Petri Nets and Flexible Manufacturing

Manuel Silva
Departamento de Ingeniería Eléctrica e Informática
C/ María de Luna,3 (ACTUR)
E-50015 ZARAGOZA

Robert Valette
Lab. Automatique et Analyse des Systemes
7, Av du Colonel Roche
F-31077 TOULOUSE

January 24, 2005

Abstract

The aim of this survey is to introduce Computer science/Petri nets specialists to the basic system level issues brought up by the development of Flexible Manufacturing and how Petri nets are used to aid the production engineers in their work. After some terminology concerning production engineering, the hierarchical decision and control level is briefly reviewed. Finally, the role and the presence of nets in CAE (Computer Aided Engineering) and in CAM (Computer Aided Manufacturing) for FMSs (Flexible Manufacturing Systems) are considered. From the design point of view, the use of nets have many advantages in modeling, qualitative analysis, performance evaluation and code generation. From the control of the plant perspective, scheduling on nets models, the coordination of the plant (global and partial) and the presence of nets in the local control level are discussed. Petri nets theory presents a relatively mature body, nevertheless efficient performance evaluation (even for P/T net models) and qualitative analysis of high-level net models still requires important developments. The Babel-Tower-impression that the Petri net newcomer may have looking at the myriad of so-called high-level-nets formalisms may initially produce some kind of rejection. In any case, it is our opinion that Petri nets appears as a key formalism to describe, analyze and implement the control of FMSs. The merging of Petri nets and knowledge based techniques seems to be very promising to deal with large complex discrete event dynamic systems such as FMSs.

1 Introduction.

1.1 Foreword.

Manufacturing systems and more generally factory automation is probably one of the oldest application domains of Petri net theory. We remind the reader that the title of Hack's master thesis is *Analysis of production schemata by Petri nets* and that it was presented in February 1972. Eighteen years ago! However, it is not the best known domain among the Petri net community, probably because people working with Petri nets are generally more familiar with computers than with screw manufacturing.

The aim of this survey is to introduce to the non specialist the basic issues brought up by the development of Flexible Manufacturing Systems and how Petri nets are used to aid the production engineers in their work.

After presenting some vocabulary, the basic motivations of the Petri net use are explained. Then Petri net applications are detailed for each sub-domain (modeling of manufacturing system, qualitative analysis, performance evaluation, scheduling and control implementation) and at two control levels (coordination and local control).

Finally, some papers presenting works about the application of Petri nets in Factory Automation are listed. This list is obviously not complete and any complementary information and update sent to the authors will be appreciated.

1.2 Some vocabulary.

1.2.1 What is a Manufacturing System?

It is a factory where the materials which are handled are mainly composed of discrete entities, for example parts that are machined and/or assembled. Frequently, sets of similar parts are aggregated (considered as unique entities) into *batchs*. Manufacturing systems are called discrete production systems. Such systems are the opposite of continuous production systems, dealing with continuous flows of material such that paper mills or oil refineries. Some systems are hybrid because they handle continuous raw material but produce batchs of products. For example in the chemical industry when tablet boxes are produced from powders. In the sequel, we shall only consider manufacturing systems which are strictly discrete.

1.2.2 What is a job shop?

The oldest production line organization, optimized for mass production, has been described in Chaplin's Modern Times. Production was driven by a conveyor and the workers had to operate synchronously each doing a repetitive task. Such manufacturing systems can only produce a unique kind of product. The system being synchronous, its optimization consists in decomposing all the work to be done into a set of operations that have exactly the same duration.

In a *flow shop*, the production line is more versatile, and products may by-pass some machines. Such systems can only produce a unique family of products that differ slightly from one another.

In a *job shop*, there is no notion of production line, for each product a *production route* is defined. This route describes a sequence of machine operations which is not restricted by the physical layout of the machines. Such systems can handle any number of product families and are limited only by the set of operations that the machines can perform. Its operation is asynchronous. It is the more versatile and flexible kind of system, unfortunately it is also known to be the less efficient because either a large number of machines remain idle most of the time, or the in-process inventory is very large (completion of the products are unpredictable because they may remain a very long time in intermediary inventories). In order to increase its efficiency, a sophisticated schedule of the machine utilizations has to be implemented.

1.2.3 What is a Flexible Manufacturing System (FMS)?

Flexible Manufacturing Systems are an attempt to reconcile the efficiency of the production line with the flexibility of the job shop in order to satisfy a versatile demand at low cost [BUF 79, OGR 86].

Generally, two kinds of flexibility are distinguished. The *long term flexibility* corresponds to the possibility of introducing new product families in the manufacturing system during its operation and with little effort. The *short term flexibility* corresponds to the possibility of handling concurrently a large variety of product families at a given time in the manufacturing system.

In order to meet these requirements, a Flexible Manufacturing System is formed of

- a set of flexible machines,
- an automatic transport system,
- a sophisticated decision making system to decide at each instant what has to be done and on which machine.

Flexible machines have the capability of performing various operations, they have an automatic tool storage/retrieval system, and machining programs can be down-loaded. This flexibility can be called *physical flexibility* [ERS 88].

An automatic transport system is required in order to transport the parts to the machine where the next operation is to be executed. This system has to be sophisticated because, in absence of a physical production line, the layout does not correspond to the sequences of machine utilizations. Any location on the shop floor has to be reachable from any other one.

But without a suitable decision making system which allows the introduction of new family products with their routes, tolerates machine disruptions, optimizes machine utilizations, etc, the shop floor would be of little use. It is this system which has to organize the production and to schedule and synchronize the machine utilizations.

Frequently, a Flexible Manufacturing System is structured into manufacturing cells. A cell is an elementary manufacturing system consisting of a flexible machine tool (or an assembly device, or any complex device dedicated to a complex manufacturing operation), some local storage facilities for tools and parts and some handling devices such as robots in order to transfer parts and tools between the cell and the global transport system. Elementary manufacturing cells are called workstations. A collection a machines which are identical is called a machine pool.

1.2.4 What is Computer Integrated Manufacturing (CIM)?

Within a Manufacturing System, computers may be used for a lot of functions. Computer Integrated Manufacturing means that all these functions are fully integrated. Such an integration requires two basic facilities [ALB 82]:

- a Local Area Network (LAN) in order to allow the communication,
- a global Technical Data Base, which might be distributed, in order to store the data.

An important standardization effort is currently being done with the MAP (Manufacturing Automation Protocol) [GM 85] and TOP (Technical and Office Protocol) standards. The technical data base is also a key issue because it is the easiest way to allow all the functions to share information. The specificity of such a database is that some data have a very short life (the fact that a part is being machined) when others have a very long one (the product routes). Another issue that has to be resolved either within the data base management system or within the functions is that the way the data are used has to be consistent: an idle machine cannot be allocated to two different operations at the same time by two different functions. This implies that somewhere a strict synchronization between the data utilizations is implemented.

Among the main functions that have to be integrated are:

- CAE (Computer Aided Engineering),
- CAD (Computer Aided Design),
- CAM (Computer Aided Manufacturing).

Computer Aided Enginnering consists in designing new manufacturing systems or in designing modifications of an existing workshop. From the products that have to be manufactured, it is necessary to determine what kind of machines are required, how many machines are sufficient, which transport system is best, etc. CAE requires a lot of performance evaluation which may be analytical or done by simulation. At a detailed level, performance depends highly on the management of the planned system, i.e. on the decision making strategy.

Computer Aided Design concerns the description of the products, their shape, the material used, the way it will be machined. CAD defines the product families by grouping products with similar operations and produces the *product routes* (Group Technology Classification [HAM 85]). It is important to underline that the simplicity and the efficiency of the decision making strategy highly depends on how the products have been grouped into families.

Computer Aided Manufacturing consists in all the software packages required to control the Manufacturing System. For example, the machine tool programs, the packages for planning, scheduling, etc. ICAM (Integrated Computer Aided Manufacturing) is considered as a first step before CIM, the integration concerns only the computers required to control (from management to real-time control) the Manufacturing System.

It is important to distinguish CIM from FMS. The first concept is applied to systems which are highly automated. It is similar to Factory Automation. It does not implies that the system is flexible. On the other hand, a Flexible Manufacturing System is not necessarily fully automated. The various control functions may exist but their integration may be performed by the management staff, or the decision making process may be automated and the local operation may be performed by skilled workers manually. Very clearly, a skilled worker is much more flexible than any automated machine tool!

In the sequel, we shall focus on CIM in the context of FMS. More precisely on the following issue: why and how can Petri nets be an aid to meet at the same time the flexibility requirements and the computer integration of Manufacturing Systems. Let us just remark that the key word synchronization has been encountered already.

2 Hierarchical decision and control level.

2.1 The hierarchy.

Due to its complexity, Manufacturing System Control is commonly decomposed into a *hierarchy of abstraction levels* [ALB 81, JON 85, KET 89, GER 89]. A coarse grained hierarchy is made up of the following levels: *planning, scheduling, global coordination and real-time monitoring, sub-systems coordination* and *local control*.

Each level operates on a certain *time horizon*, on a certain view of the manufacturing system. At the upper level, the time horizon is long (may be ”infinite”) and the global system is considered, but its representation is strongly aggregated (products families, global machining time, ratios ,etc). The response time is not required to be strictly bounded. In the hierarchy, each lower level has to be a disaggregation of the upper one [BIT 77, AXS 81, ERS 86, DAV 89]. The time horizon of the lower level is shorter, the portion of the system considered is smaller (factory, shop, cell, station, machine, etc) and on the other hand more details are taken into account (individual products, parts, operations, steps within operations, etc). Progressively, real-time constraints are introduced and at the bottom of the hierarchy (local control) they are hard. Let us consider each level in more detail.

2.1.1 Planning.

The whole plant is generally considered with an estimated demand (unexecuted orders and an estimation of future ones). The time horizon is decomposed into smaller ones and at a global level, the amount of products that can be operated in each one is computed. Frequently various levels of planning are considered for time horizons that are shorter and shorter. In the last one, information concerning the availability of raw material is used in order to determine at which time each product will be introduced in the Manufacturing System (*earliest starting time*) and at which time it has to be delivered (*due date*). This is called *Material Requirement Planning* (MRP). Frequently, it is also at this level that the resources (machines in particular) are allocated to the operations in order to distribute their work load equally. This is called *Manufacturing Resource Planning* and it can also be considered as the initial step of scheduling.

2.1.2 Scheduling.

At this level, each operation on each part or product is considered individually. The problem is to produce a schedule i.e. to decide at which date a given operation will be performed. The combinatorial explosion of the number of alternatives is enormous in the case of a Flexible Manufacturing System because each machine can perform many kinds of operations and for a given machine, once the operations are fixed, it is necessary to determine an order of execution. The role of planning is to efficiently reduce the number of these alternatives, by means of machine allocations and by fixing earliest starting times and due dates for the products. Another way of reducing the combinatorial explosion is to work with *batches* or *lots*, and to consider *machine pools*. Finally, scheduling produces a sequence of dates for the execution of each operation on each machine.

As scheduling operates on an estimated state of the manufacturing system, it generally considers some slack time: for each operation *an earliest starting time* and *a due date* are defined in such a way that the defined time interval is larger than the *operation duration time*. Sometimes the schedule is given in the form of a total ordering of all the operations on a each machine. Another way of avoiding explicit references to dates is to elaborate a set of rules which specify, each time a machine becomes free, which operation will be performed next. This can be seen as an implicit schedule. These approaches can be combined in any way.

2.1.3 Global coordination.

Its function is to update the state representation of the workshop in real-time, to *supervise* it, and *make real-time decisions*. It has to check that no abnormal update message is received in order to guarantee a certain consistency between the actual state of the workshop and the technical data base (*fault detection*). It has also to compare the set of the operations which are possible (because the required resources are free in the shop) with the set of the operations which have to be done in order to respect the manufacturing schedule and to make the right decisions in real-time. When the schedule concerns batches, it is at this level only that individual products and machines are considered. It

is the global coordination that manages in real-time the *distribution* of the elements of the *batches* on the elements of the *machine pools* for a given operation.

When the schedule is implicit, i.e. when it is a set of decision-rules, these rules are applied within the global coordination level and with the actual manufacturing shop state information. Sometimes this is called real-time scheduling. As a matter of fact, it is necessary to differentiate the use of rules with an estimation of the state in order to produce a provisional schedule, which is scheduling, from the use of rules on the actual state in order to make *one* decision which is coordination.

2.1.4 Sub-system coordination.

It realizes the coordination of subsystems such as the *transport system*, the *manufacturing cells* or *work-stations*, the storage units etc. It operates as the global coordination level supervising the behavior of the sub-system and monitoring the execution of the planned operations but in a more detailed manner. A typical example concerns the coordination of a storage unit, or a robot loading and unloading a machine tool.

2.1.5 Local control.

It implements the real-time control of the machines, the devices etc. It interacts directly with the *sensors* and the *actuators*. All the emergency procedures are implemented at this level and the real-time constraints may be very hard.

2.2 Optimal decision making in the hierarchy.

It is well known that in a hierarchical system, the optimal solution is not necessarily obtained by optimizing each level separately. However, as FMS control systems are not manageable without such a hierarchical decomposition, a global optimization is impossible. Current approaches consist in privileging a given level which is optimized first, the other ones being used only for detailing the optimal solution or for long term decision (architecture of the shop floor). They may be grouped into two trends: Taylorism and Just-In-Time production.

2.2.1 Taylor's approach.

The current organization of production systems in Europe is still heavily influenced by the ideas of Frederick W. Taylor (Philadelphia 1856-1915) who was the first to work on manufacturing system management in the past century. He found that the optimal operation could only be reached by means of a perfect organization where each person plays a precise role. Some, the management staff, have to think, calculate the optimal manufacturing plan and schedule all the operations. Others, the workers, have to faithfully execute the decisions of the management staff.

This organization is nowadays considered as bad practice because manufacturing systems such as FMSs, are too complex, disruptions are frequent and human behavior cannot be modeled by a set of equations. It often produces the so-called "dialogue" between someone blind and someone mute. The management staff elaborating the manufacturing plan has no knowledge about what is actually occurring within the shop. It therefore corresponds to the blind one. The workers in charge of the production are faced with inconsistent plans and give as few pieces of information as possible to the management staff in order to preserve some independence. They are the mute one.

Within the context of the realization of what should be an optimal hierarchical decision structure, this means that an *optimal solution* is computed at each level, taking the optimal solution of the upper level as an input. An optimal plan is elaborated, then it is *disaggregated* at the scheduling level and the optimal schedule is computed. Then the global coordination and the lower levels have to execute it. In a way, each level is blind with respect to the lower level and mute with respect to the upper one.

Even in automated manufacturing systems, disruptions occur (a tool breaks, a machine fails, etc). In such a situation, a violation of the optimal schedule occurs immediately and a new optimal schedule has to be re-computed in real-time. Due to combinatorial explosion, scheduling is very time consuming and it is generally not possible to re-schedule in real-time. Anyhow, it would be very expensive and not necessarily efficient in relation to optimality. In fact, the long term optimal control may not be obtained by the concatenation of a sequence of schedules optimized with erroneous assumptions concerning the manufacturing shop. This is why manufacturing schedules are frequently

considered as a mere aid for a human decision maker. The result of this approach is that in-process product inventory is larger than necessary in order to cope with the disruptions in the shop.

2.2.2 Push, Pull and Just-In-Time production.

Taylor's followers have privileged long term optimization with little information about the actual operation of the manufacturing shop. Another way is to establish rules at a local level implementing *inventory management strategies* [PET 79]. *No provisional schedule* is computed, the scheduling level consists only in elaborating these *rules* which are used at the *coordination level*.

In the *push* control, each time raw material is available, the input buffer of the machine is not empty and the machine is free, an operation is started. When more than one operation can be performed, some rules allow the choice of one of them. Priorities, production ratios, consideration about product due dates are mainly used. With this policy, production is mainly driven by the *Material Requirement Planning* where the demand is taken into account. Its drawback is that in-process inventories may increase a lot in the absence of real-time feedback from the demand.

In the *pull* control, each time an in-process inventory is lower than a given value, a *requisition* for new fabrication is sent to the machine which is the input of this inventory. Production is *driven by the demand*, but in order to have a good response time, in-process and raw material inventories have to be large in order to guarantee a new fabrication as soon as the requisition is received.

Just-In-Time (JIT) control can be seen as a combination of push and pull policies. The aim is that *in-process inventories have to be as low as possible*. This means that at the exact time when a requisition is received by a machine, the required raw material and in-process part has to be available in its input buffer, ready to be pushed. Slack time should be avoided and for each operation the earliest starting time should be exactly the due date minus the operation duration time. This kind of optimization can only result from a *fine grained real-time synchronization* of the operations in the manufacturing shop.

This kind of policy has been implemented first in Japan and popularized under the name of *Kanban* technique. An operation o_1 may only be performed if the raw material and the machine are available and if a certain *card* (*Kanban* in Japanese) has been received. This card is sent each time some other operation o_2 is terminated in the shop and thus implements the pull requisition.

Kanban is a way of introducing a *synchronic distance* between a pair of operations (o_1 and o_2) (i.e. of establishing a relation between the firing counts of the two associated transitions) [SUG 74, SCH 85, DIM 89]. It is clear that it can be generalized and formalized by a Petri net based description of a global synchronization of the manufacturing shop operations as it will be shown in the sequel.

In conclusion, the aim of *push* production is to optimize the machine utilization (a machine may not remain idle, production is immediately *pushed* on it). The purpose of *pull* production is to produce exactly what can be sold, production is *pulled* from the backlog. Customer's satisfaction is optimized. Finally, *Just-In-Time* production aim is to minimize in-production and raw material inventories.

2.3 Flexibility versus optimality.

The aim is to design and control the manufacturing system in order to simultaneously meet with the *flexibility* and *optimality requirements*. Unfortunately, these two requirements are frequently contradictory. Firstly, it is intuitively crystal clear that it is nonsense to compute *the* optimal long term schedule for a system likely to be deeply altered in the near future. Moreover, what is good for flexibility is frequently bad for optimality (and conversely), even at a detailed level. For example, the increase of in-process parts in intermediary storage units is good for flexibility because it allows an unexpected heavy demand to be satisfied but it is bad for optimality because large storage units have to be built and raw materials have to be available (and bought) earlier than in a *Just-In-Time* optimal solution. The economical benefits of flexibility appear only at the level of the firm policy which has to follow up as quickly as possible the market evolution.

The concept of flexibility is relatively new and unformalized. It may be addressed by defining a set of *acceptable solutions* rather than an optimal one. These acceptable solutions are defined by giving a set of constraints that have to be satisfied [ERS 88]. In doing so, the control of the manufacturing system is no longer a strict hierarchy [BAL 89].

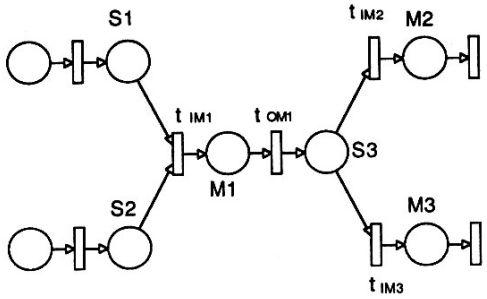


Figure 1: Petri net product route specification.

The management levels (planning and scheduling) are not privileged in relation to the real-time control ones (coordination levels and local control). Each one produces a set of *good* solutions and the decision maker chooses one of them belonging to the intersection of all these sets. Sometimes these systems are called *Heterarchical* Manufacturing Systems [HAT 85, JON 89].

As it has been seen in the case of *Just-In-Time* policy, *synchronization constraints* are important. A correct modeling of them is thus essential when people want to meet with flexibility and efficiency, that is *synchronization is one of the main issues in Flexible Manufacturing Systems*.

2.4 Why Petri nets?

First of all, a Manufacturing System is a discrete system. Apart from planning (where people work with ratios of products fabricated per week or per day), any modeling has to be based on the concepts of *events* and *activities*. An event corresponds to a state change. An activity is a black-box encapsulating what is occurring between two events. When using Petri nets, events are associated with *transitions* and, usually, activities with *places*. However Petri net is not the unique tool handling events and activities. Queuing models have been extensively used for performance evaluation [BUZ 86]. Algebras [INA 89] and formal models [RAM 89] for discrete event systems are under investigation, but Petri net is the unique tool which formally represents parallelism and synchronization. Let us consider some basic reasons explaining why Petri nets are a nice tool for FMS. In §3, advantages of Petri nets for the *design* are presented, while in §4 advantages concerning scheduling and coordination at the *plant control levels* are pointed out.

2.4.1 Petri net for Just-In-Time policy.

Whatever the control policy, an elementary inter-operation synchronization results from the part routes describing the sequences of operations that have to be performed for each product. This synchronization can easily be modeled by Petri nets and has to be implemented whatever the control policy which may be either an optimal provisional schedule computed at the scheduling level or a Just-In-Time policy.

Let us consider the Petri net in Figure 1: machine $M1$ assembles parts located in inventory $S1$ and $S2$, then the product is machined on $M2$ or $M3$.

It represents the routes, and only the routes. For some reachable markings, this Petri net shows conflicts. For example, when place $S3$ contains a token, transitions t_{IM2} and t_{IM3} are conflicting. Furthermore, tokens represent products that are not necessary identical and the choice of a given token to fire a transition is similar to a conflict.

In the case of Taylor's approach, these conflicts are resolved by the schedule which indicates on which machine and at which time a given operation is to be performed (we assume here that the schedule has been computed for individual machines and individual parts, that is there is no machine pool and no batch). It is thus necessary to associate an interpretation with the transitions (extra firing condition) and to consider that the Petri net token game is driven by its environment that is in this case by the optimal provisional schedule. It is important to point out that the Petri nets used as models of Manufacturing Systems *are always interpreted Petri nets*. Transitions attached to the beginning of an operation are fired at a date defined by the schedule. For example, when it is time to begin an operation on machine $M1$, transition t_{IM1} is fired. If it is not enabled then the schedule is violated.

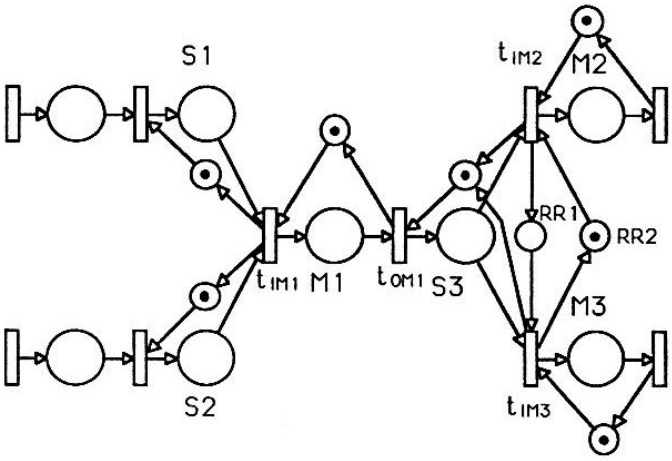


Figure 2: Petri net specified Just-In-Time policy.

Just-In-Time policies imply that synchronization rules have been defined to resolve all the conflicts in the manufacturing shop. For example, one can decide that machines $M2$ and $M3$ are to be used alternatively. Each time $M2$ is used, a card (Kanban) is sent to allow the use of $M3$, each time $M3$ is used a card is sent to $M2$. This kind of mechanism can easily be depicted by Petri nets. Figure 2 represents the same Manufacturing System as Figure 1 but with a Just-In-Time policy places $RR1$ and $RR2$ implement the cyclical priority defined above. The general structure of a Just-In-Time policy is a generalized pipeline. A path of places and transitions corresponds to the *flow of products* and a backward path implements the *control* (tokens correspond then to the cards of the Kanban approach). Time can be explicitly taken into account by associating a delay with some synchronization places (for more detail see [DIM 89]).

In a strict Just-In-Time policy, no conflict may appear in the Petri net and all the places of the product paths have to be safe places (contain at most a token, that is the in-process inventory is minimal). In fact, products have to arrive at the machines *just in time* to be operated with no delay (see Figure 2). Nevertheless, the Petri net interpretation is still necessary in order to synchronize the firing of the *end of operation* transitions such as t_{OM1}, t_{OM2} and t_{OM3} with the corresponding events.

2.4.2 Petri net for flexibility.

As it has been said previously, in order to meet the flexibility requirements, it is possible to define a set of good solutions by giving the constraints which have to be verified. The final choices are then made in real-time at the global coordination level, and they are based on the actual current state of the manufacturing shop which may be quite different from the foreseen one. Are Petri nets well suited for this?

Let us consider again the uninterpreted net in Figure 1. It describes only the fact that machine $M1$ can only operate if at least one part is present in inventory $S1$ and at least one part is present in inventory $S2$. Nothing is said about the date when transition t_{IM1} is fired. Any dated firing sequence (with a date associated with each transition firing) is a schedule which is consistent with the constraints defined by the net. In other words *uninterpreted Petri nets are the unique way of describing an infinity of schedules such that its events verify a partial order relation* (causality relation).

Clearly the net in Figure 2 is such that the set of consistent schedules (firing sequences) is included in the preceding one. Adding a Just-In-Time policy augments the economical optimality by reducing in-process inventories but reduces the flexibility. Any interpretation associated with the *operation initiation* transitions (such as t_{IM1}) and resulting from planning or scheduling has the same role.

Consequently, it can be underlined that Petri nets are well suited to describe trade-offs between flexibility requirements and control policies improving the efficiency, which is the *major issue of Flexible Manufacturing Systems*. As it will be detailed in the sequel, verification, analysis and performance evaluation can be performed from this description.

2.4.3 Why Petri nets rather than communicating processes?

At this point, we have introduced a few arguments concerning the usefulness of Petri nets. However, nets are not the unique model for describing a partial order among a set of events. Another approach could be the use of a communicating sequential process based language. It is not the case because, as it has been mentioned above, the interpretation of the net is essential to depict any scheduling policy which cannot be expressed as a pure causality relation (i.e. which is not a Kanban style policy). The extra firing conditions associated with the *operation initiation* transitions operates on dates, time interval, *and on states*.

For example, a real-time decision in order to resolve a conflict may include the evaluation of the time interval during which the value of a machine input buffer has exceeded a given value. It is thus essential to be able to handle both events and states in order to depict control policies completely. Because of the place/transition duality, Petri nets are better than communicating sequential processes where the internal states of the processes are hidden.

In a CSP like approach, each process would have to maintain a history of message exchanges in order to compute explicitly a local state and then to communicate this state to the other processes. It would be cumbersome and not natural.

To conclude this part, let us recall the points that are essential when applying Petri net in Flexible Manufacturing Systems:

- uninterpreted nets describe partial orders between events (or between classes of events),
- they handle events and states variables at the same time,
- they allow a set of schedules to be described verifying some constraints and therefore they allow dealing with flexibility,
- they may be interpreted (time or extra firing conditions) in order to restrict the set of possible firing sequences (trade-off between optimality and flexibility),

3 Nets in CAE for FMSs.

As it was pointed out, Computer Aided Engineering (CAE) for Flexible Manufacturing Systems (FMS) concerns the design of new systems or the modification of an existing one.

Returning once again to the constituents of FMSs, it can be stated that a FMS consists, basically, of a set of *workstations* (NC-machines; assembly stations; quality control systems; loading/unloading machines; etc.) *stores* (raw materials; parts; tools; finished products; etc.) and an *automatic transport system* (with automated guided vehicles, AGVs; loop conveyors; aerial chain-drivers; etc) connecting them, the whole system operating under a *computer-controlled integrated system*. Changing in the above description workstations for *processors (or computers)*, stores for *memory systems* and automatic transport system for *communication system*, the reader can observe that FMSs are, from a system level perspective, analogous to *distributed computer systems*. Therefore, many analogous design problems appear in so different technological contexts.

The manufacturing engineer usually focuses on the particular processes associated with the workstations (for example: cutting, welding, assembly, etc.). However, the FMSs designers/users must think at *system level*, looking at its entirety more than at the efficiency of individual machines.

In the next section, the domain of CAE for FMSs are briefly introduced, while the following sections consider them more in detail from a Petri net perspective. A major advantage of Petri nets is the use of a *unique family of tools* from the first stages of the design until the code generation for the real-time computer based control of the overall system.

3.1 Domains of design activity.

From its firsts steps, the design of a FMS requires extensive performance evaluation, to be able to decide among alternatives. Typical performance figures are throughputs (products rates), work in process and time span for each product, machine utilizations (use ratios), queue lengths or time distributions to access machines or transport subsystems, etc.

Alternatives to be considered at this level are: different layouts and transport system B machines in a given workstation or cell, machines of different speeds, stores of different sizes, etc.

From performance figures, by means of economic models, the cost of different alternatives is evaluated. The use of different machines can lead to different qualities for products. Therefore, product quality must be also taken into account in the subsequent economic analysis together with: workspace needs, machine investment and maintenance costs, etc.

At this point, it is interesting to observe that a FMS is essentially a distributed system in which many technical and economical tradeoffs appear to the designer. For example, the use of a faster conveyor can imply a reduction of the size of some stores and a reduction of the inventories, but may also require changing the loading/unloading mechanisms associated with it. A faster transport system may imply a more expensive investment, and a more expensive maintenance cost.

Performance evaluation phase leads to the "nominal performance" of the FMS. Once an installation is built, *scheduling* (§4.1) will try to obtain the best performance of the system under specific state and time horizon of work.

Once decisions have been made based on preliminary performance figures, a more detailed design starts. The correct synchronization to deal with concurrency and cooperation relationships becomes crucial. Deadlock-freeness, appropriate bounds for the queues, mutual exclusions in the use of machines, etc. must be verified. The outcome of this design phase must be a formal model describing parallelism and synchronization. From this model, in a more or less automatic way the skeleton of the real-time control system software can be produced.

The detailed behavior of the different machines depends on their specific programs. Placed at the system level we are at some distance from these specificities; nevertheless, it can be pointed out that the actual trend is to generate them in a more or less automatic way from the geometric/technological CAD descriptions (CAD-CAM connection).

For the system design, Petri nets offer:

1. A graphical and precise formalism which allows easy and deep dialogues about the expected behavior of the system among the different teams that participate in the design process (designers, owners, users,...).

2. A well founded theory for the qualitative verification of net model properties (liveness, fairness, boundedness, etc).

3. A reasonable framework for quantitative analysis (performance evaluation), actually undergoing important developments.

4. Implementation technology independence providing, in particular, some well understood code generation techniques for the real-time control software from the net model.

3.2 Modeling with nets.

Let us consider the coordination and local control level in the hierarchy introduced in §1.2. In addition to its graphic representation differentiating states/places and activities/transitions (or events/transitions and activities/places when transition firings are supposed to be instantaneous) and its simple and well defined semantics, Petri nets allows:

1. The modeling of *true parallelism* (instead of *interleaving* the parallel streams i.e. parallelism is clearly differentiated from non-determinism).

2. The possibility of progressive modeling by using *stepwise refinements* or *modular composition* (see, for example, [VAL 82, ALA 85, ALL 85, MAR 85]). In both cases, catalogs of well-tested subnets allow components reusability leading to significant reductions in the modeling effort.

3. The easy integration of (deterministic and/or stochastic) *timing constraints*: transitions (i.e. activities) or places (i.e. local states) can be timed in an straightforward way.

The "clean" integration of timing constraints in net models allows the *performance evaluation* of the modeled system (CAE) and the study of *scheduling strategies* for its run-time use (CAM).

The possibility of progressive modeling is absolutely necessary for FMS because they are usually large/complex systems. The refinement mechanism allows the building of *hierarchically structured* net models. If the refinement-transformation rule *catalog* is well selected and the model well designed, the design can be easily or trivially proven correct with respect to the specifications.

Therefore, *composition* and *refinements* help deal with model complexity. Obviously this work can be done with Place/Transition nets (P/T nets) or with more abstract formalisms: High Level Nets (HLN). In [MAR 86] the use of P/T nets and HLNs for the specification of FMSs is considered. The interested reader can see how a car production workshop have been modeled by means of P/T nets [VAL 82] and of Colored nets [MAR 85]. In both cases modular composition plays a major role. In the first case, extensive use of refinements is also used, while in the second one the color abstraction allows a compact model. Another structured model of a FMS is presented in [NAR 86] using colored nets. In [CAS 89] a modeling methodology based on a so called *Pregraph* (i.e. a functional description model of the production sequences) is proposed.

Figure 3 shows a simple production cell operated by means of a robot, R. When a raw part arrives (detected by the presence sensor Π_1), if the robot is free, it proceeds by loading machine 1 (*load*; *el*: end of load). The machine later performs operation OP_1, waits for deposit in the buffer (*wait dep.*) until there is an empty position and the robot is free. Machine 2 proceeds in an analogous way, but once OP_2 has been ended (eop_2), waits for the robot and $\overline{\Pi_2}$ (i.e. conveyor 2 is free) to unload the processed part.

This live and bounded model is fully reducible, so it can be generated by *refinements*. Alternatively, the model can be generated by *modules composition*:

Machine 1 (*wait raw, load,* OP_1*, wait deposit, deposit*), Machine 2 (*wait with,* OP_2*, wait free, unload*), robot (*R, load, deposit, unload, withdrawal*) and store (*empty, deposit, object, withdrawal*).

The example is a producer (machine 1)-consumer (machine 2) schema with a monitor (R). If place R is removed, the remaining net is a marked (or event) graph. When several type 1 (or type 2) machines must be used, a high level model can be easily derived.

Top-down (refinements) and bottom-up (modular composition) modeling methodologies together with the attributes abstraction of HLNs, allows a reasonable complexity for the modeling process and the model itself. But in many cases, manufacturing engineers want to use specifications languages closer to their own culture. In this sense "application domain-oriented abstractions" can be very practical. GRAMAN [VIL 88] is a graphic system for describing manufacturing applications. Figure 4 shows how a simple manufacturing and assembly cell is described with the special icons provided in GRAMAN. It needs a separate description of *work plans* (i.e. the description of operations to be performed and the resources and materials required to produce a part) and resource interconnections. *Work plans* are modeled in GRAMAN with labeled P/T nets. Using the *resource interconnection diagram* and the work plans, GRAMAN produces a single net model (hidden from the manufacturing engineer), that later can be potentially used for performance evaluation, control, scheduling or implementation.

Concluding, Petri nets allows the use of different abstraction levels models (P/T nets, HLNs) and the use of powerful modeling methodologies: *refinements* and *modular composition*. Nets can be used directly in the modeling process or can be *embedded in a higher level application oriented formalism*. In the last case, nets can be transparent to the user. Last but not least, the graphic representation of nets makes them particularly well suited for *animation purposes* (i.e. simulations that only try to illustrate how the model operates).

In this section, we have been mainly considering "normal" behavior. But in manufacturing processes there exists many special or abnormal behaviors (initialization, emergency-stop, termination,...) to be considered. Usually this problem is resolved using mutually exclusive net models, corresponding to the different behaviors. In [GEN 87], self varying (adaptative) net formalism is used in order to model with a unique net normal and abnormal behaviors.

On the negative side, it can be recognized at this point the Babel-Tower-impression that the Petri net newcomer may have looking at the myriad of so-called high level nets formalisms: numeric, predicate/transition, colored, relation, algebraic, with data structures, ... and their variants.

3.3 Qualitative analysis.

The design process of FMSs is a relatively complex task. Thus, it is very important to validate the significant steps of model building/transformation before going on to the implementation.

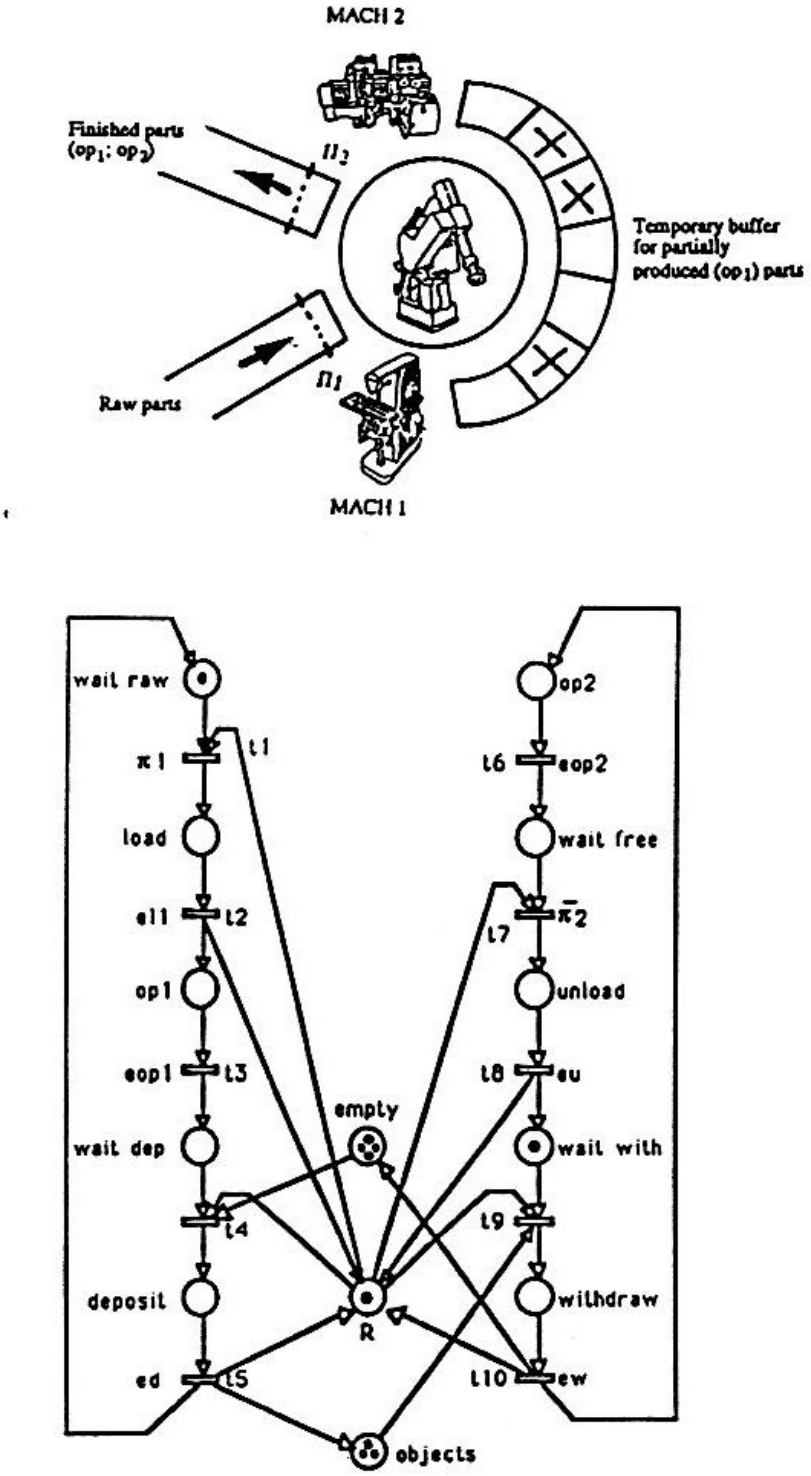
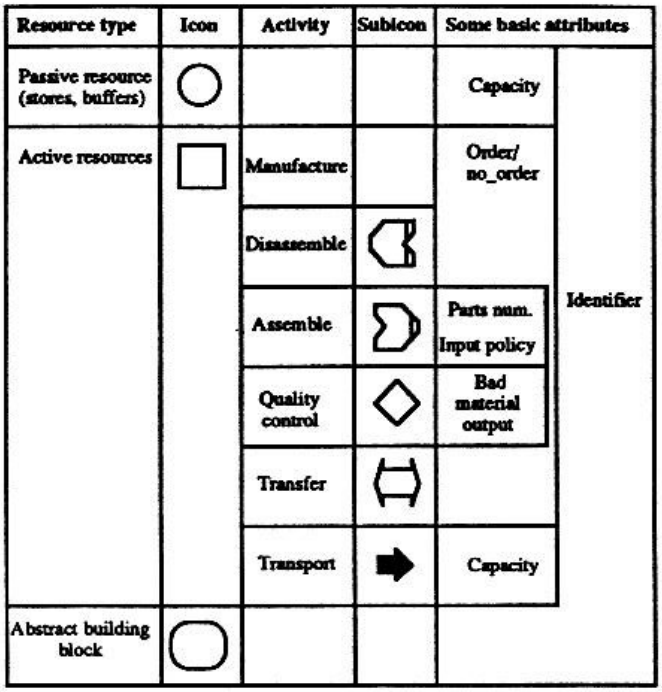
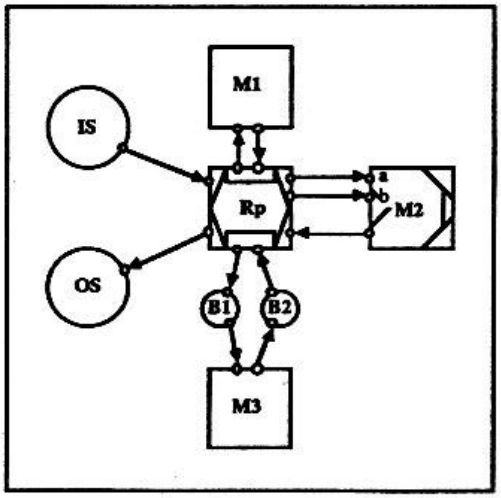


Figure 3: A production cell with two machines, one robot and a store.



a) Building blocks, activities and associated attributes.



b) Manufacturing and assembly cell,
a robot Rp feeds three machines $M1$, $M2$ and $M3$ through buffers $B1$ and $B2$
IS and OS are input and output stores

Figure 4: GRAMAN: Icons and an example of cell [VILL 88].

An outstanding aspect of Petri nets is their capacity for being analyzed to check if the constructed model verifies some specification properties. Among these, *synchronic properties* (lead, distance, places bounds, places mutual exclusions, etc.) and *activity properties* (deadlock-freeness, liveness, home states, etc.) can be studied. The first are, in temporal logic terminology, close to "safety properties", while the second group is close to so called "liveness properties".

The qualitative analysis of net models is now a relatively mature field (see, for example, [BRA 83, SIL 85, BRA 86]) for place/transition nets. Nevertheless much work needs to be done for high level nets models. The qualitative analysis allows the detection of "important" design errors, reducing risks and saving design time and costs. Nevertheless, it is important to remark that nets are interpreted when modeling a real (FMS) system. The net in Figure 3 is *interpreted*: its evolution is conditioned by the state of the environment (external world). Nets as mathematical objects are said to be autonomous, while interpreted nets are said to have *non-autonomous* behavior.

The additional constraints introduced by interpretation can lead to some important problems [SIL 85]:

1. *Synchronic functions* on the autonomous net can be greater (or equal) than the same functions on the underlying non-autonomous nets. Even more, a *synchronic relation* (e.g. boundedness) may hold on the non-autonomous net but does not hold in the underlying autonomous net. In other words, synchronic relations on the autonomous net are *sufficient* conditions for the same relation in the interpreted net.

2. *Activity properties* of the underlying non autonomous net can be nor necessary neither sufficient for the non-autonomous net. This is the case, for example, of liveness. Nevertheless, liveness in the underlying autonomous net is sufficient for liveness in the non-autonomous model if *simple nets* are considered and *fair progress* and *local fairness* properties are assumed.

The above problems leads designers of FMS to do *partial validations by simulating the net model plus its environment*. With respect to this problem, a formal validation of the underlying autonomous net assume that its behavior is correct whatever the behavior of the environment; and this is sometimes too strong.

Simulations can be done with graphical animation ("*a picture is worth a thousand words*") and/or partial observation of properties (e.g. if the set of the places associated with a structural deadlock are emptied, they will remain empty and all their output transitions will be non-live). In any case, the analysis of the underlying autonomous net model and/or the simulation of the interpreted net model plus its environment leads to more reliable results than those obtained with classical manufacturing simulation languages (e.g. SLAM).

From a theoretical point of view, analysis techniques of partially deterministic (i.e. interpreted) net models will be of interest. Because the interpretation of a net model means adding constraints, place-invariants of the autonomous model will be a subset of the invariants in the non autonomous model. So they can be used safely in any proof.

Finally, it is interesting to point out that the analysis of the autonomous underlying net model can simplify the real-time monitoring and, possibly, the fault detection plus error recovery. In fact, the underlying net frequently models sequential constraints which are verified in any case, even when an error occurs.

3.4 Quantitative analysis.

The high cost of FMSs makes crucial the quantitative analysis of these systems. In the design of new FMSs, the machines, the layout and the transport system are decided and dimensioned. Performance estimates at this stage need not to be very accurate, and relatively simplified models can be used. In the tuning of a design or of an existing shop floor, more details are needed in the model for optimization purposes. In tuning studies, huge simulation models and sensibility analysis (e.g. by means of perturbation analysis) are keywords. During the preliminary design queuing network models are used, even at the price of not considering the effect of synchronizations. These are, in general, due to *competition* (e.g. common access to a machine or subsystem) and *cooperation* (e.g. assembly of subparts) relations.

Performance evaluation in FMSs deals with issues like:

- how many machines, and/or automatic guided vehicles (AGVs),
- which transport topology and routing strategy

and are best suited to obtain:

- higher throughput, and/or
- small makespan and/or
- more balanced flows.

The synchronization among *process machines* (CNC-machines, process robots, etc.) and transport machines (loop conveyors, AGVs, pick-and-place robots, etc.) is very important because additional delays appear in the waiting states.

Performance evaluation is an extensive computation phase in the design of FMSs. Classically, it is done by *simulation* (using several formalisms: GPSS, Q.GERT, SLAM, SIMAN, RESQ-II,I) or by using simplified queueing network models. There exists an extensive bibliography for simulation concepts, techniques and languages (see, for example, [PRI 79a, PRI 79b, BAN 84, EVA 88, KRE 86, PEG 86]). Nevertheless, many problems about ergodicity (for simulation) or strong simplifications (e.g. removing synchronizations) in queuing network models leads to the conclusion that new approaches can be of interest.

The interest of Queuing Network (QN) approaches basically lies in the use of so called *product-closed-forms*, because the performances can be computed by means of low cost algorithms. Nevertheless, the *product-closed-forms* only hold for restricted queuing networks (in particular, without synchronizations).

Stochastic timing on nets allows performance evaluation of models with parallelism and synchronization. To deal with different modeling goals, several stochastic interpretations of nets have been introduced [CAM 89a]:

- Associating time with places or, more classically, with transitions.
- For timed transition models two firing rules:
  1. single phase (atomic firing).
  2. three phases (start-firing with deletion of the input tokens, delay, and end- firing with creation of the output tokens).
- Single server versus multiple server semantics: self- concurrency is avoided or allowed in the firing of a timed transition.

Analysis methods are mainly based on the generation of the underlying *Markov Chain* (so a marking enumeration approach) or in *simulation*. These techniques are integrated, for example, in the package presented in [CHI 87]. For bounded nets, the generation of the MC (Markov Chain) is of exponential complexity. Moreover, the computation of the eigenvalues of the MC's transitions matrix is delicate because many numerical problems can appears.

A recent work shows that product-form solutions seem to be possible for monoclass QN with synchronizations [FLO 89]. Other works concern the generation of aggregate markov chains corresponding to stochastic high level models (see, for example, [DUT 89]) or the use of structural theory of P/T nets to derive in polynomial time bounds on throughput for net subclasses (see, for example, [CAM 89a]). In [CAM 89b] the exact throughput is computed in polynomial time for a subclass of net models that can be viewed as some type of monoclass-complex server with synchronization queuing networks.

Analytical methods for performance evaluation based on net models are mainly adequate for preliminary designs of FMSs. In [BAL 86], generalized stochastic Petri nets (GSPN) and queuing networks are used and compared to analyze a simple FMS with one transport vehicle, three machines, one buffer, a loading/unloading station and 2-part-types. The GSPN model is more accurate than the multiple-class QN model. It allows a simplified analysis of the *scheduling policy* and of the *pallet mix*.

Other FMS performance evaluation works in which stochastic nets are used are [BRU 85] and [ALJ 88b]. The first considers a particular shop floor with two machine tools, three tools and two conveyors. The second work defines some modules as basic building blocks, later studying the performance of several basic production schemas under some machine failure assumptions. Technically, the underlying MC is always generated.

Returning to our example of Figure 3, fast throughput bounds can be computed. Let θ_i be the (average) time duration of firing t_i ($\theta_1 = \theta_4 = \theta_7 = \theta_9 = 0$). If we assume that the robot operation durations ($\theta_2 \simeq \theta_5 \simeq \theta_8 \simeq \theta_{10}$) are much smaller than the machine operations (θ_3 and θ_6), then an additional "virtual robot" will not significantly affect the performance. If there exists two such robots, place R has two tokens and becomes implicit. So it can be

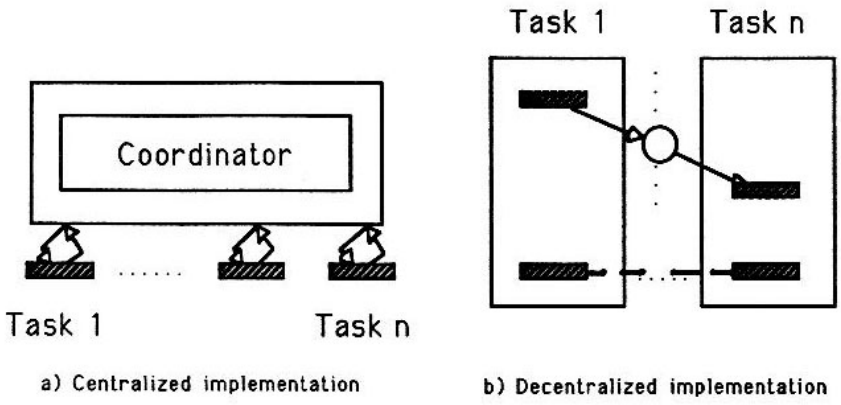


Figure 5: Implementation principles.

removed and a marked graph appears. Therefore, the following linear programming problem, LPP, allows to compute (polynomial complexity) the minimum cycle time [CAM 89a]:

$$LPP \begin{cases} \Gamma = & \max & Y^T \cdot Pre \cdot \theta \\ & \text{s.t.} & Y^T \cdot C = 0 \\ & & Y^T \cdot M_o = 1 \\ & & Y \geq 0 \end{cases}$$

where *Pre* is the previous incidence matrix, *C = Post - Pre* the incidence matrix of the marked graph, M_o the initial marking and θ the vector of average timing of transitions. If the marked graph is live, it can be proved that at least one optimal solution *Y* defines a marked elementary cycle.

If $\theta_2 = \theta_5 = \theta_8 = \theta_{10} = 1$, $\theta_3 = 10$ and $\theta_6 = 12$, the above LPP gives $14 (= \theta_6 + \theta_8 + \theta_{10})$ as the optimal solution (i.e. machine 2 is the *bottleneck* of the system).

If now it is assumed that there are two units of machine 2 while the robot operation delays are neglected, the above LPP gives $12 (= \theta_2 + \theta_3 + \theta_5)$ as the optimal solution (i.e. now the machine 1 is the bottleneck of the system).

More analytical results on stochastic Petri net models are needed. In any case these models can be used as formal specifications that can be simulated in cases where analytical methods are not computationally tractable.

3.5 Code generation for real-time control.

Once the net model of the control system has been qualitatively and quantitatively checked, an implementation must be generated to do the real-time control of the production system. To avoid error prone implementations, code generation should be automatic or semi-automatic.

Implementation of real-time Petri-net-based controllers has received considerable attention. A brief review of the software implementation of net-based models is presented in [COL 86]. Most of the referenced implementations consider industrial automation as the application domain. Usually, implementations attach some *non-reenterable code segments* to the firing of transitions, while the net model is assumed (or forced to be) without *self-concurrency*.

Centralized ([SIL 79, CHO 80, SIL 83, VAL 82, NEL 83, VAL 83, THU 85, COL 86, MUR 86]) and decentralized ([BRU 86, COL 86, TAU 86]) schemes are used. Centralized schemes (Figure 5a) can be sequential or concurrently implemented. Obviously hybrid schemes (i.e. partially centralized/decentralized) can also be found.

Centralized sequential schemes are usually employed at the local control level (frequently in special purpose real-time computers named Programmable Logic Controllers; see §4.3). A comparative study of these techniques is presented in [SIL 83]. Efficient executions can be *driven* by *the set of enabled transitions* or by *the marking of a subset of places*. The first gives the best performances for marked graph simulations, while the second is a "natural" extension of state machine implementation techniques. An additional advantage of marking-driven simulations is that

the marking is more or less directly represented, so it can be used for other purposes (e.g. when some actions are also attached to the marking of places, a very frequent fact in automation applications).

Centralized concurrent implementations are basically composed of:

- many specific tasks (possibly one per transition) and
- a coordinator/manager.

The coordinator plays the "token game" on the net model, initiating the execution of the tasks attached to the fired transitions. When a task ends its activity, it informs the coordinator to proceed with the next activation(s). The coordinator is an active (high priority) task that acts as the kernel of the application multitasking level. It provides *indirect synchronization* between the activities of specific tasks associated with the firing of transitions. The coordinator operates on an *explicit representation* of the net marking.

The simplicity and easy modification of the synchronization/communication scheme are among the advantages of centralized implementations. The basic problems with this kind of solution are:

- relatively inefficient execution (memory occupation and execution time), because even in simple cases there is neither direct intertask communication nor direct synchronization: everything is done via the coordinator,
- the coordinator is a hard point for safety; any fault is catastrophic.

Decentralized implementation schemes (Figure 5b) are usually built by using the following principles: the net is decomposed into

- a set of *sequential processes*, each one possibly grouping the code attached to several transitions in mutual exclusion (i.e. not simultaneously firable)
- some *communication/synchronization mechanisms* between these processes.

The classical communication/synchronization mechanisms are based on *asynchronous message passing* implemented by means of buffers or mailboxes, and *synchronous rendez-vous* between tasks. The required primitives concerning communication/synchronization are directly inserted, where required, in the body of the sequential processes (tasks). In decentralized implementations, communication/synchronization between sequential processes are *direct* (i.e. there is no intermediate active element). The marking of the net is partially (or even totally) *implicit* because it is implemented by the context of the sequential processes. When required, it can be made *partially explicit* by looking at the contents of the buffers.

Considering once again the net model in Figure 3, it is easy to observe that there exists four minimal place-invariants. Each place invariant leads, in this case, to a state machine. This *decomposed view* of the model allows a straightforward implementation with:

1. Two sequential tasks, implementing the machines (Machine1/producer; Machine2/consumer).
2. A mutex monitor (for the robot, *R*).
3. Two buffers (p_2 and p_3).

From a technical point of view, implementation may be *compiled* (i.e. the net model is cast into the control structures of the implementation language: ADA, PASCAL, C,...) or *interpreted* (i.e. a runtime support works on some data structures representing the net structure plus its marking). Between compiled and interpreted implementations, the classical trade-offs appears (basically memory occupation versus execution speed). Decentralized implementations are usually compiled.

Finally, it can be pointed out that the fault-tolerant implementation of net models is possible by using error detection/correction codes embedded in the net marking and/or some kind of external observers (e.g. the spy [VEL 88]).

Although considerable work has been done, decentralized and fault-tolerant implementations are relatively immature fields for place/transition net models.

Software implementation of high level net models needs considerably more work to reach industrial maturity.

4 Nets in CAM for FMS.

Computer Aided Manufacturing consists of all the software packages used during the operation of the Manufacturing System and required to control it, from planning to local control. The main areas where Petri net applications are promising are scheduling, coordination (at the global and subsystem level) and local control.

4.1 Scheduling/dispatching.

Once a net model represents a controller, its non-deterministic behavior must be reduced to produce the actual control. This non-determinism reduction is done through the *synchronization with the process to be controlled* (e.g. transition t_3 in Figure 3 is fired as soon as the eop_1 signal cames from the process) and through the *scheduling strategy* (i.e. decision about *which* and, eventually, *when* to fire the enabled transitions). The scheduler must *resolve* all effective conflicts remaining and can *delay* the firing of an enabled transition. But let us goes back to introduce a few basic ideas about scheduling.

Scheduling (see, for example, [FRE 82, GRA 81, CAR 88]) is an important activity in the control of any organization. Typically it concerns the execution planning of a finite set of *tasks* (activities) subject to a set of *potential* (i.e. precedence and temporal) and *resource constraints*. The goal is usually to optimize an *objective function* (e.g. *the makespan, a weighted flow time,...*). In other cases, a schedule must be produced such that a production *deadline* (i.e. a temporal constraint) is respected. If such a schedule is *infeasible*, a relaxation like the minimization of *tardy jobs* may become a reasonable objective.

Scheduling problems may be *static* or *dynamic* depending if the set of tasks to be performed is previously known or not. When some parameters of the problem (e.g. the duration of tasks) are probabilistically modeled, the problem is said to be of *stochastic* scheduling; otherwise *deterministic* scheduling is considered.

The easiest scheduling problems, static and deterministic, are of strongly combinatorial nature. Thus, even if well understood, usually algorithms are computationally hard (NP-complete in many practical cases). To deal with practical applications, *heuristic approaches* have been frequently developed. They consists in executing the tasks in a sequence which is not necessarily optimal but which satisfies *the constraint set*. The sequences are not pre-calculated, they are rather elaborated, commonly but not necessarily in real-time, by applying *decision rules* each time a resource has to be allocated. This can be done at *global* or at *local* levels. Many practical dispatch rules (see, for example [PAN 77]) have been derived from formal analysis or experience. This leads more recently to a widespread use of AI techniques in scheduling applications, using: (i) *Constraints directed searchs* (see, for example, [FOX 83]) or (ii) *Rule* (if-then) *based knowledge representation* (e.g. [BRU 86]). *Hybrid approaches*, integrating some mathematical programming (or analytical) and knowledge-based techniques have also been developed (see, for example, the OPAL system [BEL 85]).

The practitioner point of view with respect to scheduling is expressed in the following text quoted from [McKA 89]: "The schedulers spend very little of their time scheduling and preparing an official plan according to methods associated with the Job Shop Scheduling Problem as defined and studied extensively in academic circles. The schedulers are *problem solvers who create alternative routings and processes on the fly*. The scheduler attempt to produce a *reasonable loading* on the shop, not optimized sequences, since there are so many possible things that can happen between the time work is released on the factory floor and it is completed. They use their experience and skill, not mathematical algorithms, to make the feasible and reasonable decisions. To assist them in their decision making, the schedulers *need to know what is happening at any moment*, so that when a problem occurs, they have a reasonable view of what is happening in the shop and what the options are (if any) that exist".

The simplest scheduling problems are related to the *potential constraints* and *PERT/CPM (Program Evaluation and Review Technique/Critical Path Method)* methods. In both cases valued graphs are associated with the scheduling problem: *tasks* are represented by nodes, while the *ordering relationships* are depicted by arcs. It is assumed that there is no shared resources. Nevertheless this approach has been extended by modeling resource constraints through non conjunctive graphs which allow conflicts between tasks to be represented [BAL 69, ERS 79].

Transitions timed Petri nets allow *tasks* (represented by transitions), *resources* (represented by places) and *constraints* (potential and resources) to be modeled *within a single formalism*. Because of the complexity of scheduling techniques and of their combinatorial aspect, many results are for *timed marked* (or *event*) *graphs*. They do not allow the modeling of shared resources but represent a generalization of PERT/CPM-like graphs, allowing the study of

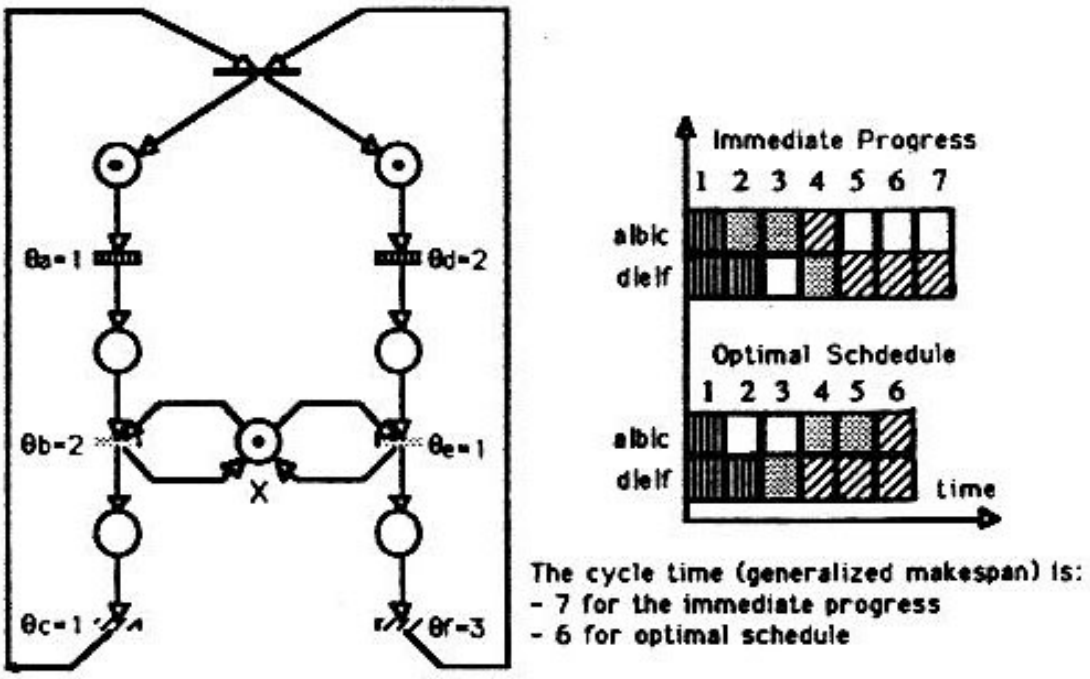


Figure 6: Immediate progress and optimal schedule.

repetitive schedulings (i.e. the case in which tasks have to be executed cyclically on parts of the same class). Clearly a Flexible Manufacturing System operates in a way which is not purely cyclical and repetitive. However repetitive behaviors are of great interest because they correspond to steady states and are thus important elements to compare FMS architectures or shop floor layouts. Moreover, at the level of subsystem coordination (workstation or cell control), the operation is commonly repetitive even in a flexible environment (routings may vary but local operations are similar). An accurate evaluation of each subsystem performance is very important for issues such as load balancing between workstations or cells [FRE 88, FRE 89].

The *Critical path* concept in PERT-like graphs can be complemented with *Critical cycles* in TMG (Timed Marked Graphs). The classical *makespan* concept (i.e. total production time), can be directly transformed into the *generalized makespan* concept (i.e. the production cycle time) [CAR 88]. Computations of these and other figures have been presented in [CHR 86][CAM 89][COH 85] [DUB 88] [COH 89]. In the last three works, an algebraic representation of timed marked graphs in terms of recurrence equations is provided. These equations are linear in a non conventional, $\{\max,+\}$, algebra (a *path algebra*). Earliest and latest starting times of activities in steady-state can be computed.

To optimize the generalized makespan in TMGs (dont forget decision-free systems!) is straightforward: fire transitions as soon as possible. Under this scheduling strategy, the analysis of TMGs can be done with polynomial algorithms. Nevertheless, extensions shows that the computation of earliest schedules is a NP-complete problem [RAM 80] even for Free choice nets (in [MAGO 87] the minimum cycle time for some live and safe free-choice nets is reduced to a set partition problem). Earliest schedules can be computed for TPNs by means of a LPP over the *earliest state graph* [CAR 88] (a reachability-like graph, that frequently is too large). *Decision Support Systems (DSSs)* based on TPNs for Flexible Manufacturing are outlined in [RAV 86] [FRE 87]. They represent help in the search of possible solutions.

Job-Shop systems applications have been considered in [COH 85, DUB 88, HIL 88, HIL 89] using TMGs models and cell ones in [FRE 87, FRE 88].

Even if TMGs represent an extension of PERT/CPM methods, shared renewable resources (i.e. concurrency relationships) cannot be modeled. When *Timed Petri Nets* (TPNs) are considered, resources can be modeled. Decision

rules corresponding to *heuristic scheduling* can be elaborated and tested by *simulation* using a Petri net based discrete event simulator [DUB 89]. However, the *immediate progress rule* for firing (i.e. fire as soon as the transition is enabled) can lead to non optimal schedules. This is nothing more than a particular case of a well known principle in optimization theory: *The optimal behavior cannot be reached, in general, using only local rules (i.e. partial information*). Figure 6 shows an example in which it is better to wait for the firing of transition θ_b, and resolve the subsequent conflict for θ_e. The different shadings in the schedules correspond to the different transitions. In this particular case, the optimal scheduling rule (fire θ_e before θ_b) can be expressed in a very simple way by adding *p*, a new place: ${}^\bullet p = \theta_e, p^\bullet = \theta_b$. If *p* is added, mutual exclusion place *X* can be removed, while the steps are preserved (i.e. *X* becomes concurrent implicit). The net obtained after removal of *X* and the addition of *p* is a TMG.

It can also be pointed out that place *p* corresponds to a *global decision rule* in an *heuristic scheduling approach* and that it may be implemented in a non-automated factory by a *Kanban* based mechanism (see §2.2.2).

Remark 1. If $\theta_a = \theta_b = \theta_c = \theta_d = \theta_e = \theta_f = 1$ for the net in Figure 6, the LPP in §3.4 produce a non-reachable bound for the cycle time: 3 instead of 4. ♢ **R1**

Remark 2. Let us consider again the net model in Figure 3. If $\Pi_1 = 1$ (i.e. there exists a part to be loaded) when t_1 is enabled, with the immediate progress rule, $\theta_1 = 0$. Analogously, if $\Pi_2 = 0$ (i.e. there exists no part waiting to exit the cell) when t_7 is enabled, $\theta_7 = 0$. Moreover, if $\theta_4 = \theta_9 = 0$, $\theta_2 = \theta_5 = \theta_8 = \theta_{10} = 1$, $\theta_3 = 3, \theta_6 = 2$ and $M_o(O) = 1$, the earliest schedule is 5. It can be reached by always solving the conflict between t_1 and t_9 in favor of the first. Because the net is simple, liveness is preserved even if local fairness is not assured! ♢ **R2**

Remark 3. If the evacuation of finished parts is frequently saturated (i.e. frequently $\Pi_2 = 1$ for some time while t_7 is enabled), the expected value of $\theta_7, E(\theta_7)$, can be positive. Assuming $E(\theta_7) = 2$, the optimal scheduling is not the one considered in remark 2, but just the reverse (i.e. fire t_9 instead of t_1). ♢ **R3**

The above remark points out that, in practice, the scheduling must know in *real-time* what happens in the controlled plant. For the considered example, the $t_1 - t_9$ conflict must be resolved as a function of the expected wait induced by the exit of finished parts. The *state of the environment* of the production cell is partially characterized, in this case, by sensors Π_1 and Π_2. In other words, the idea of *on-line* (or *closed-loop*) scheduling is pointed out against the more traditional *off-line* approach. This aspect reinforce the interest of *heuristic approaches* based on decision rules which are used in real-time i.e. at the *global coordination level*.

So far, we have seen that Petri nets are an efficient tool for *heuristic scheduling* and for the computation of a provisional schedule in the case of *repetitive* operation. When the elaboration of a schedule based on an estimated order list in a highly flexible manufacturing system is concerned, another complementary approach seems very promising: *constraints propagation* operating on time intervals *and* sequences between groups of operations produces interesting results [ERS 86b]. At each step, groups of operations which are in conflict for a given resource and have therefore to be executed in sequence are considered. New sequential constraints are deduced by considering simultaneously time and resource constraints. The introduction of this new sequential constraint is propagated through all the part routes, and production time intervals (earliest starting time, latest due date) are updated for all operations. In doing so, infeasible sequences are progressively eliminated, and the final set of constraints determines the set of all feasible schedules. Such an approach can be seen as the progressive elaboration of a time Petri net. The use of specific languages oriented to constraint propagation approaches such as Prolog III seems promising to perform this work.

As concluding remarks, it has been pointed out that Petri nets are highly suitable as a modeling formalism integrating tasks, potential and resource constraints. Important success has been obtained for the case of TMGs, generalizing in an interesting way the PERT/CPM method. Nevertheless, generalization of the net subclass leads to difficult problems and much further work is needed.

4.2 Global Coordination.

As it has been shown in §2.4, Petri nets are very useful at the global coordination level where real-time decision making is required to execute the planned and scheduled production in the manufacturing shop. These real-time decisions have to be consistent with the schedule (explicit or implicit schedule) which can be given either by time intervals associated with the tasks (*earliest starting time - latest starting time*), by an ordering (task A has to precede task B), by a set of rules (operate the shortest task first) or by a combination of the three.

The real-time decisions of the coordination level also have to be consistent with the manufacturing shop. Petri nets describe the sequential constraints which have to be respected at any time, in any situation. These are essentially *part routes* and *mutual exclusions* during machine utilizations.

On the other hand, explicit schedules and decision rules are used for optimization purposes; they are desirable, not compulsory. Some are more important than others. For example, a violation of the schedule implies rescheduling or delaying delivery. From a Petri net point of view, they allow conflict resolution (among transitions or among tokens in input places of enabled transitions).

Therefore, the Petri net can be seen as a *control structure* of a *rule-based system* where the rules operate on *object attributes* (parts, machines, tools) represented by tokens. It is why high level Petri nets have generally to be used [ALL 84, BRU 86b, CAS 89, DIL 87, KAM 86, MAR 87, NAR 86, VAL 88a].

In one case however, place/transition nets or even condition/event nets are sufficient: a strict Just-In-Time scheduling with only one class of products [BER 88, HAS 88, KOM 85]. In fact, as it has been mentioned in §2.2.2, as all the parts have to reach the input of the machines exactly in time, the corresponding places may not contain more than one token and no conflict may occur. This simple case is already used in industry [BER 88] at the global coordination level of an FMS because it is similar to the the use of a Petri net for local control or coordination of a subsystem such as a workstation.

The general case is still under investigation because it requires the implementation of high level nets. Rule-based systems are well suited to describe the part of the decision making process which does not directly involve the sequential constraints. Some criteria may be imprecise, contradictory, etc. This is why Artificial Intelligence techniques are commonly considered as necessary to solve the problem [FOX 83, BEN 86]. It must be pointed out that these rules may be the result of an *heuristic scheduling* or that they may be elaborated as a complement of a *provisional schedule*, letting some choices to be resolved in real-time (for example the result of a *constraint propagation* based scheduling). In the Petri net community, it is generally assumed that the implementation of automated decision making process in real-time should involve a Petri net interpreter and a knowledge-based inference engine working together. Mixing Petri nets and A.I. approaches are under investigation in many research laboratories [ATA 86, CAR 89, FLE 89, MAR 88, MUR 89, TON 86, VAL 88b].

Another point which has to be underlined is that Petri net utilization at the *Global coordination* level increase the *security and reliability* of FMS Control. As the Petri net based model describes the part routes and the machine allocation, if the control implementation includes a Petri net interpreter, it will necessarily include a representation of the manufacturing shop which is interpreted in real-time. This representation can be considered as an *observer* spying on the actual system and checking in real-time that its behavior is exactly the behavior foreseen by the designer [SAH 87, VEL 88]. In other words, all the update messages have to correspond to an enabled transition in order to be taken into account. If not, then a fault is detected and some diagnosis and restoration is required [CAR 89]. This aspect is essential to understand the superiority of any Petri net approach in relation to purely Artificial Intelligence ones.

4.3 Local Control: Programmable Logic Controllers, Grafcets and Petri Nets.

At the local level, many different kinds of machines can be controlled. Let us very briefly review this control level pointing out where Petri Nets (and the Grafcet) are used.

A usual trend in automation engineering is to provide specific languages for different application domains. For example, block diagrams for continuous process control and special purpose languages for CNC (Computer Numeric Control) or Robot Programming are used. In many cases, the controller is essentially a switching automaton, and a class of special purpose real-time computers named *Programmable Logic Controllers* (PLC) is used. At the beginning (some twenty years ago) PLCs were low-end replacements for relays, but actually they incorporate many additional/complex functions, approaching the functionalities of general purpose process computers. PLCs are employed in extremely different control application domains ranging from batch processes to traffic light systems or from flexible manufacturing to lift systems. Additionally, in continuous process control applications, PLCs are also used for supervisory/alarm functions, start-up/shut-down operations, etc.

One key feature of the important industrial success of PLCs is their problem- oriented-easy-to-learn programming languages [SIL 89]. They are designed with their most frequent users in mind: *process-plant maintenance people*. Thus, the most frequent programming languages are based on *ladder* or *logic diagrams* and *boolean algebra*. When the local control is of greater complexity, the above kinds of languages may not be well adapted (in fact a particular

hardware implementation is being transposed into a program, instead of a direct coding of the functionalities!). Some PLCs incorporate technology independent languages based on graphical formalisms adapted to the specification of sequential and concurrent systems. Among these formalisms, state diagrams and algorithmic state machines (ASM) are popular for sequential systems. For concurrent systems, the Grafcet is a CEI (Commission Electrotechnique International) standard derived from Petri Nets, and it is widely used.

For the definition of the Grafcet, the reader is referred to the literature (e.g. [VAL 78, BLA 79, DAV 89]). Structurally speaking the Grafcet (*Graphe de Commande Etape-Transition*), is analogous to place/transition nets, if places are substituted by "etapes". A Grafcet is an interpreted "logic" graph in which the "etapes" work as *"flip-flops"* (in P/T nets the places work as *"counters"*). So the following happens:

1. "Etapes" are by definition 1-bounded: *active* or *inactive*. The firing of the input transition of an active "etape" is possible but does not change its state.
2. If an effective conflict exists, the transitions in conflict can be *simultaneously* fired (this dangerous peculiarity is known as *"interpreted parallelism"*).

The above facts destroy most of P/T net analysis techniques (p-invariants, reduction rules, etc.). Therefore, it is always suggested that the interpreted parallelism should not be used. If this constraint is assumed, even if some differences remains, in practice the Grafcet "can be viewed" as a normalization of (contact-free) condition/event nets for switching automata applications (in particular for PLC based implementations). Actually, Grafcets are widely used in industry.

Implementation concepts and techniques for some P/T nets provided with interpretations tailored to the switching automatisms applications domain have been presented in several works (see, for example, [SIL 83, ALB 86, ATA 86, COL 86, MUR 86, CRO 87]).

5 Conclusions.

Just by having a look at the references, it is clear that the application of Petri net theory to FMS is a very rich domain. It is important to point out that the time for industrial application on a large scale seems to be close because papers written by people from industry describing actual applications can be found [BER 88, DUB 89, MUR 84].

Let us now briefly sum up some advantages of this approach:

- partial orders among events can be described, to meet with flexibility requirements,
- states and events are explicitly represented,
- a unique family of tools is used throughout the specification, modeling, qualitative validation and performance evaluation, implementation and operational process,
- the same family of tools is used within various functions of an FMS and at various levels (scheduling, global coordination and local control). It represents an important aid for integrating the whole system,
- an accurate, formal description of the synchronization mechanisms is provided, something essential to achieve security.

From scheduling to real-time control, Petri net theory offers solutions for design, performance evaluation and implementation. For example, the interested reader is pointed to the invited sessions on Petri Nets and Flexible Manufacturing [GEN 88] [SIL 87b]. Nevertheless much work is still needed to derive structured net models (i.e. how to compose, how to refine, ...), to have efficient algorithms to compute performance even for transient behaviors (i.e. non steady-state), to produce efficient and distributed code for control purposes, to dress a more complete scheduling theory, ...

Additional issues now concern systems which have to be highly flexible and may have abnormal behaviors including diagnosis in case of disruption and system reconfiguration. It is a domain where knowledge representation has to be sophisticated and sometimes has to be able to take into account uncertainty. As a matter of fact, the introduction

into software modules of the *know how* of decision makers which operate in real-time in flexible systems and have to take into account human factors is not easy. It is the reason why much work now concerns hybrid approaches using Petri nets and A.I. techniques.

Acknowledgments: The authors are grateful for the helpful comments of R. David, J. Erschler, P. Freedman, J.C. Gentina, P. Muro, J.L. Villarroel, K. Voss and many other persons.

6 References.

6.1 References on general notions in manufacturing systems.

ALB 81 J.S. Albus, A.J. Barbara, R.N. Nagel: Theory and practice of hierarchical control. *Proc. of 23rd IEEE Computer Society Int. Conf.* (September).

ALB 82 J.S. Albus, C. Maclean, A.J. Barbara, M. Fitzgerald: An architecture for real-time control in a manufacturing facility. *IFAC Conference on Information control problems in manufacturing technology*, Maryland, USA.

AXS 81 S. Axsater: Aggregation of Product Data for Hierarchical Production Planning. *Operations Research*, Vol 29, No 4.

BAL 69 Balas E., Machine sequencing via disjunctive graphs: an implicit enumeration algorithm, *Oper. Res.* 17 No 6.

BAL 89 K.E. Baldwin: Autonomous Manufacturing Systems.*IEEE International Symposium on Intelligent Control*, Albany N.Y., August, pp.214-220.

BAN 84 J. Banks and J.S. Carson: *Discrete-event system simulation*. Prentice-Hall, Englewood Cliffs, New Jersey.

BEL 85 G. Bel, E. Bensana, D. Dubois: Un systeme d'ordonnancement previsionnel d'atelier utilizant des connaissances theoriques et pratiques. *Les systemes experts et leurs applications*. Avignon.

BEL 88 G. Bel, E. Bensana, D. Dubois, J. Erschler, P. Esquirol, *A multi-knowledge-based approach to industrial job-shop scheduling*, Expert System design and management of Manufacturing Systems, Taylor and Francis Edition, p. 207-246.

BEN 86 E. Bensana, M. Corrège, G. Bel, D. Dubois: An expert system approach to industrial job-shop scheduling. *IEEE International Conference on Robotics and Automation*, San Francisco, April.

BIT 77 G.R. Bitran, A.C. Hax: On the Design of Hierarchical Production Planning Systems. *Decision Sciences, Vol 8, No 1*, pp. 28-54.

BIT 82 G.R. Bitran, E.A. Haas, A.C. Hax: Hierarchical production planning: a two stage system. *Operations research, Vol 30, No 2*, pp. 231-251.

BRU 86 G. Bruno, A. Elia, P. Laface: A Rule-Based System to Schedule Production. *Computer*. Vol 19, No 7, pp. 32-39.

BUF 79 E.S. Buffa, J.G. Miller: *Production-Inventory Systems: Planning and Control*, Richard D. Irwin Inc. 1968-1972-1979.

BUZ 86 J.A. Buzacott, D.D. Yao: Flexible manufacturing systems: a review of analytical models. *Management Science, Vol 32, No 7*, July, pp. 890-905.

CAR 88 J. Carlier et Ph. Chretienne: *Problèmes d'Ordonnancement. Modelisation, complexité et algorithmes*. Masson, Paris.

DAI 88 J.N. Daigle, A. Seidmann and J.R. Pimentel: *Special issue on Manufacturing Networking. IEEE Network, Vol 2, No 3*, May.

DAV 88 W. Davis, A. Jones: A Real-time Production Scheduler for a Stochastic Manufacturing Environment. *International Journal of Computer Integrated Manufacturing, Vol 1, No 2*, pp.101-112.

DAV 89 W. Davis, A. Jones *A functional approach to designing architectures for CIM. IEEE transactions on Systems, Man, and Cybernetics, Vol 19, No 2*, April, pp.164-174.

ERS 79 Erschler J., Fontan G., Roubellat F., Potentiels sur un graphe non conjonctif et analyse d'un problème d'ordonnancement à moyens limités, *RAIRO*, 13 No 4.

ERS 86a J. Erschler, F. Roubellat, V. Thomas: Real-time Production Scheduling for Parts with Limit Times. *TIMS/ORSA Joint National Meeting*, Los Angeles, April.

ERS 86b J. Erschler, P. Esquirol: Decision-aid in job-shop scheduling: a knowledge based approach. *IEEE International Conference on Robotics and Automation* San Francisco, April.

ERS 88 J. Erschler, G. de Tersac: Flexibilité et rôle de l'opérateur humain dans l'automatisation intégrée de production. *Rapport LAAS No 88137*, March.

EVA 88 J.B. Evans: *Structures of discrete event simulation: An introduction to the engagement strategy*. Ellis Horwood Limited Chichester.

FOX 83 M. Fox: Constraint-Directed Search: A Case Study of Job Shop Scheduling. *Ph. D. Diss. Computer Science Dept. Carnegie-Mellon University*, Pittsburg.

FRE 82 S. French: *Sequencing and Schedulig: An introduction to the Mathematics of the Job-Shop*. Ellis-Horwood, Chichester.

GER 89 S.B. Gershwin: Hierarchical Flow Control: a Framework for Scheduling and Planning Discrete Events in Manufacturing Systems. *IEEE Proceedings Special Issue on Discrete Event Systems, Vol 77, No 1*,January, pp. 195-209.

GM 85 General Motors:*Manufacturing Automation Protocol (MAP), a communication network for Open Systems Interconnection (OSI)*, Document specification, February.

GRA 81 S.C. Graves: A review of production scheduling. *Operations Research*, Vol 29, No 4, July-August, pp. 646-675.

HAM 85 I. Ham, K. Hitami, T. Yoshida: *Group technology applications in production management*. Kluwer Publishing Company, Boston MA.

HAT 85 J. Hatvany: Intelligence and Cooperation in Heterarchical Manufacturing Systems. *Robotics and Computer Integrated Manufacturing, Vol 2, No 2*, pp.101-104.

INA 89 K.M. Inan, P.P. Varaiya: *Algebras of Discrete Event Models. IEEE Proceedings Special Issue on Discrete Event Systems, Vol 77, No 1*, January, pp. 24-38.

JON 85 A. Jones, N. Whitt (Eds): *Proceedings on Factory Standards Model Conference*. National Bureau of Standards, Gaithersburg, Maryland, USA.

JON 89 A. Jones, A. Saleh: *A decentralized Control Architecture for Computer Integrated Manufacturing Systems, IEEE International Symposium on Intelligent Control*, Albany N.Y., August, pp.44-49.

KET 89 M.G. Ketcham, R.E. Shannon, G.L. Hogg: Information structures for simulation modeling of manufacturing systems. *Simulation*, February, pp.59-67.

KRE 86 W. Kreutzer: *System simulation. Programming styles and languages*. Addison-Wesley, Sidney.

McK 89 K.N. McKay, J.A. Buzacott and F.R. Safayeni: The Scheduler's Information System: What is going on? Insights for Automated Environnements. *IFAC Symp. on Control Problems in Manufacturing Technology, INCOM'89*, Madrid, September, pp. 405-409.

OGR 86 P.J. O'Grady, U. Menon: A Concise Review of Flexible Manufacturing Systems and FMS Literature. *Computers in Industry, Vol 7*.

ORL 75 J. Orlicky:*Materials requirement planning*. McGraw Hill, New York.

PAN 77 S.S. Panwalkar and W. Iskandar: A survey of scheduling rules. *Operations Research, Vol 25, No 1*, pp. 45-61.

PAR 86 Parallax: When is Pull Better than Push? *International Journal of Parallel Programming*, Vol 15, No 2, pp. 185-188.

PEG 86 C.D. Pegden: *Introduction ot SIMAN*. Systems Modeling Corporation, Pennsylvania.

PET 79 R. Peterson, E.A. Silver: *Decision Systems for Inventory Management and Production Planning*. John Wiley and Sons.

PRI 79a A.A.B. Pritsker and C.D. Pegden: *Introduction to Simulation and SLAM*. John Wiley, New York.

PRI 79b A.A.B. Pritsker: *Modeling and Analysis using Q-GERT Networks*. John Wiley, New York.

RAM 89 P.J.G. Ramadge, W.M. Wonham: The control of discrete event systems. *IEEE Proceedings (Special Issue on Discrete Event Systems), Vol 77, No 1*, January, pp. 81-98.

SCH 85 B.J. Schroer, J.T. Black, S.X. Zhang: Just-In-Time with Kanban, Manufacturing System Simulation on a Microcomputer.*Simulation, Vol 45, No 2*, August, pp. 62-70.

SUG 74 Y. Sugimori, K. Kusunoki, F. Cho, S. Uchikawa: Toyota production system and Kanban system: Materialization of just-in-time and respect-for-human system. *International Journal of Production Research, Vol 15*, pp. 553-564.

YEO 85 R.W. Yeomans, A. Choudry, P.J.W. Ten Hagen: *Design rules for a CIM system*. North Holland.

6.2 Some references concerning Petri nets.

BRA 83 G.W. Brams: *Reseaux de Petri: Theorie et pratique* (2 tomes). Masson, Paris.

BRA 86 W. Brauer, W. Reisig, G. Rozenberg (eds.): *Petri Nets* (2 vol). Advances in Petri Nets'86, LNCS 254 and 255. Springer-Verlag.

CAM 89 J. Campos, G. Chiola, J.M. Colom, M. Silva: Tight polynomial bounds for steady-state performance of marked graphs. *IEEE Workshop on Petri Nets and Performance Models, PNPM'89*, Kyoto, December.

CAM 90 J. Campos, M. Silva: Steady-State performance evaluation of totally open systems of markovian sequential processes. *IFIP Conference on Decentralized Systems (C. Girault, ed.)*. Elsevier (North Holland).

CAR 85 J.Carlier, Ph. Chretienne, J.C. Girault: Modelling scheduling problems with timed Petri Nets. *Advances in Petri Nets'84 (G. Rozenberg, ed.)*, LNCS 188, Springer-Verlag, Berlin, pp. 62-82.

CAR 88a J.Carlier, Ph. Chretienne: Timed Petri net schedules. *Advances in Petri Nets'88 (G. Rozenberg ed.)*, LNCS 340, Springer-Verlag, Berlin, pp. 62-84.

CHI 87 G. Chiola: A graphical Petri net tool for performance analysis. *3rd Int. Workshop on Modeling Techniques and Performance Evaluation*, AFCET, March, Paris.

CHR 86 Ph. Chretienne: Timed Event Graphs: a Solution to the Minimum Reachability Time between two States. *Journal of Systems and Software. Vol 6, No 1 and 2*.

DUT 89 C. Dutheillet, S. Haddad: Regular stochastic Petri Nets. *10th. Int. Conf. on Applications and Theory of Petri Nets*. Bonn, June.

FLO 89 G. Florin, S. Natkin: Solutions en forme produit matricielle pour les reseaux de files d'attente synchronisees fermees monovaluees. *IEEE Workshop on Petri Nets and Performance Models*, PNPM'89, Kyoto, December.

HAC 72 M.H.T. Hack: Analysis of production schemata by Petri nets. *Master of Science Dissertation Thesis, MIT*, February.

MAG 87 J. Magott: New NP-Complete problems in Performance evaluation of Concurrent Systems Using Petri Nets. *IEEE Trans. on Software Engineering, Vol SE-13*, No 5, May, pp. 578-181.

RAM 80 C.V. Ramamoorthy and G.S. Ho: Performance evaluation of asynchronous concurrent systems using Petri nets. *IEEE Trans. on Software Engineering, Vol SE-6, No 5*, September, pp. 440-449.

SIL 85 M. Silva: *Las redes de Petri en la Automática y la Informática*. Ed. AC, Madrid.

SIL 87a M. Silva: Towards a synchrony theory for P/T nets. *Concurrency and Nets (K. Voss et al., eds)*. Springer-Verlag, pp. 435-460.

SIL 88 M. Silva, J.M. Colom: On the computation of structural synchronic invariants in P/T nets. *Advances in Petri Nets (G. Rozenberg, ed.) LNCS, No 340*, Springer-Verlag, Berlin, pp. 386-417.

TAU 88 D. Taubner: On the implementation of Petri nets. *Advances in Petri Nets'88 (G. Rozenberg, ed.), LNCS 340*, pp. 418-439.

VAL 78 R. Valette, M. Diaz: Top-down formal specification and verification of parallel control systems. *Digital Processes, Vol 4, No 3*, pp. 181-189.

VAL 79 R. Valette: Analysis of Petri nets by stepwise refinements *Journal of Computer and System Sciences, Vol 18, No 1*, February, pp. 35-46.

6.3 References related to FMS and Petri nets.

ALA 85 P. Alanche, K. Benzakour, F. Dolle, P. Gillet, P. Rodrigues, R. Valette: PSI: a Petri net based simulator for flexible manufacturing systems.*Advances in Petri nets 1984, LNCS 188*. Springer Verlag, pp. 1-14.

ALB 86 M. Albarran: *Realization of sequencers through parallel flow control graphs. IFAC-IFIP SOCOCO'86 (Software for Computer Control)* Graz, May, pp.237-241.

ALJ 87 R. Y. Al-Jaar and A. A. Desrochers: *Petri nets in Automation and manufacturing, RAL'99, Robotics and Automation Laboratory RPI*, Troy, NY 12180-3590.

ALJ 88a R. Y. Al-Jaar, A.A. Desrochers: A survey of Petri nets in automated manufacturing systems. *IMACS World Congress Vol 2*, Paris, June, pp. 503-510.

ALJ 88b R. Y. Al-Jaar, A.A. Desrochers: A modular approach for the performance analysis of automated manufacturing systems using generalized stochastic Petri nets. *Research report, Rensselaer Polytechnic Institute*.

ALL 85 H. Alla, P. Ladet, J. Martinez, M. Silva: Modeling and validation of complex systems by coloured Petri nets, Application to a FMS. *Advances in Petri Nets 1984, LNCS 188* Springer Verlag, pp.15-31.

ALL 86a H. Alla, P. Ladet, F. Martin, J. Martínez, M. Silva: Specification de la commande des Ateliers Flexibles à l'aide de réseaux de Petri colorés. *Journées d'Etudes: Méthodes et outils modernes de conception et d'exploitation de la commande des procédés continus complexes*. Montpellier (France), May, pp. 85-96.

ALL 86b H. Alla, P. Ladet: Coloured Petri nets: a tool for modeling validation and simulation of FMS. *Flexible Manufacturing Systems: methods and studies, ed. by A. Kusiak.* Elsevier Science Publishers (North Holland).

ATA 86 H. Atabakhche, D.S. Barbalho, R. Valette, M. Courvoisier: From Petri net based PLCs to knowledge based control. *International Conference on Industrial Electronics, Control and Instrumentation, IECON 86*. Milwaukee, Wisconsin, Sept. 29-Oct 3, pp.817-822.

BAL 86 G. Balbo et al.: Generalized stochastic Petri nets for the performance evaluation of FMS. *IEEE Int. Conference on Robotics and Automation*. Raleigh, April, pp.1013-1018.

BEC 86 C. L. Beck, B.H. Krogh: Models for simulation and discrete control of manufacturing systems. *IEEE International Conference on Robotics and Automation*. San Francisco, California, pp.305-310.

BER 88 R. Berchi, G. Frosi: Design of the material handling system for a MFG line. *AIRO Workshop Coordination Management by Means of Petri Nets*. Modena, Italy, April.

BLA 79 M. Blanchard: Automatismes logiques: Grafcet ou réseau de Petri? *Le Nouvel Automatisme*, Mai, pp. 45-52.

BOU 87 J.P. Bourey, D. Corbeel, E. Craye, J.C. Gentina: Utilisation des réseaux de Petri structurés adaptatifs colorés dans l'analyse et la synthése du contrôle hiérarchisé de processus discontinus: les modéles de description. *RAIRO APII, Vol 21, No 4*, Juillet, pp. 343-362.

BRU 86 M. Bruns, H. Rake: Methods and tools for the development of software for complex realtime systems. *IFAC/IFIP SOCOCO'86 (Software for Computer Control)*. Graz, May, pp.231-236.

BRU 85 G. Bruno, P. Biglia: Performance evaluation and validation of tool handling in Flexible Manufacturing Systems using Petri Nets. *IEEE Int. Workshop on Timed Petri Nets*, Torino, July, pp. 64-71.

BRU 86a G. Bruno, A. Balsamo: Petri net based object oriented modelling of distributed systems. *Object Oriented Programming Systems Language and Applications, OOPSALA'86*. September, pp. 284-293.

BRU 86b G. Bruno, G. Marchetto: Process-translatable Petri nets for the rapid prototyping of process control systems. *IEEE Trans. Software Eng., Vol SE-12, No 2*, February, pp. 346-357.

BRU 87 G. Bruno, M. Morisio: Petri net based simulation of manufacturing cells. *IEEE International Conference on Robotics and Automation*. Raleigh, NC, pp.1174-1179.

CAR 89 J. Cardoso, R. Valette, D. Dubois: Petri nets with uncertain markings. *10th International Conference on Application and Theory of Petri Nets*, Bonn, June, pp. 35-51.

CAS 87 E. Castelain, D. Corbeel, J.C. Gentina: Apport de l'intelligence artificielle dans la réalisation d'un simulateur de réseaux de Petri adaptatifs colorés: application à la validation de la commande des processus industriels. *RAIRO APII, Vol 21, No 4*, Juillet, pp. 363-375.

CAS 89a E. Castelain, J.P. Bourey, E. Craye, J.C. Gentina: Introduction to a central tool in the design chain of the model of the control system and in the predemensionment step of flexible manufacturing cells. *IEEE International Symposium on Intelligent Control*, Albany N.Y., September, pp.221-226.

CAS 89b E. Castelain, J.P. Bourey and J.C. Gentina: Prototyping of FMS from the Design of a Pregraph Based on Some Extended Petri-Nets. *6th IFAC Symp. on Information Control Problems in Manufacturing Technology, INCOM'89*. Madrid, September, pp. 437-442.

CAV 81 B. Cavanna, F. Dolle, M. Moalla: Outils C.A.O. pour l'analyse, la spécification et la réalisation de l'automatisation d'équipements de production mécanique. *3e Journées Scientifique et Techniques de l'ADEPA*. Toulouse France, Juin.

CHO 80 D. Chocron, E. Cerni: A Petri net based industrial sequencer. *IEEE International Conference and Exhibition on Industrial Control and Instrumentation*. March 1980, pp.18-22.

COH 85 G. Cohen, D. Dubois, J.P. Quadrat, M. Viot: A linear system theoretic view of discrete event processes and its use for performance evaluation in manufacturing. *IEEE Transactions on Automatic Control, Vol AC-30, No 3*, March, pp.210-220.

COH 89 G. Cohen, P. Moller, J.P. Quadrat, M. Viot: Algebraic Tools for the Performance Evaluation of Discrete Event Systems. *Proceedings of the IEEE, Vol 77, No 1*, January, pp. 39-58.

COL 86 J.M. Colom, M. Silva, J.L. Villarroel: On software implementation of Petri nets and colored Petri nets using high-level concurrent languages. *7th European Workshop on Application and Theory of Petri nets Oxford*, July, pp.207-241.

COL 87 J.M. Colom, J. Martínez, M. Silva: Packages for validating discrete production systems modeled by Petri nets. *Applied Modeling and Simulation of Technological Systems (P. Borne and S. Tzafestas eds.)*. North Holland Publishers, pp.529-536.

COR 85 D. Corbeel, J.C. Gentina, C. Vercauter: Application of an extension of Petri nets to modelization of control and production processes. *6th European Workshop on Applications and Theory of Petri nets*. Espoo, Finland, June, pp. 53-74.

COU 83 M. Courvoisier, R. Valette, JM. Bigou, P. Esteban: A programmable logic controller based on a high level specification tool. *IEEE Annual Conference on Industrial Electronics, IECON 83*. San Francisco, USA November, pp. 174-179.

CRO 87 D. Crockett, A. Desrochers, F. DiCesare, T. Ward: Implementation of a Petri net controller for a machining workstation. Proceedings of the *IEEE Conference on Robotics and Automation*. Raleigh, NC, March 30th-April 3, pp.1861-1867.

DAV 89 R. David et H. Alla: *Du Grafcet aux Réseaux de Petri*. Hermes, Paris.

DIL 87 A. DiLeva, P. Giolito: High-level Petri nets for production system modelling. *8th European Workshop on Application and Theory of Petri nets*. Zaragoza, Spain, June, pp. 381-386.

DIM 89 M. DiMascolo, Y. Frein, Y. Dallery, R. David: Modeling of Kanban Systems using Petri Nets. *3 ORSA/TIMS Conference on Flexible Manufacturing Systems (Ed. K. Stecke and R. Suri)*. Elsevier Science Publishers, pp.307-312.

DUB 83 D. Dubois, K. Stecke: Using Petri nets to represent production processes. *22nd IEEE Conference on Decision and Control San Antonio*, Texas, pp.1062-1067.

DUB 88 D. D. Dubois and K. Stecke: Dynamic Analysis of repetitive decision-free discrete event processes: (I) The algebra of Timed Marked Graphs; (II) Algorithmic issues and application to production systems. *Graduate School of Business Administration University of Michigan, Working papers 542 and 543*.

DUB 89 D. Dubois: A flexible workshop design and optimization using SEDRIC: A Petri net simultor. *10th Int. Conf. on Application and Theory of Petri Nets*. Bonn, June, pp.

FAV 85 J. Favrel and K.H. Lee: Modelling, analyzing, scheduling and control of Flexible Manufacturing Systems by Petri Nets. *Modelling Production Management Systems (P. Falster and R.B. Mazunder, eds.)*. Elsevier Science Publ., pp. 223-243.

FLE 89 H. Fleischack, A. Weber: Rule-based programming, Predicate/transition nets and the modeling of office procedures and flexible manufacturing systems. *10th International Conference on Application and Theory of Petri Nets*. Bonn, June.

FRE 87 P. Freedman, A. Malowany: Sequencing Tasks within a Robotics Workcell: from feasibility to optimality. *IEEE Pacific Rim Conf.*, Victoria, B.C., June.

FRE 88a P. Freedman, A. Malowany: The Analysis and Optimization of Repetition within Robot Workcell Sequencing Problems. *IEEE Int. Conf. on Robotics and Automation*. Philadelphia, Penn., April.

FRE 88b P. Freedman, A. Malowany: *SAGE: A Decision Support System for the Sequencing of Operations within a Robotics Workcell, it Decision Support Systems*, Elsevier Science (North Holland), Vol 4, pp. 329-343.

FRE 89 P. Freedman: Characterizing Repetition in Workcell Applications and the Implications for Sequence Optimization. *4th Int. Conference on Advanced Robotics*. Columbus, Ohio, June.

GEN 87 J.C. Gentina and D. Corbeel: Coloured Adaptive Structured Petri Net: A tool for the Automatic Synthesis of Hierarchical Control of Flexible Manufacturing Systems. In [SILV 87], pp. 1166-1173.

GEN 88 J.C. Gentina (organizer): *Invited sessions: Application of Petri Nets to Flexible Manufacturing. 12th IMACS World Congress*. Paris, pp. 479-561.

GRO 86 E. Grötsch: Graphische Programmieroberflächen für Steurungssysteme. *Automatisierungstechnische Praxis atp, Vol 28, No 1*, pp. 27-31.

HAS 85 K. Hasegawa, H. Ohno: On programming of conventional programmable controller by using mark flow graph. *IEEE ISCAS*, Kyoto Japon, June, pp.933-936.

HAS 86 M. Hasegawa, Y. Sakaue: Programming system for work cell. *IEEE International Conference on Industrial Electronics, IECON 86*. Milwaukee, WI, USA, Sept 29-Oct 3, pp. 443-448.

HAS 88 K. Hasegawa, K. Takahashi, P. E. Miyagi: Application of the mark flow graph to represent discrete event production systems and system control. *Trans. of the Society of Intrument and Control Engineers (SICE Japan) Vol 24, No 1*, January, pp.69-75.

HIL 88 H. Hillion: Timed Petri Nets and Application to Multi-Stage production systems. *9th European Workshop on Applications and Theory of Petri Nets*. Venice, pp. 164-182.

HIL 89 H. P. Hillion, J. Proth: Performance evaluation of job-shop systems using timed event-graph. *IEEE Trans. on Automatic Control, Vol 34, No 1*, January 1989, pp. 3-9.

HOE 89 A. Hoermann: A Petri net based control architecture for a multi-robot system. *IEEE International Symposium on Intelligent Control*, Albany N.Y., September, pp.493-498.

HOL 85 D. Hollinger: Utilisation pratique des réseaux de Petri dans la conception des systèmes de production. *TSI (Technique et Science Informatique), Vol 4, No 6*, pp. 509-522.

HOR 89 Axel Horns: Job Shop control under influence of Chaos Phenomena. *IEEE International Symposium on Intelligent Control*. Albany N.Y., September, pp.227-232.

KAM 86 M. Kamath, N. Viswanadham: Applications of Petri net based models in the modelling and analysis of flexible manufacturing systems. *Proceedings of the IEEE Conference on Robotics and Automation*. San Francisco, California, pp.312-317.

KOM 84 N. Komoda, K. Kera, T. Kubo: An autonomous, decentralized control system for factory automation. *Computer*, December, pp.73-84.

KOM 85 N. Komoda, T. Murata. K. Matsumoto: Petri net based controller: SCR and its applications in factory automation. *IEEE ISCAS*, Kyoto, Japon, June, pp. 937-940.

MAR 87 F. Martin, H. Alla: Discrete event simulation by timed Coloured Petri nets. *3rd International Conference on Advances in Production Management Systems*. August.

MAR 83 J. Martínez, M. Silva: A package for computer design of concurrent logic control systems. *IFAC/IFIP SOCOCO (E.A. Puente and G. Ferrate eds.)*. Pergamon Press, pp. 243-248.

MAR 85 J. Martínez, M. Silva: A language for the description of concurrent systems modelled by coloured Petri nets: application to the control of flexible manufacturing systems. *Languages for Automation (S-K Chang ed.)*. Plenum Publishing Co., pp. 369-388.

MAR 86a J. Martínez, M. Silva: Nuevos métodos para la especificación del control de sistemas flexibles de fabricación. *Automática e Instrumentación, No 156*, pp.185-195.

MAR 86b J. Martínez, H. Alla, M. Silva: Petri nets for the specification of FMSs. *Modeling and Design of Flexible Manufacturing Systems (A. Kusiak, ed.)*. Elsevier Science Pub. pp.389-406.

MAR 87 J. Martínez, P. Muro, M. Silva: Modelling, validation and software implementation of production systems using high level Petri nets. *IEEE Conference on Robotics and Automation*. Raleigh, NC, March 30th-April 3, pp.1180-1185.

MAR 88 J. Martínez, P.R. Muro, M. Silva, S.F. Smith, J.L. Villarroel: Merging artificial intelligence techniques and Petri nets for real-time scheduling and control of production systems. *12th IMACS World Congress on Scientific Computation*. Paris, July, Vol 3, pp. 528-531.

MAS 81 R. Masuda, K. Hasegawa: Mark flow graph and its application to complex sequential control system. *13th Hawai Int. Conf. System Science*. Hawai, pp.194-203.

MER 86 A. Merabet: Synchronization of operations in a flexible manufacturing cell: the Petri net approach. *Journal of manufacturing systems, Vol 5, No 3*, pp. 161-169.

MIC 89 A. Micovsky, L. Sesera, M. Veishab, M. Albert: TORA The language of the system for rapid prototyping of discrete processes control systems *Artificial Intelligence and Information-Control Systems of Robots 89*, Elsevier Science Publishers p.315-318.

MIY 88 P. E. Miyagi, K. Hasegawa, K. Takahashi: A programming language for discrete event production systems based on production flow schema and mark flow graph. *Trans. of the Society of Intrument and Control Engineers (SICE Japan) Vol 24, No 2*, February, pp.77-84.

MOA 85 M. Moalla: Réseaux de Petri interprétés et Grafcet. *T.S.I.(Technique et Science Informatique), Vol 4, No 1*, Ed. Dunod.

MUR 84 T. Murata, N. Komoda, K. Matsumoto: A Petri net based factory automation controller for flexible and maintainable control specification. *IEEE Annual Conference on Industrial Electronics, IECON 84*, pp. 362-366.

MUR 86 T. Murata, N. Komoda, K. Matsumoto: A Petri net based controller for flexible and maintainable sequence control and its applications in factory automation. *IEEE Transactions on Industrial Electronics, Vol IE-33, No 1*, February, pp.1-8.

MUR 89 P. Muro, J.L. Villarroel, J. Martínez, M. Silva: A knowledge representation environement for manufacturing control system design and prototyping. *6th IFAC-IFORS-IMACS Symposium on Information Control Problems in Manufacturing Technology*. Madrid, September, pp. 585-590.

NAR 85b Y. Narahari, N. Viswanadham: A Petri net approach to the modelling and analysis of flexible manufacturing systems. *Annals of Operations Research, Vol 3*, pp. 449-472.

NAR 86 Y. Narahari, N. Viswanadham: Coloured Petri net models for generalized flexible manufacturing systems. *7th. European Workshop on Application on Theory of Petri Nets*. Oxford June-July, pp. 243-263.

RAV 86 R. Ravichandran and A.K. Chakravarty: Decision Support in Flexible Manufacturing Systems Using Timed Petri Nets. *Journal of Manufacturing Systems Vol 5, No 2*, pp. 89-101.

SAH 87 A. Sahraoui, H. Atabakhche, M. Courvoisier, R. Valette: Joining Petri nets and knowledge based systems for monitoring purposes. *IEEE International Conference on Robotics and Automation*, Raleigh, NC, pp.1160-1165.

SIL 79 M. Silva, R. David: Synthèse programmé des automatismes logiques décrits par réseaux de Petri: Une méthode de mise en oeuvre sur microcalculateurs. *Rairo-Automatique, Vol 13, No 4*, November, pp. 369-393.

SIL 83 M. Silva, S. Velilla: Programmable Logic Controllers and Petri nets: a comparative study. *IFAC Conference on Software for Computer Control (E.A. Puente and G. Ferrate eds.).* Pergamon Press, pp. 83-88.

SIL 86 M. Silva, J. Martínez: A software environment for system design with HLPN and their implementations. *Int. Seminar on Applicability of Petri nets to Operation Research*. Milano (Italy), May, pp.95-127.

SIL 87b M. Silva and T. Murata (organizers): *Invited sessions: Petri Nets and Flexible Manufacturing. IEEE Int. Conf. on Robotics and Automation*. Raleigh, pp. 999-1018 and 1160-1185.

SIL 89 M. Silva: Logic Controllers. *IFAC Int. Symp. on Low Cost Automation*, Milano, November, Vol II, pp. 157-165bis.

TON 86 H. K. Tönshoff, A. Horns: Petri-Netze als spezielle Inferenz-système und ihre Anwendung in der Werkstattsteurung und simulation. *IPA-IAO Forschung und Praxis T6 Produktionsplanung, Produktionssteuerung in der CIM-Realisierung*. Springer Verlag.

VAL 78 R. Valette: Etude comparative de deux outils de représentation: Grafcet et Réseaux de Petri. *Le Nouvel Automatisme, No 3*, December, pp. 377-382.

VAL 82 R. Valette, M. Courvoisier, D. Mayeux: Control of flexible production systems and Petri nets. *Application and Theory of Petri nets, Informatik-Fachberichte, No 66*. Springer Verlag, pp. 264-277.

VAL 83 R. Valette, M. Courvoisier, JM. Bigou, J. Albukerque: A Petri net based programmable logic controller. *IFIP First International Conference on Computer Applications in Production and Engineering, CAPE 83*. Amsterdam, April.

VAL 85a R. Valette, V. Thomas, S. Bachmann: SEDRIC un simulateur à événements discrets basé sur les réseaux de Petri. *RAIRO/APII, Vol 19, No 5*, pp. 423-436.

VAL 85b R. Valette, M. Courvoisier, H. Demmou, JM. Bigou, C. Desclaux: Putting Petri nets to work for controlling flexible manufacturing systems. *IEEE ISCAS*. Kyoto, Juin, pp. 929-932.

VAL 87 R. Valette: Nets in production systems. *Petri Nets: Applications and Relationships to Other Models of Concurrency, LNCS 255*, Springer Verlag, pp. 191-217.

VAL 88a R. Valette: Coordination problems in FMS control systems. *A.I.R.O. workshop on Coordination management by means of Petri nets*. Modena, Italy, April.

VAL 88b R. Valette, J. Cardoso, H. Atabakhche, M. Courvoisier, T. Lemaire: Petri nets and production rules for decision levels in FMS control. *12th IMACS World Congress on Scientific Computation*. Paris, Juillet, pp.522-524.

VAL 89 R. Valette: Monitoring manufacturing systems by means of Petri nets with imprecise markings. *IEEE International Symposium on Intelligent Control*. Albany N.Y., September, pp.233-238.

VEL 88 S. Velilla, M. Silva: The spy: a mechanism for safe implementation of highly concurrent systems. *15th IFAC/IFIP Workshop on real-time programming*. Valencia (Spain), May, pp.95-102.

VIL 88 J.L. Villarroel, J. Martínez, M. Silva: GRAMAN: A Graphic System for Manufacturing System Design. *IMACS Symp. on System Modelling and Simulation*. Elsevier Science Publ. (S. Tzafestas, A. Eisinberg, L. Carotenuto, eds), pp. 311-316.

VIS 87 N. Viswanadham, Y. Narahari: Coloured Petri nets models for automated manufacturing systems. *IEEE Conference on Robotics and Automation*. Raleigh, NC, March 30th-April 3, pp.1985-1990.

WAN 89 F.Y. Wang, G. Saridis: The coordination of intelligent robots: a case study. *IEEE International Symposium on Intelligent Control*. Albany N.Y., September, pp.506-511.

ZUR 89 Richard Z. Zurawski, Tharam S. Dillon: Specification, verification and performance evaluation of flexible manufacturing systems, *IFIP Conference on Computer Applications in Production and Engineering CAPE 89*, Tokyo p.517-525.