A RESONABLE APPROACH FOR MANUFACTURING SYSTEM
BASED ON SUPERVISORY CONTROL USING DISCRETE EVENT
SYSTEM- A REVIEW

S.C. Ramesh\textsuperscript{a}, Dr.G.Kalivarathan\textsuperscript{b}
\textsuperscript{a} Research Scholar, CMJ University, Meghalaya, Shillong.
\textsuperscript{b} Principal/ PSN Institute of Technology and Science, Tirunelveli, Tamilnadu, Supervisor, CMJ university, Shillong. Email:sakthi_eswar@yahoo.com

ABSTRACT

This paper reveals a novel and generic PC/PLC-based software/hardware architecture for the control of flexible manufacturing workcells. The proposed implementation methodology is based on the utilization of any one of the available formal discrete-event-system control theories in conjunction with state-of-the-art industrial programmable-logic controllers(PLCs). The modular control software architecture has been developed for MS-Windows environments. The effective graphical user interface provides a transparent programming environment. A detailed plant model for the testbed was created (size on the order of $10^{16}$ states) and 29 modular supervisors were designed to implement the control specifications. Several model reduction theorems were created to handle a plant this large. The clocked Moore synchronous state machines (CMSSM) was then implemented as relay ladder logic programs on the PLC. The testbed supervisors were implemented using the algorithm, and the testbed is now operational. This demonstrates that the algorithm works. The introduction of advanced information systems and machinery with communication capability and operational versatility provide the manufacturing systems with the capacity of material modification and transport, changes in material processing routes, different product insertion in the production line and plant layout reconfiguration. These characteristics give the production process great flexibility.

Keywords: Supervisory Control, Programmable logic controllers, Discrete Event System, clocked Moore synchronous state machines.
1. INTRODUCTION

Current manufacturing strategies adopted by many industrial companies necessitate the use of automated production systems which can be reconfigured and reprogrammed with great efficiency. In this context, the utilization of Flexible Manufacturing Workcells (FMCs) have been advocated for the fabrication of lower-volume, but wider-variety, products. An FMC is defined herein as a system comprising automatic processing machines typically serviced by robotic material-handling devices working under the control of a supervisor. In the academic literature, FMCs have been modeled as Discrete-Event Systems (DESs) utilizing a variety of control theories, most notably by Petri Nets (PNs) and Ramadge-Wonham Automata theory. Although, such formal control methods of workcells, modelled as DESs, have received wide attention within academia, these theories have not been widely accepted by industry and their implementations are scarce. The literature has also advocated the use of PCs for the direct control of FMC, utilizing formal control theories and, thus, providing an open hardware architecture. In and a host-computer was used as a supervisor for the control of the DES using Petri-Nets. In both cases, devices were directly wired to the host computer without the use of a PLC. On the other hand, the manufacturing workcell was controlled using Programmable-Logic-Controllers (PLCs), which were programmed using Ramadge-Wonham theory. In order to bridge the existing gap between formal control methodologies and their implementation, a software package that can be used to synthesize and implement formal supervisors in a user friendly environment is needed. Such software should allow the implementation of formal control methodologies without the need for the user to have a complete knowledge of such theories. In this paper, such a software package, developed based on the Extended Moore Automate (EMA) theory concepts is described in detail. To efficiently communicate with PLCs that in turn allows efficient communication with cell devices and their control. Software is part of a generic implementation methodology, that advocates the use of a hybrid PC/PLC supervisory control system based on the strengths of each device: namely, PCs provide users with computational and human/machine-interface flexibility in order

2. THE TESTBED COMPONENTS

The structure of the testbed is shown in Fig. 1. It contains three basic hardware components: personal computer (PC), interface and programmable logic controller (PLC).

Figure 1. The structure of the testbed
Main function blocks of these components are shown in Fig. 2. As the matrix model of an uncontrolled process is described in Simulink, PC should have installed Matlab with RTW. Furthermore, a board with digital I/Os has to be included in a PC hardware configuration. Inputs and outputs of the uncontrolled process model in Simulink are connected to modules, which communicate with the I/O board. As levels of signals on the I/O board and PLC are different (TTL versus 24V) the board sends/receives signals to/from PLC through an interface. A PLC configuration depends on the process to be controlled and the implemented control algorithm. Fast processes with numerous states require PLCs with high computational power and large number of I/O units.

Figure 2. Main function blocks of the testbed.

3. DEVELOPING THE MODEL

Important issues in developing plant models are how to reduce the modeling of a large plant into several small plant models, and how to avoid unnecessarily complicating the model. It is important to model large plants as a group of smaller sub-models for the following reasons:

- For large systems like the testbed (plant model on the order of 10^16 states) it would be extremely difficult to come up with a single model.
- It is more accurate to design several small models as opposed to keeping track of all the intricacies of one large model.
- It is quicker to design several small models than one large model.

The difficulty arises in deciding how to break down a large plant model into component models. A useful approach is to break the sub-models into two categories: fundamental sub-models, and interaction sub-models. The fundamental models are the models of the basic elements of the plant by themselves, ignoring how these elements interact with each other. For the testbed, the fundamental models are the trains, cranes, sensors and switches. The interaction models describe how the fundamental models interact with each other. For the testbed, the interaction
models are the switch request handlers, the sensor interdependencies, and the sensor dependencies on the trains, and switches. Breaking the plant model down into these categories is intuitive and clearly shows the components of the plant and how they relate to each other. Often there will still be large models within these categories that require further decomposition into component models. For the testbed, this was the case with the interaction models for the sensors, trains, and switches. These three elements are intimately related. For a given train, sensors can only be reached in a certain order, starting from the train's initial position. Depending on the current location of the train, certain sensors may be unreachable if a switch is in the wrong position. To model this for the six switches and one train would require a plant model of over 300 states. Unfortunately, it was not immediately obvious how to reduce this model. The trick is to remember that not every detail of the plant's behavior is required in every model, but every detail must be in at least one model. When the models are combined using the sync operation, the various components of the plants behavior are combined. This means that the order in which the sensors can be reached for a given train can be modeled while ignoring the effect of the switches. Then, a model can be created for each switch for a given train. This model only needs to show how the particular switch effect access to the sensors immediately before and after the switches. It is possible to compartmentalize the plant behavior this way, because the effect of the sync operation is to restrict the occurrence of common events until all sub-models that contain the event declare that the event can occur.

4. MODEL VALIDITY

The most important thing about developing a model is that the model is valid for the physical plant. If this isn't the case, there is no point in verifying if a supervisor is controllable and nonblocking for the plant. The problem is that you have no guarantee. It could apply a formal method to verify the accuracy of your plant model, but this would require that you start with some assumptions about the plant. To verify these assumptions would require simpler assumptions. The point is that there is no guarantee that your plant model will be accurate, but this will be the case however you tackle the problem. When you formalize the problem with a plant model, you at least know that you can guarantee the accuracy of your work from this point. By modeling the plant as a group of smaller sub-models as described above, you are able to work at a level that you can reasonably come up with an accurate model.

![Figure 3. Example of an Expanded Model](image-url)
5. DESIGNING SUPERVISORS

When designing supervisors, it is imperative to remember that a supervisor enforces the specification given, not necessarily the specification intended. There is no guarantee that the specification used actually solves the intended problem. Designers must ensure that they understand the problem well and that they cast their solution correctly in terms of a DES control specification. This problem will exist in any other design method. At least when the solution is formally specified, you can guarantee that the supervisor you design meets the specification. When designing DES supervisors, it is necessary to think in terms of event sequences. Since designers live in a world of space and time, they must remember that time is not represented in standard RW theory. The plant model doesn't say when an event can occur, only if it is able to occur at that point. Supervisors cannot be based on an event occurring within a given time, but on events occurring signifying that something about the plant has changed. Designers must and an event (or sequence of events) that signifies that a desired condition regarding the plant is true.

6. SUPERVISORY CONTROL THEORY MODELS

The constructed models and synthesized supervisors were designed and synthesized by using the local modular control approach. The AIM system was modeled dividing it into segments and cells, where supervisors were synthesized for each segment and machine. As the paper’s focus is to present the interaction between the SCT models and CPN, only the models where deterministic decisions are required are presented.

![Figure 4. Partial view of the transport system](image)

The first model is the Shunt (represented in Figure 4 by SHx, where x is the number of the device), the place where it is decided whether or not a pallet should change its path. The shunt segment is composed by three devices: One Shunt device and two Stoppers (represented in Figure 4 by STx, where x is the number of the device). The Stoppers are devices in the transport system that prevent collisions and provide controllable pallet releases throughout the system. In the SH1 device a decision is made whether the arriving pallet must continue to the welding station, or skip that station and continue its path to the other parts of the system. This is decided by the manufacturing sequence. The second model is the resulting supervisor for the welding machine, capable of performing two different types of welds. The cell is composed by a clamping device to keep the pallet in place and the machine itself.
7.0 CONCLUSIONS

In this paper it was presented how a hybrid approach for designing a control environment for flexible manufacturing systems could result in a solid and formal based solution, taking advantage of what SCT and CPN theories have to offer. Knowing that manufacturing systems complexity will continue to increase, the development for a consistent formal method for designing control systems becomes mandatory. Computational power will continue to grow, and with this, more possibilities of expanding the theories proposed. While intensive research has focused on the control of workcells, many of these projects have been purely theoretical and lack the implementation of the corresponding workcell controller. The implementation methodology of formal-DES-theory-based supervisors, as proposed in this paper

REFERENCES