A product-driven reconfigurable control for shop floor systems

David Gouyon, Jean-François Pétin, Gérard Morel

A product-driven reconfigurable control for shop floor systems

David Gouyon
Jean-François Pétin
Gérard Morel
Nancy Research Center for Automatic Control, UMR 7039, Nancy Université, CNRS
Faculté des Sciences et Techniques – BP 239 – Vandoeuvre-les-Nancy Cedex
FRANCE

David.Gouyon@cran.uhp-nancy.fr, Jean-Francois.Petin@cran.uhp-nancy.fr, Gerard.Morel@cran.uhp-nancy.fr

Abstract: The intensive deployment of Information Technology in manufacturing systems gives manufacturers an opportunity to promote make-to-order business models and mass customization of products. Facing this wide range of customer needs requires manufacturing control systems to be adaptable to variable demands in terms of product specifications or intrinsic system changes. To this end, the concept of product-driven control considers the product as pivotal to the automation rationale. This approach consists in providing the product with information, decision and communication capabilities in order to make it active in the scheduling and the execution of its manufacturing operations. This paper presents a formal reconfiguration framework for the development of a product-driven shop floor control system and its integration within the context of industrial automation. This approach is illustrated using a case study based on the flexible assembly cell of the AIP-PRIMECA Lorraine university workshop.

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

1 Product-driven enterprise system control

Advances in the use of Information Technologies in manufacturing systems give manufacturers an opportunity to promote make-to-order business models and mass customization of products [7]. Facing this wide range of customized customer orders impacts the whole set of enterprise information and control systems [29], which integration capability has to be improved according to the Enterprise Integration Capability Model [13] (EICM, Figure 1), in a dynamically moving context.

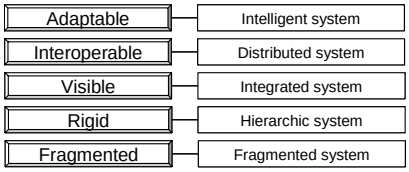

Figure 1. Enterprise Integration Capability Model [13]

Standards[1] [16] enable manufacturing enterprise-control system integration from the business level to the process level in order to meet industry-led Business-to-Manufacturing issues [26] (Figure 2a). In this context, Manufacturing Execution Systems ensure information flow synchronic gateway between enterprise and shop floor control systems and diachronic integration between execution activities (service flows). The main issue is then to ensure consistency of information and product flows.

A possible alternative, in order to reach the 'interoperable' level of EICM, is to put into question the hierarchical/integrated vision of the enterprise-wide control for a more interoperable or intelligent one by postulating the customized product as the 'controller' of the manufacturing enterprise resources [25,27] (Figure 2b). The product, seen as a good by manufacturing systems, and as information and service supplier by business systems, ensures consistency between physical and informational flows.

Another alternative, as promoted by the IMS community[2], leads to the development of new architectures based on the consideration of highly distributed, autonomous, adaptable and efficiently cooperating units integrated by a plug-and-operate approach, as done in multi-agent [24] and Holonic Manufacturing Systems [10] (Figure 2c).

[1] http://www.mesa.org/, http://www.isa.org/, http://www.mimosa.org/, http://www.opcfoundation.org/
[2] Intelligent Manufacturing Systems international initiative, http://www.ims.org/

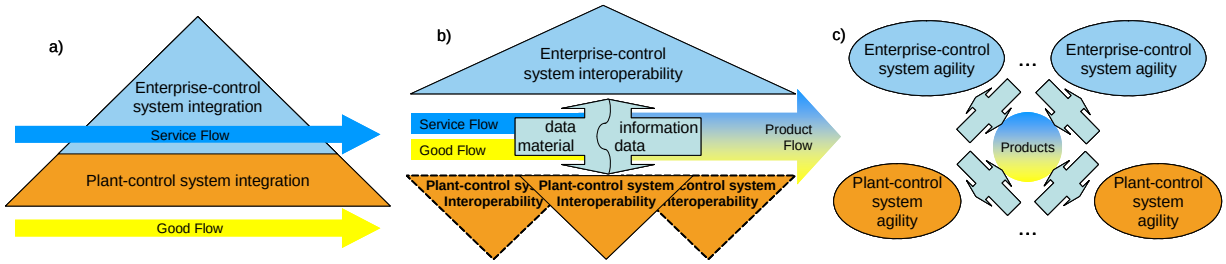

Figure 2. From integrated to agile manufacturing control

Emerging infotronic technologies embedded into product-driven control [25] bring more or less research results closer to actual deployment: RFID, wireless networking, modern PLC and industrial PC support of multi-agent systems…

This paper focuses on the impact of product-driven control on the shop floor, and proposes in this way a formal reconfiguration framework. The structure of this paper is the following: section 2 presents issues of shop floor product-driven control and a particular reconfiguration framework, section 3 and section 4 respectively detail the configuration management application and the control application of this framework.

2 Shop floor product-driven control issues

The main contribution of this paper is to explore the possibilities of dynamical reconfiguration by product-driven control.

2.1 Intelligent versus smart product

Considering an active role of the product leads to give it a form of technical intelligence [18], which corresponds, according to Wong *et al.* [46], to:

1. Possess a unique identity,
2. Be capable of communicating effectively with its environment,
3. Be able to retain or store data about itself,
4. Deploy a language to display its features, production requirements etc.,
5. Be capable of participating in or making decisions relevant to its destiny.

In function of these points, two levels are defined by Wong *et al.* [46]:

- Level 1 Product Intelligence allows a product to communicate its status (form, composition, location, key features), i.e. it is *information oriented*. Level 1 essentially covers points 1 to 3 of the intelligent product definition above.
- Level 2 Product Intelligence allows a product to assess and influence its function (e.g. self-distributing inventory and self-manufacturing inventory) in addition to communicating its status, i.e. it is *decision oriented*. Level 2 therefore covers points 1 through to 5 of the intelligent product definition above.

From an operational/logical point of view, things can be different because it seems to be difficult to implement directly into smart products all aspects of product intelligence. At this time, embedded devices have not enough processing power and the ability to communicate all the required information for the manufacturing. For these reasons, other cases can be envisaged if active entities reside in computers and are remotely linked to physical products and machines. Indeed, some multi-agent manufacturing systems are already implemented in real industrial environment [25], but there are some constraints, related for example to the reliability of RFID: successful read rate is not yet 100%, and for this reason, the system may not be fully observable.

In such an approach, the product is considered as central to the automation rationale, and is logically provided with information, decision and communication capabilities in order to make it active in the scheduling and the execution of its manufacturing operations (point 5 of Wong *et al.* [46]). The system is then said « product-driven ». Holonic Manufacturing Systems (HMS) constitute a repository to formalize this concept of product-driven control.

2.2 Holonic Manufacturing Systems

Koestler [19], based on Simon's work [37], introduced the concept of the *Holon*, which is an entity capable of functioning as a whole, while simultaneously acting as a part of a whole in a hierarchically ordered system. In other words, an Holonic system is a combination of an heterarchical system with centralised elements. Based on this concept, the IMS community, especially in the area of Holonic Manufacturing Systems [41,8] promotes conceptual architectures, which tend towards providing manufactured product with an intelligent behaviour. These HMS [2] are distributed systems which consider holons, which can be autonomous production units, cooperating to make products in a dynamically reconfigurable environment [25]. In the HMS reference architecture PROSA [42], types of holons are resource holons, order holons, staff holons and product holons. This last concept shows explicitly the active role of product.

2.3 Product-driven automation

Following conceptual guidelines of HMS, this approach focuses on the design of a product-driven distributed control system (Figure 3), which is based on the cooperation between:

- product controllers which control the manufacturing routes according to a scheduled list of operations the product has to undergo; these controllers are specific for each product occurrence in order to take into account their customization,
- resource controllers 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, or on downloading specific control policies embedded into products.

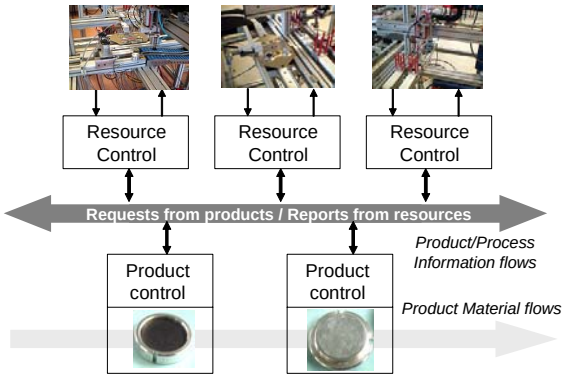

Figure 3. Product-driven control architecture

This cooperation consists in the exchange of requests of operations (noted *RQ*) emitted by product controllers to resource controllers, and reports of operations (noted *RP*) emitted by resource controllers to product controllers.

The definition of these controllers 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. A unified modelling framework is required to facilitate joint design of product and resource controllers. According to the Discrete Event Systems (DES) control theory [6], the design of control systems consists in defining the (unknown) control rules of the (known) dynamics of a physical system which satisfy some (known) behavioural goals while satisfying the predicate [11]:

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

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 [33]. In the context of a product-driven automation, this predicate can be refined into two consistent interpretations. The first interpretation, based on resources capabilities and product manufacturing 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 manufacturing resource capabilities.

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

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

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

2.4 Product-driven control reconfiguration framework

According to Brennan *et al.* [5] implementing a dynamic reconfiguration of control requires a configuration loop involving informational, decisional and operational activities (Figure 4) for:

- monitoring and diagnosis (Execution Agent) in order to produce information about the control environment and to define when and where a reconfiguration is required (e.g. performance or product changes). Monitoring [43,48] and diagnosis [40] 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 [4];
- definition of the most appropriate control policy (called Configuration Management Application),
- operational execution of the reconfigured control actions (Configuration Agents).

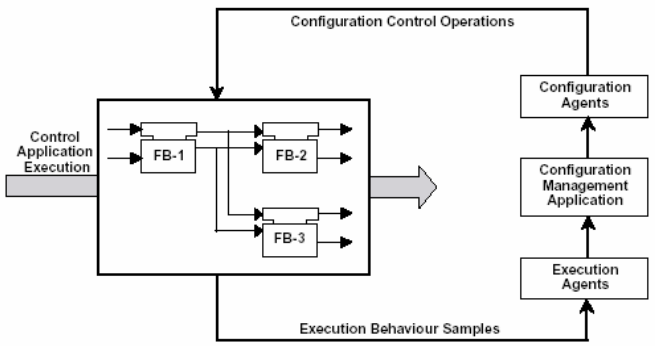

Figure 4. Brennan *et al.* reconfiguration framework [5].

Such a reconfigurable system requires [5]:

- the definition of a configuration manager defining when and which a new control configuration must be applied according to the environment changes (*Configuration Management Application* in Figure 4),
- the definition, within each control station, of a generic infrastructure monitoring and managing the execution of an evolving set of function blocks (*Execution Agent* and *Configuration Agent* in Figure 4).

Moreover, infotronics technology such as RFID tags embedded on the products enables individual identification of product occurrences which open a way towards the customization of control rules for each product occurrence [25].

2.4.1 Definition of a new configuration

Flexible system control often suffers from an inevitable state explosion problem, related to the number of machine and product variability. As underlined by Muhl *et al.* [28], if all possible manufacturing trajectories have been modelled (including the various product manufacturing routings and the functional redundancies between machines), 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. To solve this problem, major objectives of the *Configuration Management* are: to define when a new configuration is required, to develop a new configuration plan and, at least, to find the most appropriate system state from which the reconfiguration can be realised.

The definition of these functions blocks can be done from a FB library [44] or “on the fly” [35]. This last approach avoids state explosion problems, 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 [20] which enable automatic and on the fly generation of control rules.

Among candidate approaches for control synthesis, such as techniques using Petri Nets [1] or synchronous languages [22], Supervisory Control Theory (SCT) [36] 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*). In the field of SCT, the benefit of modularity in avoiding the state space explosion generated by the synthesis algorithms has been widely demonstrated [10]. Several extensions of the original framework have been proposed, such as modular *supervisors*, with a distributed or hierarchical decomposition [45], and/or modular *generators* based on a structured model of the process [47].

2.4.2 Operational execution of a new configuration

Operational aspects of control reconfiguration are generally concerned with class C interoperability[3] issues [17] 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. From an industrial point of view, this property of distributed applications has been addressed through a standardisation effort. Key objectives were to define a common set of control services (behaviour and communication interfaces) devoted to the various devices involved in a manufacturing system and to standardise the mechanisms for the distribution of these services [31].

In this way, modelling with functional block [21] has been promoted. IEC 61499 standard [14] provides a framework which represents a distributed *system* as interconnected *devices* which support one or more distributed *applications* executed on one or more *resources*. *Applications* are divided in one or more functional blocks, interacting to build functions. Functional blocks (FB) could be basic (behavioural description), or composite (composition of basic FBs) [8]. A basic Function Block is composed of a control management part and a control processing part. The first one manages the execution of algorithms to be executed in the processing part according to inputs events and the ECC (Execution Control Chart) described using finite state automaton. It also generates output events representing algorithms execution state which can be reused by others FBs. The second part contains control algorithms which can process data input to generate data outputs. These algorithms can be defined according to IEC 61131-3 standard languages [15]. For the validation of models using IEC 61499 standard, various case tool can be used: FBDK[4] or Corfu FBDK[5] [39].

This standard seems particularly adapted to master flexibility within distributed control system [30] thanks to the clear distinction between the management of the algorithms to be executed and the description of the algorithms themselves. It enables the tuning of the control rules within the blocks or even the selection of the accurate algorithm in order to face a more or less important variation in the control system environment. However, this approach assumes that the whole required function blocks are available within the existing control structure thanks to an efficient control design.

Dynamic control reconfiguration establishes a step forward with regard to this flexibility by promoting deeper changes in the control behaviour and architecture to be applied during program execution. In such a dynamically reconfigurable environment, the control program embedded in distributed stations (industrial PC or PLC) and products must be designed in such a way that any change in function blocks execution (FB addition, remove or modification) should be possible without a complete re-design of the program. One solution consists in implementing in the whole control stations a common integrating infrastructure. The aim of this infrastructure is to schedule the execution of a variable set of function blocks whose I/O interfaces are defined in a standardised library such as OOONEIDA [44].

2.4.3 Adaptation to product-driven control

The reconfigurable architecture presented in this paper (Figure 5) is an extension of the framework first introduced in Pétin *et al.* [32], based on Brennan *et al.* [5] framework which defines configuration agents. In order to avoid confusion, note that the word ‘agent’ used in this framework means behavioural independent modules. With regard to this last framework, originality of the present proposal relies on the introduction of product-driven control synthesis [34], and on the definition of the *Execution Agent* as the coordination of several elementary and local controllers for actuators and sensors.

[3] SEMATECH*: Device interoperability guideline for sensors, actuators and controllers*, (1995), Technology Transfer Standard 94102567A-STD, http://www.sematech.org
[4] http://www.holobloc.com/
[5] http://seg.ee.upatras.gr/corfu

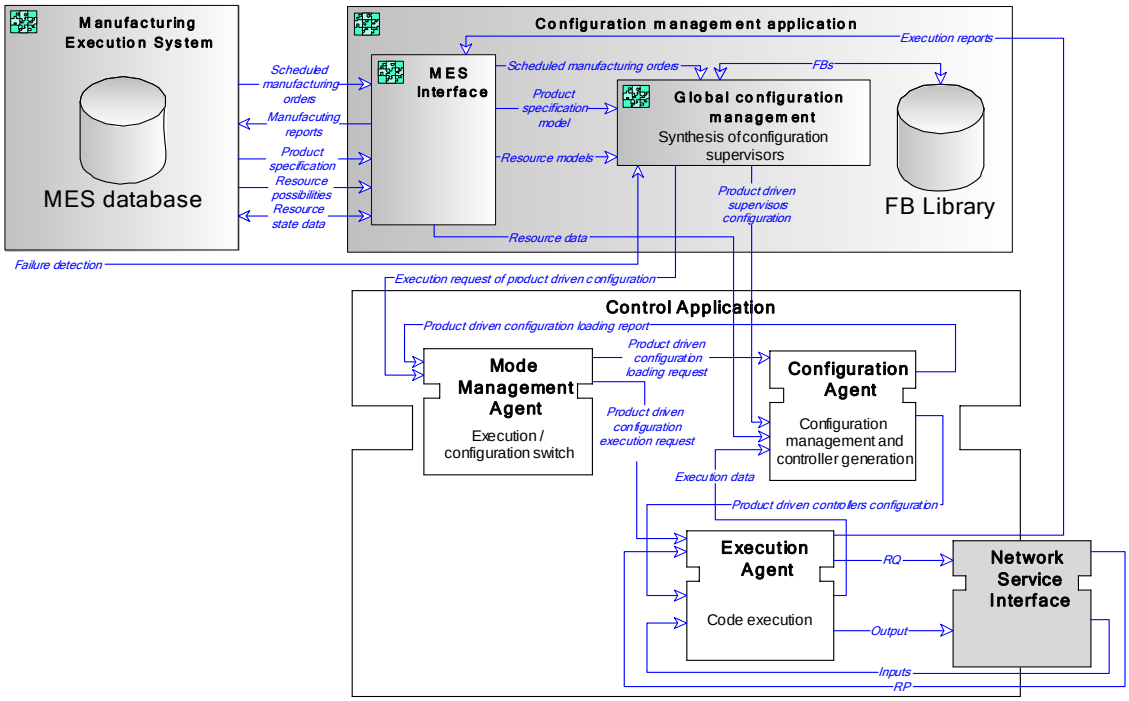

Figure 5. Function block model for product-driven dynamic reconfiguration

The function block model proposed for the dynamic reconfiguration of product-driven control system is based on a two level structure:

- the "Configuration Management Application" level where configuration and reconfiguration plans are developed according to predicates (2) and (3) with a global vision of the system; this level is assumed to be supported by a centralised resource of the control architecture.
- the "Control Application" level, which corresponds to the control infrastructure embedded in each resource stations of the control architecture (PLCs, industrial PCs, remote intelligent I/O, …); this level is composed of an *Execution Agent* which executes synthesised function blocks, a *Mode Management Agent* which ensures consistency control during reconfiguration operations and a *Configuration Agent*, which builds and transfers new configurations (new function blocks);

Reconfiguration is, in our case, initiated to master product variability but could also be considered in answer to a fault detection leading to reduce the process capability and availability. This scenario can be described as follows: when a product embedding a specific control program to be executed is presented to a workstation, a request for a new configuration is processed in the *Global management application*. The role of *Global Configuration Management Agent* is then, using the FB library or synthesis techniques, to create, delete or modify an optimised ordered list of functions blocks, event and data connections which specify the new configuration. This new configuration is sent to the *Configuration Agent* in charge of generating the control code that will be implemented within the *Execution Agent* in charge of code processing. In the same time, a request for reconfiguration is sent to the *Mode Management Agent* which manages the runtime modification (stops order to the *Execution Agent* so as to reach reconfiguration points, requests for downloading the new configuration to the *Configuration Agent*, and when done starts request to the *Execution Agent*).

2.5 Product-driven control 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 6). Six different product families can be assembled. Workstations 0 to 4 are able to perform from 1 to 4 assembly operations and involve a vacuum generator and three air cylinders to handle parts and products. As workstation 5 is able to unload product from pallets, its behaviour can be different for each pallet instance. Pallets are equipped with RFID tags. 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.

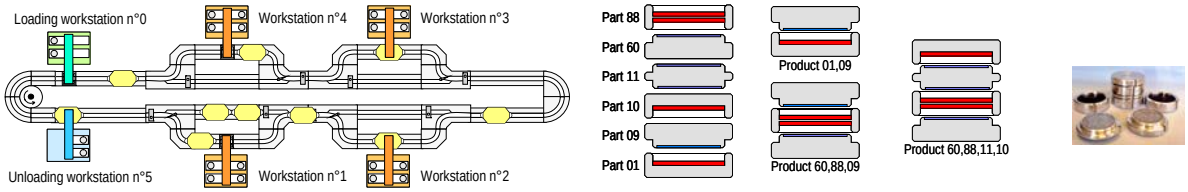

Figure 6. AIPL Flexible Assembly Cell and product types

The specific scenario which is used to illustrate the approach is the following:

- 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 .

3 Configuration Management Application

Assuming that a need for reconfiguration has been identified (new product or operation requested), the *Configuration Management Application* has to define a new specification of configuration for the *Execution Agent*. Its role is to define from an FB library and/or to synthesise from MES data the product-driven supervisor configuration:

- resource supervisors: which basic function blocks are required as Services Interface (corresponding to all layer 1 elementary action which must be controlled), and which are the possible behaviours of *Coordination Agents* which schedule sequences of elementary actions to build more complex functions (controllers of layers 2 to n).
- product supervisors: which product routing are possible regarding product specification and resource possibilities

In a similar way than Wong *et al.* [46], two different cases can be encountered to define the most appropriate product-driven control policy. In the first case, the set of operation control policies required to master product variability are designed and pre-integrated in resource control rules [4,40]. These control rules, which can remain somehow constant through time, are developed from Function Block databases (components on the shelves), or synthesized. Reconfiguration requires only “on the fly” synthesis of specific product routings and tuning or selection of already designed resource control rules. Similar to the concept of the virtual production line [35], this approach logically 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 [34]. Products embed control rules which are specific to their manufacturing.

In the second case, the development of a reconfiguration plan consists in synthesizing “on the fly” product routing, and defining the complete list of Function Blocks which have to be executed. These FB can be selected from an existing FB database, or developed by an “on the fly synthesis”. In this case, “on the fly synthesis” is extended to machine control, products embedding control rules which are specific to their manufacturing. In this second case, a specific architecture of control, as the one proposed by Brennan *et al.* [5], is needed, in order to permit the dynamical reconfiguration of resource controls.

The synthesis framework used in this paper [34] 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 7) which cooperate. 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.

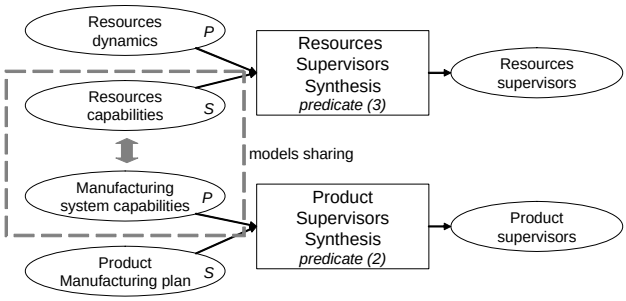

Figure 7. The product-driven synthesis framework

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 which are defined independently from the location where they are executed,

– consistent models (dashed square of Figure 7) 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 proposed synthesis processes. This corresponds in fact to an input/output interpretation of SCT theory as proposed by Balemi *et al.* [3].

3.1 Definition of models used for synthesis by the MES Interface

As presented in the previous paragraph, the definition of resource and product supervisor is done from models of resource dynamics, resource capabilities and product process plans. The *MES interface* of the *Configuration Management Application* provides such models to the *Global Configuration Management Agent* by extracting them from the MES database. By this way, the consistency is ensured between the shop floor control level and the MES level.

In the same idea of integration, but at higher level, ISO62264 [16] standard provides models and terminology to define interfaces between an enterprise's business systems and its manufacturing control systems. Using the guidelines proposed by the standard, a (partial) logical model of MES database product manufacturing plan can be extracted (Figure 8).

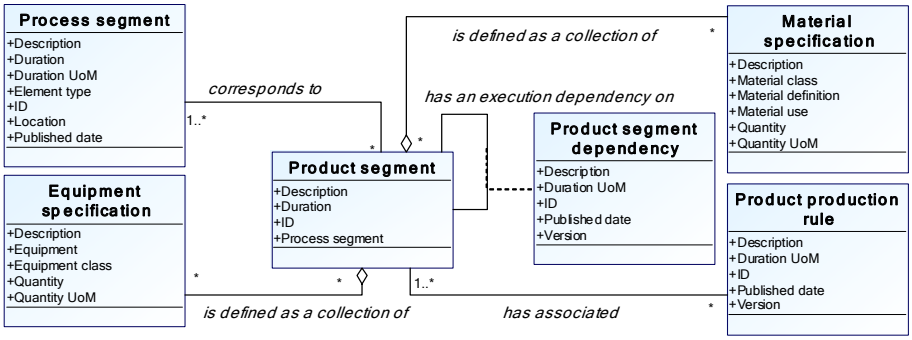

Figure 8. Logical model extracted from ISO 62264

Deriving from this logical model, a relational model including tables directly extracted from classes of the standard model (e.g. Material specification, Process segment …), has been enriched with tables related to Requests (RQ) and Reports (RP) events (Figure 9). These events correspond to the product-driven control level: products are able to *request* an operation performed by a resource, and the resources *report* to products operation ends.

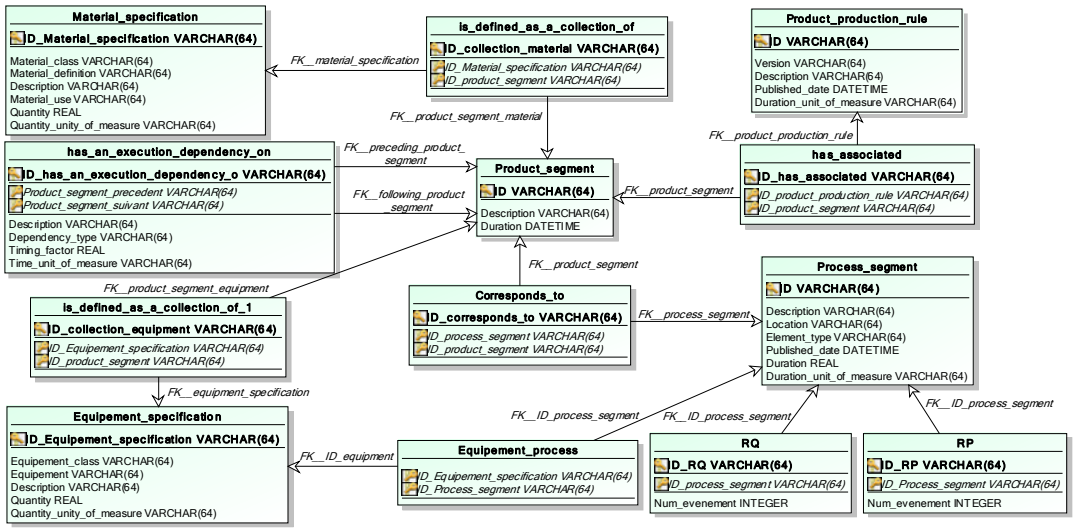

Figure 9. Relational model of prototype MES database, inspired by ISO 62264

A prototype case tool, called MES2TCT, has been developed in Java. It is able to extract from the ISO 62264 MES database models of product manufacturing plans, and manufacturing system capabilities, under the form of finite state machine used by TCT[6].

3.1.1 Models of product manufacturing plans

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 10), 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). 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.

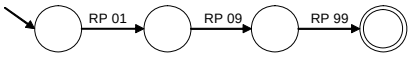

Figure 10. Product manufacturing plan for 01-09 product (*specification*)

3.1.2 Models of manufacturing system capabilities

To model the manufacturing system capabilities, resource generic models and a specific composition operator, based on the cell topology, are used. To obtain these generic models and the specific operator, based on equivalence classes of states, are further detailed in Pétin *et al.* [34]. From a control point of view, the description of resource capabilities is limited to the enumeration of the operations a resource is able to perform, controlled by the product. 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 (Figure 11).

[6] 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/)

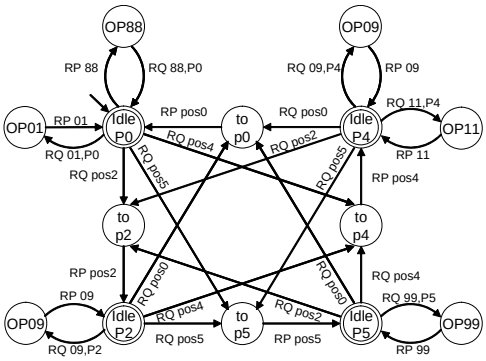

Figure 11. Model of manufacturing system capabilities (*plant*)

3.2 Synthesis of product-driven supervisors

From these models extracted by the MES interface, supervisors of product and resources can be synthesized.

3.2.1 Product supervisor synthesis

Classical synthesis algorithms [36] are applied to generate a supervisor which controls the alternate routes of a given product within the cell. Using this approach, a 01-09 product supervisor can be generated using the TCT tool for synthesis procedures, from the automata of Figure 10 (as *Specification*) and Figure 11 (as *Plant*) (Figure 12). It defines all acceptable routings of the 01-09 product within the assembly cell according to the different assembly operations this product has to undergo (i.e. according to the product specification presented in Figure 10). 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. 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. To manage flexibility, optimization criteria are used by Configuration Agents to choose statically (before manufacturing), or dynamically (during manufacturing) one trajectory from all possible (detailed in §4.1).

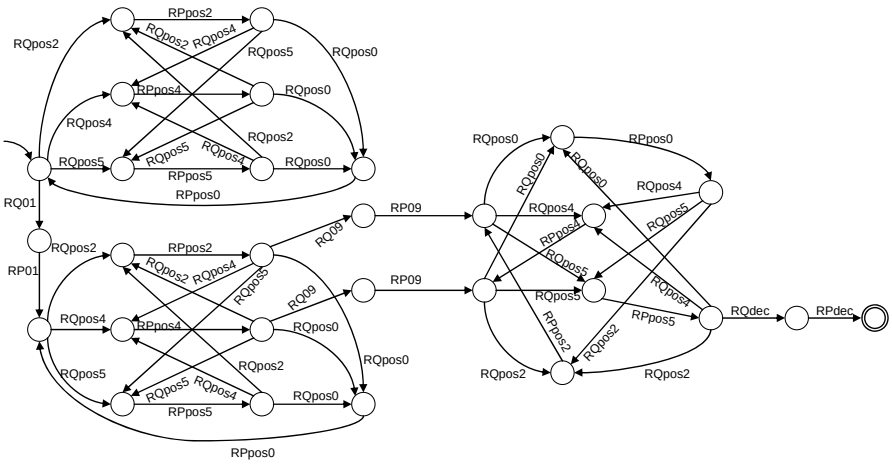

Figure 12. 01-09 product supervisor

3.2.2 Resource supervisor synthesis

The synthesis of resource supervisors approach (first presented in Gouyon *et al.* [12]) combines object-oriented automation rationales and modular synthesis techniques. Modularity criteria are not only driven by state-space explosion issues, but must be justified by the structure of the physical process itself. To this end, this structured modelling starts with the elementary actions which can be executed by resources using actuators (level 1 Service Interfaces). These actions are progressively coordinated in a bottom-up manner to perform actions which 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 (Figure 13).

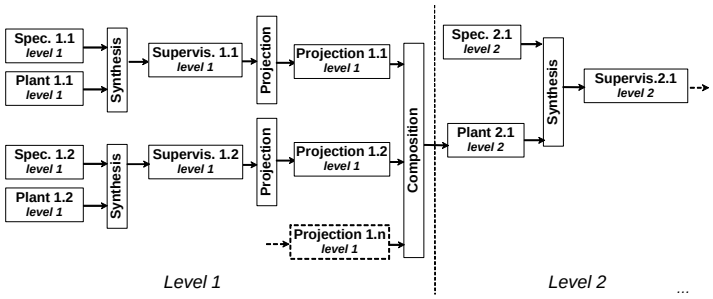

Figure 13. Iterative synthesis method [12]

Application to the assembly cell of this iterative synthesis method 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. From "layer 1" models of the operative part (double-acting air cylinders with their control valve and sensors, and of corresponding *specifications*, this method enables synthesis of layer 1 supervisors (Figure 14).

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

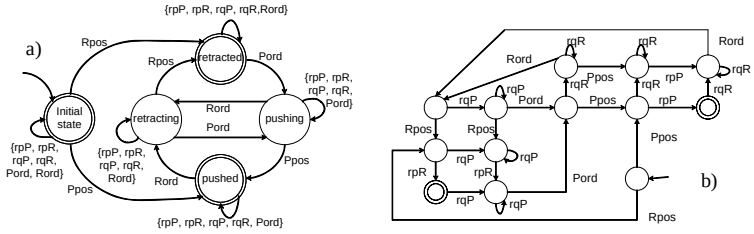

Figure 14. Air cylinder *generator* (or *plant*) (a) and supervisor (b) models

Iteratively, at each layer n, supervisors of layer n-1 are used to build layer n *generator* models, with a mechanism of projection and composition presented in Gouyon *et al.* [12]. Supervisors of layer n are then synthesized from such models and from a corresponding model of coordination rules *specification* (Figure 15) with TCT.

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

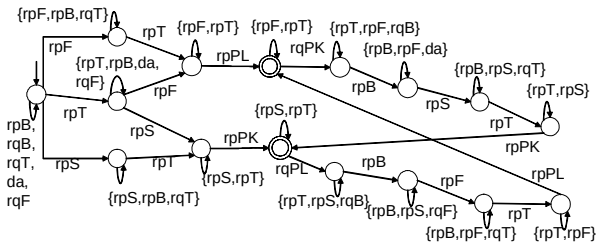

Figure 15. *Pick & place* specification model

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.

4 Control application

The role of the Control Application is to transform supervisors synthesized by the Global Configuration Management Agent into implementable controllers (*Configuration Agent*), to synchronise agents in order to ensure consistency control when reconfigurations are being processed (*Mode Management Agent*), and to execute the new configuration (*Execution Agent*).

4.1 Configuration agent

The implementation of product and resource controllers requires a specific agent to transform supervisors 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 [49]. 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. Bridging the gap between product supervisor and product controller requires removing indeterministic situations by selecting one of the acceptable manufacturing trajectories.

In the case of product supervisors (control of product routing), such a selection can be done according to external criteria such as resource performance, resource availability, transport time to resource or status of the resource, and then call for the corresponding resources. Data used to make choice comes from a centralised MES database and/or from Execution Agent reports ('Resource data' and 'Execution data' of Figure 5), in order to optimize trajectories for each product. The problem is similar to path research within an automaton and solutions have been proposed using static or dynamic costs associated with each transition [23], or optimisation algorithms, such as Dijkstra's algorithm [9]. Static cost will help in defining the chosen trajectory before production while dynamic costs will help in defining in real time the accurate trajectories, taking in account failure of resources ('Failure detection' in Figure 5). 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 by the Configuration Agent to choose the next state among all admissible states. Figure 16 shows an example of a manufacturing route which results from successive dynamic choices based on the product supervisor given in Figure 12. In other words, this trajectories optimisation is similar to a centralised control of the production, in the same way than the Staff Holon of the PROSA architecture, having a global view on all the product (states and location on the plant) and the resources (capability and availability) to optimise locally the product trajectory.

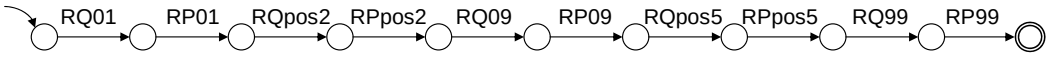

Figure 16. Selected manufacturing route

In the case of resource supervisors, translation into deterministic controllers provided with an input-output interpretation [3] is based on a priority allocation mechanism described in Gouyon *et al*. [12]. 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 which activate transitions toward states in which outputs are produced and maintained until these states are deactivated. These coding rules are based on algebraic equations which 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 PLC standard programming languages such as Ladder Diagram (LD) or Structured Text (ST), in IEC 61499 function blocks. 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 17 shows an example of a supervisor implementation in LD language.

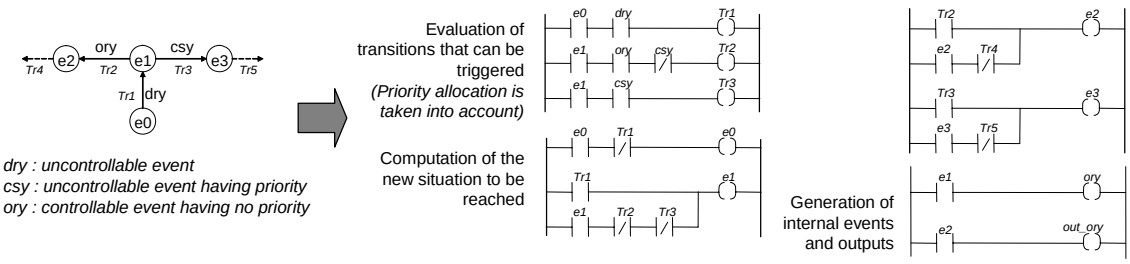

Figure 17. From supervisor to Ladder Diagram

4.2 Mode management agent

The Mode Management Agent has to ensure consistency control when the reconfiguration is being processed. It acts like a switch by giving the hand either to the *Configuration Agent* or to the *Execution Agent*. The control structure of this agent can be represented through the automaton given in Figure 18. It controls starting and stopping processes of the Execution Agent, according to the constraint of reconfiguration points provided by *Mode Management Agent* (variable *EXEC_END*), and triggers the *Configuration Management Agent* for generating and downloading code.

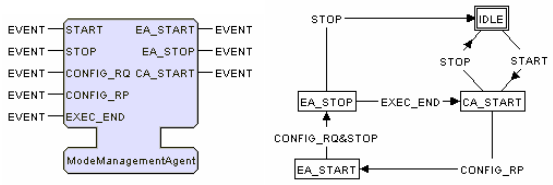

Figure 18. Mode Management Agent interface and ECC

4.3 Execution agent

The EA objective is basically to correctly execute and monitor embedded control code. Code structure is built as a network of IEC 61499 function blocks which operate in a client/server mode. Some function blocks, named *Service Interfaces* (SI), are considered as servers. They provide elementary functions for the control and the monitoring of elementary field devices such as actuators and sensors. They correspond to an implementation of layer 1 controllers (section 3.2.2). An example of such a function block, which can be used in the Services Interfaces for the control and the monitoring of an air cylinder, is given in Figure 19. Note that the reconfiguration points are set during the design phase, ensuring that reconfiguration occurs when the system state is stable, without any action running.

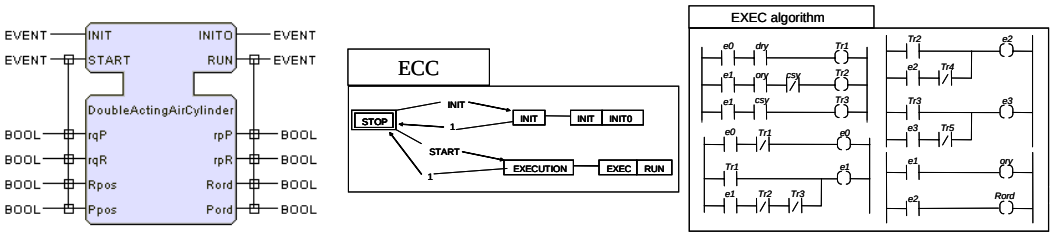

Figure 19. FB example of Service Interface

Services provided by the *Services Interfaces* are sequentially called within the *Execution Agent* algorithm (corresponding to a supervisor of layer 2), included into a coordination FB (called *Coordination Agent*). This sequence is represented by a finite state automaton where each state is associated to a service call (request to lower lever FB) and where a transition depends on the service acknowledgment (report from lower lever FB). These sequences are synthesized by *Configuration Agents*. An example of devices coordination of a manipulator of the case study (two air cylinders and a vacuum cup) is given in Figure 20.

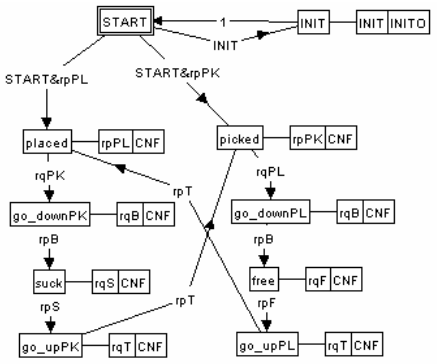

Figure 20. Example of EA coordination sequence

The *Execution Agent* control part must be triggered and stopped by external events, respectively START (a request of an upper level FB) and STOP (a report to an upper level FB). The internal synchronisation within the *Execution Agent* is performed using a global clock which schedules Coordination and Service Interfaces FB execution by successively transferring an internal START event among the different function block ECC.

5 Conclusion

This work is part of a research project on product-driven automation for business-to-manufacturing purposes [26]. This paper focuses on the design and implementation of a product-driven control system in a reconfigurable environment. As reconfiguration is the main property which is addressed to face variability of customized products, it justifies the use of automatic synthesis techniques.

A formal product-driven reconfiguration framework, using synthesis techniques of the Supervisory Control Theory, and their integration within the context of industrial automation (MES and IEC 61499), is presented. The first main objective is to provide a modeling and synthesis method to ensure the interoperability between product controllers which manage product routings within the manufacturing systems, and structured resource controllers which manage the execution of manufacturing operations. The second main objective is to provide a reconfigurable control architecture to dynamically integrate synthesized controllers.

This approach proposes product routes which respect *a priori* individual specifications using the synthesis techniques, and which can be considered as safe with regards to the specifications. Resource and product control architecture has been statically successfully simulated and implemented on the Flexible Assembly Cell case study. However, proving that no 'live-locks' can occur when various products are considered at the same time in a same manufacturing system, requires the use of tools supporting a discrete event simulation of the production flows, such as Arena[7], or the use of a centralized agents that could ensure coordination of the product controllers as done by Staff Holons in the PROSA framework [42]. This point must be part of on going work to prove efficiency of product driven control. Next step to demonstrate the feasibility of the concept should also bridge the gap from synthesis and simulation of the product and resource controllers toward the implementation on industrial devices of the Mode Management Agent, Execution Agent and Configuration Management Agent.

6 References

1. ACHOUR Z., REZG N., XIE X., **Supervisory control of marked graphs with partial observations**, International Journal of Production Research, vol. 42, n° 14, 2004, pp. 2827-2838
2. BABICEANU R. F., CHEN F. F., **Development and applications of holonic manufacturing systems: a survey**, Journal of Intelligent Manufacturing, 17, 2006, pp. 111-131.
3. BALEMI S., HOFFMANN G.J., GYUGYI P., WONG-TOI H., FRANKLIN G.F., **Supervisory control of a rapid thermal multiprocessor,**. IEEE Trans. on Automatic Control, 38, 7, 1993
4. BERRUET P., TOGUYENI A. K. A., ELKATTABI S., CRAYE E., **Toward an implementation of recovery procedures for flexible manufacturing systems supervision**, Computers in Industry, Vol. 43, Issue 3, 2000
5. BRENNAN R. W., ZHANG X., XU Y., NORRIE D. H**., A Reconfigurable Concurrent Function Block Model and its implementation in Real-Time Java**, J. of Integrated Computer-Aided Engineering, 9, 2002
6. CASSANDRAS C.G, LAFORTUNE S., **Introduction to discrete event systems**, ISBN 0-7923-8609-4, 1999
7. DA SILVEIRA G., BORENSTEIN D., FOGLIATTO F.S., **Mass customization: literature review and research directions**, Int. Journal of Production Economics, 72, 2001, pp. 1-13.
8. DEEN, S.M. (Ed.), **Agent-Based Manufacturing - Advances in the Holonic Approach**, Springer ISBN 3-540-44069-0, 2003
9. DIJKSTRA, E.W., **A note on two problems in connexion with graph**, Numerische Matematik, 1, 1959
10. ENDSLEY E. W., ALMEIDA E. E., TILBURY D. M., **Modular finite state machines: development and application to reconfigurable manufacturing cell controller generation**, Control Engineering Practice, Vol. 14, issue 10, pp. 1127-1142, 2006
11. FUSAOKA A., SEKI H., TAKAHASHI K., **A description and reasoning of plant controllers in temporal logic,** Proceedings of the 8th Int. Joint Conference on Artificial Intelligence, Karlsruhe, Germany, 1983

[7] http://www.arenasimulation.com/

12. GOUYON D., PETIN J.-F., GOUIN A., **Pragmatic approach for modular control synthesis and implementation,** International Journal of Production Research, vol. 42, n° 14, 2004

13. HOLLOCKS B.W., GORANSON H.T., SHORTER D.N., VERNANDAT F.B., **Assessing Enterprise Integration for Competitive Advantage,** ICEIMT'97, International conference on Enterprise Modelling and Modelling Technology, Berlin, 1997

14. International Electrotechnical Commission, **IEC 61499: Function blocks for industrial-process measurement and control systems**, 2000

15. International Electrotechnical Commission, **IEC 61131-3: Programmable controllers – part 3: programming languages** (2nd edition), 2003

16. International Organisation for Standardization, **ISO 62264: Enterprise-control system integration**, 2003

17. IUNG B., NEUNREUTHER E., MOREL G., **Engineering process of integrated-distributed shop floor architecture based on interoperable field components**, *International Journal of Computer Integrated Manufacturing*, Vol. 14, No. 3, 2001, pp. 246-262

18. KARKKAINEN M., HOLMSTROM J., FRAMLING K., ARTTO K., **Intelligent products – a step towards a more effective project delivery chain**, Computers in Industry, 50, 2003, pp. 141-151.

19. KOESTLER A., *The ghost in the machine*, ISBN 0-14-019162-5, 1967

20. LAUZON S.C., MILLS J.K., BENHABIB B. (1997). **An implementation methodology for the supervisory control of flexible manufacturing workcells,** J. of Manufacturing Systems, vol. 16, n°1, 1997

21. LEWIS R, **Modelling control systems using IEC 61499: Applying function blocks to distributed systems**, IEE Control Engineering series, n°59, 2001, ISBN 0-85296-796-9

22. MARCHAND H., BOURNAI P., LE BORGNE M., LE GUERNIC P., **Synthesis of discrete-event controllers based on the Signal environment,** Discrete Event Dynamic Systems: Theory and Applications, 10, 2000, pp. 325-346

23. MARCHAND H., BOIVINEAU O., LAFORTUNE S**., On the Synthesis of Optimal Schedulers in Discrete Event Control Problems with Multiple Goal,** SIAM Journal on Control and Optimization, 39(2), 2000, pp. 512-532.

24. MARIK V., LAZANSKY J., **Industrial application of agent technologies,** Control Engineering Practice, 2006, doi:10.1016/j.conengprac.2006.10.001

25. MCFARLANE D., SARMA S., CHIRN J.L., WONG C.Y., ASHTON K**., Auto ID systems and intelligent manufacturing control**, Engineering Application of Artificial Intelligence, 16, 2003, pp. 365-376

26. MOREL G., PANETTO H., ZAREMBA M., MAYER F**., Manufacturing enterprise control and management system engineering: rationales and open issues,** IFAC Annual Reviews in Control, 27-2, 2003

27. MOREL G., VALCKENAERS P., FAURE J.M., PEREIRA C., DIEDRICH C., **Survey paper on manufacturing plant control challenges and issues,** 16th IFAC World Congress in Prague, 2005

28. MUHL E., CHARPENTIER P., CHAXEL F., **Optimization of physical flows in an automotive manufacturing plant: some experiment and issues**, Engineering Application of Artificial Intelligence, Vol. 16, 2003, pp. 293-305

29. NOF S. Y., MOREL G., MONOSTORI L., MOLINA A., FILIP F., **From plant and logistics control to multi-enterprise collaboration**, IFAC Annual Reviews in Control, vol. 30, 2006, pp. 55-68

30. OLSEN S., WANG J., RAMIREZ-SERRANO A., BRENNAN R. W**., Contingencies-based reconfiguration of distributed factory automation,** Robotics and Computer Integrated Manufacturing, Vol. 21, Issues 4-5, 2005, pp. 379-390.

31. PÉTIN J.-F., IUNG B., MOREL G., **Distributed intelligent actuation and measurement (IAM) system within an integrated shop-floor organisation**, Computers in Industry, 37, 1998, pp. 197-211

32. PÉTIN J.-F., KLEIN T., MOREL G., **Function block model for dynamic reconfiguration of Discrete event Systems**, 17th IMACS World Congress, Paris, France, July 11-15, 2005

33. PÉTIN J.-F., MOREL G., PANETTO H., **Formal specification method for systems automation**, European Journal of Control, vol.12, n°2, 2006, pp. 115-130.

34. PÉTIN J.-F., GOUYON D., MOREL G**., Supervisory synthesis for product-driven automation and its application to a flexible assembly cell**, to appear in Control Engineering Practice, 2007

35. QIU R. G., WYSK R., XU Q., **Extended structured adaptive supervisory control of shop-floor controls for an e-manufacturing system,** International Journal of Production Research, Vol.41, N° 8, 2003

36. RAMADGE P.J., WONHAM W.M., **Supervisory control of a class of discrete event processes**, SIAM J. Control and Optimization, Vol. 25, n° 1, 1987

37. SIMON H. A., **The sciences of the artificial**, 3rd edition, 2001, ISBN 0-262-69191-4

38. STAROSWIECKI M., BAYART M., **Models and languages for the interoperability of smart instruments**, Automatica, Vol. 32, Issue 6, 1996, pp. 859-873

39. THRAMBOULIDIS K., TRANORIS C., **Developing a CASE Tool for Distributed Control Applications**, Journal of Advanced Manufacturing Technology, Vol. 24, 1-2, 2004, pp. 24-31

40. TOGUYENI A. K. A., CRAYE E., SEKHRI L., **Study of the diagnosability of automated production systems based on functional graphs**, Mathematics and Computers in Simulation, Vol. 70, Issues 5-6, 2006
41. VALCKENAERS P. (Ed.), **Holonic Manufacturing Systems**, Computer In Industry, 46 (3), 2001
42. VAN BRUSSEL H., WYNS J., VALCKENAERS P., BONGAERTS L., PEETERS P., **Reference architecture for holonic manufacturing systems: PROSA**, Computers in Industry, 37, 1998, pp. 255-274.
43. VOGRIG R., BARACOS P., LHOSTE P., MOREL G., SALZEMANN B., **Flexible Manufacturing Shop Operation,** Manufacturing System, Vol. 16, 1987, pp 43-55.
44. VYATKIN V. V., CHISTENSEN J. H., MARTINEZ LASTRA J. L., **OOONEIDA : an Open, Object-Oriented kNowledge Economy for Intelligent Distributed Automation**, IEEE Transactions on Industrial Informatics, Vol. 1, n° 1, 2005
45. WONG K.C., WONHAM W.M., **Modular control and coordination of discrete-event systems**, Discrete Event Dynamic Systems: Theory and Applications, 8(3), 1998
46. WONG C.Y., MCFARLANE D., AHMAD ZAHARUDIN A.A., AGARWAL V., **The intelligent product-driven supply chain**, IEEE International Conference on Systems, Man and Cybernetics, 2002
47. YOO T.-S., LAFORTUNE S**., A general architecture for decentralized supervisory control of discrete-event systems**, Discrete Event Dynamic Systems: Theory and Applications, 12, 2002
48. ZAMAÏ E., CHAILLET-SUBIAS A., COMBACAU M., **An architecture for control and monitoring of discrete events systems**, Computers in Industry, Vol. 36, Issues 1-2, 1998, pp. 95-100
49. ZAYTOON J., CARRE-MENETRIER V., **Synthesis of a correct control implementation for manufacturing systems**, Int. Journal of Production Research, 39, 2001, pp. 329-345.