Control Law Synthesis and Reconfiguration using SCT

Gregory Faraut, Laurent Piétrac and Eric Niel

***Abstract*— System evolution, such as addition or replacement of a component, may necessitate complete re-design. Such re-design may be needed to respect new or updated requirements. The models then have to be modified. In this paper, we present a procedure for reconfiguration of a discrete event system (DES) controller. Based on supervisory control theory (SCT), the objective of this work is to show how the SCT is convenient in order to reconfigure the controller to take into account the new and updated requirements without re-verifying the requirements that do not change because they are still respected.**

I. INTRODUCTION

Faults/failures cause undesired reactions and consequences as damage to technical parts of plants, to human life or to the environment and a profound impact also on production cost and product quality ([1], [2], [3], [4]). In discrete event system, this often implies a modification of the control law to take into account the updated behavior occurred by a failure and to be able to ensure a minimal production and protection ([5], [6], [7]). However, few of them use formal approaches to ensure that the requirements are respected. The supervisory control theory (SCT) guarantees that the controller respects all specifications ([8], [9]). Some works based on SCT express conditions to modify a controller when a reconfiguration is required ([10], [11], [12], [13]). Furthermore, we proposed in previous works the use of a modal standpoint to design a system and reduce complexity of it ([14], [15], [16]). The system is decomposed into number of modes where each mode represents a part of the control law. The switch between mode is occurred by a fault event. Nevertheless, all controllers, one by mode, are designed at once and do not change after. Until now, if the system evolves, to take into account a new behavior, this needs a complete re-design of the controllers.

In this paper, we propose a procedure, based on SCT, to reconfigure a controller in order to avoid a complete re-design. The reconfiguration keeps the behavior that does not change, removes the behavior that is become useless due to the evolution, and adds the new behavior. Furthermore, the SCT ensures that the requirements are respected when a synthesis is done. In consequence, a partial reconfiguration in using SCT, keeps respecting the requirements that do not change, and ensures the new requirements are also respected in the new control law. The next section of this article recalls some basic notions of SCT and the framework to build the controlled process. In section III, we present a procedure based on SCT devoted to the reconfiguration of the controller. The proposed procedure aims to modify the controller by replacing the outdated requirements by the updated ones and preserving those which are still needed. The proposition is illustrated by a conventional example in section IV.

The authors are with AMPERE Lab. INSA de Lyon, University of Lyon, 20 av. albert Einstein, 69100 Villeurbanne, France firstname.lastname@insa-lyon.fr

II. OVERVIEW

Initiated by Ramadge and Wonham, the Supervisory Control Theory has significantly improved results in the discrete-event systems (DESs) domain. This theory is based on the separation the numerous models, each one representing a part of the expected behavior of the system. The model of process, called G, represents the full behavior that the system can potentially do. This behavior is uncontrolled. The model of specification, called E, represents the requirements that have to be respected by the system. Formally, these models are designed by automata. The automaton A is a $5-$tuple and is defined by : $A := (Q, \Sigma, \delta, q_0, Q_m)$ where Q is a set of the states, Σ is a set of events, $\delta : Q \times \Sigma^* \to Q$ is the transition function, q_0 is the initial state and $Q_m \subseteq Q$ the set of marked states. Based on language theory, $L(A)$ is the language generated by the automaton A written as $L(A) := \{s \in \Sigma^* | \delta(q_0, s)\ is\ defined\}$ and $L_m(A)$ the marked language written as $L_m(A) := \{s \in \Sigma^* \mid \delta(q_0, s) \in Q_m\}$. $L(A)$ represents the set of all possible trajectories - i.e. all possible system behaviors, whereas $L_m(A)$ represents the subset of trajectories, leading to a marked state. Furthermore, in SCT, the set of events Σ is partitioned into two disjoint subsets Σ_c and Σ_{uc} which comprise controllable and uncontrollable events respectively. From the language generated by the process, $L(G)$, and the language generated by the specification, $L(E)$, a supervisor called S is adjoined to restrict the behavior of G in a feedback manner in regard to the behavior expressed by E. Nevertheless, the supervisor S can only forbid the controllable events included in Σ_c. Formally, S is a function defined by $S : L(G) \to 2^{\Sigma}$. If the supervisor S exists, i.e. there exists a supervisor able to sufficiently forbid controllable events to obtain the admissible behavior. Consequently, there exists a model H representing the controlled process. H is a sub-behavior of G as it respects the requirements imposed by E. In this case, H is controllable according to G. Formally, $H := G \times E$. In the opposite, if H is initially not controllable according to G, due to uncontrollable events in Σ_{uc}, the controlled process H will be the most permissive sub-language that respects the requirements and that is controllable according to G. Further discussion has been made in [17] and [18]. As yet, $H := [G \times E]^{\uparrow_c}$ where $\uparrow_c$ is the function that computes the supremal controllable sub-language. To resume, in order to build the controlled process

H, the designer uses the framework illustrated in "Fig. 1". The green book represents the text requirements.

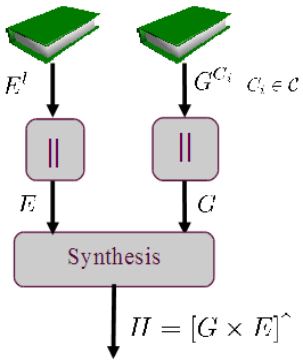

Fig. 1. Framework of the SCT.

Each component C_i in a system is modeled by an automaton G^{C_i}, and all components are included in the set of components $\mathcal{C}$. The global process G is built by parallel composition of all components. Each model E^l represents one requirement. All requirements are synchronized by parallel composition in one model E, this latter represents the global model of specification. From G and E, the controlled process H is then performed.

Different functions exist to manipulate an automaton, and its generated language. One of them is the projection function. This function is performed on strings or languages from a set of events, Σ_i, to a smaller set of events, Σ_j, where $\Sigma_j \subset \Sigma_i$. Formally, the projection is defined as follows:

Definition 1
Let $P : \Sigma_i^* \to \Sigma_j^*$ *such as* $\forall \sigma \in \Sigma_i$ *and* $\forall s \in \Sigma_i^*$*:*

$$P(\varepsilon) = \varepsilon$$
$$P(s\sigma) = \begin{cases} P(s)\sigma \text{ if } \sigma \in \Sigma_j \\ P(s) \quad \text{if } \sigma \in \Sigma_i \backslash \Sigma_j \end{cases}$$

In words, this function takes a language defined over the alphabet Σ_i and erases the events that are not included on the alphabet Σ_j. More properties of the projection function are shown in [19].

III. PROPOSITION

In this paper, a procedure is presented to reconfigure the control law, represented by the controlled process H, when a part of the system and its requirement change. Usually, depending on the design, a little change may occur a complete new design to obtain the new control law, in particular to ensure the requirements that had not been modified are still respected and the updated requirements are well formalized. The proposed procedure uses SCT. The procedure decomposes in five steps as illustrated "Fig. 2".

Procedure 1

- *Identify the part of the system that does not change,*
- *Remove the others part of the control law by the projection function,*
- *Reduce the model by language-equivalent,*
- *Extend the model with the updated behavior of the system*
- *Synthesize the new control law to respect the new or updated requirements.* ♦

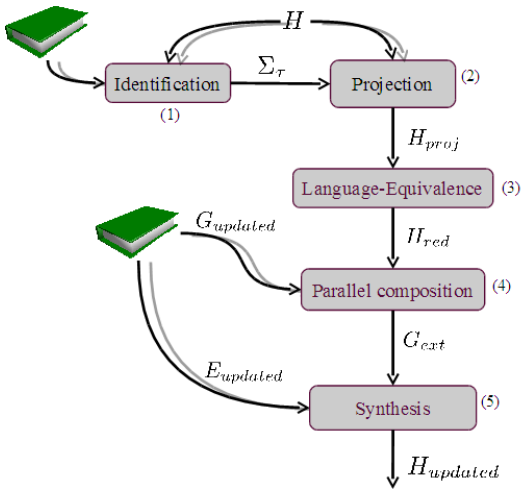

Fig. 2. Proposed procedure to reconfigure control law

From a current control law represented by the controlled process H, the designer identifies the components that preserve their original behavior in the new system, and the components that have to be updated or the new components.

In the first step, the designer has to select components that have the same behavior in the new system and remove the others. In the alphabet Σ, Σ_τ represents the alphabet of events that are generated by the outdated components. Furthermore, the outdated components have to share no events with components which the designer has selected.

The second step removes the behavior generated by the events included in the alphabet Σ_τ by using the projection function. Formally, if H is the controlled process that the designer wants to re-use for a part, the step 2 is performed as follows:

Definition 2

$$P : \Sigma^* \to (\Sigma \backslash \Sigma_\tau)^*$$
$$P[L(H)] = \{\sigma \in (\Sigma \backslash \Sigma_\tau)^* : (\exists s \in L(H))[P(s) = \sigma]\}$$

All events over the alphabet Σ_τ are replaced by an empty string ε. However, the behavior generated by the events included in the alphabet $\Sigma \backslash \Sigma_\tau$ is still there and this behavior keeps respecting the requirements on the system.

Due to the projection function and the empty string ε, the projected language $P[L(H)]$ generated by the automaton H_{proj} may become non-determinist and/or non minimal. In order to reduce the size of the model and transform the non-deterministic automaton into deterministic automaton, the third step performs a language-equivalence reducing. This transformation erases the empty-string of the projected language and checks the minimal number of states to generate the language. More explanation can be found in [9] and [19].

The reduced and determinist automaton, called H_{red} only represents the behavior of the components that preserve the original behavior, including the specifications between these components.

In the fourth step, and from the reduced controlled process H_{red}, the designer extends it with the behavior of the updated or new components. Formally, if the automaton representing the updated behavior is called $G_{updated}$, the extended process G_{ext} is defined as:

$$G_{ext} = H_{red} || G_{updated}$$

Finally, in the last step, the designer performs the synthesis as usual in SCT. From the new process G_{ext} and the updated or new models of specifications $E_{updated}$, the updated controlled process $H_{updates}$ is built. Furthermore, in the case where the model is not controllable, the supremal controllable is performed. Formally, the controlled process is defined by:

$$H_{updated} = [G_{updated} \times E_{updated}]^{\uparrow_c}$$

At the end, the new controlled process $H_{updated}$ represents a model where a control law of the system can be extracted and in which the updated and new components and behaviors are taken into account. The outdated behaviors were removed by the projection function and the language equivalence. Furthermore, thanks to projection function, the behaviors wich the designer wants to keep are not removed and the specifications about these components are still respected. This point is important to reduce the calculability. The designer has only to focus on the new or updated components and their associated specifications.

IV. EXAMPLE

In this section, an example is presented to illustrate an evolved system, and the modification of its control law.

A. System before reconfiguration

The manufacturing system illustrated in "Fig.3.(a)" the system comprises three components and one buffer. The components are used to process a part and the buffer is used as a storage between the components with a maximal capacity of 1. The components C_i are modeled by the automaton denoted G^{C_i} and are shown "Fig.3.(b)" for the component C_1 and "Fig. 3.(c)" for the components C_2 and C_3. The events s_i and e_i represent a new task and the end of the task respectively. Whilst all these events are observable, events s_i and r_1 are controllable whereas e_i and f_1 are not.

The system has two functioning modes. The first one is a nominal mode where only the components C_1 and C_2 are used. However, the component 1 may fail. It is the degraded mode. In this mode, the component C_1 is replaced by the component C_3. This malfunction is modeled with the event f_1 while the repairing is modeled with the event r_1. This change is modeled by the automaton shown "Fig. 3.(e)".

Formally, the global process G is defined by:

$$G = G^{C_1} || G^{C_2} || G^{C_3}$$

and the model of the global specification E is defined by:

$$E = E^{buf} || E^{C_1 \to C_3}$$

The controlled process H is defined by:

$$H = [G \times E]^{\uparrow_c}$$

The controlled process H is illustrated by the figure 6 without dotted and dashed lines. The name of states has been changed to reduced the size of the model.

B. Reconfiguration of the system

A system may change for various reasons. In the proposed system, the designer wants to modify the control law to include the next requirements:

- The component C_2 may now fail;
- The component C_2 will be replaced by a new component C_4 if a malfunction happens.

The evolved system is now illustrated in "Fig. 4.(a)". The updated behavior of the component C_2 is modeled by $G^{C_{2,updated}}$ and is illustrated in "Fig. 4.(b)". The component C_4 is shown "Fig. 4.(c)". The updated requirement is modeled in "Fig. 4.(d)" for the buffer and the new requirement is modeled in "Fig. 4.(e)" for the activation of the component C_4 when a malfunction happens in the component C_2.

1) Identification of useless behavior: In the first step, the designer has to build the set of events Σ_τ that includes all the events generated by the component where the behavior has to be updated. In the example, $\Sigma_\tau = \{s_2, e_2\}$.

Due to SCT, that uses the language theory and automata for modeling, it is not possible to extend automatically the behavior of the component C_2 with the fault event f_2 and the repair event r_2. The reason is the admissible behavior of the controlled process H already takes into account some requirements about the current behavior of the component C_2. Furthermore, extending it manually no more ensures that the controlled process keeps respecting requirement. It is for this reason the mathematical functions are taken to modify the control law.

2) Projection function: The projection is used as written in the definition 2. The automaton used in the projection is the controlled process H. The projected automaton, called H_{proj}, is illustrated in "Fig.5.(a)".

3) Language Equivalence: In order to reduce the model size, and potentially to transform the previous automaton into a determinist automaton, the language equivalence is performed. The result is shown in "Fig.5.(b)".

4) Extension of process: From the reduced controlled process and the models of updated and new components, the designer builds the updated process. Technically, the process $G_{updated}$ is defined by:

$$G_{updated} = G^{C_{2,updated}} || G^{C_4}$$

and

$$G_{ext} = H_{red} || G_{updated}$$

The extended process now includes the previous behavior generated by the components C_1 and C_3, and respects the

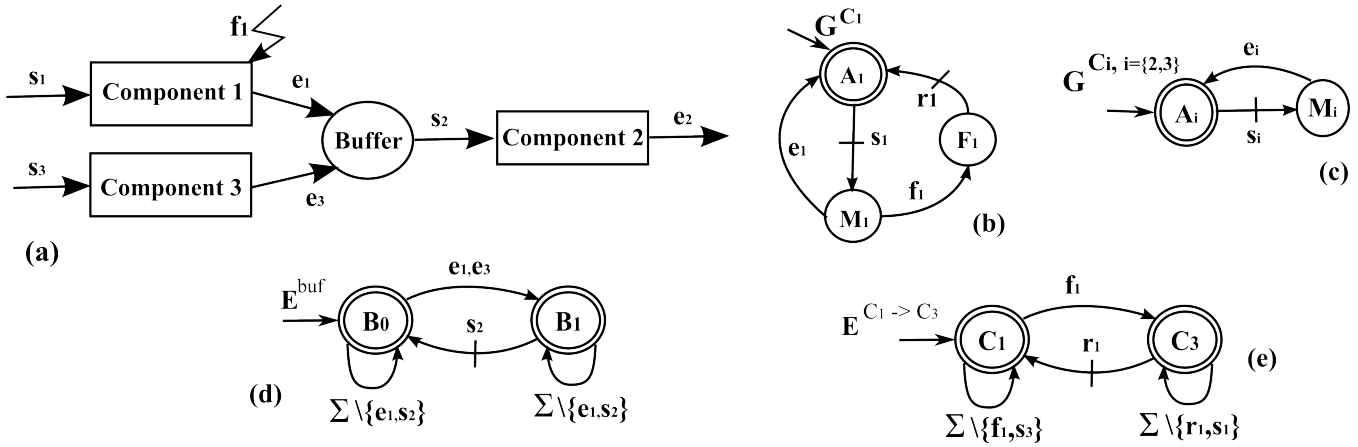

Fig. 3. Manufacturing system example : (a) the studied system; (b,c) process of components C_i; (d) model of the specification of the buffer; (e) model of the specification of the replacement when a malfunction in the component C_1 happens.

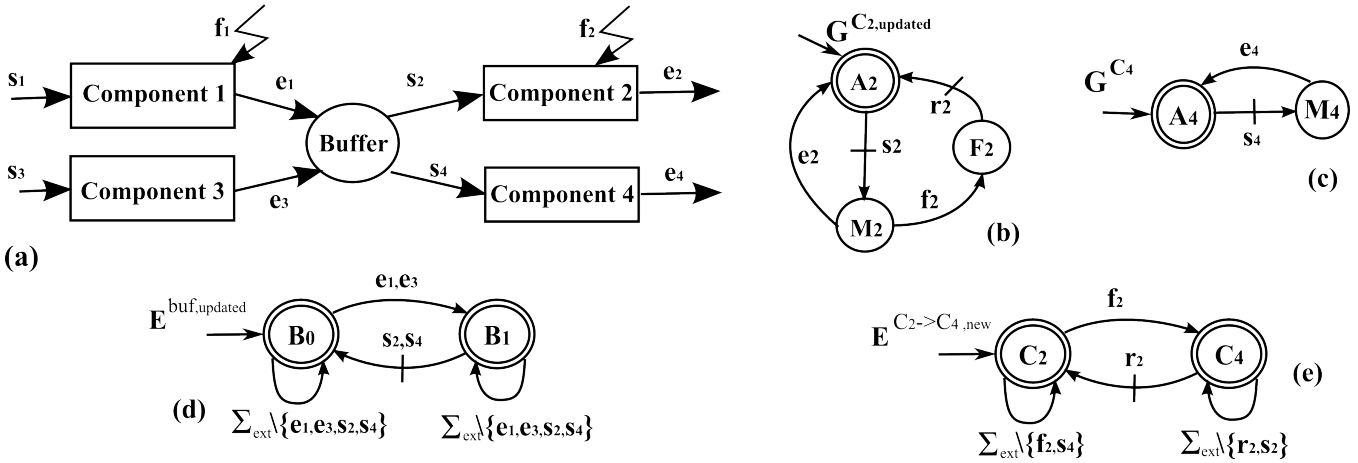

Fig. 4. Manufacturing system example : (a) the evolved system; (b) the updated behavior of the component $G^{C_{2,updated}}$; (c) process of the new component G^{C_4}; (d) model of the specification of the buffer; (e) model of the specification of the replacement when a malfunction in the component C_2 happens.

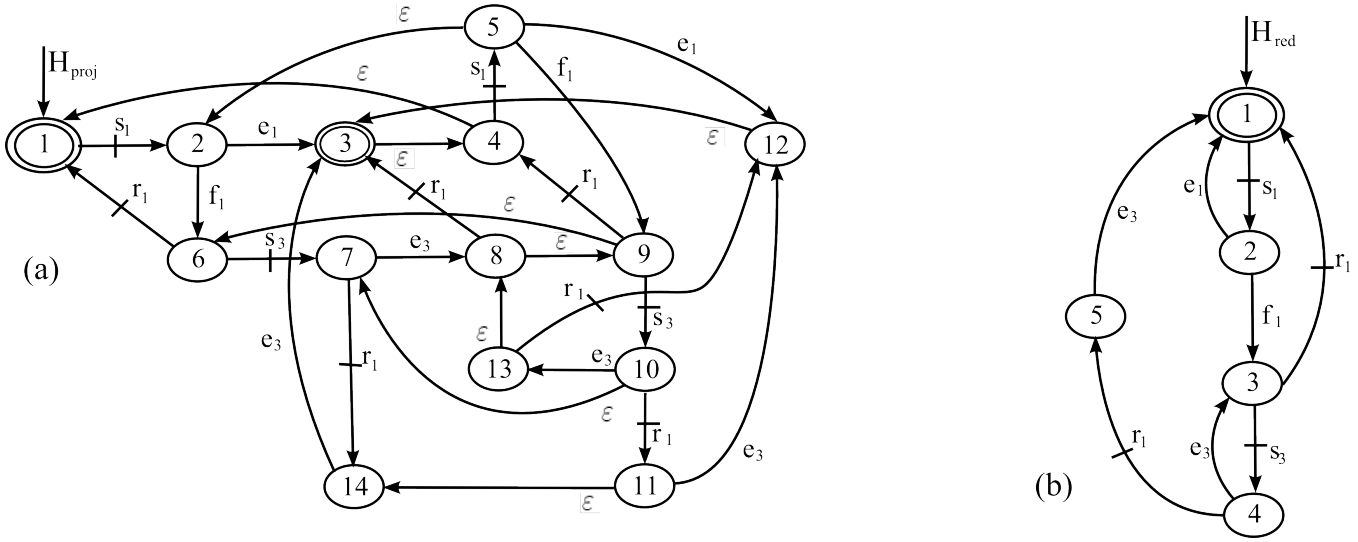

Fig. 5. Manufacturing system example : (a) H_{proj} : the controlled process after the use of the projection function; (b) H_{red} : the projected controlled process after a reduction by language equivalence.

requirement between them. Furthermore, it also includes the updated or new behaviors generated by the components $C_{2,new}$ and C_4

5) Synthesis of the new controlled process: In the same way that the usual framework in SCT, the updated controlled process is performed by synthesis. In the case where the controlled process is not controllable in regards to the process, the supremal controllable sublanguage is computed. The figure 6 illustrates the updated controlled process $H_{updated}$. The complete behavior is obviously larger than the controlled process H (without dotted and dashed lines) and it will be difficult to extend it manually.

V. CONCLUSIONS AND FUTURE WORKS

In this paper, we have proposed a procedure in order to reconfigure the control law only with the updated or new behaviors and their associated requirements. Thanks to SCT, the complete redesign has been avoided and the initial requirements that do not change are ensured to be respected, even after the reconfiguration and without re-verifying them. Then, the SCT is convenient to reconfigure automatically the controller of a system. The future works may focus on two parts; the first one is identifying the conditions to have $H_{updated}$ optimal. Indeed, with SCT, avoid a complete redesign implies the final model is not optimal. The second is about the state space explosion that is one of the problems in SCT to design huge systems. In particular, the procedure proposed in this paper is adapted to be used with a modal standpoint. The goal is to build a new mode wherein the system will be when a malfunction happens. The modal approach reduces the complexity in decomposing the model of the system into number of smaller models, one by mode. Each model of mode respects its own requirements. The requirements may then opposite between modes. Furthermore, adding a new mode to design a reconfiguration is a convenient approach to reduce complexity, design huge system and to modify only a small part of the system.

REFERENCES

[1] K. K. N. Fourlas, G.K.; Kyriakopoulos, "Fault diagnosis of hybrid systems," *Intelligent Control, 2005. Proceedings of the 2005 IEEE International Symposium on, Mediterrean Conference on Control and Automation 2005*, vol. ,, pp. 832–837, 2005.

[2] P. E. Miyagi and L. A. M. Riascos, "Modeling and analysis of fault-tolerant systems for machining operations based on petri nets," *Control Engineering Practice*, vol. 14, pp. 397–408, 2006.

[3] S. Pleisch and A. Schiper, "Approaches to fault-tolerant and transactional mobile agent execution—an algorithmic view," *ACM Comput. Surv.*, vol. 36, no. 3, pp. 219–262, 2004.

[4] Z. Yang and D. Hicks, "Synthesis of robust restructurable/reconfigurable control," in *Control, Automation, Robotics and Vision, 2006. ICARCV '06. 9th International Conference on*, 5-8 2006, pp. 1 –6.

[5] S. Chenhuan Wang; Zad, "Fault recovery in discrete-event systems using observer-based supervisors," in *INDICON, 2005 Annual IEEE*, 2005, pp. 442 – 445.

[6] M. F. Kristin Andersson, Bengt Lennartson, "Synthesis of restart states for manufacturing cell controllers," in *Dependable Control of Discrete Systems*, 2009.

[7] E. Niel, B. Brandin, S. Boukhobza, and M. Nourelfath, "Operational-safety supervisory control: an approach to supervisor activation," in *Emerging Technologies and Factory Automation, 1995. ETFA '95, Proceedings., 1995 INRIA/IEEE Symposium on*, vol. 2, Oct 1995, pp. 553–561 vol.2.

[8] P. J. Ramadge and W. M. Wonham, "The control of discrete event systems," *Proceedings of the IEEE*, vol. 77, no. 1, pp. 81–98, Jan 1989.

[9] W. M. Wonham, "Supervisor control of discrete-event systems ece 1636f/1637s 2009-10," 2009, course notes, departement of Electrical and Computer Engineering, Univeristy of Toronto. [Online]. Available: www.control.toronto.edu/people/profs/wonham/

[10] S. F. L. Yi-Liang Chen, Laortune, "How to reuse supervisors when discrete event system models evolve," vol. 3, Dec 1997, pp. 2964 – 2969 vol.3.

[11] H. Jing Liu, Darabi, "Control reconfiguration of discrete event systems controllers with partial observation," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 34, no. 6, pp. 2262 –2272, Dec. 2004.

[12] R. Sampath, H. Darabi, U. Buy, and L. Jing, "Control reconfiguration of discrete event systems with dynamic control specifications," *Automation Science and Engineering, IEEE Transactions on*, vol. 5, no. 1, pp. 84 –100, Jan. 2008.

[13] S. E. V. A. d. P. M. Silva, D.B., "Application of the supervisory control theory to automated systems of multi-product manufacturing," Sept. 2007, pp. 689 –696.

[14] O. Kamach, L. Pietrac, and E. Niel, "Design of switching supervisors for reactive class discrete event systems," *Information Control Problems in Manufacturing 2006*, vol. 12, no. 1, pp. 289–294, 2006.

[15] G. Faraut, L. Piétrac, and E. Niel, "A new framework for mode switching in sct," in *European Control Conference 2009 - ECC09*, August 2009.

[16] G. Faraut, L. Pietrac, and E. Niel, "Formal approach to multimodal control design: Application to mode switching," *Industrial Informatics, IEEE Transactions on*, vol. 5, no. 4, pp. 443–453, Nov. 2009.

[17] W. M. Wonham and P. J. Ramadge, "On the supremal controllable sublanguage of a given language," *SIAM Journal of Control and Optimization*, vol. 25, no. 3, pp. 637–659, 1987.

[18] R. Kumar, V. K. Garg, and S. I. Marcus, "On controllability and normality of discrete event dynamical systems," *Systems and Control Letters*, vol. 17, no. 3, pp. 157–168, 1991. [Online]. Available: http://home.eng.iastate.edu/ rkumar/

[19] C. G. Cassandras and S. Lafortune, *Introduction to discrete event systems [Second Edition]*, C. G. Cassandras and S. Lafortune, Eds. Springer, 2007.

Fig. 6. Controlled process H : figure without dotted-lines; Updated controlled process $H_{updated}$: complete figure. The dotted and dashed lines represent the part of behavior that has been added by the updated or new behaviors and specifications. Precisely, the dashed-lines represent the event f_2 and where the updated behavior starts.