(Draft) Annual Review in Automatic Programming. 1994, vol. 19, p. 313-317

AIRTC 94 Valencia. Oct. 94

INCREASING A KNOWLEDGE REPRESENTATION SCHEMA FOR FMS CONTROL WITH FAULT DETECTION AND ERROR RECOVERY CAPABILITIES

J. A. PEREZ, J. PUJOL and P. R. MURO-MEDRANO

Dpto. de Ingeniería Eléctrica e Informática,
Centro Politécnico Superior, Universidad de Zaragoza
María de Luna 3, E-50015 ZARAGOZA, SPAIN

Abstract. This paper presents the integration of Artificial Intelligence representation techniques and high level Petri nets in a knowledge based manufacturing modeling and analysis system. By means of Petri nets and its implementation based on rules, a model of the system can be constructed. In order to verify the real behavior, the control system interchanges signals with the plant. When an abnormal situation is detected, the control system must take control decisions to allow the system to work in the new situation. Error management in these cases includes detection, diagnosis and reaction. In this paper the hierarchy of object classes to control the system is enlarged to face with the problems produced by abnormal behavior.

Keywords. Artificial Intelligence, Failure Detection, Flexible Manufacturing, Knowledge Engineering, Simulation, System Failure and Recovery

1. INTRODUCTION

AI knowledge representation schemas go beyond classical model formalisms in allowing dimensions of representation such as inference of new knowledge, associative and other accesses to previous knowledge, matching of patterns, and metaknowledge (Muro-Medrano *et al.* 1992). 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). Rules can be easily understood by designers and have enough expressive power to represent a useful range of domain dependent behavior specifications. Nevertheless, production rules are inadequate for defining terms and describing manufacturing entities and static relationships between them. The solution for this lack of expressivity has been accomplished in commercial AI environments (such as LOOPS[1] or KEE[2]) by integrating frames and production rule languages to form hybrid representation facilities. Additional problems arise in the knowledge representation for discrete event systems applications. These problems are related with resource sharing, concurrent and parallel actions, states, action pre- and post-conditions, conflicts and etc. Some of these concepts are easily represented by means of rules, however when designing the model it become necessary some analysis and design aids such as a clear graphical representation, clear semantics and some kind of formal features to facilitate analysis. The last features have been accomplished by integrating high level Petri nets within the representation schema. We have develop a software application to support this representation (KRON (Muro-Medrano 1990)) and we have expanded the object taxonomy with objects specialized in the manufacturing domain.

The aim of this paper is to show a first step to increase the expression power of our representation schema with some real time capabilities in the field of fault detection and error recovery. Sections 2 is intended to introduce the representation schema whereas sections 3 and 4 show some fault detection and error recovery functions.

[1] LOOPS is a trademark of Xerox Corporation.
[2] KEE is a trademark of IntelliCorp.

(Draft) Annual Review in Automatic Programming. 1994, vol. 19, p. 313-317

2. THE BASIC REPRESENTATION SCHEMA

The kernel for the representation is a frame based knowledge representation language intended for discrete event systems, it belongs to the object oriented programming paradigm and integrates HLPN to represent the dynamic aspects.

The information related to a concept or physical entity, having a dynamic behavior, is articulated around three classes of specialized objects. **Transitions** are rules defining possible activities of a dynamic entity that can produce a state change. They are equivalent to the transitions of a HLPN. **State objects** represent the dynamic entities. They contain the state information and additional knowledge (physical description, relations, historical information, etc.). State knowledge is located in place slots representing single Petri net places. The activities that can produce a state change are also saved in transition slots. **Tokens** are the information contained in the place slots. From the Petri net point of view they are the tokens evolving in the net. Working in this way, dynamic behavior of system elements is modeled by Petri nets, and Petri nets are represented by frames and rules. Petri net transitions are mapped into rules whereas arcs are represented by rules premises and conclusions. Premises are composed by the conditions defined by the input arcs to the transition, and the conclusions are the output arcs from the transition (Muro-Medrano *et al.* 1993). The Petri net graphical representation appears as a graphical representation of the relations among the rules that implement the transitions. Finally, there is an inference engine which determines the evolving of the system. It is implemented by means of a RETE algorithm in order to improve its efficiency.

To be more suitable for the application domain, the basic object hierarchy is expanded with manufacturing entities (Villarroel-Salcedo *et al.* 1992). The designer can use specialized objects to create models of manufacturing plants (machines, stores, transports ...) process plans (operations) and products. The new goal, is to improve the representation schema here presented with fault detection, reaction and error recovery capabilities, so that the object hierarchy and modeling methodology must be enlarged, still supporting the same representation style.

In order to illustrate the different concepts we will follow an example of a real system. The system, shown in figure 1, consists of a transport system commonly used in the automobile industry, based on movable tables. The components of the system

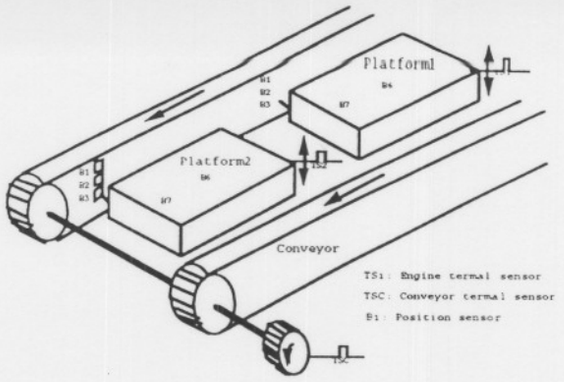

Fig. 1. Drawing of the transport system based on movable tables.

are tables (platforms which can move up and down), some of them can accept cars coming from other conveyor, and others which only act as intermediate buffers capable to store one car on themselves. Each platform has sensors used to detect different locations in its route (B1, B2, B3). Other sensors are used to detect cars upon the platform (B6, B7). These sensors provide the necessary signals to set up and stop the platform. Additional sensors are placed in electrical engines to detect overloads due to blockages.

In figure 2 a drawing of the Petri net modeling a single platform is shown. The example modeled consist of an entity of the system. It has some places (up, move-down1, move-down2, down, move-up1, middle, move-up2 and loaded-car) and transitions (request-next, unload-car, come-down, accept-previous, come-middle, load-car and come-up). By means of tokens marking these places, the status of the platform is completely described. In the example figure, token in state up minds that the platform is in its upper position, and its next possible operation is to pass to the state move-down1 by means of the transition request-next. Tokens can have additional information, as this in state loaded-car. It minds that the platform has a car loaded on it, and slots in the token object hold information about the type of car loaded.

Once a single object has been modeled, it is necessary to connect objects between them. This is done by means of a special relation mechanism called synchronization relations. **Synchronization relations** provides a simple way for interconnecting state objects by establishing synchronization between their transitions. When a synchronization is carried out, the rules modeling the synchronized transitions are enlarged with new pre and postconditions, so that the new one merges constraints from the previous transitions.

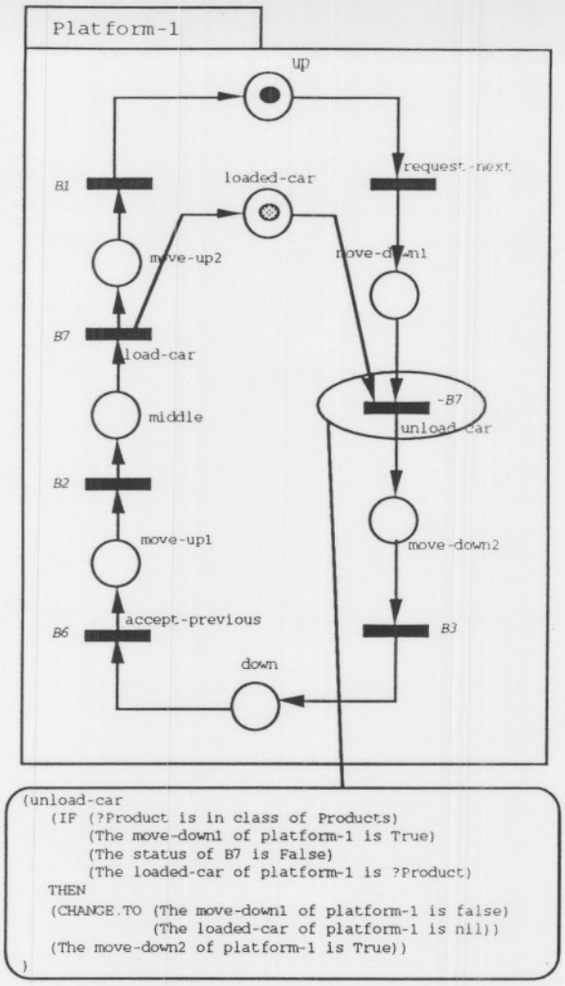

Fig. 2. Petri net modeling a part of the transport system.

3. MONITORING AND FAULT DETECTION

Detection of faults and error recovery becomes a necessity in real systems, so that the modeling method must be upgraded to a more powerful one that supports error detection and management functions.

When a fault or an anomalous situation is detected, a reaction mechanism must be automatically activated to react against the contingency. A very large set of possible faults and contingencies for a particular entity. It is not a good strategy to have a very complex Petri net modeling all these possibilities for all possible objects. It would be too difficult to model and not very efficient to be a practical system controller. The performance level of the controller would be decreased and the reusability of the objects would be lost.

Detection of faults in the real system is based in a model of the process that is contrasted with the informations received from the plant. When these situations are detected, new objects appear, either to correct the system state when possible, or to led it to a safe state until the control could be recovered again.

The work presented here has been influenced by several research works: (Sahraoui *et al.* 1987, Toguyeni *et al.* 1990, Zhou and DiCesare 1989) where Petri nets are used for the modeling of control systems with monitoring and error recovery capabilities; (Fox *et al.* 1983) uses schema-based representation for the sensors; and (Holloway and Krogh 1990) where actors are used for fault detection and diagnosis in FMS. Some additional information about Petri nets used as controllers can be found in (Krogh and Holloway 1991, Zhou *et al.* 1992). There are some related works which deal with topics about monitoring and modeling of systems in the context of AI (Sahraoui *et al.* 1986, Castelain and Gentina 1988).

To be able to perform fault detection in this model, the object hierarchy is expanded with a new kind entities Alarms and sensors in a real time environment, or events in a simulation environment. These signals are the base for detection. They inform the model about the actual state of the system. The information provided by these elements is used in preconditions of rules modeling transitions which need a particular sensor status to be fired.

Control objects can be connected to the model of the system to perform a checking and validation of the correctness of the operation ordered to the system. Two kind of entities are recognized: **Watch-dogs**: A watch-dog is a control object that constraints the maximum and minimum time allowed for an operation. When these constrains are violated, the system must react immediately (as in the alarm case) **Time-outs**: They are a particular kind of watch-dogs that only restrict the maximum allowed time for the operation. They act in the same way as the watch-dogs.

The fact that a Petri net is modeling the system has some advantages. The marking of the net mirrors the actual state of the physical system. This marking is updated in real time by means of the information provided by sensors in the real plant. In this way, the net is a monitoring system. To know the state of the plant, the only thing to do is to look at the Petri net and its marking. In the model, the state and position of each table can be known by looking at places in the net.

Monitoring objects, as said before, are the alarms and sensors. When one of these signals arrives to the model, the Petri net brings up to date its

status to reflect the actual one. These signals must arrive at the model within a concrete period of time. To verify that signals arrive when expected, and to detect signals out of time, time-outs and watch-dogs can be connected to the transitions modeling the beginning and the end of the operations whose execution time has to be controlled. Timed transitions become necessary to model these objects, but they can be easily implemented by adding additional rules with time conditions.

4. METHOD FOR ERROR RECOVERY

When an alarm is arisen, or a control object detects an operation having a duration out of the allowed margins, something wrong is happening in the physical system. The first reaction in the example case must be to stop the system, looking after safety in the plant (one possible cause of the anomalous behavior can be a person trapped by a movable table or by the conveyor belt). Once the system has been stopped, the situation is notified to the plant supervisor, who decides either to continue in automatic operation mode or to change to manual mode until the fault can be fixed. If operation mode is changed to manual, a new Petri net modeling the current dynamics of the system is necessary. The new Petri net minds that a new set of rules are activated when some fault is detected. The old set of rules modeling the Petri net in its correct status, is disabled, and the auxiliar rules are enabled. This is the physical reaction mechanism. This new set of rules containing the Petri net for the new mode of work is a kind of physical reaction object, because the new Petri net imposes the new physical constraints for the system. Using this object, the system can be modeled and controlled when an abnormal situation occurs.

When the system is running in a degraded mode of work, it is modeled by means of a Petri net; therefore, there is a marking which represents the state of the system. In this way, a fault that occurs in the system only changes its configuration and its model, but the model system continues being a valid one. The reaction happens in a foreground process in such a way that the system only knows that something has happened because of the alarm signal raised.

When the damaged model has been carried to a safe state, and the plant supervisor decides that the problem has been fixed, the correct status can be recovered, and the initial net can be rebuilt. In this step, the token marking the safe state where the system has been led (down in the platform net) is translated to the initial net and marks the corresponding place. These operations are performed when the *rebuild* transition is fired.

There is also another kind of reaction that could be called *functional reaction*: In a higher control level, a set of actions related to production goals or scheduling must be taken. Operation plans must be modified to avoid stopped resources in the plant. A new production strategy must replace the old one that works in correct states, in a similar way as was done for physical resources replacement. New decisions can be taken at a higher control level, imposing new scheduling policies. The input of new cars can be prevented in the cell that models the transport system, or an unload of this cell can be ordered when a fault is detected. A new production strategy can replace the old one.

5. CONCLUSIONS

In this paper a knowledge representation scheme has been described. This schema is based on frames to store knowledge, Petri nets as a formal tool to describe the behavior of the system, and rules to implement the transitions of the Petri net. Some of these rules have additional constraints to take into account time in order to detect abnormal behavior of the system. Some specialized auxiliar rules are activated when a fault is detected in the system, and the old ones are disabled, until the correct status of the system is recovered.

REFERENCES

Castelain, E. and J.-C. Gentina (1988). Petri nets and artificial intelligence in the context of simulation and modelling of manufacturing systems. In 'Proceedings of IMACS International Symposium on System Modelling and Simulation SMS'88'. Cetraro, Italy. pp. 28–33.

Fox, M., S. Lowenfeld and P. Kleinosky (1983). Techniques for sensor-based diagnosis. In 'Proceedings of the International Joint Conference on Artificial Intelligence'.

Holloway, L. and B. Krogh (1990). Fault detection and diagnosis in manufacturing systems: A behavioral model approach. In 'Second International Conference on Computer Integrating Manufacturing'.

Krogh, B. H. and L. E. Holloway (1991). 'Synthesis of feedback control logic for discrete manufacturing systems'. *Automatica* **27**(4), 641–651.

Muro-Medrano, P. R. (1990). Aplicación de Técnicas de Inteligencia Artificial al Diseño

de sistemas Informáticos de Control de Sistemas de Producción. PhD thesis. Universidad de Zaragoza.

Muro-Medrano, P. R., A. Ramírez, J. Pujol and J. L. Villarroel (1993). A frame-rule based approach for petri nets integration in fms contro models. In 'Proc. of the IMACS on Qualitative Reasoning and Decision Technologies'.

Muro-Medrano, P. R., J. Ezpeleta and J. L. Villarroel (1992). A rule-petri net integrated approach for the modeling and analysis of manufacturing systems. In 'Proc. of the IMACS on Robotics and Flexible Manufacturing Systems'.

Sahraoui, A., H. Atabakhche, M. Courvoisier and R. Valette (1987). Joining petri nets and knowledge based systems for monitoring purposes. In 'Proc. of IEEE International Conference on Robotics and Automation. Raleigh, North Carolina'. pp. 1160–1165.

Sahraoui, A., M. Courvoisier and R. Valette (1986). Some considerations on monitoring in distributed real-time control of flexible manufacturing systems. In 'Proc. of IEEE IECON, Milwaukee, USA'.

Toguyeni, A., E. Craye and J. Gentina (1990). A method of temporal analysis to perform online diagnosis in the context of flexible manufacturing. In 'Proceedings of the IECON 1990'. IECON.

Villarroel-Salcedo, J. L., J. A. F. Lladó, J. S. Martín and P. R. Muro-Medrano (1992). 'Entorno de diseño de software de control en cim'. *Informática y Automática.*

Zhou, M. and F. DiCesare (1989). 'Adaptative design of petri net controllers for error recovery in automated manufacturing system'. *IEEE Transactions on Systems, Man and Cybernetics.*

Zhou, M. C., F. DiCesare and D. L. Rudolph (1992). 'Design and implementation of a petri net based supervisor for a flexible manufacturing system'. *Automatica.*