APPLYING A SOFTWARE FRAMEWORK FOR SUPERVISORY CONTROL OF A PLC-BASED DISCRETE EVENT SYSTEM

B. Curto, V. Moreno
Department of Informática y Automática, Salamanca University, Plaza de los Caídos, Salamanca, Spain

C. Fernández-Caramés, R. Alves, A. Chehayeb
Department of Informática y Automática, Salamanca University, Plaza de los Caídos, Salamanca, Spain

Keywords: Supervisory Control, Automated Manufacturing Systems, Software Framework, Discrete Event System.

Abstract: In this paper we propose a software framework where the main aim is to make easier the implementation of supervisory control. The main idea is that functionalities must be offered with no effects on the robustness of the system. We prove that our methodology has a solid base, so the approach can be applied to any kind of industrial process unit. In this way, a real application of a developed framework is presented. The implementation is done over well-known devices (such as PLC with a conveyor belt) so the technical feasibility of the procedure is guaranteed.

1 INTRODUCTION

Due to the market evolution, manufacturing industry suffers some pressures in order to reduce product prices, to increase the model complexity and to support a model proliferation (Beck, 2000). To achieve these goals, it is necessary to accomplish shorter production cycles and lower manufacturing costs. More flexible and intelligent workcells are needed in order to find a competitive manufacturing process. Thus, as a part of a Computer Integrated Manufacturing (CIM) goal, Flexible Manufacturing Systems (FMS) offer the best promise of reducing costs and increasing flexibility.

Current robotic workcell systems can generally be classified as large, monolithic and centralized systems. The main reason comes from the fact that, in a typical industrial workcell, a Programmable Logic Controller (PLC) acts as an operation sequence controller of the cell. Several problems arise with these systems: limited functionality and flexibility, low levels of intelligence, and so on. As a solution we will propose the use of a distributed architecture design.

In (Curto, 2001), a proposal of a software architecture is realized where the development of distributed systems is taken into account. In a natural way, the development of a framework is the next step where some wrong situations appear when the provided services at the system becomes blocked due to an erroneous sequence of service invocations. Thus, it is necessary to achieve that these blocked situations will be avoided.

We propose the use of the Supervisory Control Theory (SCT) (Ramadge, 1987) that is proposed initially by P. J. Ramadge and W. M. Wonham in the late 80s. This theory has received a special attention from the academic environment in such a way it has reached a great evolution. Nevertheless, actually just a few applications at industrial environment can be found that applies the SCT due to mainly a complete model of the system (with a sound mathematical load) and a corresponding implementation is needed.

Although some works exist where the real implementation of controller is obtained using the SCT, they are not too much. Some of the most representative are (Chandra, 2000) and (Mušić, 2002). The first one describes the way to design a control system for assembling line whereas the second one is focused on the implementation of a concrete task on a PLC. In both cases, the controller formally obtained using the SCT is restricted to reach a particular goal at a concrete task.

This approach presents two main disadvantages. First, if a modification of the task is made, it is
necessary to perform the complete redesign and implementation of the controller. Second, a unique controller is not feasible when the task implies a large set of sensors and actuators. In this situation, the physical restrictions imposed by the communications and a possible overload of the controller are two questions that have to be taken into account. Consequently, we propose to make a separation between the particular device control and the task to be performed at the plant.

In (Chandra, 2000), when the restriction of the plant behaviour has to be done, two kinds of specifications can be distinguished: security specifications and progress specifications. The first ones are considered to prevent that no plant section takes undesirable actions and the second ones are concerned to reach the finalization of the task. Security specifications can affect to a unique plant element or several elements.

In this work we propose to perform the controller design for each device taking into account the service that the component provides with no consideration on the task that has to be done. The key element will be the development and use of a software framework where it will be necessary to model and implement a set of specifications.

The rest of the paper is organized as follows: In section 2 we review the Supervisory Control Theory. Next, the main relevant topics of the proposed architecture and the resulting framework will be presented. In section 4, in order to prove the validity of our proposal we will present a case of study where a real industrial element will be considered. Finally the main conclusions are presented.

2 MODELING DISCRETE EVENT SYSTEMS

The SCT of Discrete Event Systems is based on the use of automata and formal language models. Under these models the main interest is on the order in which the different events occur. In this way, a plant (Phoha, 2004) is assumed to be the generator of these events. The behaviour of the plant model is described by event trajectories over the finite event alphabet $\Sigma$. These event trajectories can be thought of as strings over $\Sigma$, so $L \subseteq \Sigma^*$ represents the set of those event strings that describe the behaviour of the plant.

The SCT restricts the behaviour of a plant $G$ by disabling temporarily certain events that can be created by $\Sigma$, so the goal is that the plant cannot create undesired or illegal event chains in $L(G)$. In the following, a few basic definitions will be posed.

The plant $G$ will modelled by the deterministic finite state automaton (DFSA)

$$G = (X, \Sigma_G, f_G, \Gamma_G, x_0, X_m)$$

where,

- $X$ is the set of states;
- $\Sigma_G$ is the finite set of events over $G$;
- $f_G : X \times \Sigma_G \rightarrow X$ is the state transition function;
- $\Gamma_G : X \rightarrow 2^{X_m}$ is the active event function;
- $x_0 \in X$ is the initial state;
- $X_m \subseteq X$ is the set of marked states, which represent the completion of a certain task or a set of tasks.

$\Sigma_G^*$ is used to denote the set of all finite length strings over $\Sigma_G$ including the empty string $\epsilon$. $f_G$ can be extended from $X \times \Sigma_G$ to the $X \times \Sigma_G^*$ domain by means of recursion:

$$f_G(x, e) = x,$$

$$f_G(x, se) = f_G(f_G(x, s), e) \quad \text{for} \quad x \in X, \quad e \in \Sigma_G \quad \text{and} \quad s \in \Sigma_G^*.$$

The SCT distinguish between the controllable set ($\Sigma_c$), which can be disabled, and the non-controllable set ($\Sigma_n$) which cannot be disabled. The following relations are fulfilled: $\Sigma = \Sigma_c \cup \Sigma_n$ and $\Sigma_c \cap \Sigma_n = \emptyset$.

In order to restrict the behaviour of the plant $G$, one or more specifications can be defined, that are usually modelled by a DFSA

$$H = (Y, \Sigma_H, f_H, \Gamma_H, y_0, Y_m)$$

A restriction limits the behaviour of the plant by means of one of the two composition operations defined by the SCT. Informally, it can be said that a composition operation allows two DFSA to run synchronously, which means that certain events will only be created if both DFSA are capable of doing so. In this work we will consider (synchronous) parallel composition of $G$ and $H$, defined as:

$$G \parallel H = \text{Act}(X \times Y, \Sigma_G \cup \Sigma_H, f_{G\parallel H}, \Gamma_{G\parallel H}, (x_0, y_0), X_m \times Y_m)$$

where
3 OVERVIEW OF DEVELOPED FRAMEWORK

Traditionally, when changes in the production process occur, the PLC that control the process need to be reprogrammed as well. In order to get highly flexible manufacturing systems, we propose a framework with a core element: a service repository. In this proposal, the services can be invoked dynamically based on the production needs. Therefore, if the production process is altered, then the sequence of invocation of the services will be modified. In this way, the main advantage is that the software running on the PLC does not need any modification at all, if all the required services have been taken into account.

In our framework, if an input event \( e_i \) is triggered by an external entity for a certain service, then \( e_i \in \Sigma_e \); if an internal event \( e_i \) is created by the plant, then \( e_i \in \Sigma_{ic} \). When certain services are invoked, the system can stop working properly. In order to avoid this, our proposal considers supervising the behaviour of the system using SCT. As stated in the previous section, once the plant \( G \) is modelled, its behaviour is restricted by means of one or more specifications \( H_k \) that can temporarily disable certain input events. Consequently, no undesired actions will ever take place.

Our framework (Figure 1) consists of the following components:

- **Interface with external entities.** Clients invoke the services provided by the system through this interface, and the corresponding input event will be triggered.

- **Supervision:** it is made up of a) the active event function \( \Gamma \) of \( A \), being \( A = \{ G \cup H_1 \cup \ldots \cup H_k \} \) and b) the synchronous composition \( G \parallel H_1 \parallel \ldots \parallel H_k \), where the input events can be disabled if needed.

- State transition functions, which evolve in accordance with 1) the input events that have not been disabled previously in the supervision and 2) the internal events of \( G \). In the plant, if an input event causes a transition \( f_G \), it will activate the execution of the corresponding service.

- Services provided that gather a specified functionality. Due to design criteria they are isolated from the supervision role so it is possible to perform modifications on the functionality regarding supervision consideration. This characteristic can be seen as the main advantage.

In the following, the main behaviour will be described. When an input event \( e_i \in \Sigma_e \) is triggered, it is processed by the supervisor. The composition operation, taking into account \( \Gamma_i \) for each DFSA, decides if \( e_i \) must be disabled or not. If \( e_i \) is enabled, it is then processed by the \( f_G \) of each DFSA, including \( G \). Therefore, if a transition is defined in \( f_G \) for \( e_i \) and the current state of \( G \), the corresponding service will be run. Otherwise, if \( e_i \) is disabled, the transition does not happen, and the requested service will never take place. As a result, we make sure that the system behaves according to its specifications at all times.

In what follows we discuss how to use our proposed framework in a typical element of a FSM cell: a conveyor belt. We have chosen it because it is a component found in every cell instance and because the working specifications are changed frequently, given that the conveyed elements must be moved to
different target positions using different velocities as well.

The plant is made up of a BOSCH conveyor belt with encoder, dynamo and position sensors. The control is carried out by the Allen Bradley SLC 500 series with the IMC 110 motion control module.

Depending on the operation mode of the IMC, the conveyor belt can be moved manually or automatically. Movements in manual mode are used for initialization and maintenance by an operator and are always executed at constant velocity. Movements in automatic mode are performed when some of the MML programs loaded in the IMC 110 memory is running, and it is possible to modify both the velocity and the acceleration of the motion. In order to work in automatic mode and know where the tray is located it is needed to perform a home operation. For security reasons, emergency stops must be considered when in automatic mode.

Let $H_1$ and $H_2$ be the two DFSA that model the restrictions R1 and R2, respectively.

According to the $H_2$ state transition diagram (Figure 3), the state $y_1$ represents the action “performing home operation”. If it is completed successfully ($e_2$ event), the state changes to $y_2$. Given that $e_1 \in \Gamma_{H_1}(y_1)$, then it is not possible to perform any type of movement to any position.

Table 1: $G$ plant states.

<table>
<thead>
<tr>
<th>State</th>
<th>Mode</th>
<th>Conveyor belt state</th>
<th>e-stop</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_0$</td>
<td>Manual</td>
<td>Stopped</td>
<td>Yes</td>
</tr>
<tr>
<td>$x_1$</td>
<td>Manual</td>
<td>Stopped</td>
<td>No</td>
</tr>
<tr>
<td>$x_2$</td>
<td>Manual</td>
<td>Moving</td>
<td>No</td>
</tr>
<tr>
<td>$x_3$</td>
<td>Automatic</td>
<td>Stopped</td>
<td>No</td>
</tr>
<tr>
<td>$x_4$</td>
<td>Automatic</td>
<td>Moving</td>
<td>Yes</td>
</tr>
<tr>
<td>$x_5$</td>
<td>Automatic</td>
<td>Moving</td>
<td>No</td>
</tr>
</tbody>
</table>

Taking into account the state transition diagram (Figure 4) for $H_2$, it is only possible to change to automatic mode ($e_4$ event) if a home operation has been completed successfully.

Diagrams $H_1$ and $H_2$ are structurally identical, and both restrictions over $G$ could have been modelled by only one specification $H$. However, modelling several modular specifications $H_i$, structurally simpler than $H$, is easier to verify and understand (Cassandrass, 2007).
In order to avoid these possible incidents, we use the Supervised Control Theory. The framework we propose allows modelling one or more specifications which guarantee that the system behaves properly at all times. This is achieved by temporarily disabling the input events which could put the system at risk. Our framework also differentiates between the specifications and the functionality provided. Thus, it is possible to modify both parts independently.

To prove the main features of our framework, some implementation issues must be solved, due to the synchronous nature of PLC.

ACKNOWLEDGEMENTS

The results presented at this paper have been obtained from the tasks of research that have been partially supported by the Spanish Ministry of Science and Innovation through DPI2007-62267 proposal and by the Castilla y León Council with SA030A07 project.

REFERENCES


