(Draft) Proceedings of the III IMACS International Workshop on Qualitative Reasoning and Decision Technologies: QUARDET'93. 1993

A Frame-Rule Based Approach for Petri Nets Integration in FMS Control Models

P.R. Muro-Medrano, A. Ramírez, J. Pujol and J.L. Villarroel

Departamento de Ingeniería Electrica e Informática
University of Zaragoza
María de Luna 3, Zaragoza 50015, SPAIN
Tel: +34.76.517274 Fax: +34.76.512932
email: muro@etsii.unizar.es

EXTENDED ABSTRACT

keywords: Flexible Manufacturing Systems Model, Petri Nets, Decision Making, Knowledge Representation

referred topics Knowledge Based Decision Technologies, Intelligent Systems in Process Control, Applications to Manufacturing

1 Introduction

In recent years, several researches have proposed the use of Artificial Intelligence representation techniques for the description and manipulation of manufacturing system control models [FS84, FH87, RFHM86, ZZ89]. The most popular representation schema for declarative descriptions of domain dependent behavioral knowledge in knowledge based systems has been pattern/action production rules (rules are used in most expert systems). Nevertheless, production rules are inadequate for defining terms and describing manufacturing entities and static relationships between them. The solution for this lack of expressiveness has been accomplished in commercial AI environments (such as LOOPS or KEE) by integrating frame and production rule languages to form hybrid representation facilities.

However, the specification and modeling of the control logic in discrete event system such as a manufacturing system is a quite complex problem. Our conceptual approach for model construction is based on the integration of formal techniques, such as high level Petri nets (HLPN), in a knowledge based model. The main goal of this strategy is to take the advantages of the well known capabilities of Petri nets to express discrete event behaviour (states, activities, pre- and post-conditions, synchronization, concurrency) and its graphical features, and integrate them with other representation and reasoning possibilities provided by rule and frame based systems.

In the following paragraphs we propose a design approach for FMS control models Petri nets are used to define the behaviour of the FMS as a discrete event system and they are implemented using rules (the KEE rulesystem is used with this purpose). Interesting aspects of this design are related with our methodology to create dynamic entities, the inheritance mechanism used to synchronize objects and the strategy used to find decision problems and the incremental approach which is allowed to solve them.

2 Representation of isolated entities

In generating the manufacturing system model, each individual or class of manufacturing entity (products, machines, stores, transport devices, etc.) is represented by a frame, that we generically call object. An object can contain values, relations, metaknowledge, procedures and active values (demons). In addition

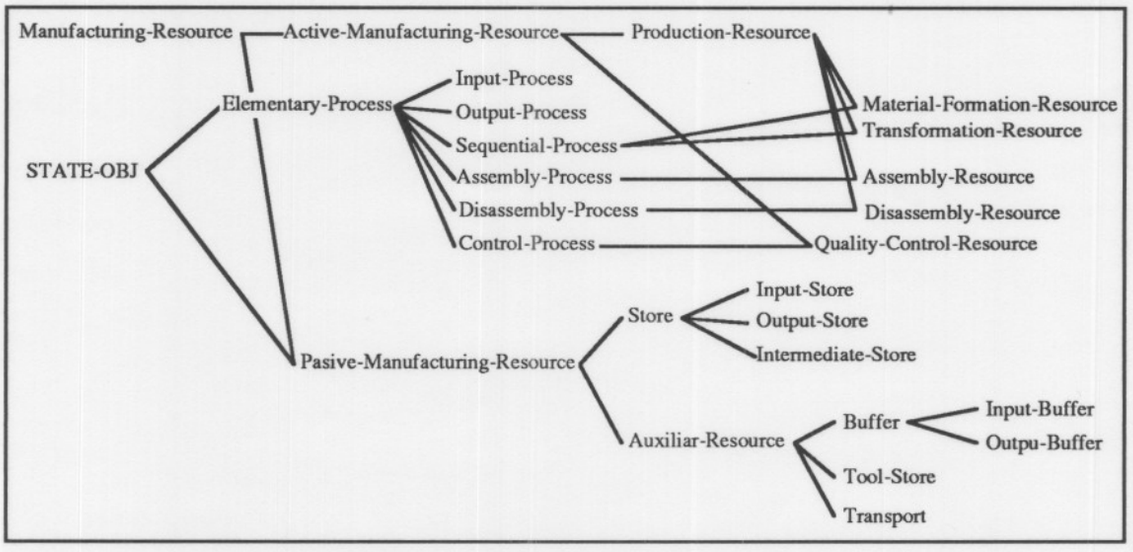

Figure 1: Part of the manufacturing resource hierarchy.

to these programming features supported by other frame and object oriented languages, our modeling approach includes a set of semantic primitives implementing a high level Petri nets [T.90, Jen86, MMEV] (HLPN) based formalism. HLPN are used as a formal mechanism to represent behavioral information of a dynamic entity, from a discrete event system point of view. Petri net places are mapped into specialized slots called state slots. Petri net transitions are mapped into rules whereas arcs are represented by rule premises and conclusions. On the other hand, colors in places are determined by objects stored in state slots. The rule system inference engine articulates the dynamic behaviour.

Following the object oriented programming methodology, the design process start by creating class hierarchies whose elements will be further instantiated in other to build the manufacturing system model. Three main hierarchies are used to deal with three different aspects: resources (physical), operations (functional) and products.

In order to understand the modeling mechanism available to create isolated entities, let us focus on a representative object of the manufacturing resource hierarchy which is shown in figure 1 called `Transformation-Resource`. This object holds the prototypical knowledge about a generic transformation resource. It has relations locating the entity in the abstraction hierarchy within the system model (`precision-level`, `has-resources` and `resource-of`). It has slots to store predicted information (e.g. `available-capacity` and `pending-operations` for example to be used by the scheduler), for data collection (e.g. to save information about functioning statistics), to describe physical parameters (e.g. set up and processing times and procedures) ...

On the other hand, it holds knowledge about its behaviour as a discrete event entity. Figure 2 illustrates a graphical representation of this behaviour using Petri nets graphical features. Places `loading`, `operate`, `unloading`, `in-process` and `capacity` are represented by their correspondent state slot in object `Transformation-Resource` (figure 2). Transitions `begin-operation`, `end-operation`, `start-process`, and `end-process` are implemented by rules which are identified in the corresponding action slots. Figure 3 shows the external representation of some of these rules which have been implemented in KEE. As is typical in object oriented systems, part of `Transformation-Resource` behaviour is inherited from its ancestors (in this case only from `Sequential-Process`.

Following an analogous approach, operations are defined by objects with an embedded Petri net. The behaviour of a typical transformation operation is represented by simple sequential net with three transitions (`start-process`, `start-operation` and `end-process`) and two places (`going-to-resource` and `in-process`).

```
{Transformation-Resource
  is-a : Sequential-Process Production-Resource
  states : loading operate unloading in-process
           avilable-capacity
      loaded :
      operate :
      unload-ready :
      in-process :
      available-capacity :
  actions : begin-operation start-process
            end-operation end-process
      begin-operation : begin-operation-M
            kind : input
      end-operation : end-operation-M
            kind : begin-op
      start-process : start-process-M
            kind : end-op
      end-process : end-process-M
            kind : output
  pending-operations : ; predicted info., data collect.
  completed-operations :
  statistics :
  max-part-size :          ; physical character.
  constraints :
  set-up :
  average-processing-time :
  precision-level :        ; abst. hierarchy relations
  has-resources :
  resource-of :
  input-synchro :          ; interface possibilities
  output-synchro :
  icon :                   ; graphic repres. parameters
  wickname :
  comment :     }
```

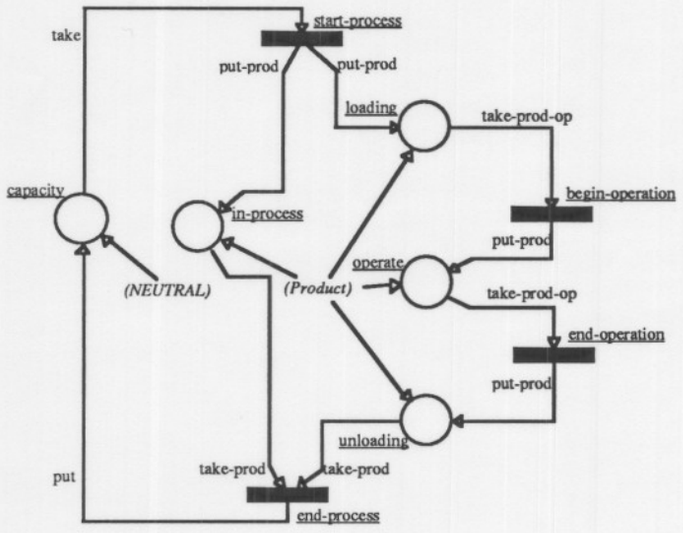

Figure 2: `Transformation-Resource` prototype.

2.1 Association of decision policies in isolated entities

It has been shown so far that the dynamic behaviour of an object is specified by an underlying Petri Net. One problem arises at this point due to non deterministic aspects of Petri nets. This non determinism produces decision problems which must be solved. Within the manufacturing systems domain most of these problems correspond with scheduling decision making: e.g. if a machine can store several available parts it may be necessary to choose one to be processed next.

In our approach, decision problems are identified by what we call *conflicts*. Conflicts are specialized control objects having information, about transitions with structurally related decision problems from the net point of view, as well as about how to solve the problem (control policy associated to the conflict). In order to make decision in an expert manner, conflicts can be characterized by its scheduling problem features, entities involved and production goals.

```
{START-PROCESS
    (IF (?PRODUCT IS IN CLASS PRODUCTS)
    (THE CAPACITY OF TRANSFORMATION-RESOURCE IS TRUE)
    THEN
    (CHANGE.TO (THE CAPACITY OF TRANSFORMATION-RESOURCE IS FALSE))
    (THE IN-PROCESS OF TRANSFORMATION-RESOURCE IS ?PRODUCT)
    (THE LOADING OF TRANSFORMATION-MACHINE IS ?PRODUCT))
}

{BEGIN-OPERATION
    (IF (THE LOADING OF TRANSFORMATION-RESOURCE IS ?PRODUCT)
    THEN
    (THE OPERATE OF TRANSFORMATION-RESOURCE IS ?PRODUCT)
    (DELETE (THE LOADING OF TRANSFORMATION-RESOURCE IS ?PRODUCT)))
}
```

Figure 3: KEE system rules representing transitions `begin-operation`, and `start-process`.

3 Adding constraints due to entity connections

Once the object class hierarchy is available, the user is able to choose and instantiate the entities which are needed to build the system model. The next step is to establish connections between objects, between physical resources to build the plant resource layout, between operations to build the different process plans, and then between plant layout and process plans to get the dynamic behaviour of the whole system.

Dynamic entity connections are established by means of transition synchronization. Only two synchronization types have been needed so far in our models, normal and bilateral. A normal synchronization implies the substitution of the transition by other transition with the premises and conclusions of the transitions been synchronized (the original one is lost), further synchronization of the transition are affected by this one. On the other hand, in a bilateral synchronization a new transition is created with the premises and conclusions of the transitions been synchronized (the original one is preserved), further synchronization of the transition are not affected by this one.

In KEE rules can be manipulated as objects within an abstraction hierarchy. Following the object oriented approach, both synchronization can be easily implemented using the inheritance properties of the rules. A normal synchronization implies the substitution of the rule representing the transition by another rule having as ancestors the ones of the synchronized transitions, all of these ancestors will be considered for further synchronization. In a bilateral synchronization a new rule is created with all the involved ancestors and only the ancestors belonging to the initial rule will be considered later on.

4 Decision making issues

The incremental approach shown to build the dynamic definition of the model allows an incremental approach for the design of decision making policies. Although this incremental strategy does not guarantee the best decisions, it is good enough to automatically find a default policy at early development steps (during prototyping), leaving the optimization of that policy for more advanced design decisions.

New decision points appear with the connections between dynamic objects. Each synchronization, normal and bilateral, produces a new conflict. Normal synchronization produces a more constrained conflict whereas bilateral produces two interrelated conflicts, one more constrained and another new one due to the creation of a new path in the net (of course coupled conflicts may appear applying the transitive closures).

Control policies from old conflicts are maintained and related with new conflicts. In this way, control policy of a new conflict can be as simple as the election between old control policies. An interesting example appears when synchronizing plant layout and process plans, the new control policy can choose between resource and operation strategies allowing in this way opportunistic reasoning.

References

[FH87] A.M. Flitman and R.D. Hurrion. Linking discrete-event simulation models with expert systems. *Journal of the Operations Research Society*, 38:723–733, 1987.

[FS84] Mark S. Fox and Stephen F. Smith. Isis: A knowledge-based system for factory scheduling. *Expert Systems*, 1(1):25–49, July 1984.

[Jen86] K. Jensen. Colored petri nets. In Goos G. and Hartmanis J., editors, *Petri Nets: Central Models and their Properties*, volume 84 of *Lecture Notes in Computer Science*, pages 248–299. Springer Verlag, 1986.

[MMEV] P. R. Muro-Medrano, J. Ezpeleta, and J.L Villarroel. A rule-petri net integrated approach for the modeling and analysis of manufacturing systems. In *To appear in the Proceedings of the 13th IMACS World Congress.*

[RFHM86] Y.V. Reddy, M.S. Fox, N. Husain, and M. McRoberts. The knowledge-based simulation system. *IEEE Software*, pages 26–37, March 1986.

[T.90] Murata T. Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, 16(1):39–50, January 1990.

[ZZ89] G. Zhang and B.P. Zeigler. *Artificial Intelligence, Simulation and Modeling*, chapter The System Entity Structure: Knowledge Representation for Simulation Modeling and Design, pages 47–74. Wiley Interscience, New York, 1989.