Intelligent Field Devices and Field Buses: Impact On Applications Design Methodology

Carlos Cardeira
Grupo de Controlo, Automaçao e Robotica
Instituto Superior Tecnico-DEM-Sistemas
Avenida Rovisco Pais,
P-1096 Lisbon
PORTUGAL

Françoise Simonot-Lion
C.N.R.S. - Centre de Recherches en Informatique
de Nancy, URA 262-ENSEM
2, avenue de la Forêt de Haye,
F-54516 Vandoeuvre lès Nancy
FRANCE

Mireille Bayart
C.N.R.S. - Laboratoire d'Automatique et d'Informatique
Industrielle de Lille, URA 1440
Bâtiment P2-Cité scientifique
59650 Villeneuve d'Ascq Cedex
FRANCE

Abstract : The paper sets the problem of the design step in the context of time constrained systems, presents a modeling of this activity, and proposes a classification of scheduling algorithms in order to support it.
Keywords : design process, real time system, validation, time constraints, scheduling algorithm

Carlos Cardeira (M.Sc.'90 , Ph.D' 94) received the M.Sc. degree from the Universidade Tecnica de Lisboa, Portugal and the Ph.D degree from the Institut National Polytechnique de Lorraine, France. In 1991 he was a researcher at the Centre de Recherche en Informatique de Nancy. Currently he is a Professor at the Instituto Superior Tecnico in Lisbon. His research interests include Real-Time Systems, Scheduling and Local-Area Networks.

Françoise Simonot-Lion was born in France. She received the M.Sc. degree in Applied Mathematics in 1971 and the Ph.D degree in Computer Science from the University of Nancy 1 in 1981, respectively.
Since 1974 she has been teaching Computer Science , first at the University of Nancy 1, and then at the Institut Polytechnique de Lorraine, Nancy. She is currently member of the Centre de Recherches en Informatique de Nancy, a Laboratory associated with C.N.R.S. in France.
Her current research includes the development of industrial production systems, particularly, the modelling and validation of distributed real-time applications (time constraints' specification, validation activities).

Mireille Bayart (Engineer '82, Ph.D' 88) received the French National Authorization to supervise Research (HDR) in 1994. She is Professor at Lille University (France) and works in the Laboratoire d'Automatique et d'Informatique Industrielle de Lille (LAIL). Her research interests are in smart instrumentation, fault diagnosis and isolation algorithms and distributed intelligent automated production systems.

1. Introduction

An automated production system is composed of three entities : the physical process equipment, the human operator(s) and the computer-based control system. We will focus on the design of the latter one.

First, the digital technology has extended; this comes with a favourable price to performance ratio for the computer based components and with the growth of their reliability. This technological evolution is particularly significant for sensors and actuators and this leads to the concept of intelligent devices communicating by means of networks (field buses). On the other hand, the goals of the control system are extended to the supervision, maintenance, quality management, and other functions [Mor 94]. So the control system becomes more complex and sophisticated and is supported by several heterogeneous computers (PLC, sensors, actuators, ...) connected by means of different networks.

Moreover, because of the context of their use and of economic considerations, these control systems are subject to stringent constraints. So their design consists first in defining an eligible system and secondly in producing the best one

according to one criterion or several criteria. .– Next, we identify the main characteristics of these systems, then we analyze the development process. Finally we present a model to support some activities of the design step.

2. Characteristics of Automated Production Systems

As we assumed in the previous part, the specific characteristics of the systems under study are :

a) they are composed of heterogeneous components

- different computers and operating systems
- different communication networks and protocols

b) the set of functions to be handled by the systems are distributed on the system components,

c) the end users requirements are expressed in terms of

- safety (men, physical equipment, environment, ...)
- quality of products, quality of services
- productivity, ...

and more generally in terms of dependability constraints, in particular, time constraints. In this paper we will focus on the latter ones.

Furthermore the development process of such systems must observe cost restrictions. So automated production systems are real-time applications distributed on heterogeneous components and the way how they are built is relevant to the design of predictable systems. Hence the keywords are constraints expression, validation and verification, optimization.

3. The Design Step of Distributed Control Systems

Figure 1 represents a model of the development process for automated production systems [Cal 92], [Ccg 92], [Sim 92]. We study specially the control system.

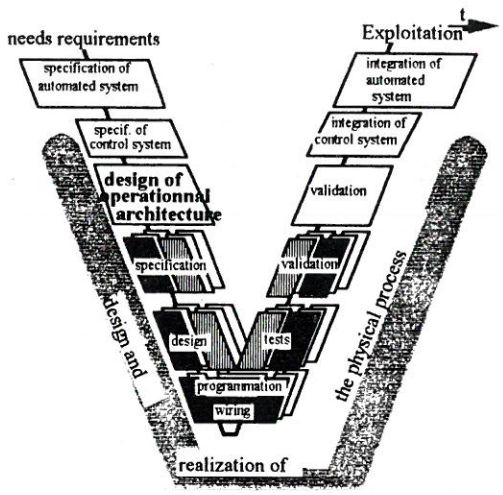

Figure 1 Model of the Development Process

We suppose here that the control systems specification may proceed irrespective of the distribution problem. At this step some properties can soon be proved as correctness of the specification. We must assume that at the integration step we obtain a control system with the same properties. So we must make sure that :

- first, the partitioning of the specified application into intercommunicating modules and the allocation of the modules to computing assets and communication systems will satisfy different constraints,
- then, the realization will still observe these constraints.

For economic reasons it is better to assume validation before the realization of the system. So the design step is the key stage in the development process and we define it as a set of elementary activities which must be executed in order to construct the solution [Sim 95]. Figure 2 shows the design process.

To analyze this step, we need three points of view on the control system,which are briefly defined here:

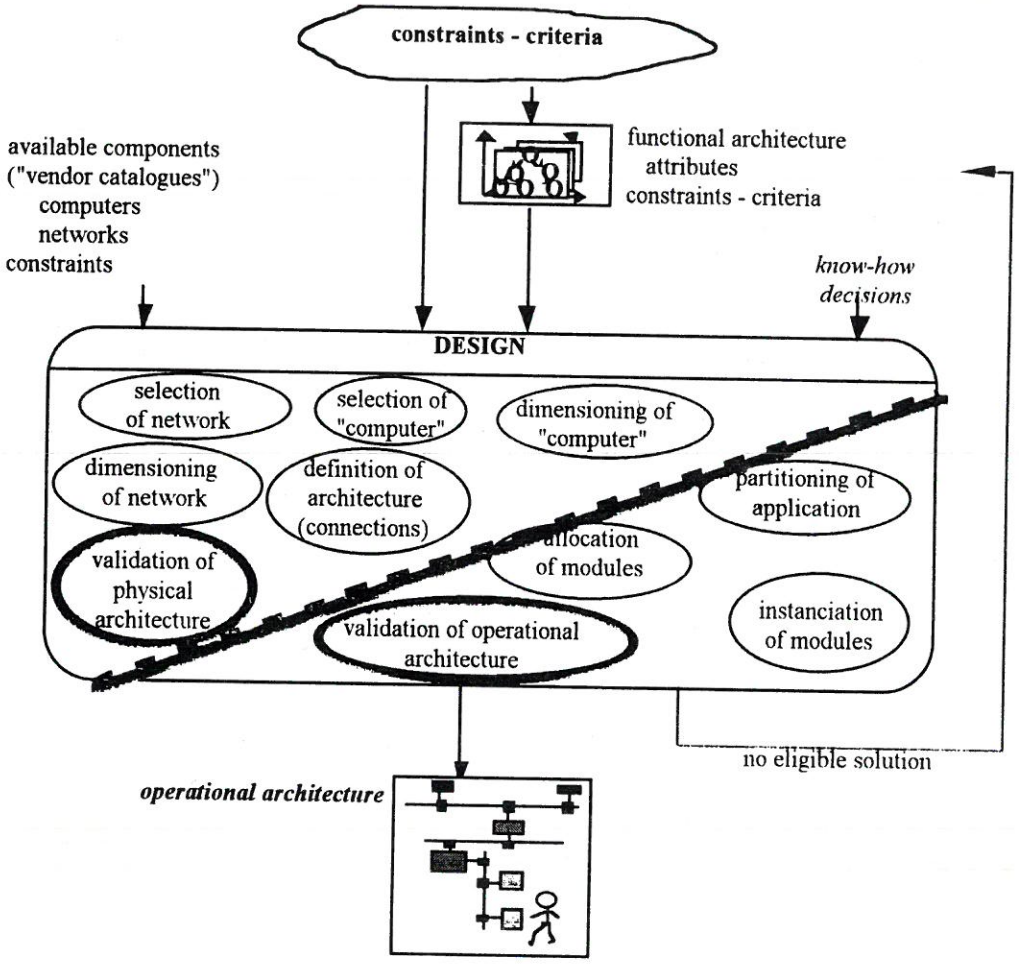

Figure 2. The Design Process Activities

• The functional architecture is the result of the specification stage expressed in a formal way ; this model is built disregarding the choice of computers and networks, and the distribution problem. It describes :

- the elementary functions that a control system has to assume (we call them "atoms"),
- the set of data exchanged between functions (data flows),
- the behaviour of the set of functions (control flows).

Each of these elements is characterized by attributes, eventually subject to constraints and/or considered as optimization criteria. A model for describing atoms is proposed in [Bay 95].

• The physical architecture is composed of :

- a set of computers (PLC, sensors, actuators, computers, ...) with their operating systems,
- a set of communication networks, with their protocols,
- the definition of the connection of different computers on the networks.

Each entity of this architecture is characterized by the provided resources (memory size, simultaneously allowed connections, ...) and by temporal performances. This information is used to perform the elementary activities of the design stage.

- The operational architecture is set up by the mapping of different elements of the functional architecture onto a physical architecture. The optimal and validated operational architecture is the result of the design stage.

The elementary activities which take place along the design stage are relevant to three different classes :

- selection and dimensioning of physical components,
- partitioning of the set of intercommunicating atoms and allocation of different parts to computing elements and networks,
- validation of the result after executing each elementary activity ; that is to check on this result meets the required properties.

The main benefit of this analysis is in pointing out that, at the design step, we must take into account both :

- executive mechanisms and services provided by a physical architecture, in qualitative and quantitative manners,and
- needs and constraints expressed in the functional architecture.

For example, we must study the eventually applied constraint, or a scheduling algorithm in regard of deadline constraints or bounded lifetime of data as illustrated in the next part.

4. Classification of Scheduling Algorithms

To guarantee that the application timing constraints will always be met, one must do a pre-runtime schedulability analysis of the system. From the scheduling point of view, distributed real-time systems impose much more problems and constraints than centralized systems do. Whereas in centralized monoprocessor based systems the main problem is to schedule tasks on a single processor, in distributed systems the problem is not only to decide on "when" executing a task, but also on "where" should the task be executed. It is up to the global scheduler to decide which is (are) the most suitable node (or nodes) to execute a given task and to give a warranty that it will be able to schedule all the tasks and network traffic, meeting all the application timing constraints [Cas 88].

We make a classification of the existing work about scheduling algorithms according to the following criteria: temporal aspects, priorities, preemption, end of tasks execution, precedence constraints, resource constraints, scope of the algorithm (global versus local), execution of the algorithm (on-line versus off-line), type of hardware (monoprocessor versus multiprocessor), optimization criteria, fault-tolerance, and the resolution technique used by the algorithm [Car 94b].

4.1 Temporal Aspects

Existing task scheduling algorithms consider tasks with the following timing constraints [Xu 91], [Car 94b]:

- ready time (date before which it is impossible to start a task),
- deadline (date on which the execution of the task must be finished),
- execution time (time necessary for one task to be executed by the processor without preemption),
- maximum execution time (maximum interval between the start time and the finish time of the task).

The previous task features may be applied to messages without change. Messages may be transmitted after a given ready time and/or before a given deadline, as well as they have a transmission time (similar to task execution time) which is the time they take to be transmitted without interference through the medium. Critical messages must be transmitted within a maximum transmission time which is the

maximum effective time for transmitting the message, considering all the interferences that other messages may provoke.

The activation of a task may be:

- at a fixed date (for instance "at 8:00 PM ..."),
- periodic (for instance "every 20 ms ... "),
- sporadic (for instance "when temperature is higher than 90° ...").

4.2 Priorities

One way to scheduling tasks is to assign priorities to them (statically or dynamically). Tasks are then scheduled obeying to previously established priorities.

4.3 Preemption

Preemption is the operation of interrupting a task to execute an(other) one(s), and of resuming the execution of the first task some time afterwards. Preemptive scheduling has better performance (it enhances the processor utilization) than non-preemptive scheduling. However, if tasks have resource constraints, preemption may lead to deadlocks or priority inversions [Raj 91].

4.4 End of Tasks Execution

Most of the existing task scheduling algorithms consider that a task has to be executed till the end which sounds logic. But there are some tasks of which results are produced by successive iterations, and may have a large (or infinite) execution time. However, after some iterations the solution is quite good and there is no need for continuing to execute the task. Examples of such tasks are, for instance, the classical example of a task calculating the exact π value or, more realistically, a pattern recognition task of which result is more and more accurate when its execution time increases. These tasks are called incremental tasks. Algorithms for scheduling this type of tasks minimize the overall error produced by an application composed of a set of incremental tasks.

4.5 Precedence Constraints

Up to now we have only dealt with constraints of a task, considered isolated from the others. However, in a distributed system tasks co-operate with each other to achieve an overall goal. By this we mean that tasks are not independent, they must be executed in a predefined order, they share critical resources, etc., i.e. they have precedence, resource and allocation constraints. Is it the same for messages ?

4.6 Resource Constraints

To be executed, tasks may require a set of resources (processors, memory, I/O ports, etc.). Similarly, to be transmitted, messages require a set of resources (physical medium, memory, etc.).

Resource constraints hardly complicate the scheduling problem, leading to undesirable situations like deadlocks [Raj 91]. In the case of tasks, a deadlock arises, for instance, when a task uses a resource R1 and requires a resource R2 which is being used by another task requiring the resource R1: the two tasks are blocking each other.

4.7 Monoprocessor Versus Multiprocessor

A node may be based on a monoprocessor or a multiprocessor architecture, and a task located on this node may be executed on any processor.

4.8 Local Scheduling Versus Global Scheduling

A real-time system may be composed of one node or several interconnected nodes. Task scheduling algorithms may be local or global. Local scheduling concerns scheduling on just one node. When nodes communicate with each other, scheduling becomes global and schedulers must answer such questions as where to execute a given task and when to do it (note that local schedulers only answer the latter question).

4.9 On-line Versus Off-line

An on-line scheduling algorithm is executed whenever a task execution request is recorded.

An off-line algorithm defines the schedule before run-time, using the information specifying the future behaviour of the tasks. Off-line scheduling is well-suited to periodic task scheduling, as on-line scheduling algorithms are more suited to handling aperiodic requests.

4.10 Optimization Criteria

The quality of the result of a scheduling algorithm is an important feature. There are algorithms which guarantee the constraints satisfaction and an optimal proof (an algorithm is optimal in the sense that if the algorithm fails no other algorithm would be able to provide a schedule satisfying the constraints). Moreover, some algorithms may still optimize some criteria. Of course, some of these criteria may be important for message scheduling too.

Let us see some of the most frequently used criteria and translate their meaning when talking about message scheduling.

a) Minimization of the schedule length

Some task scheduling algorithms optimize the schedule length, i. e. the elapsed time between the activation of the first executed task and the end of the last task. The main idea is to use the processor during a minimal time period.

b) Load Sharing

An optimization criterion of global scheduling is load sharing among the nodes. This is an important criterion for systems where migration is allowed.

c) Minimization of the communication load

To minimize the communication load is a criterion that tends to bring together tasks that interact a lot. As communication among tasks placed on the same processor is faster than communication among different nodes, this is a criterion that, in principle, tends to increase the overall system performance.

d) Minimization of the number of tasks that do not meet the timing constraints

If the scheduler cannot meet the timing constraints then one criterion to optimize may be the number of tasks that do not meet their timing constraints. Of course, this is a criterion for soft real-time systems, because it is assumed that some constraints will not be met.

4.11 Fault -tolerance

In a distributed system, fault tolerance may be obtained by the execution of several copies of the same task in different processors. The result is chosen by vote or consensus. Very few algorithms consider this constraint, in spite of its major importance in real-time systems.

4.12 Resolution Techniques for Finding A Solution

Most of the existing techniques used to elaborate exact or approximate solutions for task scheduling problems may still be used by message scheduling algorithms. Among the existing techniques, we may cite graph theoretical, heuristics [Zha 87], [Ram 89], queueing theory [Shi 89], simulated annealing [Aar 88], [Tin 92], genetic algorithms [Hol 85], [Tal 91], fuzzy logic [Zad 65], [Ish 92] and neural networks [Car 94a].

A complete presentation of all the algorithms would be out of the scope of this paper. However the reader is invited to the reference [Car 94b], where this classification is organized in a table containing the constraints and the existing algorithms for each case.

5. Conclusions

This paper has set the problem of the design step in the context of computer based systems, composed of heterogeneous components (sensors, actuators, computers, ...) and subject to stringent time constraints. It demonstrates that in order to accomplish some of its elementary activities (choice of components, dimensioning, partitioning, ...), it is necessary to study executive mechanisms in a quantitative manner in regard of needs and constraints expressed on the functional architecture. To illustrate this fact the problem of tasks scheduling is presented. For the scheduling viewpoint, we developed a

systematic approach to choosing a given scheduling strategy. We have proposed a classification of scheduling algorithms as a function of the tasks constraints of the system, to guide the choice of a scheduling algorithm (or of a set of scheduling algorithms) in function of the system features.

We may conclude that the use of such a taxonomy can be in the base of an interactive tool to automatically guide the design of an application, because in spite of the last 40 years when scheduling problems have been studied , it is nowadays still impossible to propose a strategy to handle all the constraints of every system.

BIBLIOGRAPHY

[Aar 88] AARTS, E. and HORST, J., **Simulated Annealing and Boltzmann Machines**, WILEY-INTERSCIENCE, New York, 1988.

[Bay 95] BAYART, M., **Architectures des Systèmes Automatisés de Production à Intelligence Distribuée**, Actes des Journées d'étude SAPID, Paris, France, May 1995.

[Cal 92] CALVEZ, J. P., **Embedded Real-Time Systems. A Specification and Design Methodology**, JOHN WILEY, December 1992.

[Car 94a] CARDEIRA, C. and MAMMERI, Z., **Neural Networks for Satisfying Real-Time Task Constraints**, Proceedings of SPRANN'94 IMACS Symposium on Signal Processing, Robotics and Neural Networks, Lille, April 1994.

[Car 94b] CARDEIRA, C. and MAMMERI, Z., **Ordonnancement de tâches dans les systèmes, temps réel et répartis: Algorithmes et critères de classification,** AUTOMATIQUE, PRODUCTIQUE ET INFORMATIQUE INDUSTRIELLE , 27 (4), septembre 1994, pp. 353-384.

[Cas 88] CASAVANT, T. and KUHL, J., A **Taxonomy of Scheduling in General-Purpose Distributed Computing Systems**, IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, 14 (2), February 1988, pp. 141-154.

[Ccg 92] **CCGA, Projet de Norme Française : Base PTA-version A1**, 1992

[Hol 85] HOLLAND, J., **Adaptation in Natural and Artificial Systems**, UNIVERSITY OF MICHIGAN PRESS, 1975.

[Ish 92] ISHII, H., TADA, M. and MASUDA, T., **Two Scheduling Problems with Fuzzy Due-Dates**, FUZZY SETS AND SYSTEMS, 46, 1992, pp. 339-347.

[Mor 94] MOREL, G., IUNG, B., GALARA, D. and RUSSO, F., **Prototyping a Sub-concept of Computer Integrated Manufacturing Engineering : The Integrated Control, Maintenance and Technical Management Systems**, ISRAM'94, USA, August 1994.

[Raj 91] RAJKUMAR, R., **Synchronization in Real-Time Systems: A Priority Inheritance Approach**, The Kluwer International Series in Engineering and Computer Science, KLUWER ACADEMIC PUBLISHERS, Boston/ Dordrecht/London, 1991.

[Ram 89] RAMAMRITHAN,K., STANKOVIC, J. and ZHAO, W., **Distributing Scheduling of Tasks with Deadlines and Resource Requirements**, IEEE TRANSACTIONS ON COMPUTERS, 38 (8), August 1989, pp. 1110-1123.

[Shi 89] SHIN, K. and CHANG, Y., **Load Sharing in Distributed Real-Time Systems with State Change**

Broadcasts, IEEE TRANSACTIONS ON COMPUTERS, 38 (8), August 1989, pp. 1124-1142.

[Sim 92] SIMONOT-LION, F. and VERLINDE, C., **Importance d'un cadre de référence dans la mise en place d'une démarche de développement d'un système automatisé de production**, Actes de la Conférence "Automatisation Industrielle", Vol. I, Montréal, Canada, juin 1992.

[Sim 95] SIMONOT-LION, F., THOMESSE, J.P., BAYART, M. and STA-ROSWIECKI, M., **Dependable Distributed Computer Control Systems : Analysis of the Design Step Activities**, Proceedings of IFAC - DCCS'95, Toulouse-Blagnac, France, September 1995.

[Tal 91] TALBI, E. and BASSIERE, P., A **Parallel Genetic Algorithm for the Graph Partitioning Problem**, ACM INTERNATIONAL CONFERENCE ON SUPERCOMPUTING, Cologne, Germany, June 1991.

[Tin 92] TINDELL, K., BURNS, A. and WELLINGS, A., **Allocating Hard Real-Time Tasks: An NP-Hard Problem Made Easy**, THE JOURNAL OF REAL-TIME SYSTEMS , 4, 1992, pp. 145-165.

[Xu 91] XU, J. and PARNAS, D., **On Satisfying Timing Constraints in Hard Real-Time Systems**, SOFTWARE ENGINEERING NOTES, 16 (5) December 1991, pp. 132-146.

[Zad 65] ZADEH, L., **Fuzzy Sets**, INFORMATION AND CONTROL, 8, August 1965, pp. 338-353.

[Zha 87] ZHAO, W., RAMAMRITHAN, K. and STANKOVIC, J., **Scheduling Tasks with Resource Requirements in Hard Real-Time Systems**, IEEE TRANSACTIONS ON COMPUTERS, 13 (5), May 1987, pp. 564-577.