

Verification of functional constraints for safe product driven control

Pascale Marangé*, David Gouyon*, Jean-François Pétin*, Bernard Riera **

Centre de Recherche en Automatique de Nancy, UMR 7039 CNRS Nancy-Université
Faculté des Sciences et Techniques, BP 239, Vandœuvre-lès-Nancy Cedex, FRANCE, pascale.marange@cran.uhp-nancy.fr,
david.gouyon@cran.uhp-nancy.fr, jean-francois.petin@cran.uhp-nancy.fr

** Centre de Recherche en STIC, URCA, UFR Sciences Exactes et Naturelles de Reims, BP 1039, 51687 REIMS, France,
bernard.riera@univ-reims.fr

Abstract: This paper deals with dynamic reconfiguration of DES control within the context of manufacturing system to answer product variability or production resources availability. More precisely, it focuses on product-driven control (PDC) that embeds decisional and adaptation capabilities within the product. PDC can be defined in a static or dynamic way: in the first case, all switchable manufacturing trajectories are predefined and implemented in the product control while, in the second case, PDC can be seen as a non-deterministic and “on the fly” choice among control trajectories to fit with the manufacturing system availability and the product state. In this last case, PDC safety is ensured thanks to a filtering approach that enables to maintain PDC within a state space that satisfy functional and safety constraints whatever the control generated by the reconfiguration process is. This paper focuses on the definition and verification of a minimal set of constraints that ensure the sufficiency and the non blocking properties of the constraints. A case study illustrates the approach.

Keywords: Product driven control, Model Checking, Reconfigurable manufacturing system

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 itself (or to reconfigure itself) to variable demands in terms of product specifications or manufacturing system changes (Koren 2005).

As an answer to this industrial challenge, Product-Driven Control (PDC) consists in providing each product occurrence with information, decision and communication capabilities in order to make the product active in the scheduling and the execution of its manufacturing operations (McFarlane *et al.* 2003).

Control strategies embedded in the product can be defined in a static way including all possible trajectories (Qiu *et al.* 2003) or in a dynamic way to adapt “on the fly” control decisions to fit with the manufacturing system and the product state. In the first case, the estimated scheduling is defined before implementation; this definition can be very sensible against environment variation (failure, unpredictable events, unavailable resources, ...). In the second case, we assume that control could be non-deterministic but limited to an ad-hoc choice between legal trajectories. Decision making is intrinsically a longer job, but the decision made is more robust against unpredictable events. This is compliant with our objective which is to facilitate the online reconfiguration of production. So, to ensure safety, it is necessary to verify that, whatever the “on the fly” control is, it is compliant with the functional characteristics of the product to be manufactured and with the process safety conditions. This verification can be done using a filtering approach (Marange *et al.* 2007) which enables the only decisions that satisfy a set of functional and safety constraints.

This paper aims at presenting a proved definition of the constraints which are necessary to execute such a safe product-driven control, and gives guidelines for their implementation. Section 2 introduces the formal foundations of PDC and an overview of existing design techniques. Section 3 deals with the design of PDC constraints and their verification using model checking. The proposed approach is illustrated in section 4 using a simplified case study. Conclusions and open issues for future research are discussed in section 5.

2. STATE OF THE ART

2.1. Product-driven control

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 & 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 for:

- monitoring (Vogrig *et al.*, 1987, Zamai *et al.*, 1998), diagnosis (Toguyeni *et al.*, 2006) or even prognosis 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,
- operational execution of the reconfigured control actions to manage switching from an obsolete control strategy to a new targeted configuration.

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. Products and resources are then interacting to perform manufacturing operations. Based on these conceptual guidelines, this paper focuses on the design of a product-driven distributed control system, shown in Fig. 1, which is based on the cooperation between:

- *product control* which controls the manufacturing routes according to a scheduled list of operations the product has to undergo; these controls are specific for each product occurrence in order to take into account their customization,
- *resource control* which ensures correct execution of manufacturing operations and provide the product controllers with accurate reports.

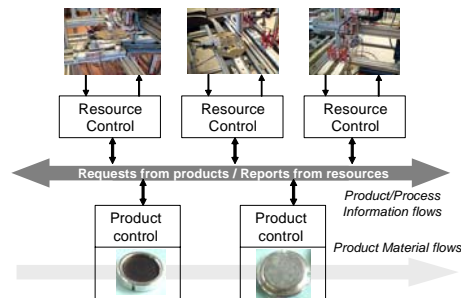


Fig. 1: Product-driven control architecture

For implementation aspects, infotonics technology such as RFID tags embedded on the products enables individual identification of product occurrences that open a way towards the customization of control constraints for each product occurrence (McFarlane *et al.* 2003).

2.2. Product-driven control engineering

The PDC definition is founded, on the one hand, on the modelling of the manufacturing system capabilities which describes 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. Two different ways of thinking the design of PDC architecture can be considered.

The first one is based on the specification of a set of switchable control policies using a graph that represents all legal product routings and the functional redundancies between the machines (Berruet *et al.* 2000, Toguyeni *et al.* 2006). Generations of such a graph can rely on synthesis techniques within the framework of Supervisory Control Theory (Qiu *et al.* 2003, Pétin *et al.* 2007). Then, operational PDC implements choices between the legal routes that results from the execution of a short path optimisation algorithm. This approach assumes that all possible manufacturing trajectories have been previously embedded in product-driven controllers and suffers from an exponential complexity in terms of number of control states, which naturally arises as the number of machines and product variability increase.

Second way of thinking consists in defining PDC as a non deterministic evolution within a state space guarded by a set of constraints. In other words, it means that the control decisions are not completely predefined but are postponed when a product ask for a transformation by selecting, even in a non-deterministic way, a control action compliant with the set of constraints. Consequence is that no strong hypothesis on the reconfigured control properties can be assumed. Consequently, a filter has to be placed before the execution in order to dismiss the evolutions that do not comply with safety and functional constraints. This approach strongly depends on the constraints definition and more especially on their covering features. For the safety constraints, an approach has been proposed by (Marange, 2008) to specify a set of constraints for each plant elements and to verify, using model checking, that this set of constraints is preserved by any controller. This paper enriches this work by considering aspects related to product functional constraints. In this case, the constraints must be efficient enough to ensure the reachability of transformation control states according to given product features but they also must be permissive enough to ensure the existence of at least one manufacturing sequence (non-blocking constraints).

3. PROPOSED APPROACH

The proposed approach considers products and resources interacting in order to execute product manufacturing plans. To ensure safety and correctness of the execution, a set of constraints are designed and used as filtering conditions that maintain PDC reconfigured control in a legal state space. This section begins by giving an overview of the approach which illustrated with some models, then proposes a method to design and verify constraints sufficiency and non-blocking properties, and finally explains how these constraints can be implemented in a product driven control system.

3.1. Overview of the approach

In order to ensure that the functional manufacturing plan of the product is respected, a four steps approach is proposed (Fig. 2):

- functional constraints writing;
- verification of constraints sufficiency. This consists in the verification that the minimal set of constraints ensuring the proper sequencing of operations, is included in the identified set of constraints;
- verification of constraints non-blocking to check that the constraints are not too restrictive, i.e. there is at least a way to achieve the product manufacturing plan;
- implementation of the constraints which can be used “on line” to filter the only legal PDC actions.

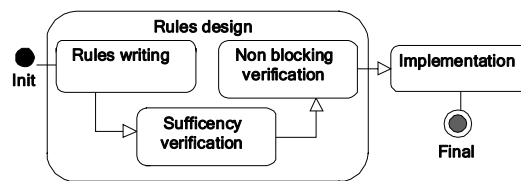


Fig. 2: Activities of the proposed approach

To perform constraints verification, the following hypotheses are considered:

- all plant evolutions are observable by the PLC,
- functions sequence are considered only as logical constraints, and do not include time aspects,
- the plant is considered without failure,
- for the trajectories choice, the cost is not yet taken in account. In future works, choice criteria will integrate costs linked to resource performances or to transfert time between resources,
- we consider that the transfer resource is always available (as in the case of a conveyor).

3.2. Constraints design

During control execution, a function can be in several states: to be executed Fi_{ie} , in execution Fi_{ie} , or executed Fi_{ex} . The request for a function execution is noted Fi_{rq} . To authorize the function execution, a filter using information contained in the product validates a set of constraints. These functional constraints, which can be defined as logic equations, are of two types: one is related to the product features and the other is related to the manufacturing resource capabilities.

3.2.1.1. Constraints writing

For each product, a set of constraints is defined to ensure that the good product is made. Each product is considered independently of each other, and execution conditions on other functions are defined for each function Fi :

- the functions that should have been executed before Fi , condition noted Fi_{before}
- the functions that cannot be executed at the same time than Fi , condition noted Fi_{during}
- the functions that cannot be executed after Fi , condition noted Fi_{after}
- the function that cannot be executed on the product, noted $Fi_{prohibited}$

Each condition can be extended according to the function states. For example, $Fi_{before\ ex}$ represents the condition Fi_{before} with the parameters Fi_{ex} .

To ensure the respect of these precedence conditions (Allen, 1983) functional constraints are defined for each function Fi :

- $Fi_{rq} \wedge \overline{Fi_{before\ ex}} = 0$ ensures that prior functions needed before the execution of Fi have been executed,
- $Fi_{rq} \wedge (Fi_{during\ rq} \vee Fi_{during\ ie}) = 0$ ensures that the function included in Fi_{during} are not executed at the same time that Fi , i.e.

in a PLC cycle in which Fi is requested, functions included in Fi_{during} , should not be already in execution or requested for activation,

- $Fi_{after\ rq} \wedge Fi_{ex} = 0$ ensures that the functions included in Fi_{after} cannot be executed after the Fi execution,
- $Fi_{prohibited\ rq} = 0$ ensures that the functions included in $Fi_{prohibited}$ cannot be executed.

In addition to these precedence constraints, we have to verify that a given function ($F_{i_{rq}}$) is not requested when the product is present on a resource (noted pp_j) that does not provide this given function: $pp_j \wedge F_{i_{rq}} = 0$.

Furthermore, the deactivation of function is forbidden before the end of its execution, what is represented by the constraint: $\downarrow F_{i_{rq}} \wedge \overline{F_{i_{ex}}} = 0$.

The goal of these constraints is to ensure a correct manufacturing plan execution, but if they are not well defined, they can be too permissive or bring some blocking situations. So, once they are written, it is necessary to proceed to some verification.

3.2.1.2. Sufficiency and non-blocking property verification

Once product requirements given and resources capability built, the verification process helps the designer to identify and to formalise a set of necessary and sufficiency constraints. Verification of the constraints aims to ensure, on the one hand, that constraints are efficient enough to disable all the manufacturing plans which are incorrect, and on the other hand, that the constraints are not too restrictive, i.e. that it exists at least one possibility to manufacture the product. The verification of the sufficiency and non-blocking property is done on a system model. This model is defined in a modular way considering the execution environment, the control and function execution. In order to reduce the risk of state space explosion, the modelling tool which is used is UPPAAL timed automata (Alur & Drill 1994, Behrmann *et al.* 2002).

3.2.1.3. Generic model of environment and resources

To verify that functional constraints are correctly defined, their sufficiency and non blocking properties are verified by model checking (Schnoebelen *et al.* 1999). The system is then defined by models of the execution environment, tag reading and update, functions activation/deactivation control and functions execution, which are detailed below.

The model of the environment (Fig. 3) represents the cyclical evolution of the PLC, and will synchronize the evolution of the different models with synchronization messages. In a product point of view, the PLC cycle is defined as follows: PLC inputs reading, tag reading, functions evolution to identify which are the possible functions, application of the constraints to verify if properties are respected, functions execution and tag and PLC outputs update. Before a new cycle is started, a verification step is added to verify sufficiency and non-blocking.

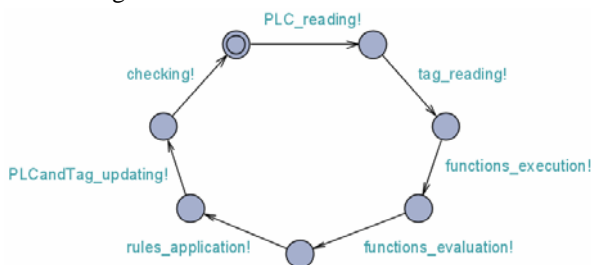


Fig. 3: Model of the execution environment

When a product is present on a resource, information contained in the tag is read (Fig. 4a) to update functions state on each resource (requested, in execution ...), then the control is executed, and finally the tag is updated (Fig. 4b) to trace the execution of functions. Reading and update can be done during each PLC cycle to ensure control consistency.



Fig. 4: Generic models of tag reading and update

As no assumption on the implemented control can be made, functions can be activated or deactivated during the period between the reception of messages *function_evolution* and *constraint_application* (Fig. 5).

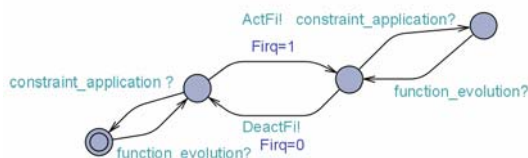


Fig. 5: Generic model of function F_i activation/deactivation

Before functions execution, the function evolution phase tests functions activation/deactivation using the constraints set. For that, when the message *constraint_application* is received, the guard representing the constraint must be false; if not, the variable *errori* is updated to “true” (Fig. 6), and then forbid the function execution.

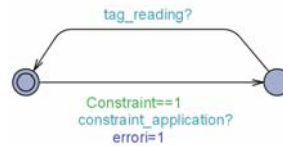


Fig. 6: Generic model of constraints

Functions can be of two types: shape processing or transport function. Shape processing function (Fig. 7) is modelled by a 3 states automaton: {*Not executed*, *In execution*, *Executed*}. The automaton goes from *Not executed* to *In execution* when the activation is made ($F_{i,q}=1$), if a product is present on the resource ($ppi=1$) and if all constraints are met ($errori=0$). The function is executed when the time *function_time* passes. With transitions crossing, function state values are updated.

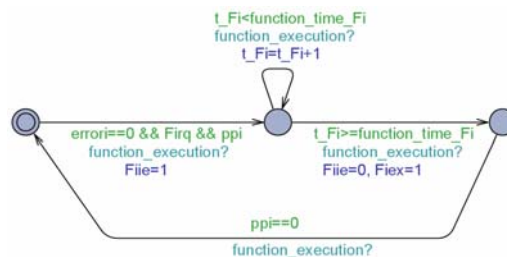


Fig. 7: Generic model of shape processing function (F_i)

Transport function is represented with a 3 states automaton: {*Not executed*, *Left from Start position*, *At End position*}, transitions crossing is made according to time values (Fig. 8).

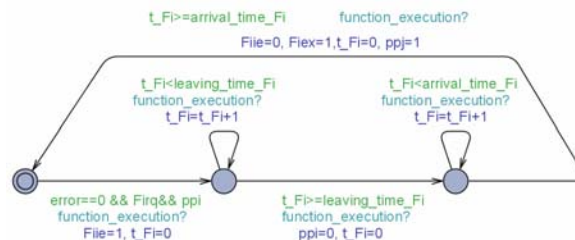


Fig. 8: Generic model of transport function (F_i)

3.2.1.4. Verification of constraints sufficiency

Once functional constraints are defined, the constraints sufficiency has to be verified, to ensure that all the evolutions which do not bring a correct product will be filtered.

To verify constraints sufficiency, an observer, including an error state representing the paths that do not respect constraints, is designed for each constraint. If constraints are sufficient, then the property “regardless the way defined by the control, the error state is not reachable”, is always true.

If it is simple to express the property to be checked, main difficulties remain in the definition of the observer that is prone to mistakes on the part of the expert (bad definition, forgetting elements ...) and in the risk of state explosion in model definition and in verification calculation. This fact justifies our observer modularity (i.e. one observer for each constraint).

3.2.1.5. Verification of constraints non blocking property

The last step of constraints design is to verify that they are not too restrictive to achieve the process plan. To verify the non-blocking control, the process plan is modelled with a *finished product* state. The verification of the non-blocking property of constraints consists then in finding at least one path reaching to the *finished product* state.

3.3. Control implementation

The set of specified and verified constraints can be implemented within a filtering supervisor that enable or disable the PDC actions according to the expected manufacturing of a product for any reconfigured control.

In this way, the control is implemented in resources, information on product state (functions executed, in execution ...) and functional constraints are embedded in the product. The verification of constraints respect is made during control execution (Fig. 9):

- the resource detects product presence,
- the resource read the tag on the product: information on product state and functional constraints,
- the resource proposes a function to execute,
- if the constraints are respected, the function is executed and the product is updated,
- otherwise, the function execution is filtered and another function is proposed.

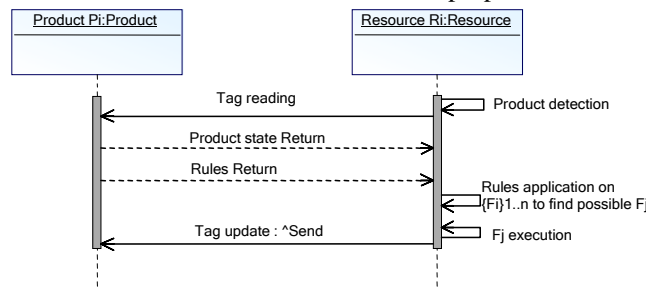


Fig. 9: Interaction scenario with on line control verification

This approach is interesting in the case of flexible or reconfigurable systems. As the control is not completely defined *a priori*, the execution of a function is not linked to a specific resource, thus some flexibility in the execution of the process plans is possible. However, the constraints do not ensure that the product will eventually be manufactured.

4. EXAMPLE: CONDITIONING SYSTEM

The considered conditioning system allows granules packaging, with or without label. It is composed of 4 partially redundant resources (R_1 to R_4) allowing 4 functions (Fig. 10): F_1 represents the bottle closing; F_2 represents the label collage, F_3 and F_4 represent the distribution of types A and B granules. These functions are completed by 4 transport functions. The variable ppi represents the product presence on the resource R_i .

From functions location on the resources, 8 resources constraints are defined:

- Resource (R_1): $ppi_1 \wedge F_{3rq} = 0$ and $ppi_1 \wedge F_{4rq} = 0$
- Resource (R_2): $ppi_2 \wedge F_{1rq} = 0$ and $ppi_2 \wedge F_{2rq} = 0$
- Resource (R_3): $ppi_3 \wedge F_{2rq} = 0$ and $ppi_3 \wedge F_{4rq} = 0$
- Resource (R_4): $ppi_4 \wedge F_{1rq} = 0$ and $ppi_4 \wedge F_{3rq} = 0$

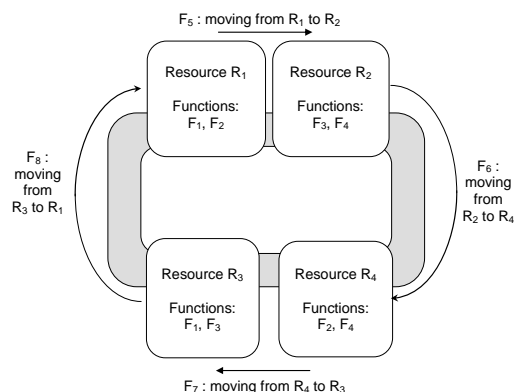


Fig. 10: Overview of the conditioning system

Table 1 describes the 10 various products that can be manufactured. According to the product type, a set of conditions is defined. For example, considering product P_9 , these conditions imply constraints on function execution (Table 2).

Table 1: Product types

| Name | Composition | Label | Name | Composition | Label |
|----------------|------------------------------------|-------|-----------------|------------------------------------|-------|
| P ₁ | A and B (without particular order) | Yes | P ₆ | A and B (without particular order) | No |
| P ₂ | A | Yes | P ₇ | A | No |
| P ₃ | B | Yes | P ₈ | B | No |
| P ₄ | A then B | Yes | P ₉ | A then B | No |
| P ₅ | B then A | Yes | P ₁₀ | B then A | No |

Table 2: Conditions on product P9 manufacturing

| Function F _i | F _i _{before} | F _i _{during} | F _i _{after} |
|-------------------------|-------------------------------------|-------------------------------------|-------------------------------------|
| F ₁ | F ₃ ∧ F ₄ (1) | F ₂ (2) | F ₃ ∨ F ₄ (3) |
| F ₂ | Forbidden (4) | Forbidden | Forbidden |
| F ₃ | ∅ | F ₄ ∨ F ₁ (5) | ∅ |
| F ₄ | F ₃ (6) | F ₃ (7) | ∅ |

According to these conditions, 9 constraints are then defined:

- Condition 1: $F_{1rq} \wedge \overline{(F_{3ex} \wedge F_{4ex})} = 0$
- Condition 2: $F_{1rq} \wedge (F_{2rq} \vee F_{2ie}) = 0$
- Condition 3: $F_{3rq} \wedge F_{1ex} = 0$ and $F_{4rq} \wedge F_{1ex} = 0$
- Condition 4: $F_{2rq} = 0$
- Condition 5: $F_{3rq} \wedge ((F_{4rq} \vee F_{1rq}) \vee (F_{4ie} \vee F_{1ie})) = 0$
- Condition 6: $F_{4rq} \wedge \overline{(F_{3ex})} = 0$
- Condition 7: $F_{4rq} \wedge (F_{3rq} \vee F_{3ie}) = 0$

In the first step of approach, the sufficiency of functional constraints is verified. The constraint $F_{1rq} \wedge \overline{(F_{3ex} \wedge F_{4ex})} = 0$ ensures that the function F_1 is activated only if the functions F_3 and F_4 have been executed. To verify this property, the observer (Fig. 11) describes the possible ways and the forbidden *error* state. Using UPPAAL model checking software, the following propriety is verified:

$$A[\text{not}(\text{observer_efficiency.error})$$

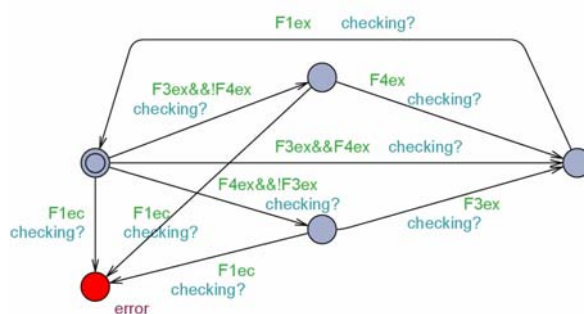


Fig. 11: Sufficiency observer

In the second step, a logical process plan (which does not specify executing resources) is defined to verify that at least a way exists to manufacture the product (Fig. 12). We define the following propriety: $E \diamond (\text{processplan.produit_fini})$ which is verified using UPPAAL.

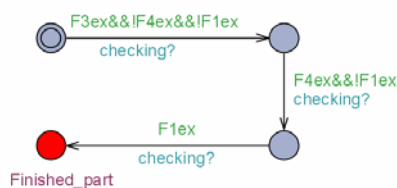


Fig. 12: Process plan

5. CONCLUSION AND PERSPECTIVES

There is a growing interest in models, methods and tools that facilitate the D.E.S. reconfigurable control. This paper focuses on the safety issues of such reconfigurable system in the case of product-driven control. Main proposal consists in a filtering approach that enables to maintain PDC within a state space that satisfy functional (with regard to the product goals) and safety (with regards to the process operations) constraints without assuming any PDC properties (i.e. whatever the control generated by the reconfiguration process is). Presented results refer to the constraints design and verification phase as shown on figure 2. Further work needs to be developed for implementation issues. Moreover, if the writing and verification of functional filter constraints has been illustrated on a simplified case-study, experiments on industrial-scale case-studies emphasize the effort that still must be employed to make the proposed engineering framework effective in practice. In future works, some hypotheses must be lifted. For example, temporal issues will be included in constraint definition, as the use of cost will be used during trajectory choice.

6. REFERENCES

- Allen J.F. (1983) Maintaining knowledge about temporal intervals *Commun. ACM*, 26(11):832–843
- Alur R., D. L. Dill (1994), A theory of timed automata. *Theoretical Computer Science*, 126(2):183-235.
- Behrmann G., J. Bengtsson, A. David, K.G. Larsen, Pettersson P., Yi W., (2002) Uppaal implementation secrets, In Proc. of 7th International Symposium on Formal Techniques in Real-Time and Fault Tolerant Systems
- Berruet P., A. K. A. Toguyeni, S. Elkattabi, E. Craye (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., X. Zhang, Y. Xu, D. H. Norrie (2002). A Reconfigurable Concurrent Function Block Model and its implementation in Real-Time Java, *Journal of Integrated Computer-Aided Engineering*, Volume 9, pp 263-279.
- Da Silveira G., D. Borenstein, F.S. Fogliatto (2001). Mass customization: literature review and research directions. *Int. Journal of Production Economics*, 72, pp 1-13.
- Koren Y., (2005). Reconfigurable manufacturing and beyond. Proceedings of the CIRP 3rd International Conference on Reconfigurable Manufacturing, Ann Arbor, MI, USA.
- Marangé P., F. Gellot, B. Riera (2007) Remote control of automation systems for D.E.S. course, *IEEE Transaction on Industrial Electronics Special Section*, pp 3103-3111
- Marangé P. (2008) Synthèse et filtrage robuste de la commande pour des systèmes manufacturiers sûrs de fonctionnement, PhD dissertation, Université de Reims Champagne Ardenne.
- Mc Farlane D., Sarma S., Chirn J.L., Wong C.Y., Ashton K. (2003) Auto ID systems and intelligent manufacturing control, *Engineering Application of Artificial Intelligence*, 16, pp. 365-376
- Pétin J.-F., D. Gouyon, G. Morel (2007). Supervisory synthesis for product-driven automation and its application to a flexible assembly cell, *Control Engineering Practice*, 15, pp. 595-614.
- Qiu R., R. Wysk, Q. Xu (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.
- Schnoebelen P., B. Bérard, M. Bidoit, F. Laroussinie, A. Petit, (1999). *Vérification de logiciels: Techniques et outils du model-checking*, Vuibert, ISBN 2-7117-8646-3.
- Toguyeni A. K. A., E. Craye, L. Sekhri (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
- Valckenaers P. (Editor) (2001), Special issue: Holonic Manufacturing Systems, *Computer In Industry*, 46 (3), pp. 233-331.
- Vogrig R., P. Baracos, P. Lhoste, G. Morel, B. Salzemann (1987). Flexible manufacturing shop. *Manufacturing Systems*, Volume 16, n°3.
- Zamaï E., A. Chaillet-Subias, M. Combacau (1998). An architecture for control and monitoring of discrete events systems, *Computers in Industry*, Vol. 36, Issues 1-2, Pages 95-100.