Supervisory synthesis for product-driven automation and its application to a flexible assembly cell

Jean-François Pétin, David Gouyon, Gérard Morel

Supervisory synthesis for product-driven automation and its application to a flexible assembly cell

Jean-François Pétin*, David Gouyon, Gérard Morel

CRAN - UMR 7039, Nancy University, CNRS
Faculté des Sciences, BP 239, 54506 Vandoeuvre lès Nancy Cedex, France

Abstract

A make-to-order business model requires manufacturing control to face customised product variability and traceability. Considering the product as central to the automation rationale provides each product occurrence with informational and decisional capabilities. This paper considers a formal framework for such product-driven automation within the context of the Supervisory Control Theory. Two interoperable classes of supervisors are synthesized. Product supervisors control routings through the manufacturing system according to customised product specifications and resource capabilities, and resource supervisors manage the execution of operations required by product supervisors. A case study, based on a flexible assembly cell, illustrates the approach and opens issues for industrial practice.

Keywords: Control system synthesis, Flexible automation, Supervisory control, Reconfiguration, Manufacturing Plant Control

1. Introduction

The introduction of Information Technology in manufacturing systems gives manufacturers an opportunity to promote make-to-order business models and mass customisation of products (Da Silveira *et al.* 2001).

Facing this wide range of customer needs requires manufacturing control systems to have the ability to adapt to variable demands in terms of product specifications or intrinsic system changes (Koren 2005).

To this end, a flexible control system may embed a maximum set of switchable control policies that have been designed to cover the various product manufacturing routings and the functional redundancies between the machines. Even if the IEC61499 standard provides a framework to handle control execution paths, this approach increases the control complexity (Ollero *et al.* 2002), in terms of the number of control states, which naturally arises as the number of machines and product variability increase.

To avoid this combinatorial explosion, challenges for modern automation are to provide methods and tools capable of designing and implementing dynamically reconfigurable control systems. It means that the control policy is not fully predefined but includes observation abilities to detect when a reconfiguration is needed, and decisional abilities based on synthesis (Qiu *et al.* 2003) or scheduling algorithms (Henry *et al.* 2004) to automatically compute a new control configuration (Brennan *et al.* 2002).

Considering that the control of elementary operations performed by the manufacturing machines seems to remain somehow constant through time, dynamic reconfiguration required by product variability is supposed to be limited to the "on the fly" generation of product routing control. Similar to the concept of the virtual production line (Qiu *et al.* 2003), this approach makes the product active in the scheduling and the execution of its manufacturing operations, and relies on a clear separation between machine control activities and product control.

This paper aims at proposing a formal modelling framework based on the Supervisory Control Theory (SCT) (Ramadge and Wonham 1987) for a modular synthesis of such product-driven reconfigurable control systems.

Section 2 introduces the dynamic reconfiguration of control systems and briefly presents the SCT theory for their synthesis. Section 3 defines a synthesis framework for product-driven automation which ensures consistency between resource and product supervisors, which respectively control the execution of manufacturing operations and product

* Corresponding author: Tel: +33-383-68-44-38, Fax : +33-383-68-44-43, CRAN, Faculté des Sciences, BP 239, 54506 Vandoeuvre les Nancy Cedex, France,
E-mail address: jean-francois.petin@cran.uhp-nancy.fr

routings. Section 4 details the different synthesis processes and the underlying models of resources, the manufacturing system and product specifications. This approach is illustrated in Section 5 using a case study based on the flexible assembly cell of the AIP-PRIMECA[1] Lorraine (AIPL) university workshop. Operational aspects of the supervisors synthesised for the case study are discussed in Section 6. Conclusions and open issues for future research are discussed in Section 7.

2. Problem definition

2.1. Dynamic reconfiguration of control

Mass customization relates to the ability to provide individually designed products and services to every customer through high process flexibility (Da Silveira *et al.* 2001).

From the process point of view, machine flexibility is the capability of machines to perform different operations. Taking into account functional redundancies between machines, routing flexibility is the ability of manufacturing a given set of part types using one or more routes through the machines (Tsubone and Horikawa, 1999).

Therefore, product customization, machine and routing flexibility impact the manufacturing control so that its online dynamic reconfiguration should make it possible to fit any process rescheduling or any customized product definition.

According to Brennan *et al.* (2002), implementing a dynamic reconfiguration of control requires a configuration loop involving informational, decisional and operational activities (Figure 1) for:

– monitoring, diagnosis or even prognosis (execution agent in Figure 1) in order to produce information about the control environment and to define when and where a reconfiguration is required,

– decision-making to define the most appropriate control policy (called Configuration Management Application in Figure 1),

– operational execution of the reconfigured control actions (Configuration agents in Figure 1).

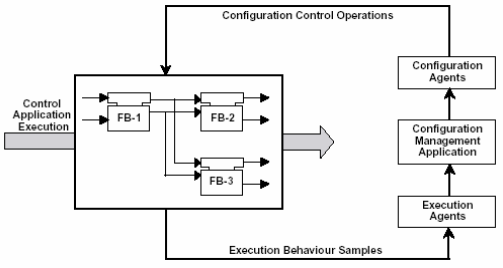


Figure 1. Reconfiguration framework (Brennan *et al.*, 2002).

[1] http://www.aipl.uhp-nancy.fr/lorraine/

Monitoring (Vogrig *et al.*, 1987, Zamai *et al.*, 1998), diagnosis (Toguyeni *et al.*, 2006) or even prognosis of manufacturing systems have been widely explored by Discrete Event System scientists and provide today material for identifying degradations or failure modes where control reconfiguration would be required (Berruet *et al.* 2000). Moreover, infotronics technology such as RFID tags embedded on the products enables individual identification of product occurrences that open a way towards the customization of control rules for each product occurrence (McFarlane *et al.* 2002).

Operational aspects of control reconfiguration are generally concerned with class C interoperability[2] issues (Staroswiecki and Bayart 1996, Iung *et al.* 2001) defined as the ability for automation components to be replaced by other ones offering similar services (class A maps the ability to exchange information while class B characterises the ability to cooperate for the execution of a given service). Indeed, operational reconfiguration has to manage switching from an obsolete control strategy to a new targeted configuration which can be obtained by tuning the component parameters or by replacing some of the control components.

These interoperability challenges lead to modular control systems promoting the use of standardised and reusable components on the shelves (Vyatkin *et al.* 2005). From a syntactic point of view, IEC 61499[3] provides a function block structure where the clear separation between execution control and algorithm description facilitates dynamic reconfiguration. From a semantic point of view, interoperability requires the standardisation of component interfaces and behaviour as well as the definition of a collaboration protocol (Endsley *et al.* 2006; Wright & Bourne, 1988; Newman *et al.*, 2000).

The informational (monitoring, diagnosis) and operational issues of control reconfiguration are beyond the scope of this paper which focuses on decisional activities required to define the most appropriate control policy. In the case of dynamically reconfigurable systems, two different methods of designing manufacturing control can be mentioned.

In the first method, the set of control policies required to master product variability are designed and pre-integrated in the control rules (Berruet *et al.* 2000; Toguyeni *et al.* 2006). It means that all possible manufacturing trajectories have been

[2] SEMATECH: *Device interoperability guideline for sensors, actuators and controllers*, (1995), Technology Transfer Standard 94102567A-STD, http://www.sematech.org

[3] International Electrotechnical Commission: *Function blocks, Part 1 – Architecture*, IEC PAS 61499-1, Geneva (2000)

modelled including the various product manufacturing routings and the functional redundancies between the machines. In this case, reconfiguration requires the tuning or the selection of already designed control rules. The main benefit is the flexibility of control policies but these approaches often suffer from an inevitable state explosion problem related to the number of machines and product variability (Muhl *et al.* 2003).

In the second method, the control policy, which is the most appropriate for taking into account product and manufacturing system features, is "on the fly" and automatically generated (Qiu *et al.* 2003). This approach avoids the state explosion problem encountered in the previous approaches but presents some limits when mixing various product lots on the same manufacturing system. Moreover, it requires using synthesis techniques (Lauzon *et al.* 1997, Lennartson *et al.* 1998) which enable automatic and on the fly generation of control rules.

2.2. Control synthesis

Among candidate approaches for control synthesis, such as techniques using Petri Nets (Achour *et al.* 2004, Basile *et al.* 2006) or synchronous languages (Marchand *et al.* 2000-1), SCT (Ramadge & Wonham 1987) has been proved to be an efficient and computer-aided framework. This theory provides a formal framework for DES analysis and synthesis based on two main concepts: the process to be automated (called a *generator* or a *plant*) and the supervisory controller (called a *supervisor*).

An automaton of the following form describes the process model:

$$G = (X_g, \Sigma_g, \alpha_g, x_{0,g}, X_{m,g}),$$

where X_g is a set of states x_g, Σ_g is a non-empty set of event labels called an alphabet, α_g is a transition function described by state transitions, $x_{0,g} \in X_g$ is the initial state, and $X_{m,g} \subseteq X_g$ is the set of *marked* (terminal) *states*. The model is assumed to generate spontaneously controllable and uncontrollable events. Controllable events can be disabled while uncontrollable events cannot be prevented and are permanently enabled.

The supervisor can affect the behaviour of the process model by enabling or disabling controllable events to maintain the process in a space of acceptable states for a given specification. An automaton of the following form describes the supervisor:

$$S = (X_s, \Sigma_s, \alpha_s, x_{0,s}, X_{m,s}),$$

where X_s is a set of states x_s, Σ_s is the alphabet used by G, α_s is a transition function, $x_{0,s}$ is the initial state, and $X_{m,s}$ is the set of *marked states*.

Synthesis algorithms (Wonham and Ramadge 1987, Kumar *et al.* 1991) automatically generate the optimal supervisor, which enables the maximal set of events that do not contradict the goal specifications. These specifications define the enabled sequences of events that belong to automaton G. Considering the generator model as the physically possible sequences, and the goal specifications as the legal sequences, the synthesized supervisor is expected to inhibit the process behaviour so that only desirable sequences are generated. Synthesis algorithms involve the synchronous composition of process and specification models and the elimination of strongly and weakly forbidden states.

In the field of SCT, the benefit of modularity in avoiding the state space explosion generated by the synthesis algorithms has been widely demonstrated (De Queiroz & Cury 2002, Vahidi *et al.* 2006, Endsley *et al.* 2006). Several extensions of the original framework have been proposed, such as:

- modular *supervisors*, with a distributed or hierarchical decomposition (Gohari & Wonham 1998),
- modular *generators* or *plants* based on a structured model of the process (Yoo & Lafortune 2002),
- mixed approaches, where both *generator* and *supervisor* are modular (Chafik & Niel 2000).

2.3. Dynamic reconfiguration using synthesis

The proposed approach aims at modelling a reconfigurable control system able to master the product variability induced by mass-customization.

It clearly appears that off-line designing of all control policies will at least be very difficult for complex customized products or even impossible if new product specifications occur. This reinforces the benefit of synthesis techniques for handling customized product control. However, considering that resource (or machine) elementary operation control, i.e. management of a manufacturing operation performed by a given resource, seems to remain somehow constant through time, "on the fly" synthesis may be limited to the routing of the product through the manufacturing system. However, to keep the flexibility of off-line design and to be able to adjust the product routing according to resource availability, objective is to synthesise a product supervisor which defines all the admissible manufacturing routes for a given customised product occurrence, and which initiates operations controlled by the resource supervisors.

This approach is part of the product-driven automation concept.

3. Synthesis framework for product-driven automation

3.1. The product-driven automation framework

The main hypotheses of product-driven automation consist in providing the product with information, decision and communication capabilities in order to make the product active in the scheduling and the execution of its manufacturing operations. In this sense, the IMS[4] community, especially in the area of Holonic Manufacturing Systems (Valckenaers 2001) promotes conceptual architectures, such as PROSA, which tend towards providing the manufactured product with intelligent behaviour. Based on these conceptual guidelines, this paper focuses on the design of a product-driven distributed control system, shown in Figure 2, which is based on the cooperation between:

– product supervisors which control the manufacturing routes according to a scheduled list of operations the product has to undergo; these supervisors are specific for each product occurrence in order to take into account their customization,

– resource supervisors which ensure correct execution of transport and transformation operations and provide the product controllers with accurate reports; control flexibility relies on tuning call parameters of the functional objects which coordinate and control the elementary operations.

The definition of these supervisors are founded, on the one hand, on the modelling of the manufacturing system capabilities which describe the system topology and the manufacturing operations performed by each resource, and, on the other hand, on the modelling of product requirements in terms of the operations it has to undergo.

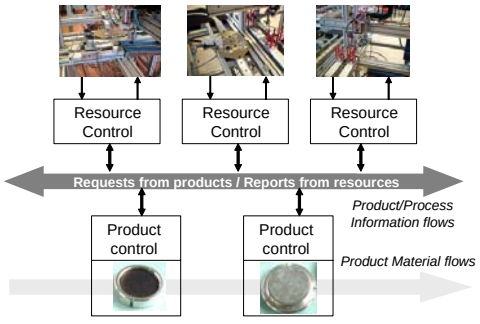


Figure 2. Product-driven control architecture

A unified modelling framework is required to facilitate joint design of product and resource controllers. According to the Discrete Event Systems (DES) control theory (Cassandras and Lafortune 1999), the design of control systems consists in defining the (unknown) control rules of the (known) dynamics of a physical system that satisfy some (known) behavioural goals while satisfying the predicate (Fusaoka *et al.* 1983):

$$Dynamics \wedge Unknown\ Control\ Rules \supset Goal \quad (1)$$

Note that the $\supset$ logic operator has the same meaning, according to Fusaoka's interpretation, as the implication operator ($\Rightarrow$). It means that satisfying behavioural properties of process and control models implies satisfying behavioural properties of the goal. In the context of product-driven automation, this predicate can be refined into two consistent interpretations.

The first interpretation relates to the product:

$$Manufacturing\ system\ capabilities \wedge Unknown\ product\ control\ rules \supset Product\ manufacturing\ plan, \quad (2)$$

where the routing control is defined from the product manufacturing plans given in terms of the orderly operations to be applied to the product, and from the manufacturing resource capabilities.

The second interpretation relates to the resources and is used to provide a modular control for the manufacturing resources according to:

$$Resource\ dynamics \wedge Unknown\ resource\ control\ rules \supset Resource\ capabilities, \quad (3)$$

where control rules are designed from resource capabilities given in terms of expected operation behaviour and resource dynamics in terms of physically acceptable states.

3.2. The product-driven synthesis framework

The framework presented in this paper applies SCT theory and its modular extensions in a structured modelling method to synthesize product and resource supervisors of a product-driven automation according to predicates (2) and (3). Correspondence of predicates (1), (2) and (3) with SCT notation is respectively the following: *Goal*, *Product manufacturing plan*, and *Resource capabilities* are models of SCT specification (*S*), *Dynamics*, *Manufacturing system capabilities*, and *Resource dynamics* are models of SCT plant or generator (*P*) while *Unknown control rules*, *Unknown product control rules*, and *Unknown resource control rules* are supervisors to be synthesized.

Predicate (2) and (3) lead to execute two separate synthesis processes in order to obtain product and resource supervisors (Figure 3) which cooperate. The product supervisor has to request a resource

[4] Intelligent Manufacturing Systems international initiative, http://www.ims.org/

supervisor to initiate an operation; if the resource is able to answer positively, the resource supervisor can execute this operation according to internal constraints of the resource, and emits a report when the operation is over. Product and resource supervisor communication is assumed to occur when products are physically connected to resources. Moreover, only one product can be processed at a time by a given resource, and initiating operations on the same resource by two different product supervisors is physically avoided.

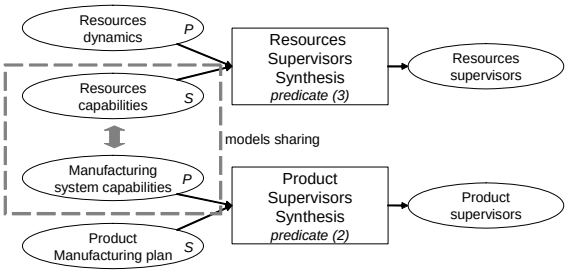


Figure 3. The product-driven synthesis framework

A necessary condition for interoperability requires standardisation of the supervisor interfaces based on request/report semantics and on a shared concept of generic manufacturing operations (transport and shape transformation) independent from their execution by a given resource. From a SCT point of view, this interface standardisation is ensured by sharing:

– a common modelling alphabet composed of request and report events related to a pre-fixed set of manufacturing operations that are defined independently from the location where they are executed,

– consistent models (dashed square of Figure 3) of the resource capabilities to be used in predicate (3) and of the manufacturing system capabilities to be used in predicate (2).

Note that this interface standardisation means that alphabet events are interpreted in terms of inputs and outputs of the product and resource control. Consequently, controllability of these events depends on the environmental context of a supervisor. For example, a request event for executing an operation is seen by a product supervisor as a controllable event while it is seen as uncontrollable by the resource supervisor. As a result, the concept of global controllability and uncontrollability of events is absent from the synthesis processes. This corresponds in fact to an input/output interpretation of SCT theory as proposed by Balemi *et al.* (1993).

4. Supervisor syntheses for product-driven automation

4.1. Product supervisor synthesis

In order to synthesize product supervisors, i.e. the unknown control rules of predicate (2), models of product manufacturing plans (specifications) and of manufacturing system capabilities are needed.

4.1.1. Product specifications

The product designer elaborates product manufacturing plans which will generate the expected features of the products. From a control theory point of view, they can be represented in terms of an orderly set of manufacturing operations which the product has to undergo. This information can be represented using an automaton (Figure 4), which represents the logical sequence between the morphological and/or spatial states of the product. Transition between two states is triggered when a report about product states occurs (*RP OPk*, where *k* is the number of *OP* operation performed on the product).

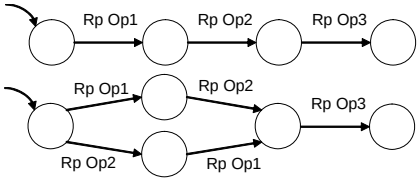


Figure 4. Product specifications for control issues

This model is a specification of the orderly states which characterise the product all along the manufacturing process (as given by reports) and does not control the requests to be sent to the resources. This logical sequence may present some flexible trajectories in case of non-orderly product transformations (second automaton in Figure 4).

4.1.2. Manufacturing system modelling

To model the manufacturing system capabilities, resource generic models and a composition operator based on the cell topology are proposed.

Models of resource capabilities

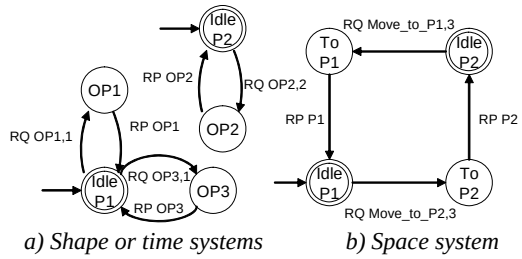
From a control point of view, the description of the resource capabilities is limited to the enumeration of the several operations a resource is able to perform. The alphabet of these models is composed of operation requests (*RQ OPk,i*) and reports (*RP OPk*) where *k* is the number of operations and *i* the number of resources. The language defined by these models details the admissible sequences of requests and reports.

According to Vogrig *et al.* (1987), Wright & Bourne (1988) and Lauzon *et al.* (1997), a manufacturing device can be described by three generic states: *working*, *idle*, and *down*. This generic model is extended by splitting the *working* state into several states which correspond to the operations the resource is able to perform. Note that the dysfunction states (*down*) of the machine are not considered to facilitate the presentation of the approach, since these states would only have as principal consequence the size of the models.

To systematise the modelling, generic models of resources are gathered using a systemic classification in terms of shape, time, or space, respectively associated to morphological transformation, storage, and transport systems.

The first two kinds of systems are modelled using an n + 1 states automaton (Figure 5a), where n represents the number of performed operations. The resource state moves from *idle* to *OPk* (k ∈ {0, n}) when the operation *OPk* is requested by the product. A move back to the *idle* state occurs when the associated report is emitted. For synthesis purposes, the *idle* state is noted as a marked state. This means that *idle* is considered as the legal end of a sequence and corresponds to a steady state.

Transport systems have been modelled in a different way because their states represent the different reachable locations, and are all considered as steady states. Consequently, the previous model has been modified to introduce several marked idle states associated to locations of the manufacturing resources (Figure 5b). Representation with a single idle state would have been adopted if the transport system had a reference position from which every move was initiated. Note that these resource models are only used to compose a system capability model from which the possible routings for one product occurrence are generated. Consequently, multiple product management is not considered for these models.



a) Shape or time systems *b) Space system*

Figure 5. Examples of resource generic models

A model of manufacturing system capabilities

The transformation and storage (shape and time) resources have to cooperate with transport (space) resources in order to ensure transformation and routings of the product. Interactions between these resources are represented by a model of manufacturing system capabilities which takes into account cell topology. Classical synchronous composition, which leads to a Cartesian product, is too permissive to obtain this model because it does not identify the localisation of the transformation resources (places that can be reached by the transport system, i.e. states of the space model).

Therefore, an operator which fuses transformation and transport automata is introduced:

– states of input automata are merged according to state equivalence classes which associate operation capabilities with their spatial localisations,

– the two transition functions – associated to the two input automata – are joint and applied on their respective state classes.

The fusion operator is defined on two automata G1 and G2:

$$G1 = (X1, \Sigma 1, \alpha 1, x1_0, X1_m) \text{ and}$$
$$G2 = (X2, \Sigma 2, \alpha 2, x2_0, X2_m),$$

with $\Sigma 1 \cap \Sigma 2 = \varepsilon$ and $X1 \cap X2 = \varnothing$.

To start, the initial automata are enriched by defining the function *sit:* $X1 \cup X2 \rightarrow \Re^3 \times F$, which associates a state $x \in X1 \cup X2$ with spatial coordinates *(x, y, z)* of the product belonging to $\Re^3$ and with in-progress operations belonging to the set $F = \{op_1, op_2, \ldots, op_n, 0\}$ of available operations.

The *sit* function allows the definition of an equivalence relationship in such a way that $x1_i$ and $x2_j$ will be considered as equivalent if $\exists (x1_i, x2_j) \in X1 \times X2 / sit(x1_i) = sit(x2_j)$.

Equivalence classes are noted with a dot above equivalence class name, i.e. the equivalence class for *x* is noted $\dot{x}$.

Then all $x2_j \in X2$ (resp. $x1_i \in X1$) which satisfy the equivalence relationship for each $x1_i \in X1$ (resp. $x2_j \in X2$) are sought. This defines state equivalence classes noted $\dot{x}1_i$ (resp. $\dot{x}2_j$) that merge $x1_i$ and $x2_j$ such as:

$$\dot{x}1_i = \begin{cases} \{x1_i, x2_j\} \text{ if } \exists x2_j \in X2 / sit(x1_i) = sit(x2_j) \\ x1_i \text{ else} \end{cases}$$

$$\dot{x}2_j = \begin{cases} \{x1_i, x2_j\} \text{ if } \exists x1_i \in X1 / sit(x1_i) = sit(x2_j) \\ x2_j \text{ else} \end{cases}$$

The resulting automaton $G = (X, \Sigma, \alpha, \dot{x}_0, X_m)$ is then defined by:

- $X = \{\dot{x}1_i\} \cup \{\dot{x}2_j\}$
- $\Sigma = \Sigma 1 \cup \Sigma 2$ with $\Sigma 1 \cap \Sigma 2 = \varnothing$;
- for $\dot{x} \in X$, $\sigma \in \Sigma$, $\alpha : X \times \Sigma \rightarrow X$

$$\alpha(\dot{x}, \sigma) = \begin{cases} \alpha(\dot{x}1_i, \sigma_k) = \dot{x}1_p \text{ for } \sigma_k \in \Sigma 1 \\ \quad \text{such as } \alpha 1(x1_i, \sigma_k) = x1_p \\ \alpha(\dot{x}2_j, \sigma_h) = \dot{x}2_q \text{ for } \sigma_h \in \Sigma 2 \\ \quad \text{such as } \alpha 2(x2_j, \sigma_h) = x2_q \end{cases}$$

- $\dot{x}_0 = (\dot{x}1_0)$
- $X_m = \{\dot{x}1_{m,i} / x1_{m,i} \in X1_m\} \cup \{\dot{x}2_{m,j} / x2_{m,j} \in X2_m\}$

The fusion operator is iteratively applied to fuse all transformation and transport models and to obtain a manufacturing system capabilities model. In the first iteration, G1 corresponds to the transport model while G2 corresponds to a transformation model. Then, for all following iterations, G1 is the previous fusion result, and G2 is the resource model to be merged.

Figure 6 shows the model of a manufacturing system which results from applying the fusion operator on two transformation resources (R1 is able to perform Operations 1 and 3 while R2 is limited to Operation 2) and one transport resource between them. Models of these resources conform to those in Figure 5.

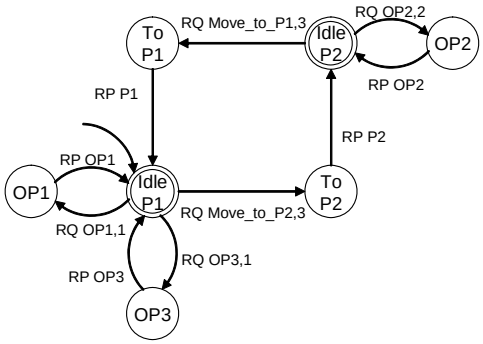


Figure 6. Result of fusion operation

4.1.3. Synthesis of product supervisors

Subsequently, classical synthesis algorithms can be applied to generate a supervisor that controls the alternate routes of a given product within the cell. The result is the most permissive supervisor which represents an exhaustive set of manufacturing trajectories which are acceptable to satisfy the product occurrence specifications. Optimality in terms of control performance or supervisor size is not sought during this phase which aims at defining all acceptable routings even if some are obviously of no interest.

This automatic synthesis of product routings is illustrated using the case study in the next section.

4.2. Resource supervisor synthesis

The synthesis of resource supervisors combines the object-oriented automation rationales and the modular synthesis techniques. Modularity criteria are not only driven by state-space explosion issues (De Queiroz & Cury 2002), but must be justified by the structure of the physical process itself (Gouyon *et al.* 2004).

To this end, structured modelling is proposed to start with the elementary actions which can be executed by a resource using actuators. These actions are progressively coordinated in a bottom-up manner to perform actions that are more complex. Applying this structured modelling to the synthesis process gives rise to a modular and iterative synthesis method in which modular models of specification and plant are used to synthesise a hierarchy of coordinated supervisors.

The resource *specification* model is split into several sub-models associated to each module function; it mainly relates to elementary action specifications or coordination specifications.

The generator or *plant* model is also split into several sub-models. The sub-models of the lower layer represent the physically admissible behaviour of the actuators and are defined by the designer. The g*enerator* sub-model of a layer *n* refers to the behavioural description of the layer *n-1* supervisors considered as the *plant* for a layer *n* coordination supervisor. This *generator* or *plant* model for the layer *n* supervisor is automatically given by (Figure 7):

– the projection of the layer *n-1* supervisors to keep only the observable events from the given layer *n*,

– enriching the alphabet of each projection by self-loops with the union of the alphabets involved in the other projections; the aim is to have a common alphabet for all projections,

– the controllability modification of each projected model, i.e. controllable events of layer *n* become uncontrollable and uncontrollable events become controllable,

– and finally, the synchronous product of the above projections (they do not share events).

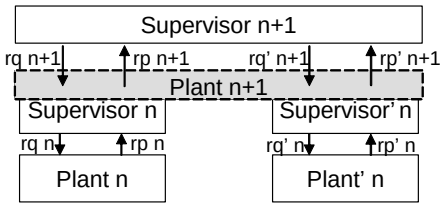


Figure 7. Iterative definition of plant models

As already shown for product and resource supervisors, controllability of events is not global for all resource supervisors but is variable according to the layer where the synthesis is applied; a controllable event (i.e., seen as an *output*) for a layer *n* supervisor will be uncontrollable (i.e., seen as an *input*) for its layer *n+1* coordination supervisor (Figure 7).

This iterative way of reasoning leads to a modular control synthesis that can be summarized by Figure 8 which shows the various stages of synthesis, projection, and composition of supervisory controllers.

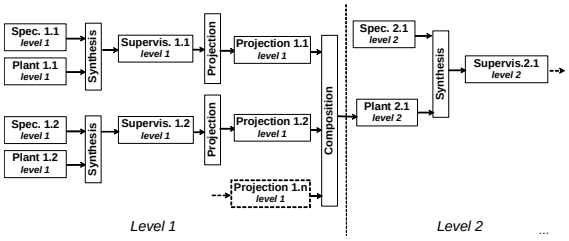


Figure 8. Iterative synthesis method

4.3. Discussion

Using coordinated supervisors cannot really be proved to be deadlock free. However, in order to limit the risk of deadlocks, the alphabets of supervisors of a given layer are disjoints. Consequently, two supervisors which belong to the same layer can be proved, according to Leduc (2002), to be deadlock free (no events are exchanged or shared between them). However, the approach does not ensure that a supervisor at a layer *n+1* is never absorbed by a strongly connected component where some events are disabled and consequently may lead to blocking in some supervisors at layer n. Open issues in the framework of modular specifications should be addressed to avoid this type of deadlock, such as the prefix-closure of the alphabets (Jiang & Kumar 2000). Hierarchical coordination is then applied until the highest layer is reached, where operations as seen by the product are processed.

Sharing resources by several products is assumed to be physically non-conflicting in the sense that only one product can initiate an operation on a given resource. The instantiation mechanism applied to the operation requests clarifies this situation on the models by ensuring that a request from a given product supervisor can only be processed by a unique resource. When a product supervisor initiates on a given resource (i) an operation (OPk) associated with the event *RQ_OPk,i*, the resource moves from the *idle* state to a *working* state and does not process any request until it returns to the *idle* state.

5. Application to the case study

5.1. Presentation of the case study

The concept of product-driven automation is illustrated in this paper using a *Flexible Assembly Cell* case study. This cell involves six workstations which are interconnected via a conveyor: one station for pallet loading, four similar assembly stations, and one station for pallet unloading (Figure 9). Six different product families can be assembled (Figure 10). Each workstation is able to perform from 1 to 4 assembly operations and involves a vacuum generator and three air cylinders to handle parts and products. Pallets are equipped with RFID tags which correspond to the informational part of product controllers. A restriction is made so that each product will only go on one pallet during its assembly. Workstations are equipped with a Programmable Logic Controller (PLC), which implements the resource controller, and with two short distance RFID tag reader/writers. One is located before the workstation by-pass and the second is located in the working area of the station.

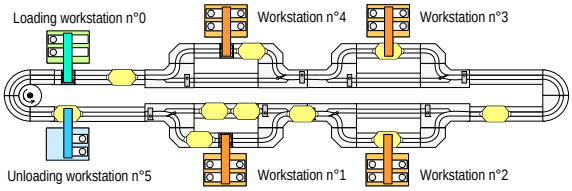


Figure 9. AIPL Flexible Assembly Cell

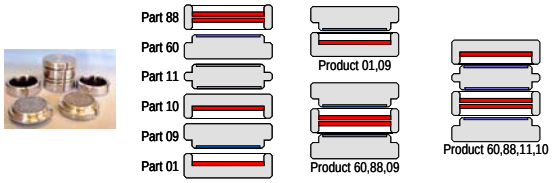


Figure 10. Product types

Applying the product-driven concept to the automation of this assembly cell leads to the following expected behaviour:

– When a product occurrence is planned to be manufactured, a model of the possible routings within the cell must be established, synthesized from product specification and the assembly workstation capabilities. This model represents a sequenced list of product states that takes into account the different admissible trajectories within the cell. Transitions between states correspond to reports which are received from the workstations or requests for transport or assembly operations. This model is embedded in RFID tags;

– Before each workstation, the RFID tag of the product is read. If a transition exiting from the active state corresponds to a transport request to a given workstation, the pallet is sent to the resource and a transport report is written on the RFID tag. If not, the pallet goes to the next workstation;

– When a product is present in the work area of a given workstation, the RFID tag is read. Transition exiting from the active state is interpreted by the resource controller as an operation request. The resource controller then executes the necessary actions to perform the requested operation. A report is emitted when the operation is correctly finished;

– Each time the RFID tag is written, a remote decision centre computes the new active state and the associated request of the routing model. Note that, if several solutions exist for a given operation (several machine are able to perform it), optimisation criteria such as workstation charge or routing time could be applied to select the requested resource.

The technical architecture of the product-driven control of the assembly cell is given in Figure 11 for the workstation 1.

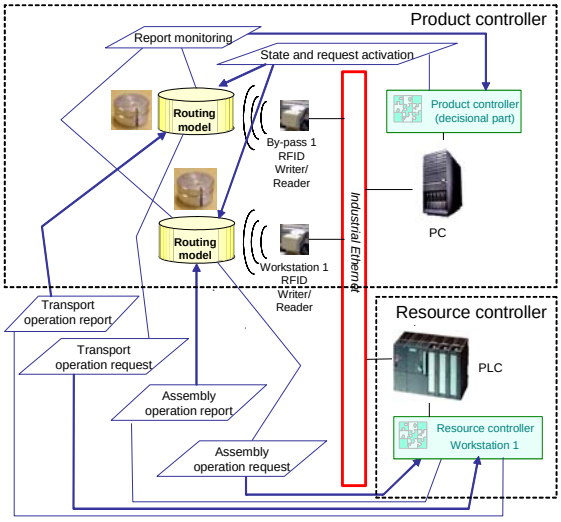


Figure 11. Product-driven technical architecture

This technical implementation aspect is not further detailed in the paper which focuses on modelling and synthesis issues for product-driven automation.

5.2. *Product supervisor synthesis*

The alphabet used in this application is given in Table 1; for the product, requests are considered as controllable events while reports are uncontrollable.

Label	Significance
RQ N,i	Request for assembling a part of type N by resource i
RP N	Report about assembling a part of type N
RQ pos i	Request for moving to position of resource i
RP pos i	Report about moving to position of resource i

Table 1. Alphabet for product supervisor synthesis

Let us consider a 01-09 type product; it is composed of an assembly of 01 and 09 parts. According to Section 3.1., product specification is given by Figure 12, where reports acknowledge the end of parts 01 and 09 assembly operations and the end of the unloading (RP 99).

Figure 12. Product manufacturing plan for 01-09 product (*specification*)

Resource models are designed by adapting the generic models of Section 3.2.1 to the following capabilities (Figure 13):

– workstations 0, 2, and 4 are able to assemble 88 and 01 parts, 09 parts, 11 and 09 parts respectively,

– workstation 5 is in charge of unloading the product out of the cell (operation noted 99),

– workstations 1 and 3 are unused,

– the transport system is a single conveyor that allows moves from any workstation to any other.

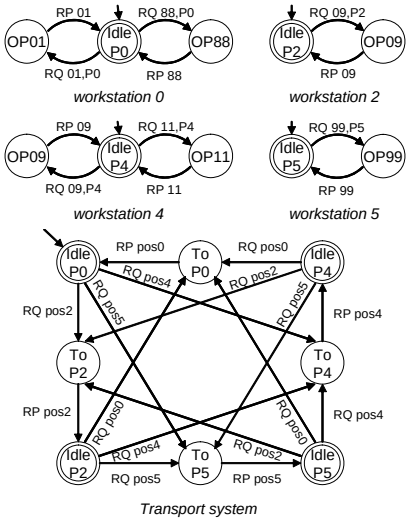


Figure 13. Workstations and transport models

The model of the assembly cell capabilities is designed by applying the fusion operator of the Section 4.1.2 to these resource models (Figure 14). It describes the set of plant admissible behaviours without control specification.

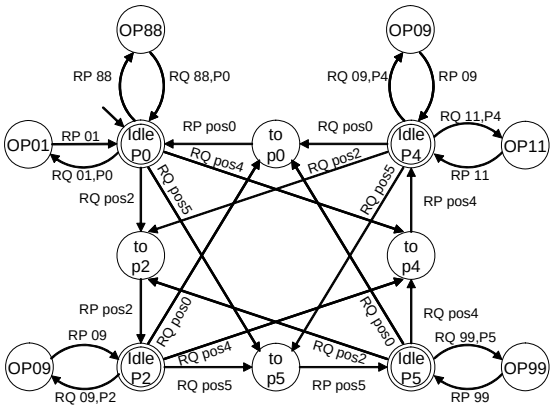


Figure 14. The model of manufacturing system capabilities (*plant*)

A 01-09 product supervisor can be generated using the TCT[5] tool for synthesis procedures, from

[5] Operations on automata (product, projection, synthesis) are made using the TCT software tool developed at the University of Toronto (http://odin.control.toronto.edu/DES/)

the automata of Figure 12 (as *Specification*) and Figure 14 (as *Plant*) (Figure 15).

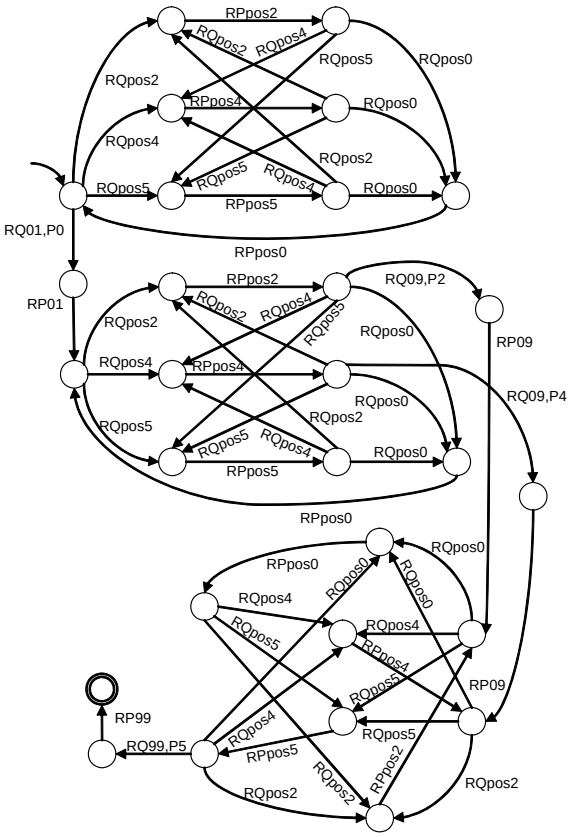


Figure 15. 01-09 product supervisor

It defines all acceptable routings of the 01-09 product within the assembly cell according to the different assembly operations that this product has to undergo (i.e. according to the product specification presented in Figure 12). In other words, it represents the maximal set of alternate routes for a given product within the cell which all lead to the expected manufactured product.

5.3. Resource supervisor synthesis

Application to the assembly cell of the iterative synthesis method shown in Figure 8 leads to three hierarchical layers for the resource supervisors: layer one concerns the actuators (air cylinders and a vaccum generator), layer two concerns the *pick and place* function (involving vertical air cylinders and a vaccum generator) and *move* function (involving two horizontal air cylinders, and the third layer supplies the part manipulation function.

Details are given below for each layer of the iterative synthesis method (complete for the lowest one and partial for the higher ones), knowing that paper length and supervisor size inhibit the complete representation of all the models used and supervisors synthesized in this case study.

5.3.1. Layer 1 supervisors

The workstations of the AIPL assembly cell are composed of air cylinders and a vacuum generator with their associated magnetic sensors to pick up, move, and finally assemble the parts.

The *generator* (or *plant*) generic model of double-acting air cylinders with their control valve is composed of two marked states in which the observable device behaviour is steady (a cylinder in a pushed or retracted position) and by two passing states, which represent the evolution from one steady state to another (Figure 16). A complete list of events is provided in Table 2. The vacuum generator is modelled using two states: *sucking on* and *off*.

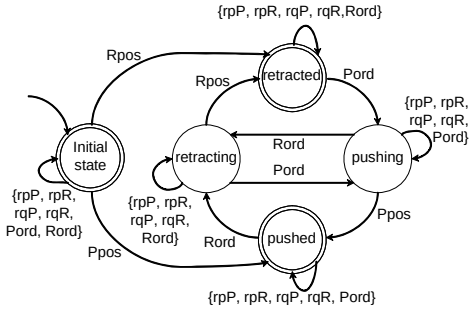


Figure 16. Air cylinder *generator* (or *plant*) model

Actuator *specifications* are then given in a modular way. For the generic model of air cylinders, two automata describe rules that filter or validate some pushing and retracting requests according to the current state of the device (first left automaton in Figure 17).

Label	Significance	Controllabilty
rqR	Retracting request	uncontrollable
rqP	Pushing request	uncontrollable
rpR	Retracted position report	controllable
rpP	Pushed position report	controllable
Rord	Retracting order	controllable
Pord	Pushing order	controllable
Rpos	Retracted position sensor	uncontrollable
Ppos	Pushed position sensor	uncontrollable

Table 2. Events for figures 16 to 18

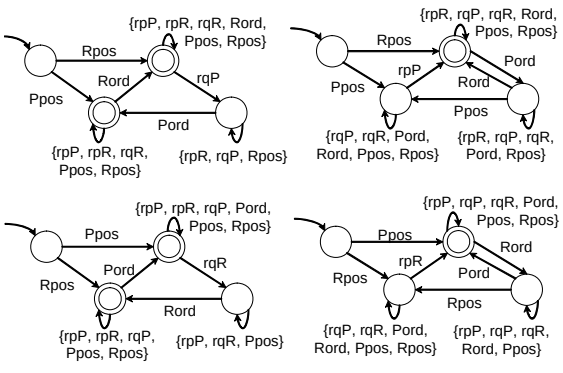


Figure 17. Air cylinder specifications

For example, after an initialisation phase considering uncontrollable events, a pushing order (*Pord*) is generated only after a pushing request (*rqP*), and there is always a retracting order (*Rord*) between two pushing orders (*Pord*). Two other automata describe the filtering of the observations. For example, a pushed position report (*rpR*) is generated only after a pushed position sensor event (*Ppos*) if a prior pushing order (*Pord*) is sent; a retracting order (Rord) invalidates this sequence (first right automaton in Figure 17).

The complete specification of the air-cylinder control module results from the synchronous product of these four automata (21 states, 68 transitions). The same approach is used for the specification of the vacuum system (8 states, 25 transitions). Note that these specifications are not natural at all and can be considered as a major difficulty of the synthesis process as further discussed in (Gouyon *et al.* 2004).

Subsequently, the TCT tool is used to compute the supremal controllable sublanguage that defines the most permissive supervisor (Figure 18), as is usually done in the SCT synthesis framework.

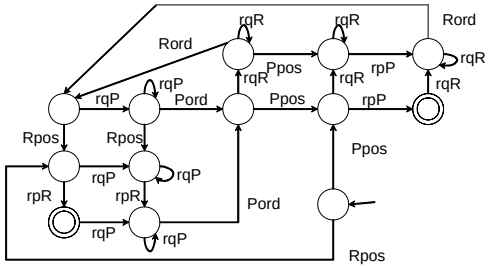


Figure 18. The air cylinder supervisor

The most permissive supervisor is determined for each actuator (air cylinders and vacuum generator) of the workstation. Note that if similar actuators are involved, previous alphabets are prefixed to be disjointed.

5.3.2. Coordination supervisors

At layer 2, two main functions manage the manipulator *move* from a position to another one and the part *pick & place*. Both functions result from coordination of the actuators (two air-cylinders for moving and one air-cylinder and vacuum generator for picking and placing).

Generator models used for this synthesis phase are automatically generated from the actuator supervisors following the mechanism presented in section 4.2: projection of actuator supervisors and synchronous composition. Figure 19 shows the projection of the air-cylinder supervisor, which only preserves report and request events (orders sent to the valve and sensor data have been eliminated), used in the synchronous composition. Note that the generic air-cylinder supervisor is instantiated to each actuator by renaming the projected events (for example by adding an 'h' for the horizontal cylinder) to create a specific alphabet for each instance.

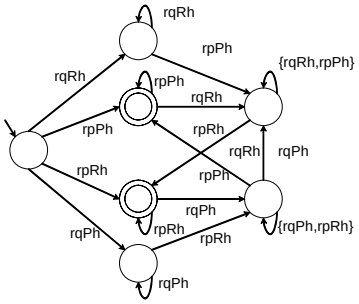


Figure 19. Projection of the air-cylinder supervisor

The *specification* model describes the actuator coordination rules. Figure 20 shows an example of the coordination specification for the *pick & place* function. The TCT tool is then used to generate the *pick & place* supervisor (20 states, 63 transitions) and the *move* supervisor (138 states, 558 transitions).

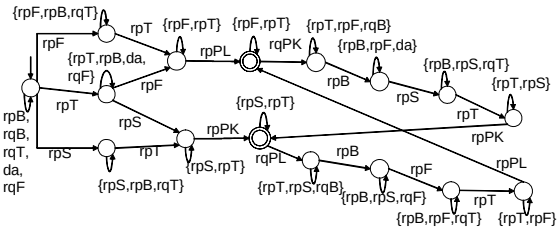


Figure 20. *Pick & place* specification model

Label	Significance	Controllabilty
rqS	Sucking request to vacuum cups	uncontrollable
rqF	Freeing request for vacuum cups	uncontrollable
rpS	Sucked Part report	controllable
rpF	Free Part report	controllable
da	Sucking request for vacuum cups	controllable
rqT	Rise request (go to top)	uncontrollable
rqB	Go down request (go to bottom)	uncontrollable
rpT	"At the Top" report	controllable
rpB	"At the Bottom" report	controllable
rqPK	Pick request	uncontrollable
rqPL	Place request	uncontrollable
rpPK	Part picked	controllable
rpPL	Part placed	controllable

Table 3. Events for Figure 20

Finally, the highest hierarchical layer (layer 3) is in charge of coordinating the *pick & place* and *move* functions to carry out a given assembly using part manipulation from one position to another. For example, assembling part 01 means moving to the place where 01 is stored, picking the part, moving back to the pallet and then placing the part on the semi-finished product.

The generator at this last layer is obtained by a projection and a synchronous composition of the *pick & place* and *move* supervisors.

The *specification* includes functional (sequence of moving and picking) and safety (no moves when picking) properties. According to predicates (2) and (3), the highest layer of resource specification makes the link between available operation requests and reports and the resource elementary actions which must be initiated to answer them. Projection of this specification by only preserving events that belong to the alphabet of the product supervisors describes the resource capabilities in term of operation requests and reports and is consistent with the *generator* model that is used in section 4.1 to synthesise product supervisors. In other words, operation *OPi* that appears in the generator model in Section 3 is further detailed in several elementary states in the resource specification model while preserving the same entering and exiting events for the refined sequence.

Figure 21 shows an example of such a specification for the Workstation 0 where operations *01* and *88* are supported. *epr* and *epo* denote picked and placed reports, *dpr* and *dpo* picking and placing requests, *ei* represents the manipulator position and *di* the request for reaching a manipulator position.

The consistency between product and resource supervisor synthesis can then be demonstrated by proving that the resource specification model in Figure 21 can be projected to obtain an associated resource model such as the ones presented in Figure 13:

– events kept by the projection are requests *rq01* and *rq88* as well as reports *rp01* and *rp88*,

– the two marked states are merged,

– the *move* and *pick & place* sequences are replaced by a single state, meaning that the resource is executing the required assembly operation.

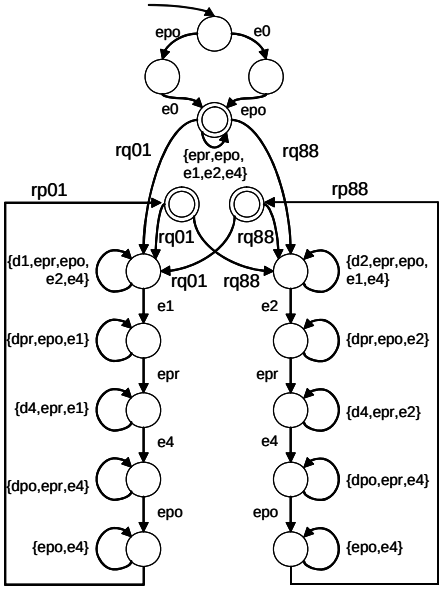


Figure 21. Highest-layer specification

Label	Significance	Controllabilty
rp01	Assembling 01 report	controllable
rp88	Assembling 88 report	controllable
rq01	Assembling 01 request	uncontrollable
rq88	Assembling 88 request	uncontrollable
epo	Product placing report	uncontrollable
epr	Product picking report	uncontrollable
dpo	Product placing request	controllable
dpr	Product picking request	controllable
e0	Position 0 reached	uncontrollable
e1	Position 1 reached	uncontrollable
e2	Position 2 reached	uncontrollable
e4	Position 4 reached	uncontrollable
d1	Position 1 request	controllable
d2	Position 2 request	controllable
d4	Position 4 request	controllable

Table 4. Events for Figure 21

The last layer of the resource supervisor can then be synthesized using the TCT tool and corresponds to an automaton (38 states, 43 transitions) that synchronises the actions required to carry out the assembly operations.

The final resulting control architecture is composed by three hierarchical layers: layer 1 involves 5 actuator supervisors (4 for the air cylinders and 1 for the vacuum generator), layer 2 involves one supervisor for part picking/placing and one supervisor for manipulator moves, and layer 3 supervisor coordinates the two layer 2 supervisors for part assembly.

6. Implementation issues

Implementation of product and resource controllers requires transforming the supervisors synthesized in Section 3 and 4 to introduce deterministic choices. Indeed, there is a clear interpretation gap between the roles a supervisor is assumed to play within the SCT modelling framework and the roles a controller has to play within current practices in real-time control systems (Zaytoon & Carre-Menetrier 2001).

Within the SCT framework, the process (generator) is assumed to generate events in a spontaneous manner. Therefore, the only way for the supervisor to affect the behaviour of the process is to enable or to disable the controllable events. Moreover, this supervisor is said to be a maximally permissive supervisor, meaning that it includes all legal process sequences for a given specification without providing choice criteria between two legal sequences of controllable events. However, a reactive control system is expected to force some events to occur, not only to enable and disable some of them. It is often based on a set of predetermined evolution rules which calculate the appropriate outputs (controllable events) to be applied to the process system according to its current state, given as inputs (uncontrollable events).

The gap between interpretations of the *supervisor* in SCT and the *controller* in the forcing events approaches (Marikar *et al*. 1998) can be bridged by an input-output interpretation of the SCT controllable and uncontrollable events (Balemi *et al.* 1993). Such an interpretation can be direct (Nourelfath and Niel 2004) if the supervisor sequences contain at least one controllable event between two uncontrollable events. This means that the supervisor reacts to a given input by emitting an output before a new input occurs. If this hypothesis is not verified, interpretation is indirect since it requires translating a supervisor into a controller according to a set of rules (Marikar *et al.* 1998, Fabian & Hellgren 1998).

6.1. Implementation of resource supervisors

Translation from resource supervisors to deterministic resource controllers provided with an input-output interpretation is based on a priority allocation mechanism. The objective is to enable two transitions t1 and t2 to exit from the same given state S1 if and only if the two events associated with t1 and t2 are uncontrollable. This strong hypothesis is justified by the fact that:

– two controllable transitions exiting from the same state mean that two actions (outputs) are acceptable for the supervisor, but one of them needs to be forced by the controller,

– controllable and uncontrollable transitions exiting together from the same state mean that the actions associated with the controllable event (output) may be triggered or not depending on the sampling period in which the uncontrollable event is seen.

In the first case, priority will be given to the event that belongs to an alphabet of a higher-layer supervisor (in Figure 22, *report* and *request* have higher priority than *action* and *observation*). In the second case, priority depends on event controllability, uncontrollable events having higher priority than controllable events (in Figure 22, *observation* has higher priority than *action,* and likewise for *request* and *report*). If two events have the same hierarchical layer and the same controllability, allocating priority refers to a design choice.

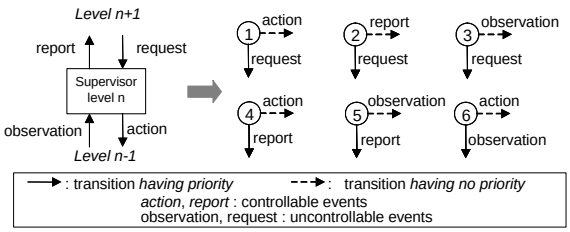


Figure 22. Priority allocation

SCT events are interpreted as inputs and outputs. Inputs are associated to uncontrollable events while outputs are associated to controllable events. More precisely, uncontrollable events are interpreted as rising edges of Boolean variables that trigger transitions. Controllable events of the supervisor are interpreted as rising edges that activate transitions toward states in which outputs are produced and maintained until these states are deactivated. These coding rules are based on algebraic equations that synchronously activate (Ai_t) and deactivate (Di_t) a state (Si_t) in accordance with:

$$Si_{t+1} = Ai_t \vee (Si_t \wedge \neg Di_t).$$

These algebraic equations can then be encoded into IEC 61131-3[6] PLC standard programming languages such as Ladder Diagram (LD) or Structured Text (ST). Each supervisor of the control hierarchy gives rise to an implementation module called Function Block (FB). Plugging these equations into FB requires using an execution algorithm that ensures equation scheduling. The most frequently used algorithm, *without stability search*, is based, initially, on the evaluation of the transitions that can be triggered, then on the calculation of the newly reached situation, and finally on the activation of the associated outputs. Figure 23 shows an example of a supervisor implementation in LD language.

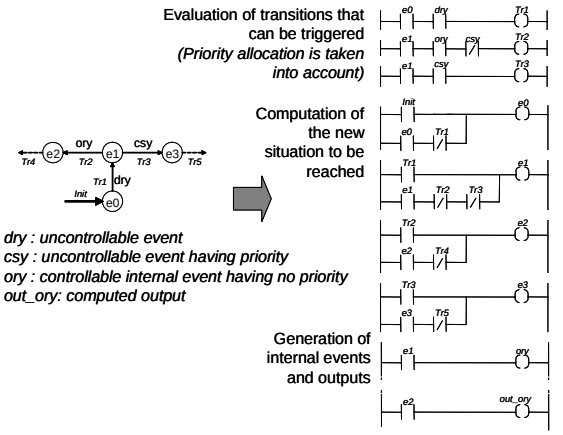


Figure 23. From supervisor to Ladder Diagram

To facilitate the implementation of the resource supervisors, a dedicated software tool (TCT2LD) has been developed:

– to read the supervisors description files given as input by the TCT tool,

– to provide an automatic (if possible) or interactive way of allocating priorities to transform the supervisors into controllers,

– to generate LD programs for downloading into PLCs.

[6] Int. Electrotechnical Commission, *IEC 6113-3 Programmable controllers, Part 3 programming languages*, 2000

Using this set of tools (TCT and TCT2LD), the resource controllers of the assembly cell have been successfully implemented and tested.

6.2. Implementation of product supervisors

Product supervisors represent all acceptable routings within the plant in such a way that the product specification is established. Bridging the gap between product supervisor and product controller requires removing indeterministic situations. This happens when there are two transitions exiting from the current product state. In this case, the product controller has to make a decision to select one of the acceptable manufacturing trajectories and then call for the corresponding resources. If the other cases, the product controller applies the only admissible sequence, as proposed by the supervisor.

Consequently, solving the determinism problem requires making a decision for choosing one of the admissible sequences according to external criteria such as resource performance, resource availability, transport time to resource or status of the resource. This problem is similar to path research within an automaton and solutions have been proposed using static or dynamic costs associated with each transition (Marchand *et al.* 2000-2), or optimisation algorithms, such as Dijkstra's algorithm (Dijkstra 1959). Static cost will help in defining the chosen trajectory before production while dynamic costs will help in defining in real time the accurate trajectories.

This technique has been applied by defining transition costs as the product of the time to move from one manufacturing state to another and the quantity of product inside the resource buffers (space between by-pass and workstation). When an indeterministic situation occurs, the optimisation algorithm is executed to choose the next state among all admissible states. Figure 24 shows an example of a manufacturing route that results from successive dynamic choices based on the product supervisor given in Figure 15.

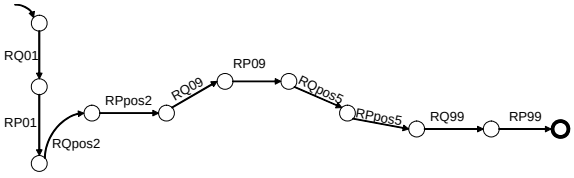


Figure 24. Selected manufacturing route

The product supervisor is then considered as the informational part of the product controllers. This part is coded into RFID tags thanks to arrays including the name and activation of product states as well as the name and the triggering state of transitions. The decisional part of the product controller is coded in Java and implemented in a remote computer.

7. Conclusion and open issues

This work is part of a research project on product-driven automation for business-to-manufacturing purposes (Morel *et al.* 2003). This paper focuses on the design and implementation of a product-driven control system. On-the-fly reconfiguration is the main property which is addressed to face variability of customized products and justifies the use of automatic synthesis techniques. The main objective is to provide an iterative modeling and synthesis method within the context of SCT to ensure the interoperability between product controllers which manage product routings within the manufacturing systems, and resource controllers which manage the execution of manufacturing operations.

However, even if the application of the approach to the case study has been successful, some limits of SCT modelling and synthesis must be mentioned.

The size of the generated supervisors is the major problem of synthesis algorithms, even if used for a pedagogical case study. Modularity is a classical way to handle this problem. For example, the modular and iterative synthesis of resource controllers of the assembly cell leads to a supervisor with a maximum of 138 states and 558 transitions. This can be considered as huge for human analysis of the result but can be easily implemented on PLCs with the use of tools such as TCT2LD. However, when the routing complexity increases, implementation of product supervisors could be a real difficulty. Splitting the product *specification* into sub-plans and isolating *plant* islands are open issues that could introduce modularity in the product supervisor synthesis.

The robustness of the synthesis algorithms is questionable (Gouyon *et al.* 2004); indeed, on the one hand, the resulting supervisor strongly depends on the way the *plant* and *specifications* have been modelled; on the other hand, human based modelling activities can give rise to a wide range of models that more or less cover the initial needs. Consequently, the modelling phase remains a major difficulty for the synthesis, especially when using finite state machines. This justifies further developments, such as synthesis based on the refinement of algebric equations (Roussel *et al.* 2004) or based on scheduling algorithms (Henry *et al.* 2004).

As addressed by Qiu *et al.* (2003), the above-mentioned problems could put into question the application of synthesis techniques for MES industrial applications but also opens onto complementary experiments.

8. References

Achour Z., Rezg N., Xie X. (2004), Supervisory control of marked graphs with partial observations. *International Journal of Production Research*, vol. 42, n° 14, pp. 2827-2838, ISSN 0020-7543

Balemi S., Hoffmann G.J., Gyugyi P., Wong-Toi H., Franklin G.F. (1993). Supervisory control of a rapid thermal multiprocessor. *IEEE Trans. on Automatic Control*, 38, 7.

Basile F., Carbone C., Chiacchio P. (2006). Simulation and analysis of discrete-event control systems based on Petri nets using PNetLab, Control Engineering Practice, In Press, doi:10.1016/j.conengprac.2006.07.006

Berruet P., Toguyeni A. K. A., Elkattabi S., Craye E. (2000) Toward an implementation of recovery procedures for flexible manufacturing systems supervision, *Computers in Industry,* Vol. 43, Issue 3, Pages 227-236

Brennan R. W., Zhang X., Xu Y., Norrie D. H. (2002). A Reconfigurable Concurrent Function Block Model and its implementation in Real-Time Java, *Journal of Integrated Computer-Aided Engineering*, Volume 9, pages 263-279.

Cassandras C.G., Lafortune S. (1999). *Introduction to discrete event systems*. Kluwer Academic. ISBN 0-7923-8609-4.

Chafik S., Niel E. (2000). Hierarchical-decentralized solutions of supervisory control. *Proceedings of the 3th Int. Symposium on Mathematical Modelling*, Vienna, Austria.

Da Silveira G., Borenstein D., Fogliatto F.S. (2001). Mass customization: literature review and research directions. *Int. Journal of Production Economics,* 72**,** pp 1-13.

De Queiroz M.H., Cury J.E.R. (2002). Synthesis and implementation of local modular supervisory control for a manufacturing cell. *Proceedings of the 6th Int. Workshop on Discrete Event Systems*, Zaragoza, Spain, pp. 377-382.

Dijkstra, E.W. (1959). A note on two problems in connexion with graph, *Numerische Matematik*, 1, pp. 269-271.

Endsley E. W., Almeida E. E., Tilbury D. M. (2006). Modular finite state machines: development and application to reconfigurable manufacturing cell controller generation. *Control Engineering Practice*, Volume 14, pp 1127-1142.

Fabian M., Hellgren A. (1998). PLC-based Implementation of Supervisory Control for Discrete Event Systems. *Proceedings of the 37th IEEE Conference on Decision and Control,* Tampa, Florida.

Fusaoka A., Seki H., Takahashi K. (1983). A description and reasoning of plant controllers in temporal logic. *Proceedings of the 8th Int. Joint Conference on Artificial Intelligence*, Karlsruhe, Germany, pp. 405-408.

Gohari P., Wonham, W.M. (1998). Hierarchical supervisory control of discrete-event systems. *Proc. of the 4th IFAC Workshop On Discrete Event Systems*, Cagliari, Italy.

Gouyon D., Pétin J.-F., Gouin A. (2004). Pragmatic approach for modular control synthesis and implementation. *International Journal of Production Research*, vol. 42, n° 14, pp. 2839-2858, ISSN 0020-7543

Henry S., Zamai E., Jacomino M. (2004). Real time reconfiguration of manufacturing systems, *Proceedings of the IEEE Conference on Systems, Man and Cybernetics*, The Hague, Netherlands.

Iung B., E. Neunreuther and G. Morel (2001). Engineering process of integrated-distributed shop floor architecture based on interoperable field components. *International Journal of Computer Integrated Manufacturing*, Vol. 14, No. 3, 246-262

Jiang S., Kumar R. (2000). Decentralized control of discrete event systems with specializations to local control and concurrent systems. *IEEE Transactions on Systems, Man and Cybernetics, part B,* 30, 5, pp. 653-660.

Koren Y., (2005). Reconfigurable manufacturing and beyond. *Proceedings of the CIRP 3rd International Conference on Reconfigurable Manufacturing*, Ann Arbor, MI, USA.

Kumar R., Garg V., Marcus S.L. (1991). On controllability and normality of discrete event dynamical systems. *Systems & Control Letters,* 17, pp. 157-168.

Lauzon S.C., Mills J.K., Benhabib B. (1997). An implementation methodology for the supervisory control of flexible manufacturing workcells. *SME, Journal of Manufacturing Systems***,** vol. 16, n°1.

Leduc R. J. (2002). Hierarchical Interface-based Supervisory Control. *Doctoral Thesis*, Dept. of Elec. & Comp. Engineering., University of Toronto

Lennartson B., Tittus M., Fabian M., Hellgren A. (1998). Specification structures for supervisory control. *Proceedings of the 4th IFAC Workshop On Discrete Event Systems,* Cagliari, Italy.

Marchand H., Bournai P., Le Borgne M., Le Guernic P. (2000-1), Synthesis of discrete-event controllers based on the Signal environment. *Discrete Event Dynamic Systems: Theory and Applications*, 10, pp. 325-346.

Marchand H., Boivineau O., Lafortune S. (2000-2). On the Synthesis of Optimal Schedulers in Discrete Event Control Problems with Multiple Goals. *SIAM Journal on Control and Optimization,* 39(2), pages 512-532.

Marikar M.T., Rotsein G.E., Macchietto S. (1998). An integrated environment for the design of procedural controllers. *Proceedings of the 9th IFAC/INCOM symposium,* Nancy, France

McFarlane D., Sarma S., Chirn J.L., Ashton K. (2002). The intelligent product in manufacturing control and management, *Proceedings of the 15th Triennal IFAC World Congress*, Barcelone, Spain.

Morel G., Panetto H., Zaremba M., Mayer F. (2003). Manufacturing enterprise control and management system engineering: rationales and open issues. *IFAC Annual reviews in Control.*

Muhl E., Charpentier P., Chaxel F. (2003). Optimization of physical flows in an automotive manufacturing plant: some experiment and issues. *Engineering Application of Artificial Intelligence*, Vol. 16, pp. 293-305

Newman W. S., Podgursji A., Quinn R.D., Merat F. L., Branicky M. S., Barendt N. A., Causey G. C., Haaser E. L., Kim Y., Swaminathan J., Velasco V.B. Jr., (2000), Deign lessons for building agile manufacturing systems. *IEEE Transactions on Robotics and Automation,* Vol. 16, n° 3, pp. 228-238

Nourelfath M., Niel E. (2004). Modular supervisory control of an experimental automated manufacturing system. *IFAC Control Engineering Practice.* Vol.12, pp 205-216.

Ollero A., Morel G., Bernus P., Nof S.Y., Sasiadek J., Boverie S., Erbe H., Goodall R. (2002). From MEMS to Enterprise systems. *IFAC Annual Reviews in Control*, vol. 26, issue 2, pp. 151-162

Qiu R., Wysk R., Xu Q. (2003). Extended structured adaptive supervisory control of shop-floor controls for an e-manufacturing system. *International Journal of Production Research*, Vol.41, N° 8, pp. 1605-1620.

Ramadge P.J., Wonham W.M. (1987). Supervisory control of a class of discrete event processes. *SIAM J. Control and Optimization*, Vol. 25, n° 1.

Roussel J.M., Faure J.M., Lesage J.J., Medina A. (2004). An algebraic approach for dependable logic control systems design. *International Journal of Production Research*, vol. 42, n° 14, ISSN 0020-7543

Staroswiecki M., Bayart M. (1996). Models and languages for the interoperability of smart instruments, *Automatica*, Volume 32, Issue 6, Pages 859-873

Toguyeni A. K. A., Craye E., Sekhri L. (2006), Study of the diagnosability of automated production systems based on functional graphs, *Mathematics and Computers in Simulation*, Vol. 70, Issues 5-6, Pages 377-393

Tsubone H., Horikawa M. (1999). A comparison between machine flexibility and routing flexibility, *The International Journal of Flexible Manufacturing Systems*, 11, Pages 83-101

Vahidi A., Fabian M., Lennartson B. (2006). Efficient supervisory synthesis of large systems. *Control Engineering Practice*, Volume 14, issue 10, pp 1157-1167.

Valckenaers P. (Editor) (2001), Special issue: Holonic Manufacturing Systems, *Computer In Industry*, 46 (3), pp. 233-331

Vogrig R., Baracos P., Lhoste P., Morel G., Salzemann B. (1987). Flexible manufacturing shop. *Manufacturing Systems*, Volume 16, n°3.

Vyatkin V. V., Chistensen J. H., Martinez Lastra J. L. (2005). OOONEIDA : an Open, Object-Oriented kNowledge Economy for Intelligent Distributed Automation, *IEEE Transactions on Industrial Informatics*, Vol. 1, n° 1

Wonham W.M., Ramadge P.J. (1987). On the supremal controllable sublanguage of a given language. *SIAM J. Control and Optimization*, vol.25, No.3.

Wright P.K., Bourne D.A. (1988). *Manufacturing intelligence*. Addison-Wesley, ISBN 0-201-13576-0.

Yoo T.-S., Lafortune S. (2002). A general architecture for decentralized supervisory control of discrete-event systems. *Discrete Event Dynamic Systems: Theory and Applications*, 12.

Zaytoon J., Carre-menetrier V. (2001). Synthesis of a correct control implementation for manufacturing systems. *Int. Journal of Production Research*, 39, pp. 329-345.

Zamaï E., Chaillet-Subias A., Combacau M. (1998). An architecture for control and monitoring of discrete events systems, *Computers in Industry*, Vol. 36, Issues 1-2, Pages 95-100