Conceptual and Pragmatic Tools for Design and Control of Manufacturing Systems

Petri Nets and Ramadge-Wonham Discrete Event Systems

Marco Antoniotti
Robotics Lab
Courant Institute of Mathematical Science
New York University
marcoxa@cs.nyu.edu

July 14, 1993

Contents

List of Figures

1 Introduction

In this survey I explore two conceptual tools that have been used in recent (and not so recent) years to build *models* of and to *control* various **Manufacturing Systems** and processes.

The paper is organized by three logical subtopics:

1. Manufacturing Systems (MS) basic definitions and typical problems appearing in the field;

2. Petri Nets (PN) concepts and tools and

3. Ramadge-Wonham Discrete Event Systems.

1.1 Manufacturing Systems

Manufacturing Systems are a very important sector of any form of economy. It is not surprising that many research efforts are spent in the field, in order to ameliorate their implementation and management.

Our aim is to investigate some *computational frameworks* (or *models*) that could be interpreted and used to (partially) perform the tasks of

- *simulation* of an (arbitrary) MS;

- preventive *verification* of some of its properties;

- *evaluation* of some of its characteristic parameters.

- *control* of its behavior.

These tasks require well defined computational models, that should be easily equipped with an appropriate "semantics", capable of expressing several properties of MS's. Petri Nets and RW-Systems satisfy these requirements.

1.1.1 Petri Nets

The research on Petri Nets encompasses a wide variety of fields and approaches. The main characteristic of PN's is their inherent ability to model *concurrency* and *synchronization* in complex systems.

The exposition will follow a "genealogical tree" of Petri Nets models as they appeared in literature, with special attention to *High Level* models and to *modularity* and *time related* issues.

1.1.2 Ramadge-Wonham Discrete Event Systems

This relatively new approach to the modeling and control of Manufacturing Systems has its origin in the field of "Control Theory". Using well known techniques and results from formal language theory, the creators of the approach develop interesting notions and are able to define methods and algorithms for the synthesis of "controllers" for Discrete Event (hence Manufacturing) systems .

As we will see, the two approaches share some common traits in the developments they have undergone and in the problems they address, both from theoretical and pragmatic viewpoints.

Needless to say, these are by no means the sole possible approaches to the modeling and control of Manufacturing Systems (and of Concurrent Systems in general). As an example, the field of Temporal Logic has many appeals as well as advocates.

Time, the Ultimate Tyrant, is not only the problem addressed by the researches that will be described here. It is also the main constraint on the resources of the writer.

2 Manufacturing Systems: Definitions and Typical Problems

Over the decades, many technological changes contributed to improve the quality (as well as the quantity) and variety of the goods produced by Manufacturing Systems. An historical account can be found in [WB88] and [Mit91] (chap. 2). In [WM89] and [Mer85], a thesis is put forth that the introduction of computers (themselves a product of a manufacturing process) do constitute a significant opportunity to further improve, if not radically change, the nature of the manufacturing activities.

This section contains some basic terminology used to indicate the components, processes and aims of Manufacturing Systems (MS for short) along with definitions of the typical problems arising in the domain. We also discuss several paradigms used to tackle the problems.

These definitions of tasks and problems will be used in the next section as taxonomical criteria to classify the conceptual tools under consideration.

2.1 Terminology and Definitions

I use the following, very broad, definition of Manufacturing System.

> *A* **Manufacturing System** *is a collection of interacting men and machines whose goal and function is to* produce *and/or* transform *goods of the most disparate sorts.*

The "generic place" where a Manufacturing System is set up is the **factory** or **shop floor**. The term *floor organization* will be used to distinguish also among different kinds of productions taking place in a factory.

As usual we can indicate other components of an MS, such as *input*, *output* and *internal behavior*, following a standard "black box" methodology. A MS has the following components:

- a set of **machines**;
- a group of **workers**;
- a set of **manufacturing processes**;
- a set of **inputs**;
- a set of **outputs**.

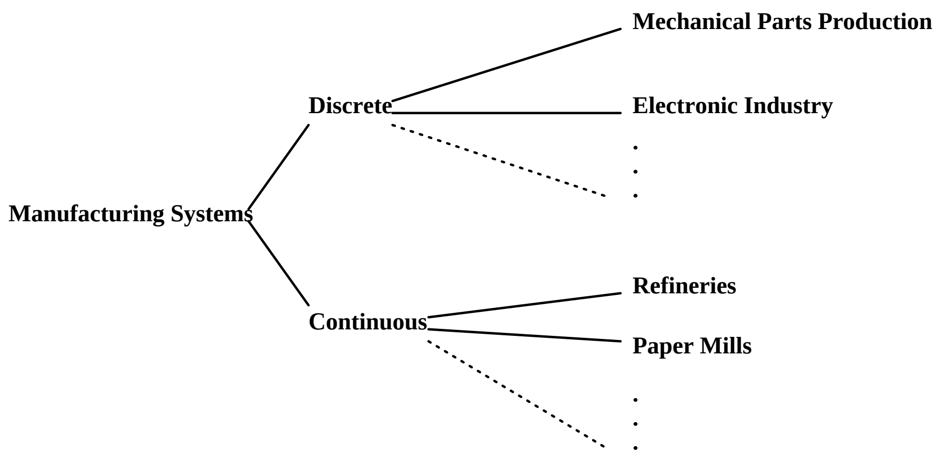


Figure 1: A Coarse Classification of Manufacturing Processes

The nature of the processes is used to classify the type of the MS. We are interested in **discrete** systems, whose main characteristic is that the materials handled and the division of the processes can be viewed as a collection of discrete entities. **Continuous** production systems, such as paper mills and oil refineries, conveniently do not fall under this classification. A metal cutting shop is therefore a discrete MS according to these definitions. Of course, there are mixed situations where part of the processing is continuous and part is discrete.

The set of machines is specifically designed for the process. *Physical* inputs and outputs are usually clearly identifiable. Moreover they are usually organized in *batches*, which actually constitute the elementary entities at a certain level of abstraction.

Human Presence on the Shop Floor

The workforce is *trained* to work with the given machines. The tasks assigned to humans vary according to several factors. Following this assignement, the major distinctions are then made between a "fully automated" and a "partially automated" factory floor.

The fully automated approach is the assumed goal of most of the research done in the field of factory automation [AGM88]. Yet the inherent complexity of the MS, makes more desirable the presence of human workers on the floor. The **supervisory** task becomes therefore central in the partially automated factory.

Flow Shops and Job Shops

The most important kind of production organization that historically imposed itself is named after its creator: Taylor. The main objective of the tayloristic factory organization is to optimize the work done by humans and machines in order to maximize the output of the system. This organization has been used specially in the situation of "mass production". The actual floor organization in this framework is the "production line"; unfinished parts are transported by a conveyor and *synchronously* acted upon by the human worker or by other machines. The optimization is achieved by subdividing each (synchronous) operation into elementary steps, each of approximately equal duration.

The tayloristic organization of the mass-production shop floor is recognized to be "unflexible". In a **flow shop** the production line can be split according to the necessities of the manufacturing process. An unfinished product may undergo several, but not necessarily all, transformations which are built in the plant. A flow shop is therefore capable of producing a "family" of similar products.

In a **job shop** the production "line" is not defined anymore. Groups of machines (or **workstations**), organized in **cells** are able to perform a set of different operations. The cells can receive parts from anywhere on the floor and send their product to any other cell. The sequence of operations is therefore not determined by the "physical layout" of the production line, rather it is dynamically mapped on a **production route** across the floor.

This arrangement makes the shop floor organization the most versatile and flexible, at the price of being very difficult to control and to use efficiently.

Flexible Manufacturing and Computer Integrated Manufacturing

The job shop is the most "flexible" organization for a factory floor. The term **flexible manufacturing system** (FMS) usually denotes a job shop organization, but not exclusively.

While there is no clear definition of *flexibility*, the term FMS usually denotes the capability of the system to deal with *long* and *short term* flexibility [SV89]. The first is seen as the capability to introduce a "new" product in the system without a major reorganization of the shop floor. The second refers to the possibility to deal with many different families of products at the "same" time.

The components of a FMS include (with respect to the definition of MS previously given) a set of *manufacturing cells* (which are themselves composed of *manufacturing workstations*) and a *transport subsystem* whose task is to shuffle parts of the produce among the cells on the shop floor.

The production route for each different product and the tasks assigned to each cell or machine have to be decided elsewhere. This is the core of the many complexities a FMS designer/implementor has to address.

The actual implementation of a FMS does not "formally" require the presence of a computerized support system. A FMS could be efficiently implemented by organizing procedures and protocols for the human workforce[1].

Computer integrated manufacturing (CIM). The presence of computers in the factory poses new problems for the structured organization of Manufacturing Systems. Most definitions of CIM that appeared in the literature (e.g. [WM89], [AGM88], [SV89], [Mer85]) agree that its main goal is the realization of a *common technical database* (TDB) which can be accessed through a *computer network*. The content of the database should serve as a support system for the diverse areas where computer applications are already common. Typical terms for different applications are *computer aided design* (CAD), *computer aided engineering* (CAE), *computer aided manufacturing* (CAM) and *computer aided process planning* (CAPP).

The realization of the TDB and the set up of the local network on the factory floor are by themselves challenging problems[2]. Yet they are only the building blocks upon which more sophisticated functionalities can be built. As an example, the interconnected computer environment should also serve as the substrate upon which to construct the more desirable and less understood "production manager" software[3].

2.2 Problems and Research Areas with respect to Computerized MS's

The research literature on MS's and related topics is extremely vast and fragmented in a myriad of levels and subfields. Historically, the first formal studies (dating back to the period between the two World Wars) con-

[1]A *Kanban* organization ([Mit91]), though not necessarily FMS related, is an example of this approach.

[2]As an example the whole notion of *group technology* has been defined with the objective of the TDB in mind.

[3]Note that CIM systems do not need to be "flexible". A production line or a flow shop may be fully computerized, while a FMS may (in principle) not.

centrated of the mathematical modeling of the MS's. The introduction of computers in the factory has produced new subfields addressing the *support systems* for various aspects of the manufacturing activity. The scope of such researches ranges from experimental applications to abstract formal models manufacturing processes.

One of the main research topics concerning MS's is the construction of a framework capable of dealing with both

- the **modeling** problem and
- the **control** problem.

The *modeling problem* is important because we want to reduce the complexity of implementation of a MS. The presence of a mathematical and/or symbolic *computational model*, is necessary in order to manage all the logical components of the system being analized and/or built. The "computational" nature of the models is necessary because we like to use computerized tools to carry out the modeling task.

The *control problem* deals with the "optimization" of an MS behavior in its "production regime". A typical embodiment of the control problem for MS's is the *real time scheduling* problem: i.e. how to promptly react to a change in the "demand" for the product.

The aim of this paper is to survey two computational tools with respect to their "usefulness" and "expressiveness" in performing some typical tasks arising from the modeling and control problems for MS's.

2.2.1 MS's Decompositions with respect to Modeling and Control

The modeling and control problems suggest some sort of structural decomposition of an MS, in order to be able to manage the complexities of the design and implementation. There are several such "decompositions" proposed; often differing only in certain minor details. Davis and Jones [DJ89] note that most proposed decompositions are "de facto" hierarchical; [Mit91], [SV89], [Mal89], [BHH82] offer diverse examples of analysis either to propose solutions or just for the sake of exposition.

There is no general agreement on the levels that are actually needed and on the functionalities and their distribution. Moreover the diverse (hierarchical) decompositions employ different criteria; *time* seems to be the most common, i.e. the "lower" levels usually have a shorter "time horizon", which actually gives a "natural way" to evaluate the proposed solutions for the level.

A decomposition of an MS according to a combination of "function" and "time" has the following components.

- **Low level direct control** of equipment. Methods for direct real time control through sensors and actuators, e.g. *Computer numerical controls* (CNC).
- **Coordination** among cells and or workstations. *Synchronization* among cells/workstations and personnel.
- **Cooperation and "internal" information processing.** "Real time" scheduling and reaction to cell/workstations, personnel, communication failure.
- **Strategy and "external" information processing.** This component structures the relationships with the "external world", i.e. keeps track of the demands and produces the information associated with the finished goods.

An extra level could be added where functions like marketing and general "policy making", are grouped.

Mitchell ([Mit91] chap. 1 and 3) uses a different decomposition. The first cut decomposition views an MS from three points of view.

- **Physical models**: i.e. machines, transporters, material I/O and so on.
- **Functional models**: i.e. what functions are to be performed and are assigned to the physical "layout" of the MS.
- **Organizational models**: i.e. how people relate to the other models and to the organization of the MS activity.

This decomposition captures the variety of "models" useful for describing and design diverse aspects of a MS.

2.2.2 Paradigmatic Tasks in Modeling and Control of Manufacturing Systems

The modeling and control activities of an MS are supported by a number of "generic" tasks which can be applied to every step in the analysis and synthesis of each of the levels previously listed. Four major tasks are considered here.

- *simulation* of an (arbitrary) MS;
- preventive *verification* of some of its properties;
- *evaluation* of some of its characteristic parameters;
- *control* of its behavior.

With the possible exception of the fourth one (behavioral control), the tasks do require some form of *formal computational model* as a basis of their operations. The following brief definitions will describe what form of formal computational model is associated to each.

Simulation and Performance Evaluation

Simulation is often performed in the modeling stages of an MS (or of a complex system) setup. It is also used when the mathematical analytical models are computationally too expensive to be used directly. The aim of running a simulation of a system is to gain an understanding of some of its parameters by observing the evolution of a suitable model.

The main support offered by a simulation environment consists of a set of"representation" sublanguages, that allows the user to specify the *structure* of the system, *events* that can happen, *states*, *timing* information and so on.

Verification

The verification task deals with the issue of *correct design* of a system or of a device. The verification task is performed in order to "prove" that the system being designed has some desired "properties". E.g. *deadlock freedom* for a distributed environment and *delivery guarantee* in a communication setting. To achieve these goals, verification systems are usually equipped with some form of theorem prover or model checker.

Most systems have appeared in the field of Logic Circuit design, but some theoretically inherent limitations (e.g. $\mathcal{NP}$-completeness of some typical problems) seems to make their use not so handy and widespread.

Performance Evaluation

The evaluation task is listed apart from the simulation task even if they share many common traits. The structure of the modeling sublanguages can be the same in both cases.

Performance evaluation sets itself apart because of the different approach taken for the underlying computational model. Especially in the field of manufacturing, performance evaluation activity tries to directly use results from mathematical systems such as Stochastic Process Theory, Markov Chains and Queueing Networks.

Control of Behavior

This task deals with the "run–time" behavior of the system being modeled. In particular this task serves to specify the *algorithms*, the *procedures* and the *control loops* that the various components of the system have to obey.

2.2.3 Typical *Run-Time* Problems for MS's

The purpose of the "cooperation" component of the logical model of an MS is to organize the diverse activities on the floor in order to yield the "most satisfactory" throughput possible in terms of quantity, quality and speed of production[4].

The modeling and control activities in an MS environment must address many problems. The following two are the major ones.

- **Scheduling** (especially in *real time*).
- **Monitoring**, **diagnosing**, **recovery**.

A brief discussion of both problems will follow. Their introduction will serve as another dimension for classifying some of the current research efforts in manufacturing.

The Scheduling Problem

The scheduling problem, has been extensively studied since Taylor's times. Management Science, Operations Research, Artificial Intelligence and Algorithm Theory have looked at the problem from different viewpoints. Since

[4]Typical organizations go under the names of **Manufacturing Automation Protocol** (MAP), **Material Requirement Planning** (MRP), **PULL/PUSH** systems, **Kanban**, "**just in time**" (JIT). See [Mit91] for descriptions and discussion

the most general formulation of the job shop scheduling problem is $\mathcal{NP}$-complete [GJ79], many special cases and heuristics have been studied in order to approximate the optimal solution. Graves [Gra81] is a standard survey of the field. A more recent survey can be found in [RW88], while a detailed and formal classification of scheduling problems can be found in [LLRS90].

The goal of a scheduling discipline is to minimize or maximize some *objective function* with respect to the *allocation of resources of activities over time* [LLRS90]. Minimum *makespan* (i.e. time to complete all the jobs submitted) and minimum tardiness (i.e. minimizing the maximum difference between the *actual finishing time* and the *expected finishing time*) are typical criteria used to evaluate a schedule; many others can be devised accordingly to the setup of the problem.

One criticism of the "traditional" scheduling literature is that the most approaches take into account only a single performance measure and simplify the model in such ways to make it insensitive to "external factors", like *machine failures*. This last observation leads to further criticism that most scheduling approaches are *static* in nature. This is an heritage of the overall Tayloristic organization of the enterprise. A more realistic scenario would call for *real time rescheduling* methods.

A lot of research is currently being done in the field; the annual IEEE conference on Robotics and Automation [ICR] contains several sessions dedicated to the problem.

The Monitoring and Diagnosis Problems

Another problem that is to be faced when considering tools and methodologies for MS's management is the *monitoring* of the plant behavior. A related problem is that of *failure diagnosis*. Even the scheduling problem should take into account the monitoring and diagnosis activities ([RW88]).

A precise definition of the monitoring and diagnosis problem is difficult. The necessity of such activities is dictated by commonsense. No preventive analysis will foresee all the possible errors, incidents or misuses of equipment.

The monitoring and diagnosis tasks have been tackled mainly from the Control Theory ([Ger89], [DJ89]) and Artificial Intelligence viewpoints. An abstract analysis of how to cope with failures (by classifying them and by introducing *fault tolerance* in the system) is given in [Cri91].

Notably, most approaches assume the presence of some sort of abstract model of the system being monitored. Such model always reflects the field of origin of the approach. As such it is usually well "tuned" for the application of the methods specific to the field.

2.3 Summary on Manufacturing Systems and Methodology

Basic terminology regarding MS's was given as a common ground for two classification criteria for the modeling tools (explored in the next section). The first criterion uses the tasks of simulation, verification, performance evaluation and behavioral control as a basis. The second one (obviously incomplete) uses two typical problems in the field of manufacturing: scheduling and operation monitoring.

In the next section two modeling tools will be examined with respect to the above classification criteria. They are

- *Petri Nets* formalism;
- *Ramadge-Wonham Discrete Event System* formalism.

The two approaches do not obviously cover the whole spectrum of all the possible ones. Yet, they are among the most studied at the present time. As an example, the field of Temporal Logic ([MP92] and [CES86]) is another source of very interesting ideas and notions that can be used to tackle the modeling and control of MS.

3 Conceptual and Pragmatic Tools

In this section I will present two[5] *conceptual* (and pragmatic) tools used in conjunction with the *four tasks* presented in the previous section.

- Simulation
- Verification
- Evaluation
- Control

Each task can be tackled at all levels of the hierarchical decomposition of Manufacturing Systems proposed in the previous section.

The conceptual tools I will present, can be applied in more general and diverse frameworks than those addressed in this paper (Manufacturing Systems). In fact the more general viewpoint of *Distributed Real Time Systems* with *discrete* behavior, could be more appropriate.

This section is organized as follows. First we classify the approaches and tools with respect to the four tasks. Next, we summarize the main characteristics of two conceptual and pragmatic tool that have appeared in literature: **Petri Nets** and **Ramadge-Wonham discrete event systems**. Finally, we describe briefly a different classification of the tools with respect to how they have been used in conjunction with the problems mentioned in the previous section.

3.1 A Classification with respect to the Four Tasks

In the field of Manufacturing there are two approaches which seem interesting at present. We concentrate on these approaches and just mention other prominent ones. The first approach comes from the field of Petri Nets and the second from the field of Control Theory. The later approach is widely used in "Process Control" and Robotics, but was used only recently to model *discrete system*.

Following is a classification of tools and techniques. A similar approach (more problem driven) is taken in [AGM88]. As the previous section, the common denominator of each task is that it requires some *model* of the system, which determines its properties or drives its behavior.

[5] No pretence of exhaustivity made.

3.1.1 Simulation

A lot of literature has been published on the topic of simulation (see [Mit91] for a collection of references on theoretical works and popular tools: SIMSCRIPT^{TM}, SIMFACTORY^{TM}, SLAM^{TM}, GPSS^{TM} etc.). In Winston's introductory book on Operation Research [Win91], other references and technical material can be found.

Petri Nets are a very good simulation tool for discrete event systems. Their compact representation of what is intuitively a finite state automata[6] and their graphical behavior makes it easy to implement simulations of the systems modeled. Standard randomization techniques can be used in connection with Petri Nets to introduce a stochastic behavior in the simulation.

The field of Artificial Intelligence has contributed recently to the simulation topic. As an example see the work done by Fishwick [Fis88]. Another approach, from the viewpoint of Temporal Logic, can be found in [Tuz92].

3.1.2 Verification

Models of discrete systems like MS's serve many purposes. One of them is to "formally prove" that certain characteristics of the systems modeled are "correct" with respect to a given interpretation and/or that they have certain properties.

A typical property that is taken into consideration is **liveness**. Its definition usually depends on the conceptual tool used. Intuitively a system is "live" when it can make *progress* toward its *purpose*. E.g. a finite state automata in a "dead" state is not live. A related notion is that of **deadlock**. An opposite, yet very important, notion is that of **livelock**, where a system is not making any progress while consuming its *resources*.

The research in the field of Distributed and Real Time Systems has come up with many different formalisms to tackle the verification task in conjunction with many different problems from many areas outside the MS's field; these include Logic Circuit and VLSI design in Electronic research and protocol testing in the Communication field.

The techniques used in verification are mainly based on various forms of **Temporal Logic** (TL), most of the time *propositional*. The *models*[7] of the diverse formalisms are usually built from an automaton description

[6] This is not totally correct. Some Petri Nets models are strictly more powerful than standard finite state automata (see [Pet81]).

[7] In a mathematical logic sense.

(e.g. [CES86]). Both *proof calculi* and *model checking* techniques have been proposed in the literature in connection with TL specification.

Other verifiable specification techniques are under investigation. A recent collection of works is in [Vyt92]. Typical approaches use a variant of CSP or CCS as the underlying model.

Petri Nets can be used for the verification task. Part of the theory deals with algebraic methods to find the so called *invariants* in a net. These invariants can be used to check some "structural" properties of the model.

The correlation between Petri Nets and Logic formalisms (especially TL ones) has been known for a long time (see again [Rei85]). More recently *Dynamic Logic* and TL verification approaches have been further investigated in conjunction with Petri Nets: see [Bra91] and [Tuo89].

3.1.3 Performance Evaluation

The "performance evaluation" task shares many characteristics with the task of simulation of MS's. Its principal aim is to **measure** specific quantities of the system modeled, either by analytical methods or by simulation.

Queuing Theory is the main analytical approach taken in the MS's field. Models of MS's are constructed with *networks of queuing stations* and their performance is measured with respect to different operating conditions (distribution of "arrivals" and "departures"), formalized by well-established mathematical results as the *product form BCMP Theorem*.

Petri Nets variants called *Stochastic Petri Nets* have been used for performance evaluation. Ajmone Marsan [Ajm89] gives an account of the approach, which is currently used and is under further investigation ([FM86]) An open problem in this field is to find a counterpart of the BCMP theorem that would apply to Stochastic Petri Nets.

3.1.4 Control

The "control" of a system (be it a "soft" model or the "real" thing) consists in *constraining* its behavior along some more or less desirable *trajectories*. The field of *Control Theory* studies these problems ([Won85]). There are obviously other approaches that have tackled the problem. As an example, some Artificial Intelligence approaches were successful in the Control of industrial processes.

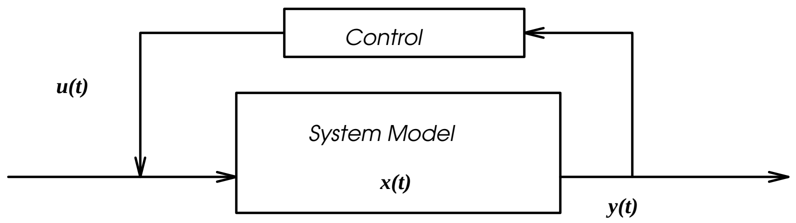


Figure 2: Black box model of a feedback loop

Control Theoretic Approaches

The generic formulation of a control problem is to find a function representing the constraints of the trajectory of the modeled system, given a functional representation of the *state* of the model and of the possible *inputs*. More formally, the functional form $\Pi(x(t), u(t))$ represents some characteristic of the process, while we denote with $x(t)$ a time dependent quantity representing the state of the process (usually also some of its derivatives are constrained by other functional forms). If $u(t)$ is the *control input* of the system, then the standard control problem[8] is to modify the input u, in order to modify the input/output behavior of the system. u is obtained as a result of the application of a function g (the so called *control law*) to the state $x(t)$ of the system. Such "control law" depends on the application and it can assume different forms.

As Ho points out ([Ho89a]) in the introduction of a special issue on "Dynamics of Discrete Event Systems" of [Ho89b], Control Theory has mainly dealt with *Continuos Variables Systems*. On the other hand 'discrete event systems" (especially MS's) pose different and challenging problems from the Control Theory viewpoint. Davis and Jones [DJ89] and Ramadge and Wonham [RW89] are representative of some approaches to the control problem for discrete event systems.

Open and Closed Architectures

One of the design problems to be faced when designing a complex system is the specification of the **control architecture**. A major choice to be made is whether the control architecture should be *open* or *closed*.

A standard picture depicting a "control feedback loop" is in fig. 2. The `control` component decides what kind of "control input" is to be supplied

[8] I.e. how "the loop is closed".

to the system, in order to modify some of the measures under scrutiny.

A *closed* control architecture implements the control component (in fig. 2) by a "fixed" law, possibly built in hardware. Instead, we have an *open* control architecture when the control component is "flexible", i.e. it can be changed or multiplexed according to empirical results or transient needs. Such "flexibility" is achieved in a number of ways, the presence of human intervention being the main one (cfr. *supervisory systems*). Coupling the control component with a software component yields a very flexible architecture and creates opportunities for an enhanced human interaction when desired. An ongoing "open architecture" project at NYU is described in [GHFP89].

Artificial Intelligence And Other Approaches

The complexities of the control task for MS's have led many researchers to investigate "weak methods"[9], i.e. methods based on heuristics applied within the framework of "generic state space search" architecture (a classical reference is [Win92]). The AI literature contains many examples of these approaches. It is interesting to note that in many cases the architectures proposed in the AI literature can be classified as "supervisory systems", where human intervention is considered as part of the system.

Since the modeling of discrete event systems is the major concern for the Petri Net community, it is not surprising that many research and applicative efforts did come from the field.

According to [SV89] there are many attempts to integrate Petri Nets within control architectures for MS's: from the *programmable logic controls* (PLC) to "higher" levels of control on a shop floor. Many other research reports can be found in [ICR] and elsewhere.

[9]In the standard terminology, due to A. Newell and H. Simon.

3.2 Petri Nets

Petri Nets have become a widespread tool for the modeling and analysis of discrete event systems, and thus of MS's. In this section, I will illustrate their typical use and the problems that arise in their analysis. In doing so, I will follow the material contained in [Rei85], [GLT80] and [Pet81].

3.2.1 Petri Net Classes and Basic Analysis Techniques

The two notions of **state** and **transition** (or *state change*, *change*, *movement*) beyond their "common sense" interpretation, play a central role in the description of any conceivable system. As Petri himself states [Pet79], their unified treatment, despite the many different possible interpretations, was also the main drive for the development of *Net Theory*.

Such basic notions are at the heart of the theory of *State Machines* (Automata, both Finite and Infinite), which has been extensively studied over the years (e.g. [HU79]). Finite Automata (FA's) have been used in various fields to model the most disparate systems. As an example, the "Supervisory Control Systems", by Ramadge and Wonham ([RW89]) described in the next section, use state automata (and the notion of "recognized languages") to develop their approach.

The many varieties of Petri Nets introduced in the literature[10] develop the two notions of state and transitions at different levels of abstraction. The main strands of PN's systems add different components to the basic model, with the objective to enhance its expressiveness and/or usability. A "genealogical tree" of different net Models is shown in fig. 3.

From a certain point of view, the basic idea of Net systems is that, in order to describe *distributed* systems the "monolithic" notions of state and transition should be broken down to a finer granularity. The aim being to express properties like *causal dependence*, *choice* and *concurrency*.

An Informal Definition of Petri Nets

A Petri Net is similar to a *bipartite graph*. Two sets of vertices are defined: the set of **places** (traditionally called S) and the set of **transitions** (called T). The usual graphical notation uses circles, $\bigcirc$, to represent places and boxes, $\square$, to represent transitions. For each element in both sets we can define the so called *pre* and *post* sets, with respect to the edges in the graph

[10] An encyclopedic literature reference (as of 1990) can be found in [PR91])

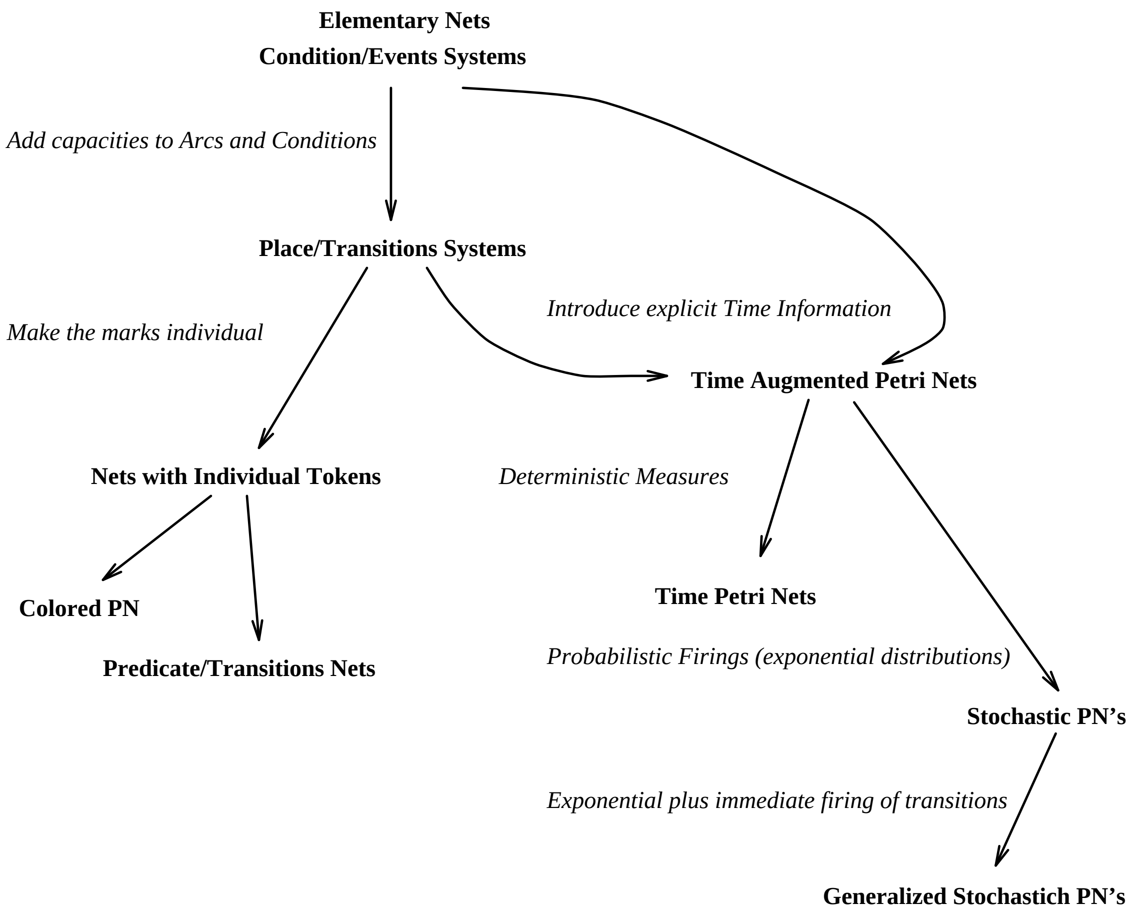


Figure 3: A "Genealogical Tree" of Petri Nets Models.

(which compose the so called **flow relation** F). It is also standard practice (when no confusion arises) to write N for $S \cup T$.

We can associate a **capacity** to each place ($K(s)$) and a **weight** to each arc ($W(t,s)$ and $W(s,t)$). Typical cases of interest are when the capacity is restriced to 1 or is totally unrestricted (equal to ∞). Also we may want to restrict the waight of an arc to 1. A very simple and standard example is the *4-Seasons Net* (in fig. 4. The states are the four seasons (represented by the four places), while the transitions actually represent the four transition changes.

Given the above basic definitions we now tackle those of *state* and *state-change*.

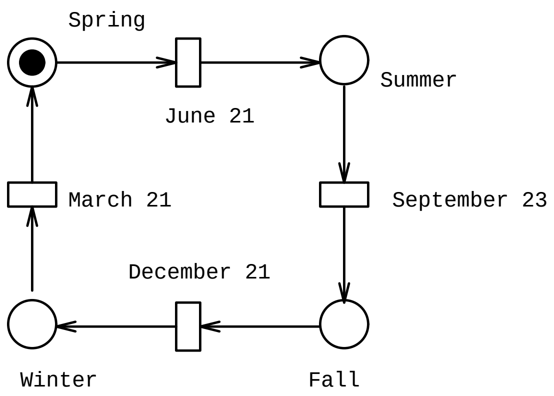


Figure 4: The 4 Seasons Net.

Distributed State The notion we want to model is that of a *distributed state*. Each place can contain a certain number of tokens or **marks** (the capacity is the limit that cannot be exceeded)[11].

In each "moment" it is possible to denote the *distributed* state of a Net by simply "counting" the number[12] of marks in each place.

State Transitions The **occurences** of transitions change the state in the net by changing the **marking** of the places.

An transition t can occur in a state M if and only if the places in its preset contains enough tokens with respect to the weight of arcs from them to t. Moreover, the places in the postset must have "enough room" to be able to receive the tokens that will be generated *and passed on the arcs from t to them. A transition that* may *occur is said to be* **enabled** *in the current marking.*

The figures 5, 6 and 7 pictorially represent these situations.

The **firing** (or *occurrence*) of transitions leads from one marking to another and so on. Given a marking M many transitions may be enabled and may occurr. If two transition fire in marking M they can be combined in one **step**. Yet, the occurrence of a set of transitions at M is possible if and only if they have their pre- and post- set disjoint or they contain enough

[11]This notion of *marking* makes attractive the nets with $K(s) = 1$, since then each place can be seen as a condition that either *holds* or not. This class of Nets is named *Condition/Events Systems*.

[12]In Petri Nets of "higher level", we will also consider the *type* of the tokens.

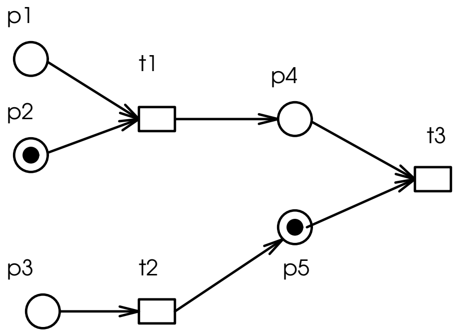


Figure 5: A *marking* in a simple Petri Net (actually a C/E system): p5 and p2 each contain one token.

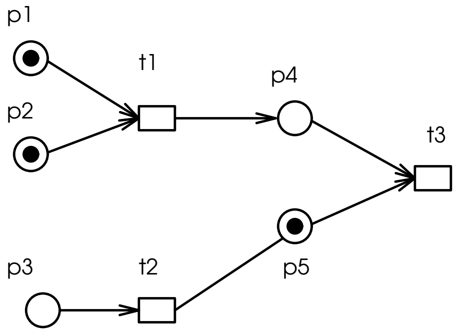


Figure 6: Representation of enabledness. The same Petri Net with marking $\{\mathsf{p5}, \mathsf{p1}, \mathsf{p2}\}$. Transition t1 is *enabled*, while t2 and t3 are not.

tokens to be consumed. When two transitions *may* fire at a certain marking they are said to be **concurrent**.

Since the notions of case (state) and step (transition) imply the existence of a *reachability* relation among markings, by the firing of transitions, it immediate to define the notion of *case graph* (or *reachability graph*.

This notion carries on to models of greater expressiveness as the Predicate Transitions system.

Linear Algebraic Representation for P/T Nets

The P/T nets have a nice linear algebraic representation that eases many inquiries on the structure of the systems modeled.

If N is a net then we can define the following items

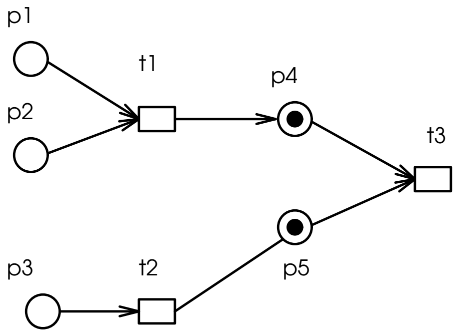


Figure 7: Firing of a transition. Transition t1 has fired yielding the new marking $\{\mathsf{p4}, \mathsf{p5}\}$. Transition t3 is enabled in this new case.

- given a transition $t \in T$, we define the vector $\bar{t}$ that associates the following elements to each transition t:
  - $W(t, s)$ for s that are only in the postset of t,
  - $-W(s, t)$ for those s that are only in the preset of t,
  - $W(t, s) - W(s, t)$ for those s that are in both pre and post sets of t (i.e. those forming structural "slef loops"),
  - 0 otherwise.
- given $\bar{t}$ we can define the matrix $\bar{N}(s, t) = \bar{t}(s)$

Given some extra conditions on the structure of the net N, it can be shown that the behavior of a net can be completely expressed with its linear algebraic representation. In particular, the *firing rule* for a M-enabled transition t becomes

$$M[t\rangle M' \equiv M + \bar{t} = M'$$

where M' is a marking of the net reachable from M by the firing of t[13].

Properties of Net Systems and Their Analysis

Given the above definitions (of C/E and P/T systems), we would like to be able to define and analyze some properties of the models we build. Some of the properties we want to be able to analyze are

- **liveness**,

[13]The notation I used is standard in Petri Net literature.

- **boundedness**,

Liveness Properties. In the nets literature a much studied property is that of **liveness**. Given a P/T net N and a transition $t \in N$

1. t is called *live* iff $\forall M \in [M_0\,\rangle$ there $\exists M' \in [M_0\,\rangle$ such that t is *enabled* in M'.

2. N is called *live* iff $\forall t \in T.\, t$ is live.

where the notation $[M\rangle$ denotes the set of reachable marking starting at M. Such definition is adjusted for all kinds of nets and is extendable to markings.

Given the definition of live transition the we can now denote the *dead* ones; next we use both notions to define *deadlocks* and *traps* as subsets of S.

Deadlocks are sets of places which will host no marks from a certain point on. Traps, on the other hand, are sets of places which never loose their marking.

Boundedness Properties. The *boundedness* property of a P/T net tells us that, given an initial marking M_0, each place can contain at most k marks. A net where each place has a finite associated capacity, is obviously bounded. Yet it might be the case that boundedness is a "structural" property of a model. In this case, it is important to compute how many marks can be held in a single place at any time.

As an example consider the model of a manufacturing transportation subsystem with an in-transit buffer. Either we want to impose a limit as a *system specification*, or we want to *evaluate* what is the maximum buffer size we can reach (with a maximum of ∞ being highly undesirable).

Analysis Types for Nets. The analysis techniques developed over the years can be roughly classified into three categories. We distinguish the following

- *enumeration*, state/space techniques,

- *reduction* techniques,

- *structural* techniques.

The enumeration technique uses the so called **reachability/coverability** graphs for P/T nets. The reduction techniques have the aim to "size down" a net, in order to facilitate its analysis by means of an enumerability approach. The structural techniques use the so called **invariants** methods, which give good insight on some behavioral aspects of a net model, prior to some interpretation effort.

Simulation techniques can obviously be used to gain statistical data about the behavior of a modeled system.

In the following the *coverability graph* tool and the *net invariants* techniques will be introduced. Various reduction schemes (preserving properties of original Nets) can be found in [FM86] who cite a work by Silva.

Enumeration Techniques: Coverability Graphs The formal study of reachability can be carried on by means of the so called **coverability graphs**.

A P/T net will usually generate an infinite set of reachable markings from a given initial one (M_0). The related *reachability problem* can simply be stated as "given an arbitrary marking M of a (P/T) net N, does $M \in [M_0\rangle$?".

The set or reachable marking can easily be arranged in an infinite tree (or graph). It turns out that from such infinite tree a *finite* graph can be derived, without loosing information. The idea is to collapse "regularly structured paths" into nodes with a particular inscription.

Given a path in the marking reachability tree

$$\ldots M_i \xrightarrow{t_{i+1}} M_{i+1} \xrightarrow{t_{i+2}} M_{i+2} \ldots \xrightarrow{t_j} M_j \ldots$$

such that for all $s \in S$, $i < j$ and $M_i(s) \leq M_j(s)$, then we can collapse such a sequence into a single node. The label will be a tuple of length $|S|$ built in the following way: for each $s \in S$ such that $M_i(s) = M_j(s)$ then $label(s) = M_i(s)$; if $M_i(s) < M_j(s)$ then $label(s) = \omega$. The value ω indicates that the "value" of S increases unboundedly[14] within the above path. For a P/T N, the resulting coverabilty tree $CT(N)$ (graph $CG(N)$) is *finite*.

The justification for doing so, is that such a sequence $t_{i+1} \ldots t_j$ is repeatable infinitely often in the reachability tree. Note that all the transitions enabled in M_i are also enabled in M_j.

[14] In this discussion, places with unbounded capacity are assumed; ω is the way to indicate it.

Liveness and Boundedness as a Reachability By–product. Liveness and Boundedness properties are definable in terms of *reachability* of markings (see Peterson [Pet81]). The problem of reachability is therefore a central one in the development of Net theory.

Unfortunately, it turned out that the reachability problem is decidable (see [RC90]), but with a very high complexity. In the survey by Jantzen [Jan86] a result by Cardoza, Lipton and Meyer is reported, placing it at best in EXPSPACE (w.r.t. the size $||N||$ of a net N).

Other related decidability and complexity results assume the construction of a coverability graph (given a coverability graph standard search and "x–connected" components algorithms can be used to test for properties like liveness). Their complexity is still very high. The boundedness problem for P/T nets is decidable in $2^{c\,||N||\log ||N||}$ space, but not in $2^{\sqrt{||N||}}$ space.

Most of these results stem from the fact that it is possible to build nets whose coverability graph are extremely unwieldy.

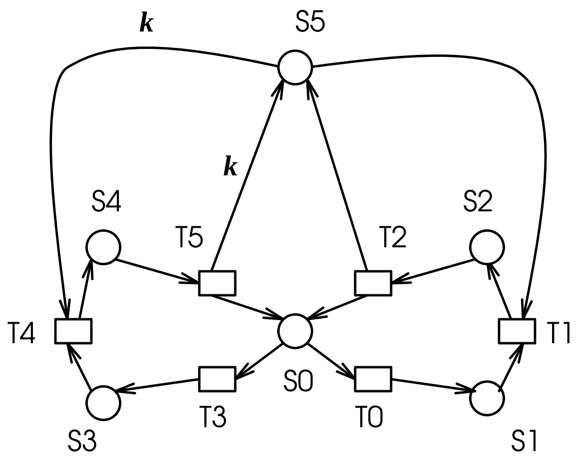
Recent Results in Complexity Reduction. While it is generally accepted that coverability graphs are not very useful for all but very simple nets, many studies have been done on the problem of representing reachability graphs in a compact way, or to find average low complexity reachability analysis methods. Valmari [Val90], Janicki and Koutny [JK91] describe methods and algorithms to reduce the size of an internal representation of the state space of P/T (and also of "higher level" Nets)[15].

***S*-Invariants and *T*-Invariants.** A means of analysis for P/T nets and higher level ones is the Invariant calculus. The first form of invariants is the S-Invariant, which informally, is a set of places where the mark count is constant during transition firing. Given the flow matrix $\bar{N}$ of a P/T net the **place vector** i that is the solution of the equation

$$\bar{N}^t \cdot i = \mathbf{0};\ i \geq \mathbf{0}$$

is called a ***S*-invariant** of the net N. As an example consider a simple reader/writer model shown in fig. 8 (from [Rei85]). In this example we have n processes trying to read from or write into a buffer. Each Si represent a possible state for the processes. Each process is represented in the initial

[15] It is worth noting that the same state space reduction problem appears in Temporal Logic verification systems using the "model checking" technique. See for example [Bry86] and [BCM+90] for an account of their approach.



Description of the places:
S0: inactive processes
S1: processes which are ready to consume
S2: reading processes
S3: processes ready to write
S4: writing processes
S5: synchronization

Figure 8: A simple reader/writer system.

marking as a token in place S0. In order to represent the number of processes which are allowed to read at the beginning we put k tokens in place S5. The weights on the arcs (T5, S0) and (S0, T4) ensure that only k processes at a time will be reading[16].

If we compute the S-invariants for the net in fig. 8 we get

$$\begin{aligned} i_1 &= (1,1,1,1,1,0) \\ i_2 &= (0,0,1,0,k,1) \end{aligned}$$

By interpreting the two invariants found we have that for each marking following M_0 the number of processes (reading or writing: i.e. in states S0 ... S4) remains equal to n; moreover the number of writers at any time is

[16]Note how the net actually contains two "conflicts", since it does not specify a policy for choosing between readers and writers.

equal to 1. This can be inferred from the equation

$$M(s_2) + k\,M(s_4) + M(s_5) = M_0(s_2) + k\,M_0(s_4) + M_0(s_5) = k$$

by using invariant i_2.

The second form of invariant is the T-invariant. A T-invariant for a net N tells us if a given marking is *reproducible* after each transition has fired exactly $v(t)$ times.

Again the T-invariant is defined to be the solution of the equation

$$\bar{N} \cdot v = \mathbf{0};\ v \geq \mathbf{0}$$

where v is a **transition vector**.

Such information is definitively useful in testing liveness of particular subsets of a Net[17].

As a final note on Net invariants, we can observe that the algebraic equations systems that must be solved to find them give rise to a typical Integer Programming Problem; as such the complexity of the solution algorithms is high[18].

Place Transitions Nets Summary So far we have seen the basic C/E and P/T models, along with the basic notions encountered in the Petri Nets literature. C/E systems have been used mainly for foundational studies on *concurrency*, while P/T Nets have been extensively studied because they offer a convenient way to represent and compute (or better: prove decidability results about) several "structural" properties of the systems modeled.

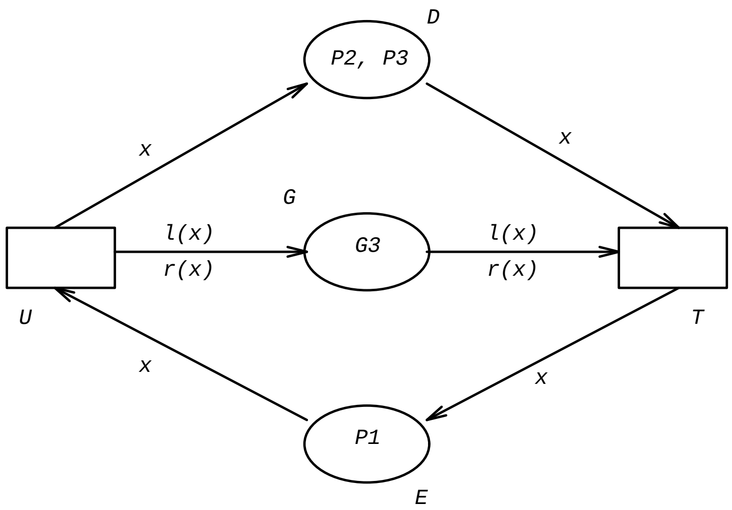
In the next section, we will see some Net models of "higher" abstraction level.

3.2.2 Higher Level Models and Modularity Problems

Despite their theoretical appeal and modeling power, C/E and P/T nets have been considered too "fine grained" to be effectively used as a "real" design tool. In order to cope with these shortcomings "High Level" Petri Nets (HLPN) models have been proposed in the literature [JR91] along with *compositionality* and *modularity* concepts.

[17] There are many other results concerning invariants, but the space is limited here. See the usual [Rei85] and [Pet81] for a more extensive account.

[18] There exist an entire subliterature on how to efficiently compute net invariants, mainly by "translating" the problem in a standard LP problem. See [IPN] and [APN].



Description of the Net elements

D: thinking philosophers,

E: eating philosophers,

G: available forks,

$\mathtt{P} = \{\mathtt{P1}, \mathtt{P2}, \mathtt{P3}\}$,

$\mathtt{G} = \{\mathtt{G1}, \mathtt{G2}, \mathtt{G3}\}$,

$\mathtt{l} : \mathtt{P} \to \mathtt{G}$ (with $\mathtt{l}(\mathtt{Pi}) = \mathtt{Gi}$),

$\mathtt{r} : \mathtt{P} \to \mathtt{G}$ (with $\mathtt{r}(\mathtt{Pi}) = \mathtt{G}\,(\mathtt{i} + 1 \bmod 3)$),

The P/T net is much less compact.

Figure 9: The Dining Philosophers as a Pr/T Net.

The two major strands of HLPN's are named **Predicate/Transitions** (Pr/T) [Gen86] and **Colored** Petri Nets (CPN) [Jen86].

Both Pr/T and CPN extend the P/T model by allowing for the marking to be composed of *individual tokens*. This change in the definition allows for a very compact representation of systems. As an example see the standard dining philosophers model as represented by a Pr/T net in fig. 9.

The introduction of individual tokens is done by associating a standard *relational structure* $\mathcal{R}$ to a net. This relational structure comprises a *domain* D of *individuals*, a set of relation symbols R and a set of functions symbols F. Next, we define a set of *first order terms* (with *variables*) in the same way we would in a formal logic framework. Finally, we redefine the notions of marking and firing, in order to take into account the given extensions.

Genrich ([Gen86]) does this by adding an annotation to the components of a net N.

- To each arc we associate a formula that is a *symbolic sum* on the domain D of individuals.
- To each place we associate a set of variables.
- To each transition we associate a well formed formula over the language induced by $\mathcal{R}$.

The initial marking M_0 (as well as all the "subsequent" markings are a "consistent" assignment of elements from D to the places in S.

The firing rule for a transition t in a Pr/T net acts in the following three steps:

1. first it checks that all the input places are marked and that the formula associated to t is satisfied by the marks in the input places;
2. secondly, it checks that the expressions on the arcs can be satisfied with the required number of tokens from the input places;
3. finally, it simply removes enough tokens from the input places and puts enough tokens into the output places.

Pr/T nets support a form of *symbolic invariant calculus* that can be used to check systems properties as in the case of P/T nets.

The complications arising from the invariant calculus for Pr/T nets are the motivations for the development of the restricted CPN model. By restricting the form of the net annotation, CPN possess an easier an "more intuitive" invariant calculus.

The CPN model has received some attention recently, mainly because of its appeal and a dedicated number of researchers. Some of the studies carried on on CPN deal with the problems of modularity and decomposition in Petri Nets [HJS90] and [Jen90].

The Modularity problem in Petri Nets

While the definition of Pr/T and CPN addressed the problem of the fine *granularity* of C/E and P/T models and it might be adequate to the formal treatment of such objects, it still does not address some of the more pragmatic issues connected to a "daily" design routine.

There are three main approaches in the literature, dealing with the modularity problem; two of them are forms of *net refinement*:

- transition refinements;
- place refinements.

The two refinements procedures (in the many variants proposed in literature) have the aim of augmenting the size of a net while maintaining the properties of the original net, which is seen as a "high level" specification of a system. The work by Huber, Shapiro and Jensen explores these notions for CPN's. Another survey can be found in [BGV90].

The third approach is that of *compositionality*. The main ideas are to build separate nets and to specify rules of combinations of transitions and/or places. Superimposed Automata Nets (SA Nets) and its variants ([DDPS82], [BDMR91]) are an example of "composition through transitions", while studies by [SM90] look into "composition through places". Once again properties like maintainance of liveness in the "composed" net, are the aim of the studies.

The overall declared aim of these researches is to yield a set of compositional and modularity notions equipped with *simulation and equivalence properties* and usable proof techniques. [19].

3.2.3 Time-augmented and Stochastic Models

The notion of *time* is present in the Net Theory only as a *derived* property of the systems modeled. The only notion of the "passing of time" comes from the *observation* of the occurrence of events.

Since many pragmatic applications do require an explicit notion of time, many extensions have been proposed to the basic Nets models (mainly C/E and P/T) in order to include some notion of time. Two main categories of extensions have been proposed: *deterministic* and *stochastic*.

Deterministic Time Extensions to Petri Nets

In order to introduce time in the PN's model, the "natural" extension to Petri Nets is to add some time information to places or transitions.

[19] A notion used in Petri Nets research is that of *bisimulation*, which can informally defined as the ability of a certain system to simulate another other and viceversa.

Two approaches used to modify transitions add either a *duration* to them or a *time interval* during which they can fire (see [Ram74] and [Mer74]. Berthomieu and Diaz [BD91] and [Fre91] advocate the second approach.

With the "time interval" extension a C/E system changes the firing rule in the following way. Let N be a net with two numbers α_i and β_i associated to transition t_i and $0 \leq \alpha_i \leq \beta_i \leq \infty$.

- α_i is the *minimal time* it must elapse after t_i becomes enabled (in the usual sense) before it can fire.

- β_i is the *maximum time* after t_i became enabled during which t_i may not fire.

I.e. if transition t_i becomes enabled at time τ then it cannot fire before time $\tau + \alpha_i$ and it *must* fire within time $\tau + \beta_i$ (unless it looses enabledness due to some other occurrences of conflicting transitions).

Obviously, the model requires the presence of some form of *clock* in the system.

The first two authors propose an analysis method for *Time Petri Nets* based on a state space enumeration that yields a set of associations between markings and timing constraints. Once again, the complexities of PN's analysis surface in the formulation of the solution methods proposed[20].

Stochastic Time Extensions

The previous extension to PN's models is *deterministic*. A number of *stochastic* extensions to PN's have been proposed. Ajmone Marsan [Ajm89] gives a good introduction to the subject, relating it to the task of performance evaluation of the systems modeled.

The specification of the so called *Stochastic Petri Nets* (SPN) and *Generalized Stochastic Petri Nets* (GSPN) brings well known results from stochastic process analysis and Queuing theory into the PN's field.

A SPN is a standard (C/E) net N, with a *firing rate* λ_i associated to each transition t_i in T. The firing rule is modified in order to take into account the firing rate. The λ_i's are the parameters of *negative exponential density function*, therefore modeling the *firing delay* as a random variable. The choice of the negative exponential density functions was dictated by

[20]Freedman proposes a different use of TPN and by slightly restricting the structure of the nets, he is able to show some nice algorithms for the computation of some characteristics of repetitive tasks.

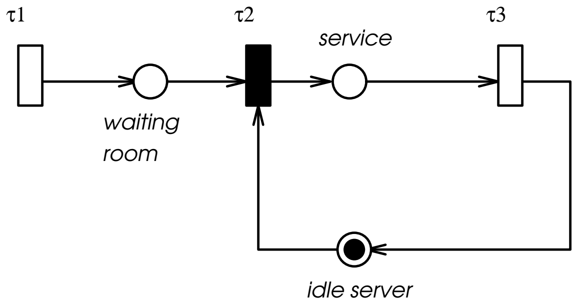


Figure 10: The representation of a M/M/1 queue with a GSPN.
The black transition is an immediate one, transitions t_1 and t_3 have rates λ and μ respectively.

the desire to recast many results from *memoryless Markov Chain* stochastic processes into Net Theory.

The main conceptual tool to analyze the behavior of a SPN is the underlying Continuous Time Markov Chain. After the "generator" for the Markov Chain has been built, the steady state behavior of the SPN can be computed. This process has the same (space) complexity as for the construction of the state space of the net.

Indices of Performance. [Ajm89] lists some aggregate indices that can be measured from the steady state distribution.

- Probability of an "event" A, described as a condition on a place marking.
- Average number of tokens in a place.
- Frequency of firing of a transition in a unit of time.
- Average delay of a token (e.g. of a part in a manufacturing shop) while traversing a subnet (a section of the shop floor).

Generalized SPN (GSPN) introduce a different kind of transition with no associated delay (*immediate transitions*). Their definition is more complicated but they can be analyzed more efficiently, thanks to a set of heuristics rules. See for example [FM86] and [ACF89].

As a final example, in fig. 10 is the GSPN representation of a M/M/1 queue.

Examples of the evaluation of a two-processor system with private memory modules, local buses and access to shared memory modules are also given in the works mentioned.

3.2.4 An Overview of Petri Nets in Manufacturing

Despite some intrinsic difficulties in their analysis, Petri Nets have been widely used in manufacturing, as testified by the great number of works appearing, as an example, in the Robotics and Automation conference [ICR]. Here is a list of many works classified by *problem* and *task* (according to the classifications described earlier).

Scheduling (in conjunction with AI techniques)

Petri Nets have been used to tackle the problem of production scheduling at the "cooperation" level introduced in section 2.2.1.

Silva and Valette (in [SV89]) report of applications of Petri Nets in this framework. Moreover they state that a Petri Net approach is well suited for *on-line*, *heuristic* scheduling for *repetitive* operations in a manufacturing system. They also claim that *constraint propagation* schema can be used in conjunction with Petri Nets in order to produce task schedules for a MS.

Heuristics approaches have seen recent developments. See for example [MMS$^+$88] and [MMS87]. Two recent works that integrate well known search techniques from the AI field with PN's in order to tackle two scheduling problems are [MS91] and [LD92].

Coordination, Control and Error management

Low level control and coordination among different machines on the shop floor have seen a number of applications of Petri Nets. The problem tackled here is Monitoring and Error Management; with respect to the four tasks the focus is on control.

A first group of applications deals with error recovery along with "sensor interpretation" (usually recalling AI techniques). Some of the work done by the group at Rensselaer Polytechnic Institute[21] is in this direction. Zhou and Di Cesare [ZD89] define some methods to augment a "PN controller" in order to provide error recovery.

A second group focuses on producing effective control strategies for robotics and manufacturing systems. Teng and Black describe a cell control

[21] Actually a very active group in all fields of applied PN's.

design in [TB91]. Zhou and Leu discuss a model for an assembly station in [ZL91]. Another interesting approach, dealing with coordination of multiple robots, can be found in [KS91] and [KS92]. In [TF92] the authors present a PN based methodology for modeling material transportation and handling in a FMS.

Modeling and performance evaluation

This subfield is the one that seems to be receiving the most attention. [FM86] contains the analysis of different FMS models.

Many of the works in this subfield mix theoretical results with example applications. See for example [JD91], [WD91] and [JSS92].

3.3 A Control Theoretical Approach: Ramadge-Wonham Systems

The modeling and control of processes is also studied in the discipline of *Control Theory*. Most of the work done in this field deals with *continuous variables systems*; the introduction of *discrete variables* in a model poses many challenging problems, especially in terms of computational complexity.

A very interesting foundational approach is the one described by Ramadge and Wonham in [RW89].

The basic idea behind the work done by Ramadge and Wonham in order to built a "control theory" for discrete events systems (DES) is to resort to the well known Languages and Automata Theory ([HU79]).

3.3.1 Basic Definitions

One of the central notions in control theory is that of *trajectory* of a system. Ramadge and Wonham translate this notion in the framework of DES by representing the trajectory of a system by a *string* over an *alphabet* Σ of *events*. This is a standard definition taken from Language Theory.

The *behavior* of a DES is represented by a *language* over Σ. An automaton G (not necessarily finite, even if it is obviously nice to have it so) is called a *generator* for the language L.

More complex systems can be theoretically modeled by *shuffling* the states diagrams of two or more generators. This approach leads to potentially big state spaces for the resulting automata, since the costruction of the "shuffle" is based on the cartesian product of the state spaces of the original automata (see fig. 11 for an example).

Once again the modularity problem surfaces when "tractable" applications are to be implemented and/or manipulated.

3.3.2 Controllable and Uncontrollable Events and the Notion of Supervisor

The generator G is used to model the *open loop* behavior of the system modeled. In order to introduce the notion of *control* into the theory, Ramadge and Wonham assume that some of the events in the alphabet Σ can be prevented from happening at certain times. The events in such subset are called *controllable* (Σ_c), as opposed to the rest, which are dubbed *uncontrollable* (Σ_u).

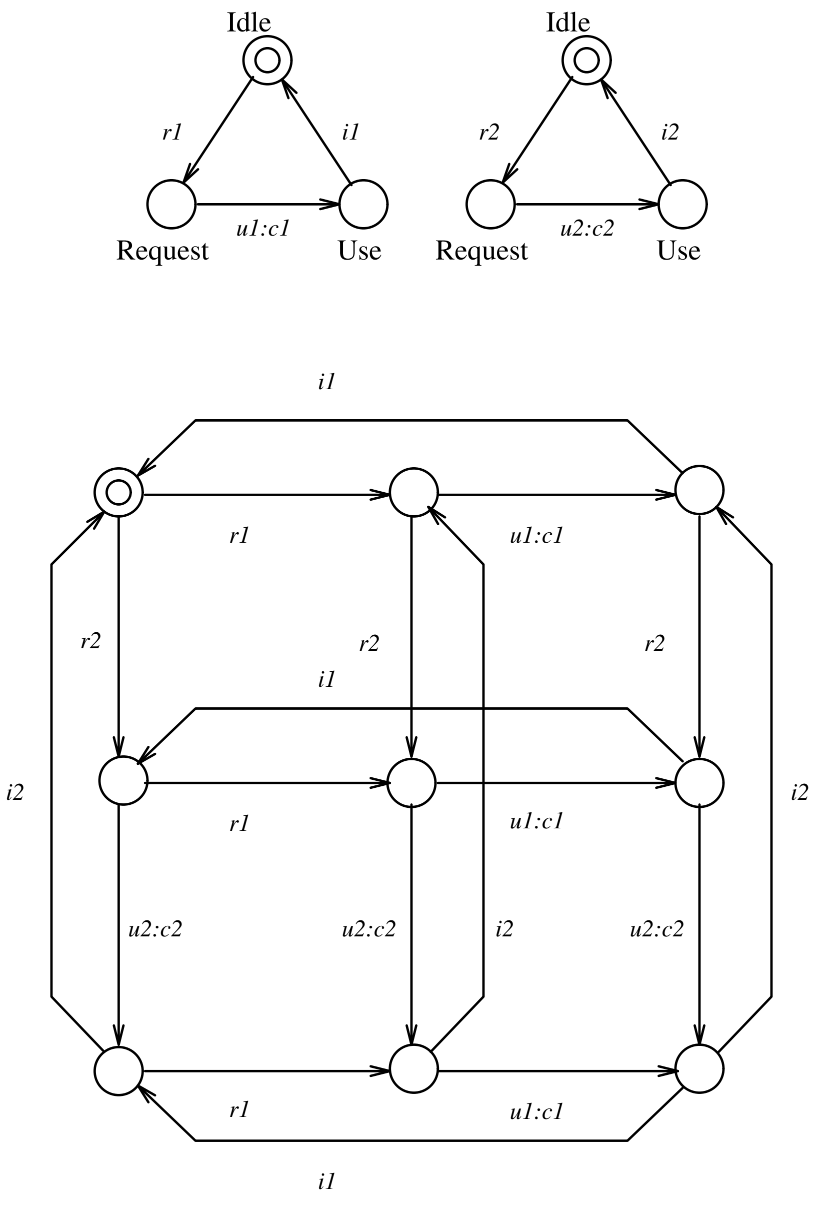


Figure 11: A FA description of two machines. Each machine can be in three states. All the possible states of the combined system are given by the "shuffle" of the two original FA's.

A *control input* for the generator G is a subset of the event alphabet containing all the uncontrollable events. If an event belongs to a control input then it is said to be *enabled*; otherwise it is said to be *disabled* by the control input.

Controlling a DES consists in choosing a sequence of elements in the set of possible control inputs in response to observed strings of previously generated events. Such process of subsequent choices is called a *supervisor* for the DES.

3.3.3 Analogies with Standard Control Theoretical Practice

If we call the set of control inputs Γ then a supervisor for the DES is simply a map $f : L \rightarrow \Gamma$. Given a sequence of events w generated by G, the supervisor f specifies what control input is to be applied in order to constrain its behavior. The control loop is therefore closed by the supervisor f.

As Ramadge and Wonham point out, if the supervisor f is realized as an automaton T with an associated "output map" to Γ (i.e. it is similar to a Moore automaton), then the analogy with standard control theoretic terminology becomes straightforward. The generator G becomes the *plant* to be controlled, the supervisor automaton T acts as an *observer* and its output map is the *feedback*.

Controllability

The problem of modifying the open loop behavior by supervisory control is cast in terms of "supervisor synthesis". Ramadge and Wonham first define the notion of *controllable language* with respect to the "open control" language representing the behavior of the system modeled. Then they state the conditions under which a controllable language is realizable by restricting the open control language by means of a supervisor f, i.e. they state the conditions under which the supervisor exists.

Another result concerns the closure of the controllable sublanguages of a (possibly uncontrollable) language K under union. This result is important because it implies the existence of a sublanguage of K which is a "maximal controllable approximation" for it. A straightforward fixpoint based construction algorithm for the supremal controllable language has been shown for regular languages.

Since finding "optimal" controls for a systems is difficult (i.e. to find an optimal language with an associated supervisor), the possibility to compute an approximation to it is very appealing; the previous result gives a method that can be followed in many situations.

3.3.4 Modularity, Observability and Complexity Issues

As in the case of Petri Nets the issue of modularity in the synthesis of systems models becomes very important in order to control the computational and expressive complexity of the automata descriptions.

From Ramadge and Wonham's viewpoint, the issue of modularity is addressed by defining an algebra of supervisors and DES's. Modular compositions of DES's and supervisors is therefore accomplished by defining more comprehensive system models by mean of the *intersection* of behaviors and supervisors. Such intersection operation must be constrained by some extra conditions that ensured that the properties of controllability are maintained by the composed model. Once this is done, a supremal controllable language can be built for the intersected languages. In particular, it can be shown that the supremal language of the constrained intersection of two languages is the constrained intersection of the respective supremal languages. This result reduces the complexity of finding the supremal language for a "composed" model.

The complexities of various problems within the theory are summarized in the following[22].

- *The controllability of a given language $K \subseteq L(G)$ (where G is the generator for the system modeled) is* polynomially decidable.
  The complexity is stated as $O(m\,n\,|\Sigma_u|\,|\Sigma|)$ time ([RW89] and [RW87]), where m is the number of states in the generator for K and n is the number of states in the generator G.
- *The problem of synthesizing a supervisor for a given language K (which should be controllable), is in $\mathcal{P}$.*
- *The problem of synthesizing a supervisor with the* least *number of states is $\mathcal{NP}$-complete.*

Even with these results at hand, the authors recognize that the main complexity problem lies in the composition methods they use to model more complex systems. The *shuffle* product and the constrained intersection operations yield an unacceptable (i.e. computationally unmanageable) increase in the number of states of the resulting models.

[22]In [Ram89] some of the results are given for languages generated by Büchi Automata.

However, the authors point out that it might not be necessary to actually build the product of the component generators to have a conjunct supervisor. The result might not be *minimal* in terms of number of states, but it could maintain some "structural properties" which might be valuable in its overall analysis and manageability.

3.3.5 Tasks Addressed and Recent Developments

The theory of DES's has some very appealing formal foundations and some nice computational aspects. I will here explore how it can be used to tackle the four tasks listed at the beginning of section 3, and how it recently developed.

Simulation. A model in the DES formalism can be used for simulation in a straightforward way.

Verification. The problem of verification is not directly addressed by Ramadge and Wonham. However, since the model representation is an automaton, it should be possible to use the theory in conjunction with the Temporal Logic and Model Checking or Proof techniques of, e.g. [CES86].

Performance Evaluation and Control The DES theory does not address the first problem directly; being all within the general framework of control theory, it instead recasts many well known notions developed for the second problem with continuous variables, in the challenging discrete events world.

One recent development to the theory of DES has been the addition of *explicit time information* to the theory (see [BWB92]). It is interesting to note that the "addition" has the same form that it takes in the framework of Time Petri Nets ([BD91]), i.e. it associates a time interval to the transitions of the automaton representing some system under study.

The notion of *distributed* supervisors has also been further developed: new results appear in [RW92].

Given the above new developments, it seems quite possible that the notions of supervisory synthesis developed in [BWB92] could be used within the Petri Nets framework.

3.4 Section Summary

There are many conceptual tools that can be used in conjunction with the four tasks of simulation, verification, evaluation and control for MS's; many of them have pragmatic – i.e. software – realizations.

The present section quickly surveyed two of them: Petri Nets and RW-DES Theory. Both tools have their appeals and research continues to be done each area. It is interesting to note that many efforts are currently devoted to augmenting both systems with time information ([BD91] and [Ajm89] for PN's and [BWB92] for RW DES). Moreover a great deal of attention is paid to reduce the computational complexity inherent to the structure of the underlying theories by defining modular and compositional concepts that reduce the sizes of the respective state–spaces.

Other tools are present in the MS's application literature, each with their own merits. However, an exhaustive discussion of all such tools would be beyond the scope of this survey paper.

4 Finale: Needs for Manufacturing Systems Design and Control Tools

In this survey, I described two approaches to the modeling and analysis problem for Discrete and Concurrent Processes: Petri Nets and Ramadge-Wonham Discrete Supervisory Control Theory (RW-Systems).

Both approaches have been studied with respect to some typical tasks occuring in the practice of designing and modeling Manufacturing Systems. These tasks were identified by a (necessarily) brief historical survey of the Manufacturing Field.

Common Problems It is my opinion that Petri Nets and RW-Systems are facing similar developments. The two main research directions are

- introduction of better *modularity* and *compositionality* concepts;
- introduction of *explicit time information*.

Colored Petri Nets [Jen92] and Decentralized Supervisory Systems within RW-Systems Theory [RW92] tackle the first problem. Stochastic and *Time-augmented* Petri Nets tackle the second from two different viewpoints; in [BWB92] explicit time information is added to a RW-System.

Missing Approaches There are many other approaches to the modeling and control problems for Manufacturing System.

A very important is the Temporal Logic approach ([MP92] and [CES86]). It is natural to look for some synthesis of Temporal Logic methods with the those described in this survey.

The scope of this survey has been wide. As such, it has just scraped the surface of many research efforts aimed to a better understanding and use of systems as complex as Manufacturing ones can be. The main result has been to form a solid ground upon which to build a hopefully useful and new research.

References

[ACF89] M. Ajmone Marsan, G. Chiola, and A. Fumagalli. *Improving the Efficiency of the Analysis of DSPN Models*, volume 424 of *Lecture Notes In Computer Science - Advances in Petri Nets*, pages 30–50. Springer-Verlag, 1989.

[AGM88] J. C. Ammons, T. Govindaraj, and C. M. Mitchell. Decisions Models for Aiding FMS Scheduling and Control. *IEEE Transactions on Systems, Man and Cybernetics*, 18(5):744–756, September 1988.

[Ajm89] M. Ajmone Marsan. *Stochastic Petri Nets: An Elementary Introduction*, volume 424 of *Lecture Notes In Computer Science - Advances in Petri Nets*, pages 1–29. Springer-Verlag, 1989.

[APN] Advances in Petri Nets. Yearly collection of significant papers in the field - appears in Springer-Verlag Lecture Notes in Computer Science.

[BCM+90] J. R. Burch, E. M. Clarke, K. L. McMillan, D. L. Dill, and L. J. Hwang. Symbolic Model Checking: 10^{20} States and Beyond. *5th LICS*, pages 428–439, 1990.

[BD91] B. Berthomieu and M. Diaz. Modeling and verification of time dependent systems using Time Petri Nets. *IEEE Transactions on Software Engineering*, 17(3):259–273, March 1991.

[BDMR91] E. Battiston, F. De Cindio, G. Mauri, and L. Rapanotti. Morphism and Minimal Models for OBJSA Nets. "Informatic Systems and Parallel Computation" Project i/4/26, C.N.R., 1991.

[BGV90] W. Brauer, R. Gold, and W. Vogler. *A Survey of Behaviour and Equivalence Preserving Refinements of Petri Nets*, volume 483 of *Lecture Notes In Computer Science - Advances in Petri Nets*, pages 1–46. Springer-Verlag, 1990.

[BHH82] G. R. Bitran, E. A. Haas, and A. C. Hax. Hierarchical Production Planning: A Two-Stage System. *Operations Research*, 30(2):232–251, March 1982.

[Bra91] J. C. Bradfield. *Proving Temporal Properties of Petri Nets*, volume 524 of *Lecture Notes In Computer Science - Advances in Petri Nets*, pages 29–47. Springer-Verlag, 1991.

[Bry86] R. E. Bryant. Graph-Based Algorithms for Boolean Function Manipulation. *IEEE Transactions on Computers*, C-35(8):677–691, August 1986.

[BWB92] B. A. Brandin, W. M. Wonham, and B. Benhabib. Manufacturing Cell Supervisory Control – A Timed Discrete Event System. In *IEEE International Conference on Robotics and Automation*, pages 931–936. IEEE, 1992.

[CES86] E. M. Clarke, E. A. Emerson, and A. P. Sistla. Automatic Verification of Finite-State Concurrent Systems Using Temporal Logic Specifications. *ACM Transactions on Programming Languages and Systems*, 8(2):244–263, 1986.

[Cri91] F. Cristan. Understanding Fault-Tolerant Distributed Systems. *Communications of the ACM*, 34(2):56–78, February 1991.

[DDPS82] F. De Cindio, G. De Michelis, L. Pomello, and C. Simone. Superimposed Automata Nets. In C. Girault and W. Reisig, editors, *Applications and Theory of Petri Nets – IFB 52*. Springer-Verlag, 1982.

[DJ89] W. J. Davis and A. T. Jones. A Functional Approach to Designing Architectures for CIM. *IEEE Transactions on Systems, Man and Cybernetics*, 19(2):164–174, 1989.

[Fis88] P. A. Fishwick. The Role of Process Abstraction in Simulation. *IEEE Transactions on Systems, Man and Cybernetics*, 18(1):18–39, January 1988.

[FM86] G. Franceschinis and G. Molinar Roet. Applicazioni delle Reti di Petri Stocastiche Generalizzate nello studio delle prestazioni dei Sistemi Flessibili di Lavorazione. (tran.: *Generalized Stochastic Petri Nets Applications in the Performance Evaluation of Flexible Manufacturing Systems*), Thesis for the Laurea degree in Computer Sciences, Universitá degli Studi of Turin, Italy (in Italian), 1986.

[Fre91] P. Freedman. Time, Petri Nets, and Robotics. *IEEE Transactions on Robotics and Automation*, 7(4):417–433, August 1991.

[Gen86] H. J. Genrich. *Predicate/Transition Nets*, volume 254 of *Lecture Notes In Computer Science - Advances in Petri Nets*, pages 207–247. Springer-Verlag, 1986.

[Ger89] S. B. Gershwin. Hierarchical flow control: A Framework for Scheduling and Planning Discrete Events in Manufacturing Systems. *Proceedings of the IEEE*, 77(1):195–209, 1989.

[GHFP89] I. Greenfeld, F. Hansen, J. Fehlinger, and E. Pavlakos. MOSAIC System Description, Specification and Planning. Technical Report 201, New York University, Courant Institute of Mathematical Sciences, Robotics Laboratory, 1989. Also listed as Computer Science report 452.

[GJ79] M. R. Garey and D. S. Johnson. *Computers and Intractability*. W. H. Freeman and Company, 1979.

[GLT80] H. J. Genrich, K. Lautenbach, and P. S. Thiagarajan. *Elements of General Net Theory*, volume 84 of *Lecture Notes In Computer Science - Advances in Petri Nets*, pages 23–164. Springer-Verlag, 1980.

[Gra81] S. C. Graves. A Review of Production Scheduling. *Operations Research*, 29(4):646–675, July 1981.

[HJS90] P. Huber, K. Jensen, and R. M. Shapiro. *Hierarchies in Coloured Petri Nets*, volume 483 of *Lecture Notes In Computer Science - Advances in Petri Nets*, pages 313–341. Springer-Verlag, 1990.

[Ho89a] Y. -C. Ho. Dynamics of Discrete Events Systems. *Proceedings of the IEEE*, 77(1):3–6, 1989.

[Ho89b] Y. -C. Ho, editor. *Special issue on Dynamics of Discrete Event Systems.* IEEE, January 1989.

[HU79] J. E. Hopcroft and J. D. Ullman. *Introduction to Automata Theory, Languages and Computation.* Addison Wesley, 1979.

[ICR] IEEE International Conference on Robotics and Automation. Yearly Conference.

[IPN] International Conference on Theory and Applications of Petri Nets. Yearly Conference - appears in Springer-Verlag Lecture Notes in Computer Science.

[Jan86] M. Jantzen. *Complexity of Place/Transition Nets*, volume 254 of *Lecture Notes In Computer Science - Advances in Petri Nets*, pages 413–434. Springer-Verlag, 1986.

[JD91] H. Jungnitz and A. Desrochers. An Overview of Analytical Performance Analysis Models for Manufacturing Systems. In *Proc. of the Winter Meeting of the ASME – PED*, volume 53, pages 71–82. ASME, 1991.

[Jen86] K. Jensen. *Coloured Petri Nets*, volume 254 of *Lecture Notes In Computer Science - Advances in Petri Nets*, pages 248–299. Springer-Verlag, 1986.

[Jen90] K. Jensen. *Coloured Petri Nets: A High Level Language for System Design and Analysis*, volume 483 of *Lecture Notes In Computer Science - Advances in Petri Nets*, pages 342–416. Springer-Verlag, 1990.

[Jen92] K. Jensen, editor. *Coloured Petri Nets.* Springer-Verlag, 1992.

[JK91] R. Janicki and M. Koutny. *Optimal Simulations, Nets and Reachability Graphs*, volume 524 of *Lecture Notes In Computer Science - Advances in Petri Nets*, pages 205–226. Springer-Verlag, 1991.

[JR91] K. Jensen and G. Rozenberg, editors. *High-level Petri Nets.* Springer-Verlag, 1991.

[JSS92] H. Jungnitz, B. Sáanchez, and M. Silva. Response Time Approximation for the Performance Analysis of Manufacturing Systems Modeled with Stochastic Marked Graphs. In *IEEE International Conference on Robotics and Automation*, pages 1000–1005. IEEE, 1992.

[KS91] V. G. Kountouris and H. E. Stephanou. Dynamic Modularization and Synchronization for Intelligent Robot Coordination: the Concept of Functional Time-Dependency. In *IEEE International Conference on Robotics and Automation*, pages 508–513. IEEE, 1991.

[KS92] V. G. Kountouris and H. E. Stephanou. Inducing Patterns of Behavior in Distributed System Processes. In *IEEE International Conference on Robotics and Automation*, pages 1070–1075. IEEE, 1992.

[LD92] D. Y. Lee and F. Di Cesare. FMS Scheduling Using Petri Nets and Heuristic Search. In *IEEE International Conference on Robotics and Automation*, pages 1057–1062. IEEE, 1992.

[LLRS90] E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan, and D. B. Shmoys. Sequencing and Scheduling: Algorithms and Complexity. In S. C. Graves, A. H. G. Rinooy Kan, and P. Zipkin, editors, *Handbooks in Operations Research and Management Science*, volume 4. North–Holland, 1990.

[Mal89] B. Malakooti. A Hierarchical, Multi–Objective Approach to the Analysis, Design and Selection of Computer–Integrated Manufacturing Systems. *Robotics & Computer-Integrated Manufacturing*, 6(1):209–228, 1989.

[Mer74] P. Merlin. *A Study of the Recoverability of Computer Systems*. PhD thesis, Dept. of Computer Science, UC Irvine, 1974.

[Mer85] M. E. Merchant. Computer-Integrated Manufactuing as the basis for the factory of the future. *Robotics & Computer-Integrated Manufacturing*, 2(2):89–99, 1985.

[Mit91] F. H. Mitchell Jr. *CIM Systems*. Prentice-Hall, 1991.

[MMS87] J. Martinez, P. Muro, and M. Silva. Modeling, Validation and Software Implementation of Production Systems Using High Level Petri Nets. In *IEEE International Conference on Robotics and Automation*, pages 1180–1185. IEEE, 1987.

[MMS+88] J. Martinez, P. R. Muro, M. Silva, S. F. Smith, and J. L. Villarroel. Merging Artificial Intelligence techniques and Petri Nets for real time scheduling and control of Production Systems. In *Prooceedings of the Twelveth IMACS World Congress On Scientific Computation*, July 1988.

[MP92] Z. Manna and A. Pnueli. *The Temporal Logic of Reactive and Concurrent Systems*. Springer-Verlag, 1992.

[MS91] H. Martins Shih and T. Sekiguchi. A Time Petri Net and Beam Search Based On-Line FMS Scheduling System with Routing Flexibility. In *IEEE International Conference on Robotics and Automation*, pages 2548–2553. IEEE, 1991.

[Pet79] C. A. Petri. *Introduction to General Net Theory*, volume 84 of *Lecture Notes in Computer Science*, pages 1–20. Springer-Verlag, 1979.

[Pet81] J. L. Peterson. *Petri Nets Theory and the Modeling of Systems*. Prentice-Hall, 1981.

[PR91] H. Plünnecke and W. Reisig. *Bibliography of Petri Nets 1990*, volume 524 of *Lecture Notes In Computer Science - Advances in Petri Nets*, pages 317–572. Springer-Verlag, 1991.

[Ram74] C. Ramchandani. Analysis of Asynchronous Concurrent Systems by Timed Petri Nets. Technical Report 120, MIT, 1974.

[Ram89] P. J. G. Ramadge. Some Tractable Supervisory Control Problems for Discrete-Events Systems Modeled by Büchi automata. *IEEE Transactions on Automatic Control*, 34(1):10–19, 1989.

[RC90] C. Reutenauer and I. Craig, editors. *The Mathematics of Petri Nets*. Prentice Hall, 1990.

[Rei85] W. Reisig. *Petri Nets: An Introduction*. EATCS Monographs on Theoretical Computer Science. Springer-Verlag, 1985.

[RW87] P. J. G. Ramadge and W. M. Wonham. Supervisory Control of a Class of Discrete Events Processes. *SIAM J. Control and Optimization*, 25(1):206–203, 1987.

[RW88] F. A. Rodammer and K. P. White, Jr. A Recent Survey of Production Scheduling. *IEEE Transactions on Systems, Man and Cybernetics*, 18(6):841–851, November 1988.

[RW89] P. J. G. Ramadge and W. M. Wonham. The Control of Discrete Events Systems. *Proceedings of the IEEE*, 77(1):81–98, 1989.

[RW92] K. Rudie and W. M. Wonham. Think Globally, Act Locally: Decentralized Supervisory Control. *IEEE Transactions on Automatic Control*, 37(11):1692–1708, 1992.

[SM90] Y. Souissi and G. Memmi. *Composition of Nets Via a Communication Medium*, volume 483 of *Lecture Notes In Computer Science - Advances in Petri Nets*, pages 457–470. Springer-Verlag, 1990.

[SV89] M. Silva and R. Valette. *Petri Nets and Flexible Manufacturing*, volume 424 of *Lecture Notes In Computer Science - Advances in Petri Nets*, pages 374–417. Springer-Verlag, 1989.

[TB91] S. H. Teng and J. T. Black. Autonomous Cell Control System Design with Petri Nets. In *Proc. of the Winter Meeting of the ASME – PED*, volume 53, pages 157–172. ASME, 1991.

[TF92] C. J. Tsai and L. C. Fu. Modular Approach for Petri Net Modeling of Flexible Manufacturing Systems Adaptable to Various Task-Flow Requirements. In *IEEE International Conference on Robotics and Automation*, pages 1043–1048. IEEE, 1992.

[Tuo89] H. Tuominen. *Elementary Net Systems and Dynamic Logic*, volume 424 of *Lecture Notes In Computer Science - Advances in Petri Nets*, pages 453–465. Springer-Verlag, 1989.

[Tuz92] A. Tuzhilin. SimTL: A Simulation language Based on Temporal Logic. *Transactions of The Society for Computer Simulation*, 9(2):86–99, 1992.

[Val90] A. Valmari. *Stubborn Sets for Reduced State Space Generation*, volume 483 of *Lecture Notes In Computer Science - Advances in Petri Nets*, pages 491–515. Springer-Verlag, 1990.

[Vyt92] J. Vytopil, editor. *Formal techniques in Real-time and fault tolerant systems*, volume 571 of *Lecture Notes in Computer Science*. Springer-Verlag, 1992.

[WB88] P. K. Wright and D. A. Bourne. *Manufacturing Intelligence*. Addison-Wesley, 1988.

[WD91] J. F. Watson III and A. A. Desrochers. Applying Generalized Stochastic Petri Nets to Manufacturing Systems Containing Non-exponential Functions. In *IEEE International Conference on Robotics and Automation*, pages 366–371. IEEE, 1991.

[Win91] W. L. Winston. *Operations Research: Applications and Algorithms*. PWS-Kent, 1991.

[Win92] P. H. Winston. *Artificial Intelligence*. Addison-Wesley, 3rd edition, 1992.

[WM89] K. P. White, Jr. and C. M. Mitchell. Manufacturing Systems Engineering: A Second Industrial Revolution? *IEEE Transactions on Systems, Man and Cybernetics*, 19(2):161–163, 1989.

[Won85] W. M. Wonham. *Linear Multivariate Control – A Geometric Approach*. Springer–Verlag, 3rd edition, 1985.

[ZD89] M. C. Zhou and Frank Di Cesare. Adaptive Design of Petri Net Controllers for Error Recovery in Automated Manufacturing Systems. *IEEE Transactions on Systems, Man and Cybernetics*, 19(5):963–973, September 1989.

[ZL91] M. C. Zhou and M. C. Leu. Petri Net Modeling of a Flexible Assembly Station for Printed Circuit Boards. In *IEEE International Conference on Robotics and Automation*, pages 2530–2535. IEEE, 1991.

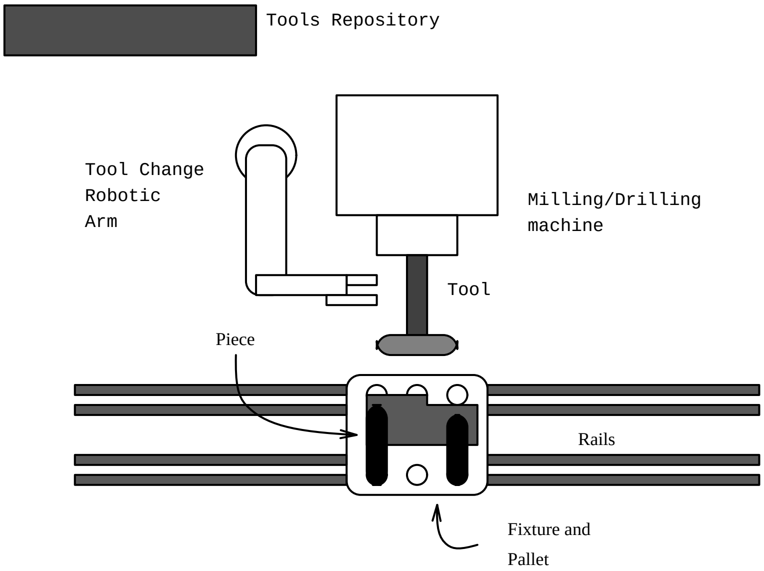


Figure 12: A depiction of the simulated Manufacturing Cell.

Appendix

A Simple Model of a Exemplary Manufacturing Cell

In the past months I realized a simple Time Petri Net (TPN) simulator. It is written in C++ and uses a simple X-Windows interface.

I tested the system by modeling the behavior of an exemplary manufacturing cell comprising

- a milling/drilling machine,
- a fixture for the pallet,
- a robotic arm for the tool change operation.

Fig. 4 shows the manufacturing cell.

The TPN model is logically divided among the three subsystems. I therefore have a net for the fixturing system, one for the milling/drilling machine and one for the toll changing arm.

The nets model both the "correct" behavior and the "erroneous" behavior of the systems. The faulty behavior is modeled by introducing "conflicts" in the net and resolving them by "external information". Figures 4 and 4

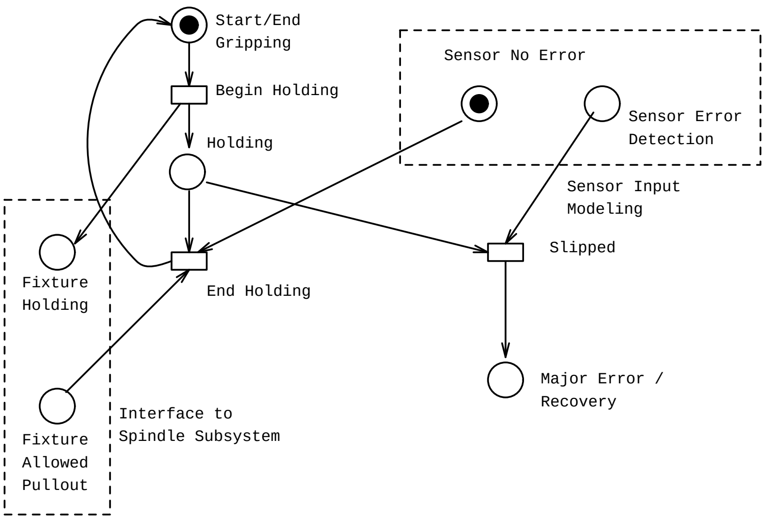


Figure 13: The TPN representing the behavior of the Fixture Subsystem.

show the models for the fixture and the spindle subsystems.

The interconnections among the subnets is done via places that are used to communicate the statuses of the components. Each Net is depicted with its initial marking and a time interval is actually associated to each transition.

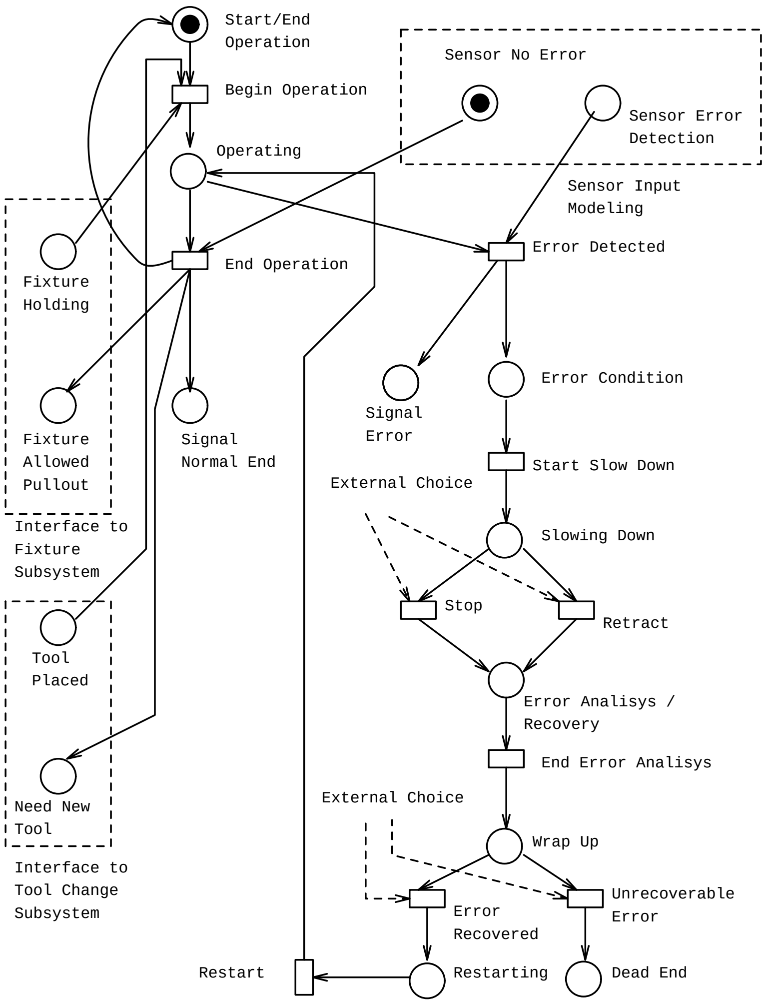


Figure 14: The TPN representing the behavior of the Spindle Subsystem.