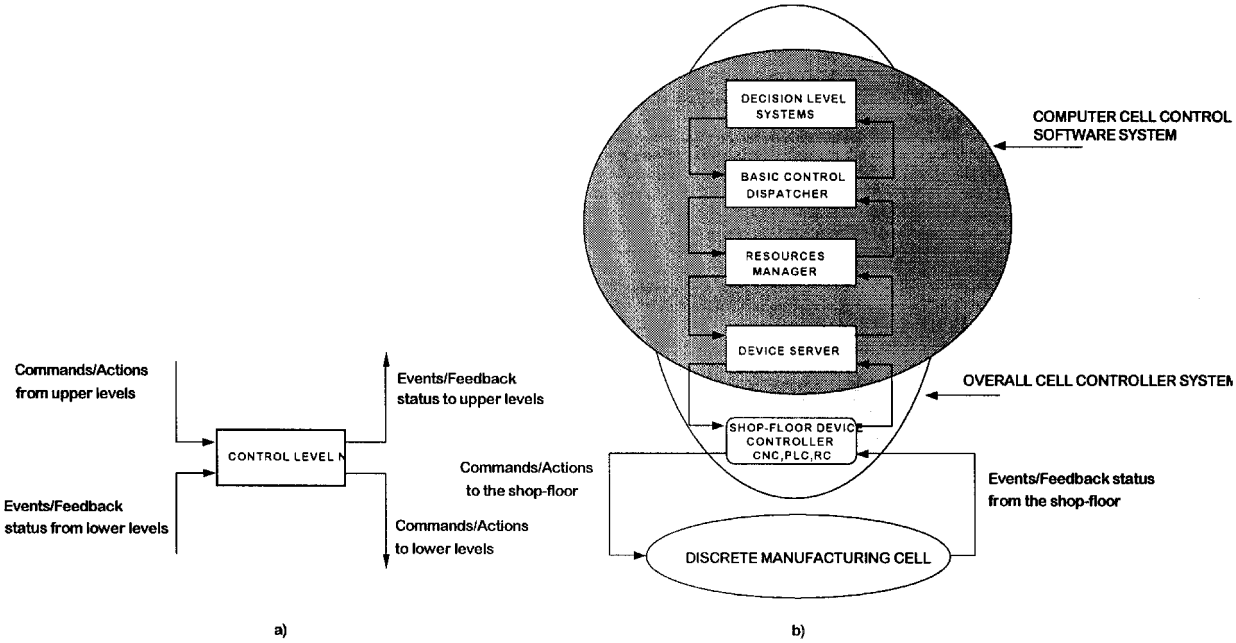

Figure 2. (a) Actions and events processing in a hierarchical architecture. (b) Example of actions and events processing for the different levels of control software within the computer and within the shop-floor controller equipment.

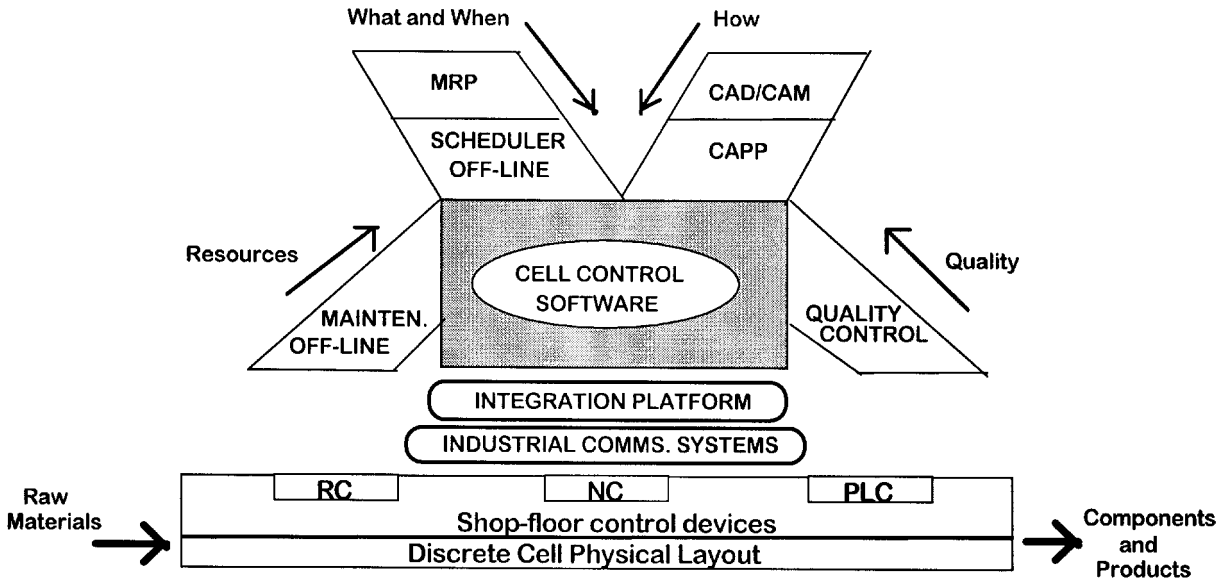

Figure 3. Environment of the cell controller software system.

manufacturing cell environment. This scheme shows, in a general way, those modules which take part and are related to the cell control system. It is a modification on the IDS design (Scheer and Hars 1991). There are three clearly different aspects:

- The information regarding what and when should be manufactured (**MRP/SCHEDULER OFF-LINE**), as well as how to manufacture each product (**CAD/CAM/CAPP**) flows towards the **Cell Control**. Both planning and the design are nurtured by the information and indications of **Maintenance** and **Quality**. The communication between the cell control and the plant takes place by means of **Device Servers**, which are the software elements within the cell controller representing the shop-floor device control equipment.
- As Figure 4 shows, the **Cell Level**, on which this paper is focused, is made up of several modules: **ON-LINE Maintenance**, **Dynamic Scheduler**, **Basic Control Dispatcher (BCD)**, **Resource Managers** (Tool Manager, DNC Manager), and **Device Servers**.
- The information regarding what, how, and when to manufacture gets to the plant devices by means of the cell control, which then acts as a link or co-ordination element between the manufacturing plant and the higher level software modules (its distributed nature and its operating capacity in real time should be underlined). Figure 5 gathers the information flow among the different modules which make up the control system within the architecture that is proposed in this paper.

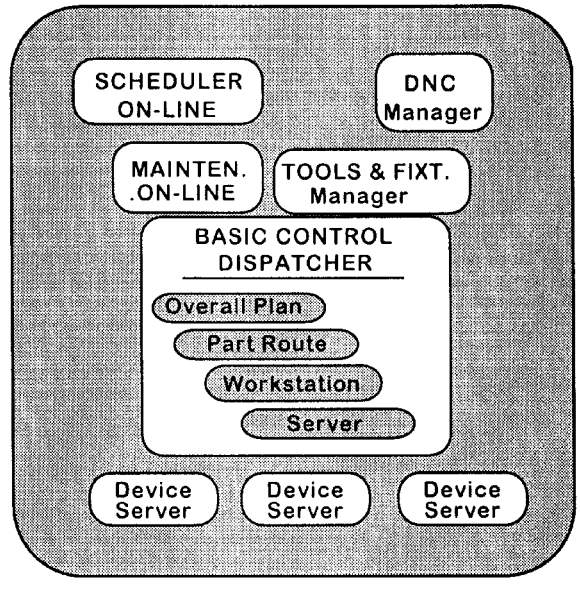

Figure 4. Outline of the modules that make up the cell controller.

2.3. *Description of the methodology used for the design and development of cell controller software systems*

Next, we describe the stages that have been followed in this paper, leading towards the definition of the software system architecture design at the cell level for a manufacturing plant, As a basis to work on, we take the model of reference architecture shown in Figure 5, which is made up of a set of device servers, resource managers, as well as a Basic Control module, together with the Dynamic Scheduling and

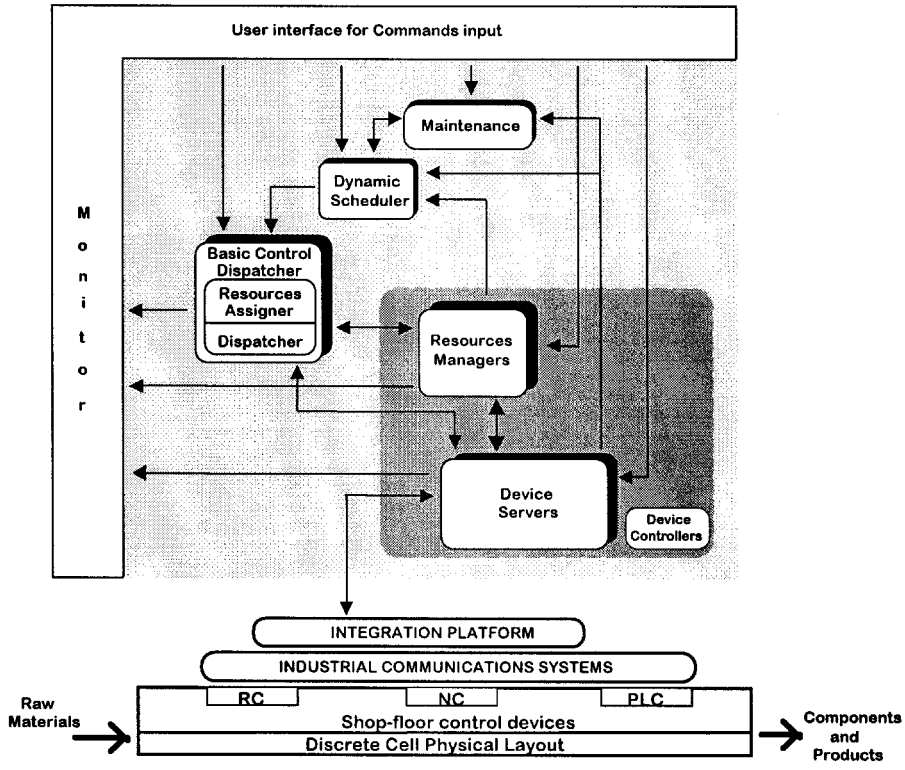

Figure 5. Proposed model of reference architecture.

Maintenance modules, whenever they are necessary. The point of departure of this process is the functional (or requirement) specification of the global manufacturing system. After this initial step, each of the stages that are defined according to this methodology are applied, starting with the Software Requirements Specification (SRS) document for the cell controller software system until we get to the software system architecture design at the cell level. This is expressed in the Software Design Description (SDD) document, according to the standards that are to be applied at every instance, for example those generated by the Institute of Electrical and Electronic Engineers (IEEE 1984) and (IEEE 1987). Figure 6 represents, graphically, these considerations.

When developing this methodology, the use of object-oriented techniques for software analysis, design, and implementation has proved to be highly advisable, since they have very interesting advantages. One such advantage is their good coverage of the concepts of client and server applications, which interoperate on the basis of service requests. Another advantage is their suitability for iterative development, which in turn makes application prototyping easier. Last, but not least, they favour flexible, robust, and reusable applications, due to the data and methods encapsulation (Adiga 1993).

The methodology that is proposed to build the

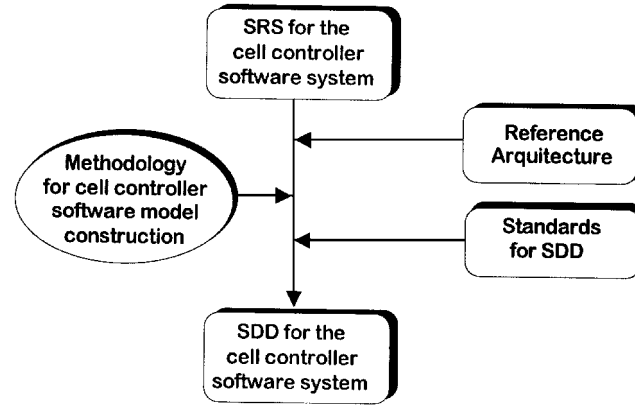

Figure 6. Producing the software design description (SDD) for the cell controller software.

model is carried out in an iterative process, in which several stages may be distinguished; in each one of these stages, given certain inputs, concrete results will be obtained by applying a series of procedures. Sometimes, system designers will think it appropriate to go back to a prior stage, as a result of the modifications that have been introduced in successive stages. The phases that have been taken into account are dealt with in detail below, and Figure 7 shows graphically how those stages are covered as the design process proceeds. The functional dependence of some stages with regard to others

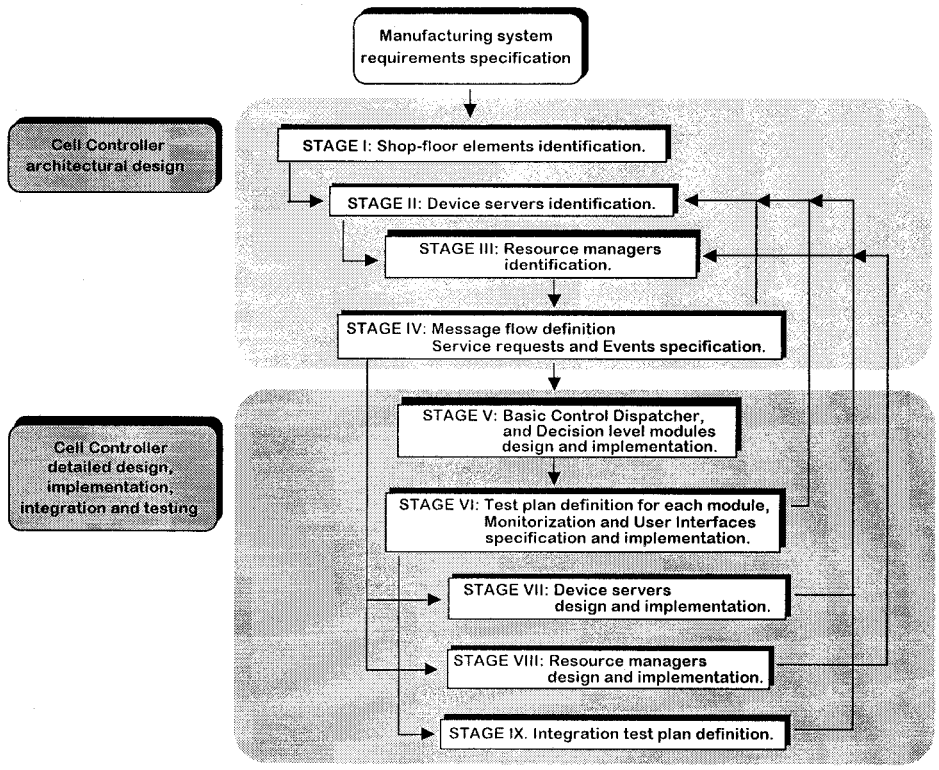

Figure 7. Stage execution sequences and their feedback.

is represented as well, although no reference to their time dependence is done at this moment.

All stages may be performed sequentially provided that we have a very detailed knowledge of the physical operation of the plant, and of the functionality of the software system. In this case, the design will rarely be subject to changes, which means that going back to previous stages would not be necessary. Figure 8 contains an improved proposal regarding the time dependence of the different stages. Thus, we can see that the first stages are really inter-dependent; however, after a certain time, stages can be concurrently developed. In practice, the design will be carried out as an iterative process in which, a basic preliminary outline (as far as requirements and operating modes of the system are concerned), is successively refined. Every stage will have an intensive development period (shown as the dark shaded areas on Figure 8).

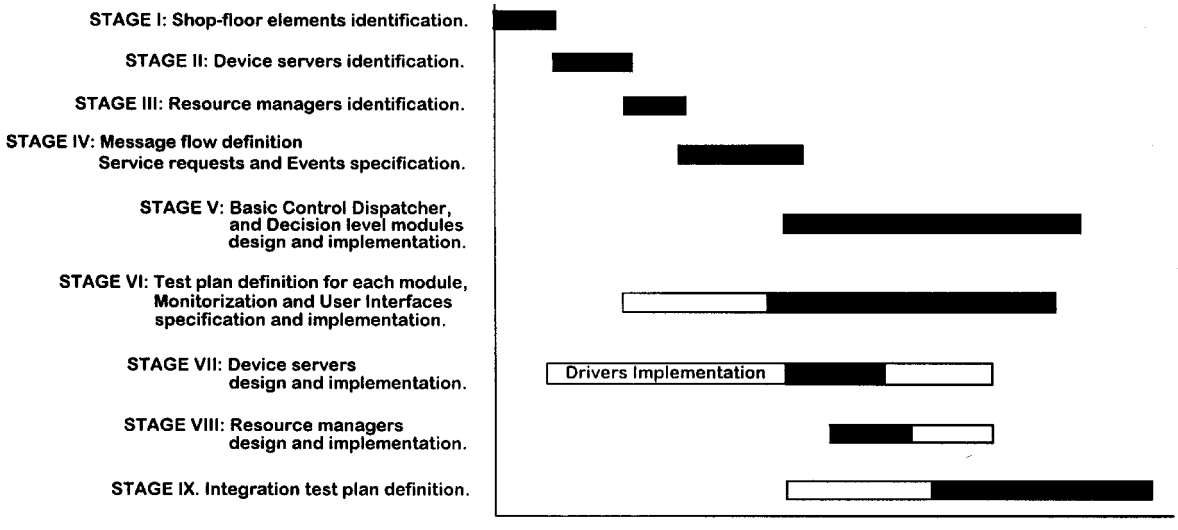

Figure 8. Time dependence among the stages.

3. Results of the practical application

In order to verify, prove and demonstrate the validity of the methodology described above, a prototype cell controller for a pilot plant (in particular, a machining flexible cell) has been built at the Laboratory of the Department of Mechanical Engineering. This pilot plant has both educational and technological goals, since students have had the opportunity to take part in the construction of such a cell, so that they have become acquainted with these new integrated manufacturing technologies.

Taking as a point of departure previous experiences with simpler systems, the pilot plant has been built in two phases. As a consequence of this, it has been necessary to adapt and re-structure the cell control software system. Thus, it has been possible to study and test the advantages of implementing the software system by means of the methodology proposed.

3.1. *Plant manufacturing equipment*

At Stage 1 of the project, the cell was made up by the manufacturing equipment that are mentioned below:

- A Kondia B-500 machining centre: it contains a 3-axis automatic tool changer, and tool magazine with 18 positions.
- A CMZ TBI 450 turning centre: it has C axis control, motorized tools, and a MONTRONIX TS200 cutting strength control system by Kennametal.
- Robot Fanuc S-10: with 6 axes, and linear movement control.
- Robot Fanuc S-700: with 6 axes.
- A Bosch TS-2 transfer line: equipped with stations that are able to read and write pallet identifying cards.
- A tool warehouse with a tool identification and tool presetting station.

At Stage II of the project, a new machining centre (Kondia Powermill (3 axes, 24-tool magazine)) was added, with the purpose of providing the manufacturing system with higher flexibility levels, so as to allow for validation of the Dynamic Scheduling module. This machining centre was incorporated with its corresponding ASEA IRB/6 robot, in charge of loading and unloading the parts. In addition, in order to feed parts onto this new workstation, the original transport system was enlarged.

3.2. *Device controllers*

At Stage I of the project, the plant manufacturing equipments described above were controlled by the following control devices: NC Fidia Copymill, NC Fagor 8010-T, RC Fanuc S-10, RC Fanuc S-700, PLC Siemens SIMATIC. At Stage II, others were added: the NC Fagor 8010-M (which belonged to the new machining centre), the RC ASEA IRB/6 (provided by the new robot); in addition, the PLC Siemens SIMATIC programs were re-designed, in order to accommodate the new operating conditions of the transport system.

3.3. *Workstations*

The pilot flexible machining cell may be divided into different functional areas, known as workstations. Thus, at Stage I of our project (Figure 9), the following workstations could be found: Palletizing, Turning, Machining, and Transport. At Stage II (Figure 10), a second machining workstation was added. This new workstation was made up of the new machining centre (Kondia Powermill), and the ASEA IRB/6 robot. In general terms, these workstations are in charge of performing some manufacturing tasks (machining, movement, palletizing, etc.), and each one of them has several plant manufacturing equipments assigned (as well as the corresponding control devices). Some of this manufacturing equipment is shared among several workstations; some tasks must also be carried out by the co-operative performance of several manufacturing equipment (for example, part and tool load and unload need RCs and NCs to cooperate).

Dividing the manufacturing plant into workstations is a great help to isolate the manufacturing system control software from the equipment modifications, re-configurations, or replacements that usually take place in this kind of manufacturing plant (characterized by its great flexibility in the face of changes).

3.4. *Functionality covered by the global software system*

The global software system is made up of several modules in order to adjust to the requirements of the manufacturing environment which is particular to this pilot plant. The modules that it comprises at Stage I (Figure 11) are as follows: MRPII (MAPICS by IBM), CAD/CAM (CATIA by IBM), Off-Line Robot Simulation and Programming (GRASP by BYG), Computer Aided Process Planning (CAPP) (CoroPLAN by Sandvik Coromant), Tool Management (CoroTAS by Sandvik Coromant), DNC Management, Basic

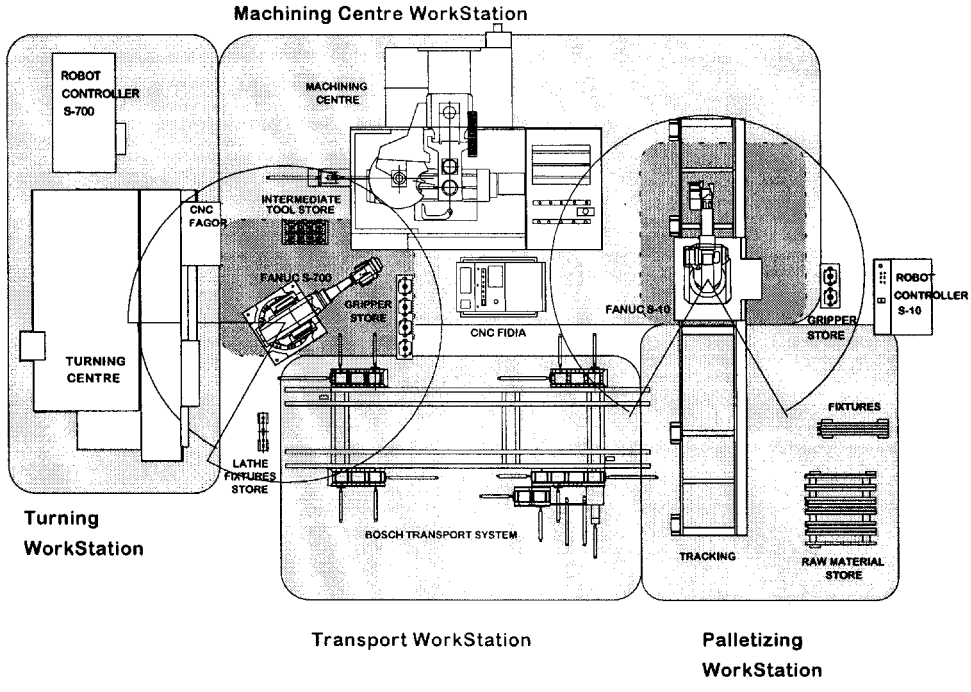

Figure 9. Layout of the flexible machining cell, and its associated functional areas (workstations). Stage I of the project.

Control Dispatcher (BCD developed with the G2 tools, by Gensym and GRAFCHART from the University of Lund), Graphical Monitoring (PLANTWORKS by IBM), as well as Device Servers for each of the plant device controllers. At Stage II of the project, together with the corresponding Device Servers, which are necessary to have access to the new plant controller equipment incorporated at this time (NC Fagor 8010-M and RC ASEA IRB/6) and to a plant operator screen, new software applications were

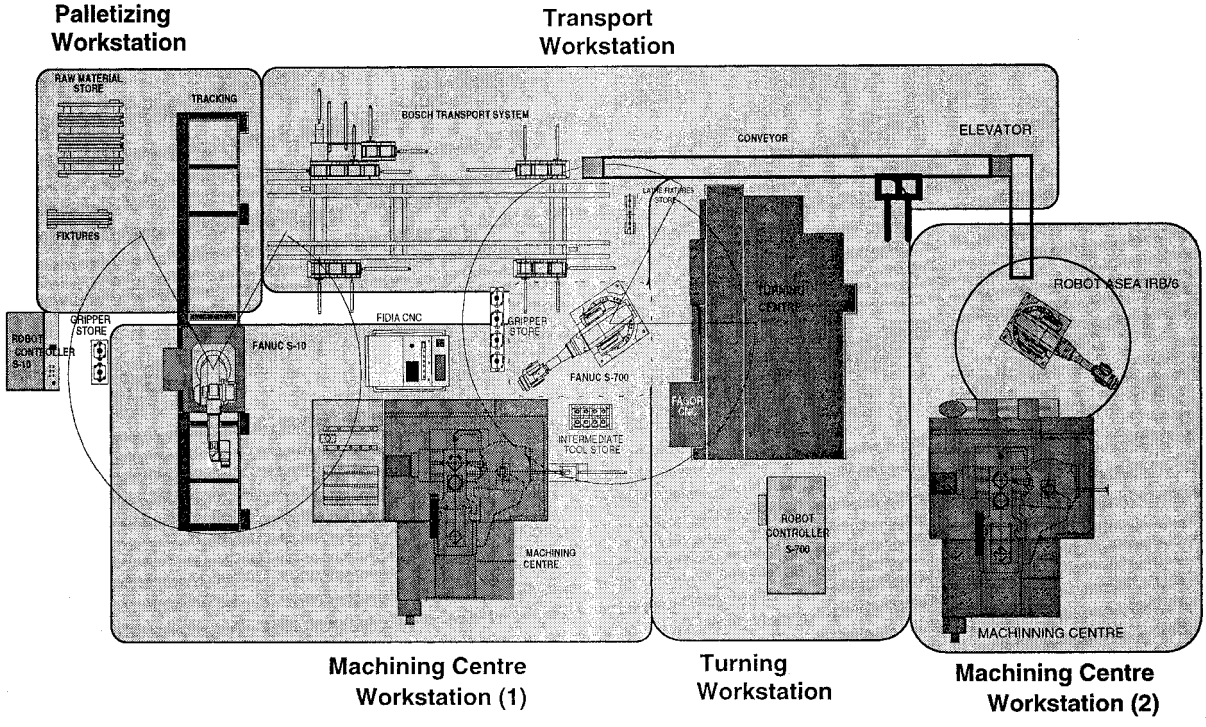

Figure 10. Layout of the flexible machining cell, and its associated functional areas (workstations). Stage II of the project.

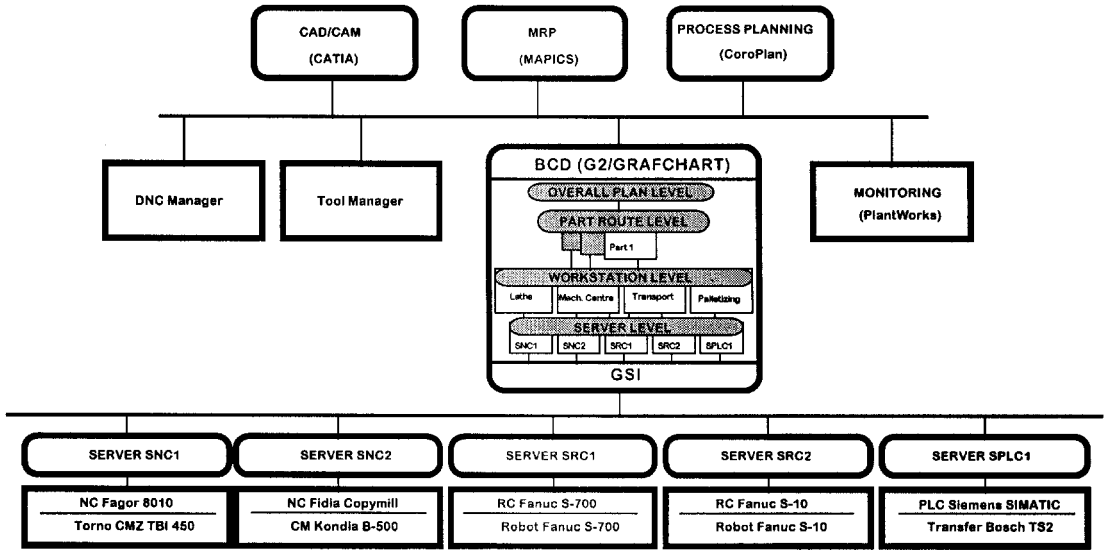

Figure 11. Modules that make up the global software system at Stage I.

added (such as Dynamic Scheduling, and On-Line Maintenance).

3.5. *Architecture of the manufacturing software system*

By using the approach proposed in this paper, we are able to deal with two main problems that are commonly found in this field, that is, the complexity of the tasks to be performed, and the uncertainty of the data and information that are handled within the cell level. We do this, by means of different software modules which work at different levels. The modules described below are found as common elements in the cell control prototypes developed for Stages I and II of the project:

- Basic Control Dispatcher (BCD): this BCD module is a high-level software component which is in charge of coordinating and synchronizing (according to a pre-established control logic) the plant activities necessary to manufacture all parts within the manufacturing plan. For the design and implementation of the BCD module, graphical tools based on the standard GRAFCET have been used. This module runs in an IBM RS/6000 graphical workstation under the UNIX/AIX operating system.
- DNC Manager and Tool Manager: these modules are designed to perform their respective tasks in a very specialized way, making the task of the BCD module easier and simpler. On the other hand, we also considered the possibility of developing additional manager modules, such as the Transport Manager, but given the pilot cell features, that option was disregarded.
- Device Servers: these modules are in charge of solving problems such as the physical connection with the plant control equipment, their communication protocols, and other peculiarities which are equipment specific. In this way, the Device Servers provide the other modules within the cell level with a homogeneous image of the plant, acting as software–hardware components, which offer this global software architecture important reusability capabilities. These Device Server modules run on IBM PS/2 computers, under the OS/2 operating system; these computers are equipped with intelligent communication cards (RIC (real-time interface coprocessor)) in order to make the connection with the plant control devices via RS-232.

All these software modules, distributed within the cell level, inter-operate in a client/server architecture, using TCP/IP on Ethernet networks and Token Ring in higher levels, and the Distributed Automation Edition (DAE) integration platform on Token Ring in lower levels. The integration of all these modules that make up the system has been performed (at both project stages) by means of a set of utilities and tools, included in the concept of 'Software Bus', which is dealt with in other research work currently in progress at the University of the Basque Country. In particular, the Device Server modules use the functions that DAE offers in order to exchange messages, and also to connect with the plant control devices, by means of RIC cards.

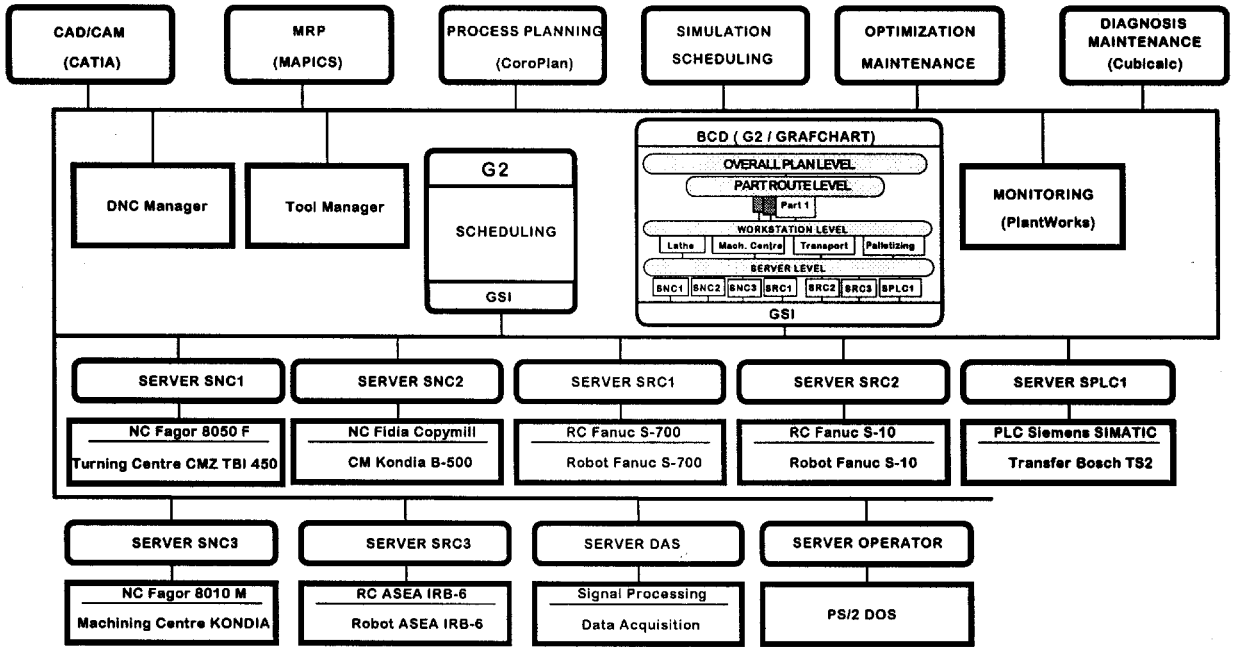

Figure 12. Modules that make up the global software system at Stage II.

3.6. *Service and event message specification*

The main goal of this task is to define Message Flow Diagrams (MFDs) on the basis of knowledge concerning the way every plant device works, and also regarding which services can be requested from each one of the servers that control them, and from the corresponding resource managers. At this stage, the interrelations among the different software modules are defined at cell level. As a consequence, the set of service and event generation requests that the plant devices, device servers, resource managers, and other control modules need to exchange, should be defined. In addition, the order in which both the service requests and the events should be generated, needs to be defined as well. It seems quite obvious that in a system of average complexity it may be almost impossible to define this message flow, when one considers the process as a whole. In this way, a first step to be taken is dividing the global process successively into manufacturing tasks which will be executed at the workstations, and which, in turn, could be sub-divided into sub-tasks, depending on their degree of complexity. Once this subdivision into simpler tasks is performed, we can proceed to represent the service request and event notification flow by means of diagrams. It is advisable to have every task into which the global process has been divided constitute a significant part of that process, with its own entity. Some examples of typical tasks in a flexible manufacturing cell or system could be the following: tool allocation and booking in order to manufacture a plan; tool load/unload in a machining centre; part palletizing; transport of a palletized part from the entry position to the machining centre; machining of a part in a machining centre, etc. These tasks are analysed next in more detail, and bearing in mind their degree of complexity, they are sub-divided into sub-tasks. In this way, for example, the task involving part machining is made up of the following sub-tasks: part load onto a machining centre, download of part machining programs, machining of the part in a machining centre, change of position of the part within the machining centre, and machined part unload. Each of these sub-tasks must have a message flow diagram (MFD) associated with it in order for the second goal of this stage to get the green light. This second goal is to define in detail each service and event. This definition should be complex enough to be implemented later on, including the kind of parameters that the client application and the server application will exchange, as well as the values that those parameters may adopt, or the error codes that are generated, etc.

When these considerations are taken into account, and aiming at making the construction of message flow diagrams easier, it is a good idea to use a computer tool. The range which is available is very wide, but the most appropriate ones are those that implement object-oriented analysis and design methodologies. In the work that we are presenting, the tool that has been

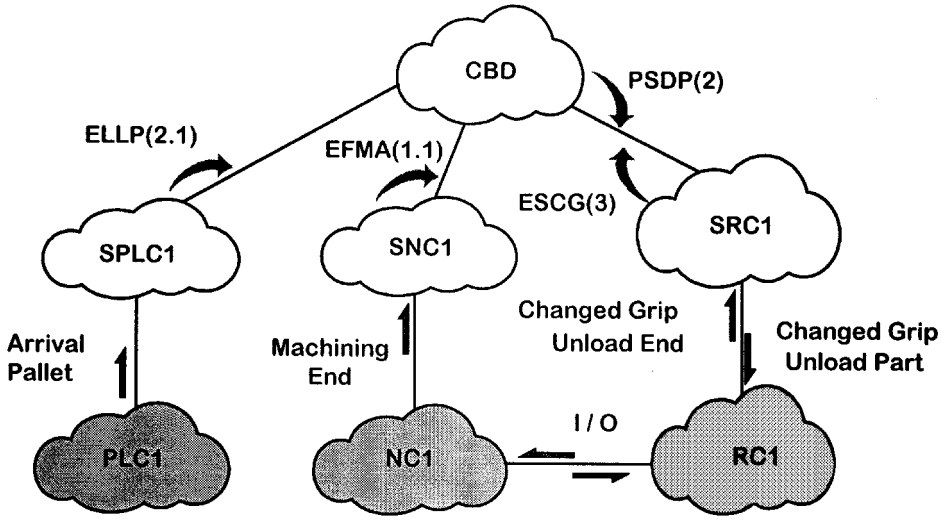

Figure 13. Part unload from the turning centre onto the pallet.

used is Rational Rose (White 1994), which implements the object-oriented methodology formulated by Grady Booch (Booch 1991). Figure 13 shows an example of diagram, specifying the services and events for the task concerning the part unload from the turning centre onto the pallet.

3.7. *Basic Control Dispatcher (BCD)*

At this point, we will analyse one of the modules of the cell controller, the Basic Control Dispatcher (BCD). In general terms, this module is in charge of both carrying out and co-ordinating the manufacturing activities within the cell, so that the actual production plan, which has been previously generated at a higher decision level, can be fulfilled.

At Stage I of the project, when the Dynamic Scheduling module did not exist, the BCD processes files with working plans, made up by sequences of parts that belong to different families. Every part family is manufactured according to a certain route within the cell. There are three possible routes within the cell layout at Stage I, that is, one for the parts which are only turned, another one for those that are only milled, and finally, one for those parts both turned and milled. The tasks that should be executed by each workstation in order to manufacture a part of a given family are clearly established within every route.

The BCD model used at Stages I and II is divided into four different levels, as Figure 14 shows. These levels are outlined below:

- Overall plan level: describes properly the operating stages of the cell control system.
- Part route level: is in charge of the actual operation state as far as part manufacturing is concerned. It describes the list and sequence of tasks that should be performed in order to manufacture each part. The basic actions that are carried out at this level are the introduction of task objects into the message queues at the corresponding workstations.
- Workstation level: contains the description of the tasks that may be performed within each of the functional areas of the cell or workstation. The task objects which are queued in the message queue at every workstation are exploited at this level in a sequence of service objects, which, in turn, are queued in the message queues of the corresponding Device Servers. Moreover, at this level we have the resource assigner available; this is an element within the BCD, which is responsible for selecting the most adequate moment to send the corresponding Service Requests to those plant manufacturing resources which are shared by several workstations, and/or which must perform tasks in co-operation with others.
- Device Server level: contains the description of the different services that are offered by every Device Server. At this level, we model the real Device Server modules (which run in IBM PS/2 computers, under OS/2 operating system), as well as the operating state of the plant devices that they supervise. The basic actions at this level consist of sending service request messages to the corresponding device server modules.

In addition, the BCD module uses a G2 Standard Interface (GSI) bridge towards a DAE integration

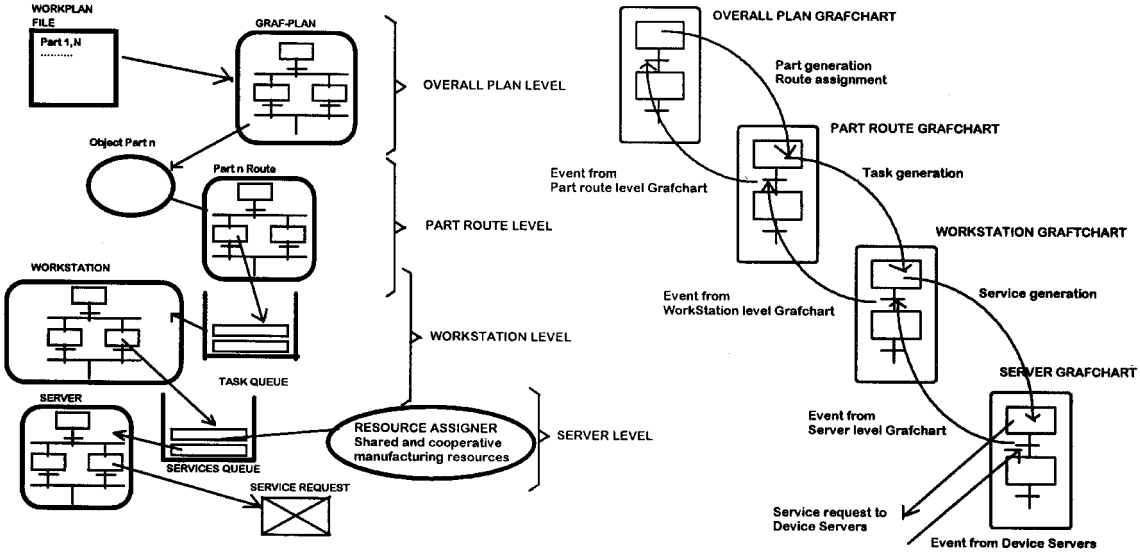

Figure 14. Architecture of BCD at both Stages I and II.

platform, in order to inter-operate with other software modules within the cell level, taking advantage of the possibilities that such an integration platform offers as far as the message exchange among distributed applications is concerned. Remote Procedure Calls (RPC) are used by this GSI bridge in order to fulfil the message exchange between the BCD module, and the other software modules of the system (DNC Manager, Tool Manager, and Device Servers), both regarding service request messages and event notification services.

At Stage II of the project, since the Dynamic Scheduling module was added to be in charge of adapting the working plan to the actual plant conditions, it has been necessary to remodel the architecture of the higher levels of the BCD module (that is, general plan, and part levels). The Dynamic Scheduling module provides the BCD with detailed information of those added value operations, and of those part palletizing and transport operations leading towards the outer part of the manufacturing system. Each one of these operations is marked by an initial scheduled starting date, which is used by the BCD in order to find the particular moment at which the operation could be started, and also to have the sequence of tasks performed in an orderly manner. In turn, generating the tasks without added value (intermediate transport, loads onto the machine, tool control, NC program control, part unload, etc.), is performed at the BCD. As the part completes its manufacturing route, intermediate tasks are generated; these tasks serve as a link among the different added value operations. These manufacturing tasks are represented graphically using a part diagram which is general enough to describe the route of any part manufactured in the system.

As far as the impact of the new on-line maintenance module on the BCD is concerned, the fact is that the BCD design is not affected at all, since the former module influences production by communicating with the Dynamic Scheduler.

At this second stage, in turn, the impact due to the incorporation of new manufacturing resources has been solved within the BCD, at the workstation and device server levels; this task has been made easier by the use of a graphical language. At both stages of the project, whenever a workstation within the BCD receives a command to execute a task, it is in charge of transforming it into the corresponding sequence of service requests for the device servers that it governs. The BCD module has at its disposal, at both stages of the project, the co-operation of two manager modules, that is, the DNC Manager and the Tool Manager, which are intermediate control elements, necessary to perform very specific tasks (NC file management, and integral tool management). At this second implementation stage, all the developments achieved at the first stage were reused, and some new server modules were added: those that correspond to the new plant devices and to the new functional modules (that is, Dynamic Scheduler and Maintenance). Few changes were introduced at the second stage; only those that we felt were necessary in order to make communication with the new functional modules easier. The use of a graphical tool has allowed for rapid implementation within the BCD of the new functionalities and the new plant equipment. Figure 15 shows the general appearance of the BCD

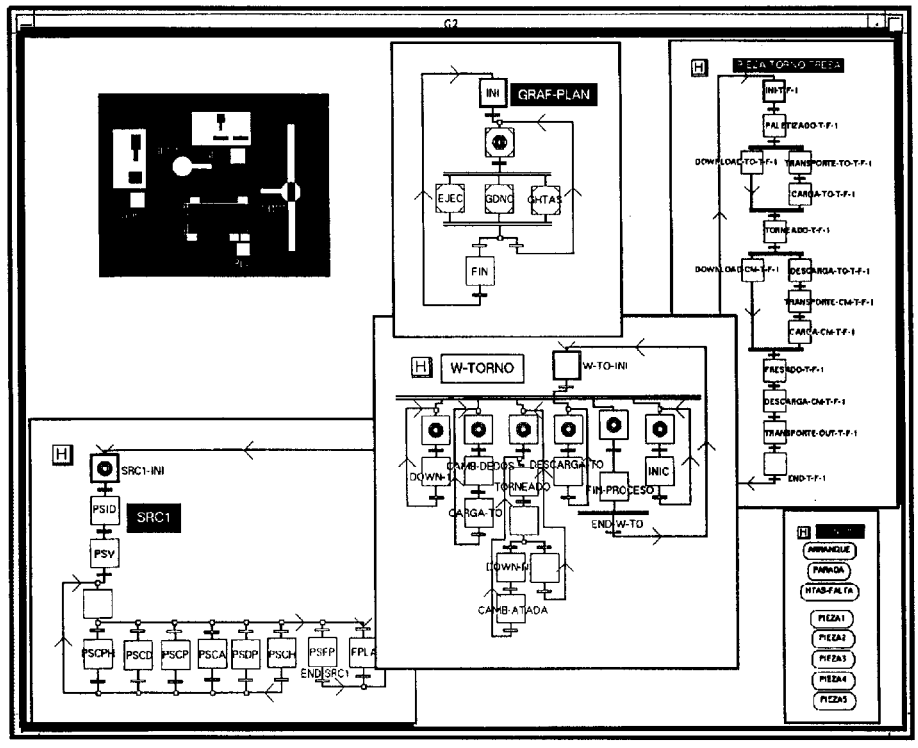

Figure 15. General appearance of the BCD user interface.

user interface, which is similar at both stages of the project.

The approach that was taken in order to design the BCD module has the features listed below:

- Client–server architecture: the interface between the BCD module and the other modules of the general software system is performed by means of service requests and by means of a continuous attention paid to the event arrival.
- Use of graphical tools, and of a prototype building environment for developing the BCD module.
- Object-oriented plant model, divided into several levels in order to describe the manufacturing plant and the characteristics of the control software system.

In order to implement the BCD module, we have used the G2 tool (by Gensym Corporation to develop real-time expert systems) and GRAFCHART (an implementation of the standard language GRAFCET IEC 848 from the University of Lund on G2). In particular, the plant model which maintains the BCD module contains, within the G2 environment, not only a description of the components of the manufacturing system which it governs, but also a description of their behaviour, and their sequencing and synchronizing elements; all this, in the form of GRAFCET diagrams implemented in the GRAFCHART tool.

4. Conclusions

By using the approach presented in this paper, important benefits are derived. The ones listed below should be highlighted:

- Graphical expression of the system control logic; this makes it easier to understand the requirements and restrictions that the manufacturing cell user/specialist may have, starting at the control system specification and design stages.
- Simulation of the cell control logic during the system specification stage. This is done through building rapid prototypes of the cell control software system.
- Graphic monitoring of control flow.
- Simplicity for configuration control, and support both for the application maintenance, and for the implementation of new improvements in the cell controller.
- Concurrent design and global implementation of flexible manufacturing cells and systems, regarding the control software and the manufacturing system.

Other aspects should also be brought out. The use of an Implementation Guide, the reference architecture established for the cell level, and the use of a development environment for real-time expert systems, such as G2, together with the GRAFCHART tool, has made the development of the Basic Control Dispatcher (BCD) module prototype much easier: its implementation cost has been reduced, and at the same time it has allowed for simpler and more efficient integration and testing of all modules in the global software system.

Acknowledgements

The authors thank the Basque Government (Gobierno Vasco, Departamento de Educación, Universidades e Investigación, project GV 145.345-0010/93), the University of the Basque Country (project 145.345-TA143/93), and the CICYT projects (references TAP94-1435-E and TAP95-0977-C02-01) for the support given to this project.

The project section which corresponds to ROBOTIKER has also received financial aid from the Basque Government (Departamento de Industria) and from the CICYT (projects ROB91-0572, TAP94-1436-E and TAP95-0977-C02-02).

References

ADIGA, S., 1993, *Object-oriented Software for Manufacturing Systems* (Chapman & Hall, London).

AEW, 1992, *Sistemas Automatizados de Producción*. (Documento de trabajo del Grupo AEW Español, 1992).

ARZEN, K. E., 1991, Sequential function charts for knowledge-based, real-time applications. In: *Proceedings of IFAC Artificial Intelligence in Real-Time Control Conference*, California, USA, pp. 91–96.

BAUMGARTNER, H., KNISCHEWSKI, K., and WIEDING, H., 1991, *CIM, Consideraciones Básicas* (SIEMENS & Marcombo Boixareu Editores, Barcelona).

BOOCH, G., 1991, *Object Oriented Design with Applications* (Benjamin/Cummings, Redwood City, CA, USA).

BOUCHER, T. O., and JAFARI, M. A., 1992, Design of a factory floor sequence controller from a high level specification. *Journal of Manufacturing Systems*, **11**, 401–417.

DTI, 1991, *Advanced Manufacturing Technology. Manufacturing Industry Applications Projects and Funding* (DTI/SERC, DTI, Departamento de Industria y Comercio del Gobierno Británico).

FAN, I. S., and SACKETT, P., 1988, An architecture for knowledge-based flexible manufacturing cell control. In *Proceedings of the Fourth International Conference on Computer Aided Production Engineering*, Edinburgh, pp. 105–112.

FRANKS, I. T., LOFTUS, M., and WOOD, N. T. A., 1990, Discrete cell control. *International Journal of Production Research*, **28**, 1623–1633.

GROSS, J. R., and RAVI KUMAR, K., 1988, Intelligent control systems. In A. Kusiak (ed.) *Artificial Intelligence. Implications for CIM* (IFS Publications, Bedford, UK).

HENDERSON, M. J., 1987, Factory communication. Siemens' experience. *The FMS Magazine*, **5**, 138–140.

IBM, *Computer Integrated Manufacturing: The IBM Experience* (Findlay Publications for IBM, Kent, UK).

IEC, 1988, *Preparation of Function Charts for Control Systems* (International Standard IEC 848, Commission Electrotechnique Internationale).

IEEE, 1984, *IEEE Guide to Software Requirements Specifications* (ANSI/IEEE Std 830-1984, The Institute of Electrical and Electronics Engineers, New York, USA).

IEEE, 1987, *IEEE Recommended Practice for Software Design Descriptions* (ANSI/IEEE Std 1016-1987, The Institute of Electrical and Electronics Engineers, New York, USA.

JAFARI, M. A., and BOUCHER, T. O., 1994, A rule-based system for generating a ladder logic control program from a high-level systems model. *Journal of Intelligent Manufacturing*, **5**, 103–120.

JOSHI, S. B., WYSK, R. A., SMITH, J. S., and PEDGEN, C. D., 1995, RapidCIM: an approach to rapid development of control software for FMS control. In *Proceedings of the 27th CIRP International Seminar on Manufacturing Systems*, 21-23 May, Ann Arbor, Michigan, pp. 51–58.

KHERMOUCH, G., 1989, How Mazak makes machines: robots, AGVs, smart tools. *Metalworking News*, **16**, No. 765, 1–33.

LÓPEZ, J. M., 1995, Metodología para el desarrollo de software de control de sistemas de fabricación. *PhD Dissertation*, Dpto. Ingeniería Mecánica, UPV.

LYDON, W. P., 1992, OOP's new role in software for control engineering. *Control Engineering*, March, 46–48.

MEYER, W., 1991, *Expert Systems in Factory Management: Knowledge-Based CIM* (Ellis Horwood/John Wiley & Sons, West Sussex, UK).

Reflex Manufacturing Systems Ltd., The University of Birmingham, BIS, DELCAM 1990, *DCC Phase 2*. (DTI).

SACKETT, P., and FAN, I. S., 1988, Goal identification for flexible manufacturing system control. *Knowledge-Based Production Management Systems* (Elsevier Science Publishers, North-Holland); pp. 311–327.

SACKETT, P., and FAN, I. S., 1989, The control of a flexible manufacturing system by short-term goal identification. *International Journal of Advanced Manufacturing Technology*, **4**, 123–143.

SAUTER, J. A., and JUDD, R. P., 1990, Applying integrated tools to the design of manufacturing systems. In: *Proceedings AUTOFACT 1990 Conference*, November 1990, Detroit, Michigen, USA, pp. 25.1–25.17.

SCHEER, A. W., and HARS, A., 1991, Enterprise-wide data modelling. The basis for integration. *Proceedings of the Seventh CIM-Europe Annual Conference*, Turin, 29-31 May, pp. 331–344.

SIEMENS, 1990, *Descripción de SICOMP: FMC Células de Fabricación Flexible* (Descripción del Producto SICOMP de SIEMENS).

STANISLAWSKI, M. P., 1990, Planificación Estratégica hacia el CIM. *Automatización Integrada & Revista de Robótica*, **50**, 48–52.

THOMAS, B. H., and MCLEAN, C., 1988, Using Grafcet to design generic controllers. In *Proceedings 1988 International Conference Computer Integrated Manufacturing*, Rensselaer Polytechnic Institute, Troy, New York, May 23-25, pp. 110–119.

WHITE, I., 1994, *Using the Booch Method. A Rational Approach* (The Bejamin/Cummings Publishing Company, Redwood City, CA, USA).

WILCZYNSKI, D., and WALLACE, D. K., 1993, OOPS in Real-time Control Applications. In: S. Adiga (ed.) *Object-oriented Software for Manufacturing Systems*, (Chapman & Hall, London).