A comparative experiment of control architectures

ELSEVIER

Computers in Industry 28 (1996) 185–193

COMPUTERS IN INDUSTRY

A comparative experiment of control architectures

Patric Timmermans [a,b,*], Laszlo Szakal [c]

[a] *Eindhoven University of Technology, Faculty of Industrial Engineering and Management Science, P.O. Box 513, 5600 MB Eindhoven, The Netherlands*

[b] *CIMRU, University College Galway, Nun's Island, Galway, Ireland*

[c] *University of Miskolc, Department of Information Technology, Miskolc–Egyetemvaros, H-3515, Miskolc, Hungary*

Received 21 September 1993; accepted 14 June 1994

Abstract

This paper presents the results of two research projects in the CIM laboratory of the Cooperative Engineering Centre (CEC) in Amsterdam. The two projects involve the design and implementation of control systems for shop floor control. Two different designs for shop floor control systems are implemented in a scale model factory for PCB assembly and test. The first system is based on a distributed control architecture, and the second system is based on a hierarchical control architecture. This paper discusses the differences between the architectures and implementations of the two control systems. The benefits and disadvantages of different architectures and implementations are discussed in terms of functionality, performance, maintenance, software reuse and required infrastructure.

Keywords: Distributed control; Hierarchical control; Control architectures; Shop floor control

1. Problem statement

An important goal in the design and implementation of shop floor control systems is to reduce complexity and increase flexibility [1,2].

Various architectures for shop floor control have been proposed in the literature and applied in industry, ranging from hierarchical oriented production planning systems to distributed systems using JIT techniques [2,3]. In addition, various models are used to implement these shop floor control systems. Literature is scarce however on practical experiences on benefits and disadvantages of various approaches. The contribution of this paper is to reduce this deficit by discussing the architectures and implementations of the shop floor control systems in the model factory of the CEC.

Two different control architectures are specified and implemented: a hierarchical control architecture and a distributed control architecture. The hierarchical architecture is characterised by a two-level scheduling and dispatching. At the highest level a work order is received and planned in a static schedule. The second level provides dispatching and detailed dynamic scheduling. Feedback is obtained by monitoring the status of the model factory. The distributed architecture is characterised by autonomous controllers for various parts of the model factory. This results in a pull oriented control of the model factory.

This paper describes the two architectures and

* Corresponding author. Email: timmermans@acm.org

SSDI 0166-3615(94)00016-J

their implementations, and makes a comparison between these implementations. We will, in contrast to [2], not discuss reference architectures. In stead we will focus on a specific application area, and the architectures and implementations in this area.

2. The laboratory

The laboratory consists of a model factory. The model factory is a miniaturized, though still complex, model of a real printed circuit board (PCB) assembly and test plant. The function of the model factory is to assemble and test pseudo PCBs. The following sections describe the product, operations, process lay-out, and hardware of the factory.

2.1. Product

The model factory produces printed circuit boards (PCBs). Each PCB consists of a board and a maximum of six components. Currently, two different types of board and three types of component are used in the model factory.

2.2. Operations

The model factory is a miniaturised model of a PCB production line. It emulates operations which are performed on real PCBs during the manufacturing process. In fact, the operations of the model factory have been derived from case studies of real PCB manufacturing facilities. These operations are:

- Screen printing: the bare PCB is positioned in the workstation, a PCB-specific screen is selected and moved into position, and a "squeegee" is reciprocated horizontally over the screen.
- Component placement: the PCB is positioned in the workstation, and components are placed on the positions according to the component-placement recipes for that product.
- Reflow and cleaning: PCBs are passed through an oven and cleaning station.
- Test and repair: the PCB is inspected to see if it contains the components in the designated position, and component and functional tests are performed. If the PCB fails, it must be routed to an off-line diagnosis and repair workstation. Upon successful repair, the PCB is routed back to the test station.

2.3. Process layout

In addition, the model factory contains some other features. Raw material to the plant is automatically supplied from a centralised Raw Material Store. The model factory can support mixed model flow production, where different types of products can be manufactured at the same time. The model factory is designed for batch production, but the batch size can vary from batch to batch, as well as from product to product. The maximum batch size in the model factory is three.

The process layout is depicted in Fig. 1. The operations are indicated by square boxes, while stockpoints are indicated by triangles. The first two stockpoints contain the two different empty board types. After the screen printing station alternative routings are possible between two component placement stations. The next station is reflow and clean-

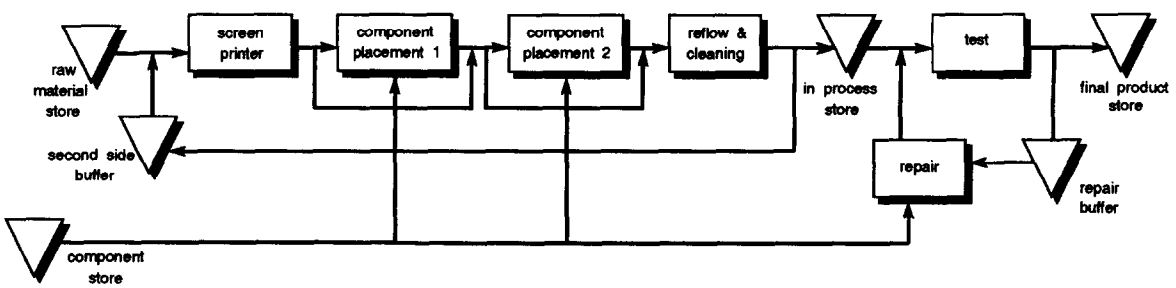

Fig. 1. Process layout of the model factory.

ing. In the in-process store a batch can be split or concatenated with other batches. The store can contain a maximum of nine products. In the test and repair cycle a maximum of one batch can reside in the buffer. Finally, nine individual products can be stored in the final-product store.

An additional feature is a loop from the in-process store to the screen-printer. This loop is necessary to manufacture double sided PCBs. These products have to pass the process twice, since only one side can be finished in one pass. The buffer in this second-side loop can contain one batch.

2.4. Hardware

All workstations in the model factory are fully automated, with the exception of the repair station, where a manual operator is required. Besides the actual operation each workstation has to manage temporary storage and retrieval of PCBs, indexing of PCBs through the process, inventory of raw materials, etc. This necessitates many sensors in the factory, in addition to solenoid stops, motors, lights, conveyers, pneumatics, etc.

Obviously, all the logical I/O signals to and from these sensors and actuators are controlled by a programmable logic controller (PLC) and its associated programs. There is also a higher level supervisory system to manage the overall production process. Hence, the implementation of the hardware consists of two levels, a PLC level and a VAX level. In summary, a considerable amount of equipment was needed to implement this I/O controller and supervisory computer system.

3. Control architectures and implementation models

Two different control architectures were specified and implemented: a hierarchical architecture and a distributed architecture. The hierarchical architecture is characterised by a two-level scheduling and dispatching. At the highest level a work order is received and planned in a static schedule. The second level provides dispatching and detailed dynamic scheduling. Feedback is obtained by monitoring the status of the model factory. Monitoring is performed on lead-times and transport times. The distributed architecture is characterised by autonomous controllers for various parts of the model factory. This results in a pull oriented control of the model factory. The last controller in the line receives a work order which is consecutively passed to the other controllers as requests for production. Both architectures have been implemented in the model factory, using the same infrastructure and model factory layout. The following sections describe the two implementations in more detail.

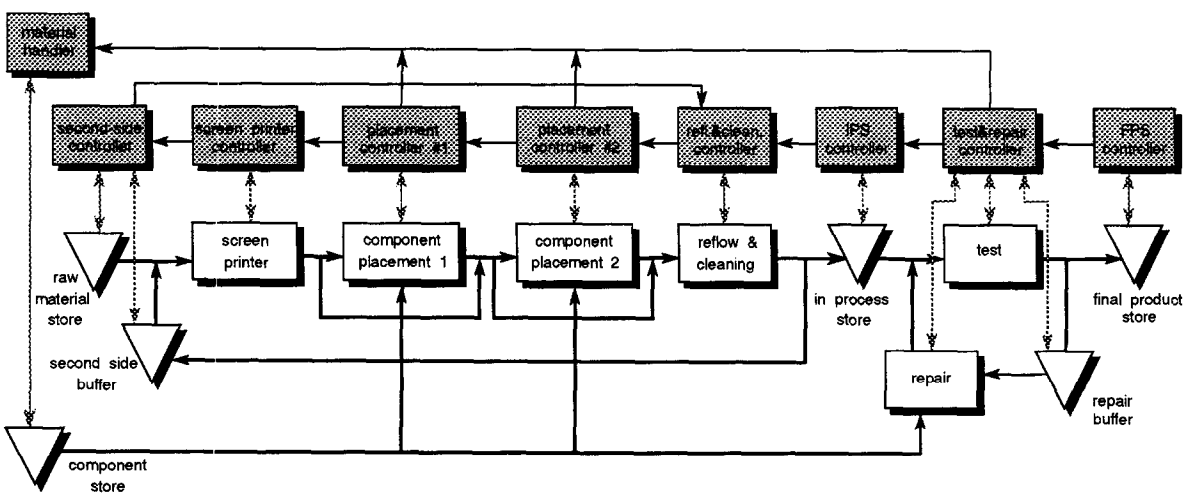

Fig. 2. Distributed control architecture.

4. The distributed control architecture

4.1. Architecture

The distributed control architecture implements a decentralised control of the process. Each operation in the process is controlled independently and very little information is taken into account about states or decisions made in other parts of the process. An operation is considered as an individual process which has one or more suppliers (the preceding process(es)), and one or more customers (the succeeding process(es)).

In a flow environment each operation has exactly one predecessor and one successor, but the production process has characteristics which hamper this approach, such as process looping, failure routings, etc. Such complexities of the process however leads to combination of certain control functions into an integral control function that controls more than one process. It necessitates a control function that does not control an operation but the product flow itself.

With this limitation in mind, nine controllers have been defined for this control system. Fig. 2 shows how these controllers actually control the product flow.

The principal functionality of the implemented control model is as follows. The last controller in line, the final product store controller, receives a work order. This controller creates a batch and a request for that batch based on Statistical Inventory Control. In the batch definition it formulates what type of products has to be produced and how many (the batch size). The request for this batch is then send to the preceding controller in line, the test and repair controller. This controller in its turn creates its own request for the same batch and forwards it to its preceding controller. This process of receiving and forwarding requests goes on until the beginning of the production line. Here, the physical batch is created by releasing empty boards from the raw material store. This physical batch is send to the requesting station, thereby eliminating the request for that batch. Hence, all requests for the batch will be eliminated when the physical batch arrives at the final product store. Here, the batch definition itself may also be removed from the information base.

4.2. Implementation

Information system modules have been defined for each autonomous controller [4]. Each controller consists of a request handler service and one or more execution manager services. The request handler manages the incoming requests for replenishment of products. The controller fulfils this request as soon as possible. If the particular workstation has a store or a buffer and can fulfil the request, it will do so. If not, the request handler forwards the request to the preceding controller and the request is propagated through the system. When the execution manager receives a batch, it performs the transformation, for example screen-printing, on the batch. Once the task is accomplished, the controller transports the batch to the succeeding controller's workstation.

The services of a module are implemented using a combination of C, SQL and a language called CIMfast. CIMfast is an event driven language that makes the callable interfaces to BASEstar much easier to use from a high level language such as C. The main functionality of the services are written in C, with SQL calls to retrieve or store data from a database, and CIMfast calls to send messages to another module's service, change a control point in the factory or react to status changes in the factory.

5. The hierarchical control architecture

The hierarchical control architecture is based on the COSIMA architecture [3] which was the name of ESPRIT Project 477. This project defined the functional structure of a production activity control system. The hierarchical control architecture of the model factory is described in this section according to the five COSIMA units (Fig. 3):

- scheduler,
- dispatcher,
- monitor,
- mover,
- producer.

5.1. Scheduler

The scheduler in the COSIMA architecture is part of the production planning—it has to plan the manpower, tooling and workstation capacity. This sched-

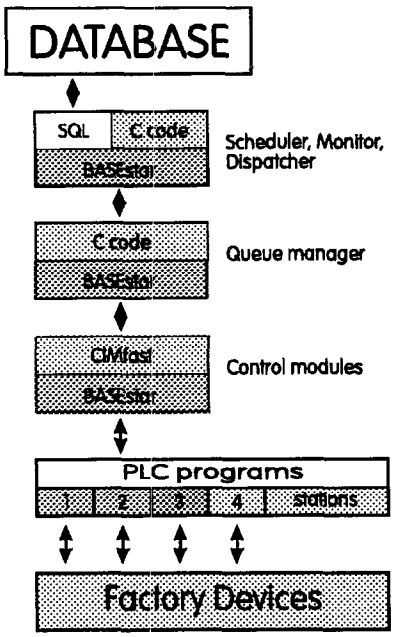

Fig. 3. Hierarchical control architecture.

uler generates a short-term detailed schedule table from the long-term strategic plans. The time scale can vary between a real-time and a daily scale. The table contains the start and finish time, the location, the transportation requests between the stations. The time values are generated according to the long-term due date data which provide the constraints for scheduling. This module is the crucial part of the system since the scheduling policy has a deterministic role in the system.

The implemented scheduler reads the customer's incoming orders from the database. The scheduler is activated on time slice basis (after every previously fixed time difference). The scheduling algorithm is a shortest processing time (SPT) which considers the actual work in process (WIP) status and the due date of the orders. The scheduling policy is simple according to the job-shop like problem in the factory. The schedule tables are written into the database and made active after the scheduler has finished the creation. Provided with feedback from the factory, this module traces the actual status of the WIP batches so the errors between the theoretical start and finish time and the real situation are corrected each time when the system is rescheduled. After a new table has been created the scheduler notifies the dispatcher to update its own internal table. The scheduler plans only the general instructions about the transformation and move requests. The table is completed by material requests in order to handle this in a similar way as the batch orders. The table created by the scheduler will be referred to later as the primary schedule table.

5.2. Dispatcher

The work basis of the COSIMA dispatcher is the schedule table created by the scheduler. The workstations start to process at the time planned in the table. The workstation and the requested material availability have been checked before the start instruction is given. If there are more than one jobs waiting for the workstation it is necessary to choose between them (on highest priority). The dispatcher is prepared to handle the unexpected bottlenecks of the factory and automatically reschedule the considered schedule table.

In the implemented system as soon as a primary schedule table is available the dispatcher loads it and generates a secondary schedule table. This schedule table resolves the move requests into elementary requests—step by step through the factory—and at proper time it sends these operation instructions to the producer level. The fulfilled job operations are reported in the database modifying the WIP entries. The dispatcher and the producer level communicate via a message queue.

5.3. Monitor

The monitor module in COSIMA collects the data on shop-floor level. It is notified about the events and data changes in the factory. The relevant information is forwarded to the modules through messages and written into a history file. For example a notification message is sent if a certain material level drops below a reorder point; an entry is generated in the history file if a workstation starts or finishes to process a job. These data can be displayed on an operator display representing the current system status giving an overview of the factory status. The history file can be processed later to generate different kind of reports and figures to support further decisions.

The monitor in the Model Factory Control system is a history file generator. It gets messages about every action which have taken place in the shop-floor. The data analysis has not been implemented yet.

5.4. Mover (material handler)

This module in COSIMA handles all the units on the shop-floor that are responsible for the material transport between the workstations. Transport device can for example be automatic guided vehicles (AGVs), robots, conveyers and human operators. The transport requests come from the monitor or dispatcher modules. The requesting processes are informed about the accomplished transport orders.

During the implementation the mover functionalities were divided into two parts: transferring the materials between the stations and receiving the requests. The first operation is done by the producer modules controlling the factory devices including the raw material stocks and conveyers. The material requests are handled in a similar way as the product requests—these requests are primary schedule table entries.

5.5. Producer (process control)

The final step in the COSIMA control system is to give low level instructions to the workstations. These devices can be CNC machines, assembly workstations, other specialized machines or human operators. These machines have their own individual control systems. When a dispatcher sends a start instruction to the producer then the producer is responsible to set up the workstations with the necessary data. In case of a CNC machine this data consists of a requested tool, a part load command and a NC program. When these preparations have completed, the machine is ready to start the operation. As soon as the transformation process has finished the part is unloaded. Thus the three basic function of the producer are: loading, transformation and unloading. The implemented producer is divided into two levels. The upper level module is called application level and the bottom level is the station control level. This partitioning separates the real-time control from the database access, queuing and communication tasks.

5.5.1. Application level

The main parts are a queue manager and two communication handlers between the dispatcher and the control level. The incoming requests are put in FIFO queues which are related to the stations. The queue manager traces the queue statuses and forwards the instructions to the control level when the stations are available. The bottom level signals provide the actual information about the conditions in the shop-floor and the batch statuses. These data are administered and sent back to the dispatcher. The workstation and product specific information are inquired from the database. This is the most bottom part in the system which needs high level database access.

5.5.2. Control level

This module directly communicates with the physical devices (more precisely with the PLC programs). The communication is completely synchronized and all the statuses are mirrored via the shop-floor data points. The modules in this level are specific for the controlled station.

6. Technology used for implementation

The control system is divided into two main parts: real-time and time- sharing processes. The operating system (VAX/VMS) on which platform the implementation took place is a time sharing category therefore the immediate responses to the unusual events were necessarily implemented in the real-time PLC environment. Today, PLCs are sophisticated devices and considered as a complete computer with CPU, memory, application programs and network connection protocols. The PLC programming methods are very different from the classical programming languages and have vendor dependent specialities. The two implemented control systems were different in their characteristics. The decentralized system PLC programs are sophisticated, almost fully operational programs so the application interface is comparatively simple. Every logical unit called "zone" has a "start" command and a few status flags (ready, busy, done). The product and batch characteristics and routes are fixed in the PLC program variables and algorithms. This makes the PLC

program package very robust and complicated to maintain.

The hierarchical system was designed in a different way. The amount of logical units are increased and simplified (modeled as finite state automats), the parameters and the algorithms are put on higher level. Thus the user interface is more complex but the PLC programs are simpler and easier to maintain than in the previous case.

In the VAX/VMS environment the shop-floor applications can be developed using the BASEstar software tool. This software package is an operating system extension especially for the factory management applications. The services provided by BASEstar are: data management, configuration, application control, application synchronization and network communication. A data operation means a transparent access mechanism, namely a logical point (BASEstar high level variable)–physical point–PLC memory chain. The applications read/write the logical points after the proper configuration has been defined. This provides a very flexible architecture. When the device configuration should change, the programs are capable to run without any correction if the environment description database has modified properly. The device specific operations are performed by the specific device connection management part of the BASEstar.

To make the programmer's work easier a special tool is available to develop the applications. This is a special event driven language called CIMfast. The events are logical point values or message queue status changes. The programmer can use this utility in two distinctive ways:

1. stand-alone mode: every action is described inside this special language;
2. embedded mode: the event handler system is described in CIMfast and the algorithms and other transactions are programmed in a classical high level language like C, PASCAL etc. The modules have the possibility to use other standard interfaces: SQL for database access or graphical user interfaces for the front-end applications.

The distributed control system (Fig. 4) is divided into two levels: the PLC programs and the control processes. In the latter case the modules are embedded CIMfast applications combined with relational database access via a SQL interface programmed in C. With the BASEstar application programmer interface these programs also have read/write capabilities on the PLC program control logical points. The communication method is exchanging messages between each other and write the status flags into the database entries.

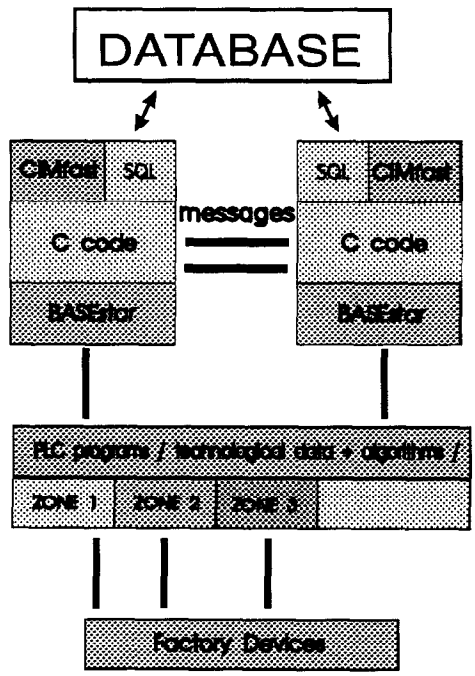

Fig. 4. Distributed control architecture.

The hierarchical control system (Fig. 3) is built up from three different kinds of layer. The bottom level is the PLC program set. The second level consists of a number of stand-alone CIMfast programs to have a full control over the PLC program modules. The required parameters and commands are received in message bodies. The higher level modules are C programs with database access and BASEstar communication interfaces. The factory dependent algorithms and the general planner functions are separated into different levels to increase the flexibility and the device independence.

The modules in both systems share a central database. The data is permanently retrieved and modified. Therefore, there is a need for a method which helps to avoid the write collisions and the deadlock situations between the different modules. The database definitions contain entities, attributes and consistency constraints.

The database in the distributed control system stores the actual information about the batches in the factory and only a minimum knowledge about the factory zone statuses. It is not needed to provide any minute details to describe the physical factory layout or product definitions because these parameters are programmed on PLC level.

The module domains have dynamic behaviour: the database access constraints depend on WIP item occurrences and they are changing during the production process according to the accomplished transformations. If a batch passes a logical unit in the factory the system modifies the "batch owner" entry in the related record. The simplification of the information storage—fixing the product routes, the transformation processes—makes the database maintainable and the application module algorithms simple. The disadvantage of the current implementation is the lack of certain flexibility. It is very difficult to introduce a new product type or to use a new workstation. Both type of modification have consequences on PLC and application level. The PLC programming language and the size of this program requires a specialist and a lot of effort to achieve the working system again.

The hierarchical control system stores static data about the products. These data define the route followed during the production process, the parameter values required for a single production step and the factory physical layout from which the possible routes of a batch can be determined. Therefore, changes in the shop-floor configuration means a few changes in the static database and the modules can remain unmodified. The algorithms are general, the parameters are retrieved from the database. A disadvantage is that the uncomplicated PLC programs demand more sophisticated information system. The database access constraints are independent from the data occurrences thus the domains are static.

7. Comparison

Both architectures have been implemented in the model factory at the CEC, using the same computer equipment and application integration platform. The characteristics, advantages and disadvantages of both implementations are quite different however.

The first control system implemented was the distributed architecture. This system was chosen to be the first implementation because of the simplicity of the architecture and available knowledge of the tools to be used. After the global control architecture was defined, as described in Section 4.1, each of the controllers was designed and implemented using the method of modular design of information systems as described in [5–7]. This method allows to design, implement and test an individual controller, while disregarding the implementation of other controllers.

The advantage of this implementation was the relatively fast implementation of the applications that constitute the controllers. Moreover, it is easy to modify the system and extend it by one or more modules. A large part of the software was reused in different controllers because of the commonality of the required services. Furthermore, the control system is easy to reconfigure or to adapt.

The disadvantage of this architecture is the difficulty of global optimalisation of production planning. This disadvantage is reduced by the fast response of the control system to events in the factory.

A problem we faced, which was not the consequence of this particular architecture, was the complexity of the PLC software. Certain functionality that was built in the PLC software restricted the functionality of the application programs. The PLC software was programmed by the manufacturer of the model factory. Because of time restrictions, it was impossible to reprogram this PLC software. This caused some problems in coding the individual controllers, and as a result the controllers are more different than initially foreseen.

Next, the hierarchical architecture was designed and implemented, and the PLC program was simultaneously updated. The characteristics of the hierarchical control system are: (1) different levels of aggregation of data, and (2) supervisory control of one level over the other. The advantage of this architecture is the overall control of the factory by a single controller. This allows optimalisation of the factory workload. A disadvantage is the high number of processes that control the factory. This high number requires considerable computational power, and increases the number of communication links between processes. As a result the control system may be more error prone. Furthermore, it is difficult to fore-

see the future consequences for various controllers of extending the factory with new functionality.

In addition, we faced some other problems in the design and implementation of the control systems that are not the consequence of a particular implementation. A problem we mentioned earlier is the limitation in the control applications due to functionality that was already build in the PLC software, or even in the physical hardware of the factory. An important conclusion is therefore that both hardware and software have to taken into consideration when designing a modular factory.

8. Conclusions

In this paper we described the design and implementation of two architectures for the control of a scale model factory for PCB assembly and test. We made a comparison in terms of the characteristics of the control architectures and their implementations. The advantages and disadvantages of the hierarchical and distributed architecture correspond to these described in [8].

In the CIM laboratory, an emphasis was put on the modular design of information systems. This includes extensibility and adaptability of the information system during operation. The implementation of the distributed control architecture satisfies the criteria for modularity in particular. This is the result of applying the method for modular design very strictly [7].

The main recommendations from the research at the CEC sofar are:

- Choose an appropriate architecture. The most appropriate architecture depends on the requirements of a particular situation. Consider in advance what the consequences of the architecture will be in terms of extensibility, modifiability, required infrastructure, available software, etc.
- Develop independent modules. Consider that a module does not consist of software only, but includes the design of physical hardware.
- Define what you mean be modularity. Usually modularity is thought of as composability only. More important is the possibility of decomposing, reconfiguring and adapting the current system.
- Consider technical limits. There are still a number of technical limits to what can be implemented. This holds in particular for the availability of standards for communication, protocols and operating systems.
- Consider available software. Methods for system design do not emphasize the availability of commercial software. This will often result in expensive 'design from scratch'. An important limitation, also encountered in the model factory, is the lack of availability of software.

A number of areas need further research. In particular we emphasize the need for research in the relation between control architectures, fault-tolerance and reliability. More research is also needed in the definition of standards for communication, protocols and operating systems for different types of manufacturing systems. Finally, we would like to emphasize the need for available commercial software and methods to select and implement this software.

References

[1] A. Bauer, R. Bowden and J. Browne, *Shop Floor Control Systems: From Design to Implementation*, Chapman and Hall, London, 1991.

[2] F.P.M. Biemans, *Manufacturing Planning and Control*, Elsevier, Amsterdam, 1990.

[3] J. Duggan, A design tool for production activity control, Ph.D. Thesis, University College Galway, CIM Research Unit, Galway, Ireland, 1990.

[4] C. Koopmans, Exchangeability of CIM components: Implementation of an information system, M.Sc. Thesis, Eindhoven University of Technology, Digital Cooperative Engineering Centre, Amsterdam, June 1992.

[5] H.J. Pels, *Integrated Information Bases* (in Dutch), Stenfert Kroese B.V., Leiden; dissertation Eindhoven University of Technology, 1988.

[6] P. Timmermans and H.J. "A case study on modular decomposition of a CIM system", *Proc. IFIP APMS'90 Conf.*, Helsinki, Finland, 20–22 August, 1990.

[7] P. Timmermans, and C. Kearns, "Modular design of a scale model factory", in: J.C. Wortmann (ed.), *Proc. Int. Conf. on Integration in Production Management*, Elsevier, Amsterdam, 1992.

[8] D.M. Dilts, N.P. Boyd and H.H. Whorms, "The evolution of control architectures for automated manufacturing systems", *J. Manuf. Syst.* 10(1) (1991).