Proceedings of the 17th World Congress
The International Federation of Automatic Control
Seoul, Korea, July 6-11, 2008

A Component Based Architecture for the Reconfiguration of Hybrid Systems Using Control Description

Ahmed Guadri * **Nathalie Dangoumau** ** **Etienne Craye** ***

* *Ecole Centrale de Lille, Villeneuve d'Ascq, 59650 France (e-mail : Ahmed.Guadri@ec-lille.fr)*
** *Ecole Centrale de Lille, Villeneuve d'Ascq, 59650 France (e-mail : Nathalie.dangoumau@ec-lille.fr)*
*** *Ecole Centrale de Lille, Villeneuve d'Ascq, 59650 France (e-mail : Etienne.Craye@ec-lille.fr)*

Abstract: Control systems are required to satisfy increasingly severe safety and performance criteria (reliability, security$\cdots$) mainly in the case of large scale systems. Such systems have to be described with a multiple abstraction levels paradigms such as hybrid systems (implementing both discrete and continuous dynamics). We present and formalize a new strategy for the reconfiguration of a hybrid system that relies on the specification of a region-based component-oriented model which is updated whenever the system dynamics are modified. The standard controller is modified whenever a fault is detected and identified by use of an objectives automaton : Failing standard controls are replaced by a new sequence of controls chosen from an input space. Finally, we illustrate this method in the case of a system of communicating tanks.

Keywords: Active approaches to fault tolerant control, Hybrid systems modeling and control

1. INTRODUCTION

Today, fault tolerant control and design offers many challenges such as the analysis complexity which prevents optimal exploitation of the dynamics of the analyzed system. Therefore, abstraction approaches may be used in order to allow the reconfiguration of a control system depending on the system dynamics evolution and objectives specifications. Such theories has been developed in Manz and Gohner [2002] to describe qualitative models of each component, so that a component-oriented model could be built in order to solve monitoring problems. This abstraction is made through a transition graph describing the evolution of the qualitative state of each component. An analog methodology is described in Su et al. [2003]. The use of supervisory theory for the control of hybrid systems has been adopted in Koutsoukos et al. [2000] with the adoption of an interface as a medium device between the process and the controller (Stiver et al. [1995]). Analog methodologies are exposed in Alur [2000], Aström et al. [2001]. In contrast, it may be necessary to use a hierarchical modeling in order to simplify the task of control and reconfiguration such as in Lorimier [2005] in the case of large scale systems.

In this article we adopt a component-oriented hybrid modeling for the description of a hybrid system where each component is described with an hybrid formalism (presenting both continuous and discrete evolutions). The proposed architecture is depicted in Figure 1 and consists of three main parts : The hybrid system, the control modules and the reconfiguration modules. This choice allows the development of efficient reconfiguration strategies that benefit from the exhaustive description of components dynamics.

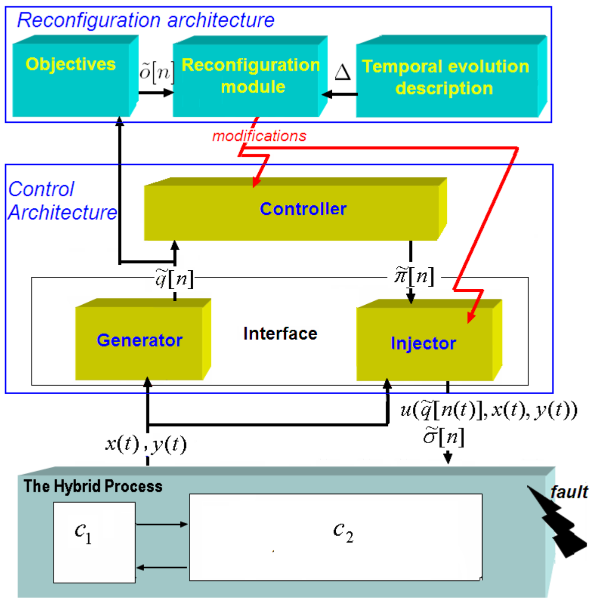

Fig. 1. The control and reconfiguration architecture

2. THE TWO TANKS PROBLEM

In order to illustrate this méthodology, we use a simple example which is represented in Figure 2 and is composed

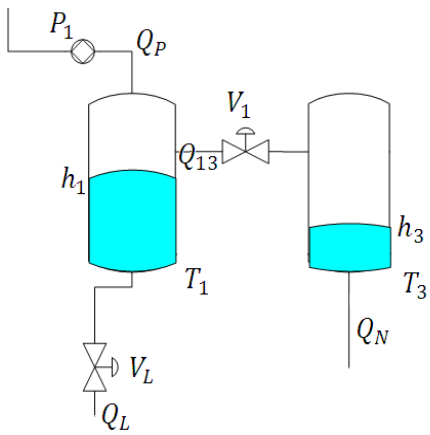

Fig. 2. Two tanks system

of two coupled tanks T_1 and T_3. T_3 is a tank that is dedicated to deliver a *suitable* flow of water to another complex system which is not represented. In practice, this flow could be provided if the fluid level in T_3 is kept in a medium interval (i.e. the level in T_3 is between 0.09 m and 0.11 m). T_3 is supplied by water flow from T_1 through the controlled valve V_1, and T_1 is filled by the controlled pump P_1. While V_1 can only be completely opened or completely closed, P_1 is controlled through a PI regulator according to an injected level. Also, a valve V_L is used to simulate a leakage in T_1. Let h_1 and h_3 be the water levels in T_1 and T_3, Q_P the water flow from the pump P_1, Q_{13} the flow from T_1 to T_3 and Q_L the flow of the lost water from T_1 in case of leakage. Finally, Q_N is the water flow delivered from T_3 to the consumer.

3. HYBRID SYSTEM MODELLING

The hybrid system S is a complex hybrid system composed of communicating components $\{c_1, ..., c_{|S|}\}$. Each component c_i is described with two automata : The behaviour automaton $(ci.A)$ and the fault automaton $(c_i.A_{Fi})$.

Notations In this paragraph, we introduce some notations that are used in this section. First, the cardinality of a set E or the number of lines in a vector v are respectively $|E|$ and $|v|$, whereas the absolute value function will be noted by $abs(x)$, $x \in \Re$. Also, we adopt the dot notation in order to describe membership.

In addition, two different temporal scales are adopted : The ordinary continuous time t and the step index n. $n(t_1)$ is the step index when $t = t_1$, and $t[n_1]$ is the moment of transition from $n_1 - 1^{th}$ to the n_1^{th} step. Recall that $\forall t \in (t[n_1], t[n_1 + 1]), n[t] = n_1$

3.1 The behavior automaton

Each component dynamics is described through a variant of hybrid automaton and possesses a number of algebraic shared continuous variables with the other components. Each state is associated with some dynamic continuous variables specifications (through ordinary differential equations) and shared variables (through algebraic continuous variables) evolution. Let $c_i.x$ be the continuous dynamic variable and $c_i.y$ the internal shared variables of c_i. Also, notice that $c_i.y$ is composed of a non modifiable read only variable $c_i.y_{read}$, modifiable read/write variables $c_i.y_{write}$ and control continuous inputs $c_i.u$. In addition, crossing a transition is detected when a guard function h is strictly positive and a control event σ is triggered. In that case, the shared read/write variables are updated with a reset function. Let H be the set of all detected hypersurfaces in the system components and Σ the set of control events.

Furthermore, we adopt $Z_i = X_i \times Y_i$ as a phase space, and $z_i = \begin{pmatrix} x \\ y \end{pmatrix}$ as the relative variable. Also, $\sigma(t[n])$ is the event which was detected at $t[n]$ (When $t[n] < t < t[n+1]$ we have always $\sigma(t) = \epsilon$). At n, $c_i.y_{write} := reinit(z_i)$.

Component behavior evolution At each step n and for every component $c_i \in S$ we define the behavior evolution as :

Definition 1. $\forall t \in]t[n], t[n+1]]$ and suppose $\tilde{q}_i[n] = q$. The continuous evolution of the internal dynamical variables of c_i is defined by :

$$\dot{x}_i(t) = f_{i,j}(x_i, y_i) = f_i(q_j, x_i, y_i)$$
$$y_i.write(t) = y_i.write(t, n) = g_{i,j}(x_i, y_i)$$

A transition from the state q to a state q' is crossed according to the definition 2.

Definition 2. Let $\tilde{q}_i[n] = q_{src}$. If $\exists e \in E_i$ such as $e =< q_{src}, q_{dest}, h, \sigma, reinit >$ and $\exists \tau \geq 0$ such that :

- $h(z_i(t[n] + \tau)) > 0$,
- $\exists \eta_0 > 0 / \forall \eta < \eta_0, n(t[n] + \tau - \eta) = n$
- $\sigma(t[n]) + \tau) = \sigma$

then e is crossed :

$$t[n + 1] = t[n] + \tau$$
$$z_i(t[n + 1]) = reinit(z_i(t[n] + \tau))$$

Notice that a trajectory which is tangent to $\{h(z(t[n] + \tau)) = 0\}$ without crossing it does not triggers the transition crossing. In practice

3.2 The fault automaton

The fault automaton is used in order to give the component qualitative state regarding faults at each moment.

Definition 3. $\forall c_i \in S$ we define a fault automaton $A_{Fi} =< Q_{Fi}, H, F_{i,0}, \delta_{Fi}, F_i >$ with Q_{Fi} : Set of automaton states, H : Set of symbols referring to detected hypersurfaces (see next section), $F_{i,0}$ is the automaton initial mode (which is the safe mode). δ_{Fi} is the transition function. Moreover, the variable $F_i(t) = c_i.F(t)$ is the current fault mode at t. Also, if $t[n] < t < n(t) + 1$, then we can simply note $F_i(t) = \tilde{F}_i[n]$. Finally, $F(t) = \tilde{F}[n] =< F_1(t), ..., F_{|S|}(t) >$, $Q_F = Q_{F1} \times ... \times Q_{Fn}$

4. CONTROL ARCHITECTURE

The control architecture consists of the controller and the interface (injector and generator). This modeling allows logic differentiation between the controller which is a discrete event system and the hybrid process thanks to the interface (see Figure 1.

4.1 The Interface

This part is composed of two modules : The injector that converts the controller symbols into a continuous and event inputs to the hybrid process and the generator which informs the controller about the qualitative changes (deduced from the crossed hypersurfaces) of the system components.

These hypersurfaces are chosen depending on physical constraints (for example a hypersurface that separates two different dynamics) or objectives requirements (for example, if an objective is to maintain the water level in T_3 between 0.09 m and 0.11 m, the generator must detect whether the water level is below or above these levels. So the component behaviour states must differentiate these regions).

The symbols returned by the generator describe the evolution of the system behaviour and faulty modes. We define a system state at t as $q(t) =< c_1.q(t), ..., c_{|S|}.q(t)$, $c_1.F(t), ..., c_{|S|}.F(t) >$. Let Q be the set of all system modes and $R = 2^Q$.

The injector converts the controller symbolic entries into continuous and event inputs to the process. For each component c, there is a finite basis $c.W = \{w_1, ..., w_{|W|}\}$ of continuous functions pondered by coefficients vector $c.A(\pi) = \begin{pmatrix} c.\tilde{\alpha}_1(\pi) \\ ... \\ c.\tilde{\alpha}_{|W|}(\pi) \end{pmatrix}$ which are chosen according to the controler current state $\tilde{\pi}[n]$. For each component, we define the admissible continuous control u as the combination of $c.W$ that is pondered with $A(\pi)$ such as $c.u(t) = \sum_{i=1}^{|c.W|} c.\alpha_i(\tilde{\pi}[n(t)])w_i(t), N(c.u) \leq 0$. In addition to continuous inputs, the injector delivers a control event $\sigma(\pi[n])$ in the beginning of each step according to the controller state $\pi[n]$.

4.2 The controller

Definition 4. The controller is a deterministic finite automaton $\Pi =< Q_\Pi, R, \delta, \pi_0 >$ with Q_Π : The set of controller states, R : Set of regions of Q. δ is the transition function and π_0 is the initial controller state. $\pi(t)$ ($\tilde{\pi}[n]$) is the controller state at t (respectively step n).

The controller automaton evolves according to the system state which is returned by the generator at each step, whereas the injector interprets the controller state to select a hybrid control for the process. Thereafter, we admit that the system process coupled with the standard control architecture always meet the objectives specifications until a fault is detected.

4.3 Controller and Injector for the Two Tanks System

We choose the continuous input base $W = \{1\}$, with 1 is the function returning 1 for all t. This input base is relative to the water level reference for the PI regulator of the tank T_1. The continuous input must verify $\|w\|_\infty = \max_t |w(t)| = \max_t \{A(t) \times W(t)\} < 0.6$.

As P_1 is controlled by a PI regulator, a constant input through w causes an oscillation around the new level reference. For example, if the tank T_1 is initially empty, injecting $\alpha = 0.5$ causes an overflow in T_1. So, we choose to make an initial control reference to bring the water closer to 0.5 ($\alpha = 0.35$ until the end of the first oscillation). After that, the control architecture starts by injecting a reference level of 0.5 to the pump, while the valve V_1 is closed. When the tank T_1 is sufficiently filled, V_1 is opened. Finally, the control architecture starts a loop (opening V_1, closing V_1 while maintaining the level reference as 0.5). The standard controller and injector iare given by Figure 3

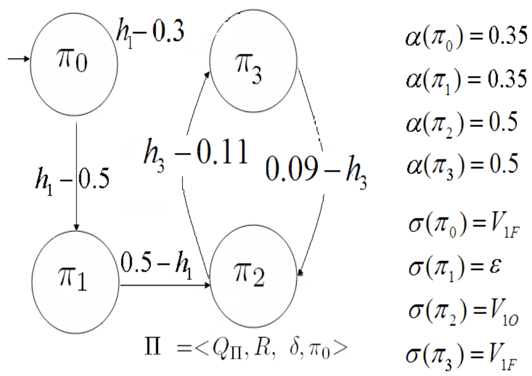

Fig. 3. The standard controller and injector for the two tanks system

5. RECONFIGURATION ARCHITECTURE

We assume that the controlled hybrid system (control architecture + hybrid process) will behave according to the objectives specifications until a fault occurs and causes a failure. In this case, the reconfiguration module uses the description database in order to update the controller and injector modules. In this section, we start by formalizing the objectives specifications through a state transition automaton. After that, we give the reconfiguration algorithms intended to compute alternative controls sequence whenever a failure is detected. This sequence is exploited in order to update the controller so that the system achieves its objectives.

5.1 Objectives automaton

At each step, the objectives automaton defines the acceptable destination of the next system state transition thanks to the automaton defined in Definition 5

Definition 5. the objectives automaton is a non deterministic finite automaton $A_O :< O, R, \delta_O, o_0, inv >$ with O : The set of fault modes. δ_O is the transition function. o_0 is the initial state. Finally, inv is a function that associates to each $o \in O$ a region $o.inv \in R$.

R is the set of system modes (a system state is a composition of component behaviour and faulty modes). In addition, $o.T_{max}$ is the maximal duration of activation for each state $o \in O$(which could be $+\infty$). Also, we define $o(t)$ ($\tilde{o}[n]$) as the current state at the instant t (step n).

According to Figure 4, the system must evolve into the invariant of o_1 in a finite duration, than it must oscillate around this region. $o_1.inv$ is the set of all system

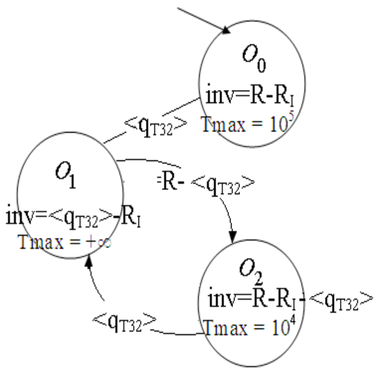

Fig. 4. The objectives automaton

states such that T_3 is in the state $< q_{T32} >$, with $q_{T32}.inv = \{0.9 \leq h_3 \leq 0.11\}$, whereas $o_2.inv$ is the system allowed states outside $q_{T32}.inv$. We choose $R_I =< q_{VLO}, q_{T34}, q_{T14} >$, ($q_{VLO}$ corresponds to the valve V_L opened, whereas q_{T34} and q_{T14} respectively corresponds to $h_3 > 0.3$ and $h_1 > 0.55$ (overflow risk)). Notice that the control architecture of Figure 3 aims at keeping the system state in $< q_{T32} >$ (the water level h_3 in $[0.09, 0.11]$) by a loop increase/decrease that causes a very brief objectives automaton transitions to o_2.

Suppose that the objectives automaton state at n is o ($\tilde{o}[n] = o$). If the controlled hybrid process respects the objectives specifications, the next system state must not be in R_I . Moreover, it must stay in its current state or reach one of the following objectives automaton states invariants. Formally, let R_{acc} be the set of system states where the system must evolve from its current state.

Definition 6. $R_{acc}(o) = \bigcup\{o'.inv$ such that $o \in O, \exists r \in R/\delta_O(o, r) = o'\}$

When a failure is detected, the first case is that the system is not in a prohibited mode ($\tilde{q}[n] \notin R_I$), then the reconfiguration control sequence must bring the system into R_{acc} without crossing R_I. In the second case ($\tilde{q}[n] \in R_I$), it may be impossible to reconfigure the system architecture without crossing other prohibited regions (For example, if we assume that the tank T_3 must not be filled more than 0.5 m, and if a failing control resulted in an overflow, it is impossible to return to $h3 < 0.11$ without crossing the prohibited region).

5.2 Failure detection

The reconfiguration is triggered whenever the system behaviour does not respect the objectives automaton evolution. In practice, such a condition is met when the system does not evolve into the awaited modes in an acceptable time, or when the system mode crosses a prohibited region.

Condition 1. The reconfiguration algorithm is triggered at t if $t - t[n(t)] \geq \tilde{o}[n(t)].T_{max}$ is detected or the system state $\tilde{q}[n]$ is in a prohibited mode from R_I

5.3 Input descriptions

This module contains informations about the qualitative the system evolution according to the current input. Its construction is performed by offline analysis of the whole system. In practice, this analysis is achieved by partitioning the continuous input according of each component according to its behavioural model and to its environment. However, if some dynamics are not fully understood, then we adopt a conservative description by describing only the fully understood evolutions, or we making an approximation of the system usual evolution. While allowing more flexibility for controller modification with a much simpler modeling, the latter option does not ensure the system qualitative evolution as expected. Nonetheless, this method could be acceptable if the nonlinearity degree is high and the transitory phases are limited.

In order to achieve objectives specifications, these informations are stored in a database containing descriptions of the components evolution depending on the hybrid entries coefficients A and the environing components modes. In practice, the description is given for each component c with a function $\Delta(r_{src}, r_{dest}, r_{neighb}, \alpha, \sigma)$ that returns an upper and lower bound (Δ^- and Δ^+) for reaching r_{dest} from r_{src} depending on the component environment state r_{neighb}. We assume that $\Delta^- = +\infty$ means that using the continuous input guarantees that the system does not evolve to r_{dest}. However $\Delta^- = 0$ describes an instantaneous evolution.

The offline analysis of each component may be achieved through exhaustive simulation (because it is easier to analyze a component behaviour rather than the whole system evolution) or dynamic approximation.

The pump P_1 descriptions The component P_1 is represented in Figure 5. This component possesses a simple descriptions as the automaton transitions are depending on hypersurfaces that depends only on component continuous control w. Thereafter, we adopt a partition $PU_{T1} = \{U_{11} = [0, 0.3[, U_{12} = [0.3, 0.4[, U_{13} = [0.4, 0.55[, U_{14} = [0.55, +\infty[\}$ be the P_1 continuous input space partition. The reason for introducing four states is that each one is associated to an interval for the pump reference level which enables some evolutions in the neighbour components.

For example, one of the descriptions of P_1 is : $\Delta(< q_{P11} >, < qP_{12} >, R, U_{12}, \epsilon) = (0, 0)$. This means that once the pump P_1 is in q_{P11} (w in [0, 0.3[), injecting w from $U_{12} = [0.3, 0.4[$ causes the component to move iummediately to $< qP_{12} >$ (w in [0.3, 0.4[), whatever the environing components modes are. For P_1, all the described evolutions are instantaneous ($\Delta = (0, 0)$) because the continuous injected input is affected directly to w_1.

5.4 Reconfiguration strategy

Whenever the controlled hybrid process fails to meet the system objectives (If the control architecture is well defined, this can not be possible unless a fault occurs) The reconfiguration module starts the reconfiguration task by trying to replace the associated hybrid control by a new hybrid controls sequence in order to reach one of the next objectives invariants or to return to the current objective invariant.

Computation of a new controls sequence Assume that at the step n, a fault in component $c_i$that caused a failure is detected (The fault automaton of c_i has evolved into

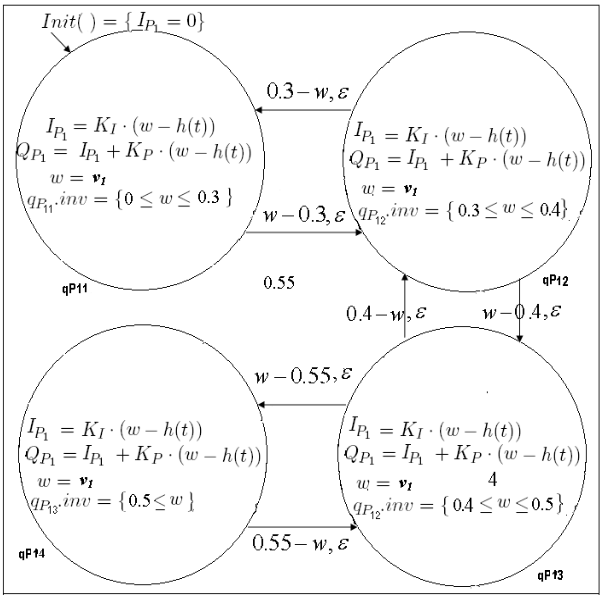

Fig. 5. bahaviour automaton of P_1

a non safe state). Let $\tilde{\pi}[n], \tilde{q}[n], \tilde{o}[n]$ be respectively the current states of the controller automaton, the objectives automaton and the hybrid process at n.

Definition 7. A sequence of hybrid controls sequence γ is defined as :

$$\gamma = \begin{pmatrix} V_1 & \sigma_1 & h_1 \\ V_2 & \sigma_2 & h_2 \\ \dots & \dots & \dots \\ V_m & \sigma_m & h_m \end{pmatrix}$$

with $V_j \in PU$ a set from the functional continuous input space, $\sigma_j \in \Sigma$ and $h_j \in H$ an hyperpsurface from Z.

The signification of injecting the hybrid sequence γ at n is : At step $n + j$, the injector uses a continuous control from V_j until the hypersurface $\{h_j = 0\}$ is crossed in the increasing direction. At that instant, the control event σ_j is delivered.

The computation of the new control sequence aims at finding the qualitative continuous inputs V_j and control events σ_j, whereas the effective controls are computed progressively with the system evolution. The idea of the hybrid control sequence generation is to find, thanks to the description database, the admissible evolutions from the current system mode to reach one of the following objective modes invariants, or to return to the current objective mode invariant within the specified duration. An important criterion for the sequence generation is the respect of the temporal evolution specification given by the objectives automaton. Controls sequence generation is achieved according to the algorithm 1.

Algorithm 1. (1) Starting from the current system mode $\tilde{q}[n]$ and objective mode $\tilde{o}[n]$, let $index = 0$ and the initial reached state $\tilde{q}[n]$ (The current system mode). Go to 2.

(2) make an inventory of all possible evolutions in step $n + index + 1$ according to the temporal description database from the last reached system states. Go to 3.

(3)
- If there is a reached region that intersects with R_{acc} or $\tilde{o}[n].inv$ within an acceptable duration, then the associated hybrid control sequence is adopted. Algorithm stopped.
- If all the last reached regions require temporal duration higher than the maximal allowed duration associated to the current objective mode, than choose the hybrid control sequence that minimizes the described duration in order to reach R_{acc}. If such a sequence exists, the algorithm is stopped. Else, go to 2.

(4) index=index+1

A component c evolution from a state r_1 to state r_2 is interpreted as possible if there is a control set from $\hat{U} \in PU$ and a region r such that the neighborhood components remains in r thanks to $\hat{U}$ (with a description $\Delta^- = +\infty$) and $\exists \hat{U}', \sigma$ such that $\Delta(r_1, r_2, r, hatU', \sigma) < +\infty$.

Global reconfiguration algorithm The generation of the hybrid control sequence is incorporated in a more general control and reconfiguration algorithme (algorithm 2).

Algorithm 2. Assume that a well defined control architecture and continuous input partition is available for the hybrid process S according to the objectives automaton A_O.

(1) Use the standard control architecture until step n when a fault is detected. Then move to 2.
(2) According to the algorithm 1, generate a new hybrid control sequence $\gamma^* = \begin{pmatrix} V_1^* & \sigma_1^* & h_1^* \\ V_2^* & \sigma_2^* & h_2^* \\ \dots & \dots & \dots \\ V_m^* & \sigma_m^* & h_m^* \end{pmatrix}$. Go to 3.
(3) update the controller automaton and the injector in order to incorporate hybrid controls according to γ^* as the system mode evolves. This is done by associating to each hybrid control $(V_j^*, \sigma_j^*, h_j^*)$ a new control state that is interpreted by the injector by injecting a continuous input from V_j^* and a control event σ_j^* when $h_j^* > 0$ is detected.

In order that the reconfiguration task succeeds, the system dynamics must not be radically affected by the fault. In fact, updating the controller and the interface modules could be very fast when only some components are faulty : The initial controller is modified in order to respect the objectives automaton without modifying the global control architecture. However, a radical change of system structure and behaviour may need a new controller synthesis.

6. APPLICATION TO THE 2 TANKS SYSTEM

Suppose the following scenario : The control architecture performs its mission until reaching π_1. Assume now that the valve V_1 is blocked opened at this step. Triggering $V1F$ fails to close the valve V_1. Then, a failure will be detected after some duration. Let n be the step of failure detection, and the controller and objectives automata are respectively $\tilde{\pi}[n] = \pi_3$ and $\tilde{o}[n] = o_2$.

Depending on the available descriptions of the components evolution, the reconfiguration module explores the reachable system modes. In fact, when valve V_1 is blocked

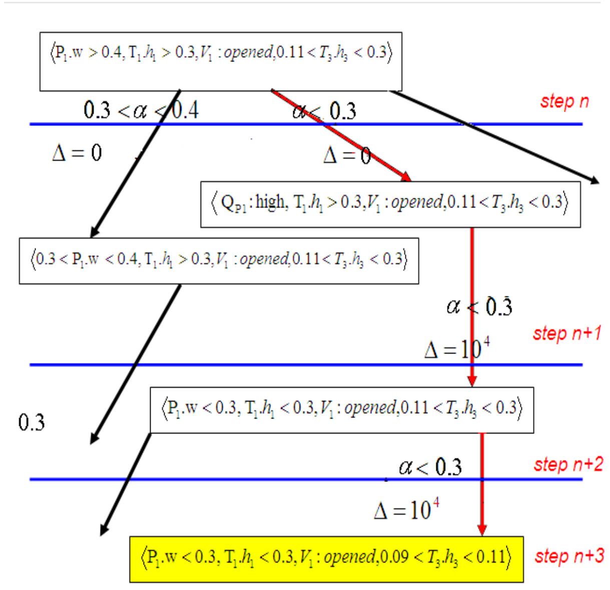

Fig. 6. A part from the controls sequence generation

opened, the control events V_{1O} and V_{1F} does not affect the system evolution. Then the only controls that could be used are the continuous inputs from $PU_{T1} = \{U_{11} = [0, 0.3[, U_{12} = [0.3, 0.4[, U_{13} = [0.4, 0.55[, U_{14} = [0.55, +\infty[\}$. As the current objective mode is $\tilde{o}[n] = o_2$, then the acceptation set is $R_{acc} = o_1.inv =< q_{T32} > -R_I$.

Figure 6 gives a part of the modes exploration graph that is the result of the algorithm 1. An acceptable sequence that leads to R_{acc} is to use a level refernce for P_1 such that $\alpha \times 1 \in [0, 0.3[$ (with 1 the basis element defined in the section 4.3, and α the relative coefficient). As the evolution description does not depend on the coefficient α, then we could choose an arbitrary level reference $\alpha^* = 0.2$. It may be possible to take a precise input coefficient if we add other restrictions to the objectives specifications. It is important to remark that returning to the invariant of o_2 because it is impossible the specified duration for staying in o_2 has been violated.

The generated hybrid control sequence is then used to update the controller automaton and the injector according to the algorithm 2. The injector is updated by assigning a continuous input $w = \alpha * 1 = 0.2$. The resulting controller and interface are depicted in Figure 7. Notice that the main difference with the original control architecture is the interpretation of controller states in order to regulate the process.

7. CONCLUSION AND FUTURE WORK

Although this approach succeeds in reconfiguring the hybrid control strategy, we notice that it's efficiency hugely depends on the initial modeling. Also, reconfiguring the system requires the handling of each component state at each step. This leads to a great increase of the algorithm complexity in the case of large scale complex systems. Therefore, a next step is to extend our methodology to exploit the uncertain informations that are delivered by the FDI modules. Also, we plan to formalize a methodology of abstracting sets of components into a single subsystem

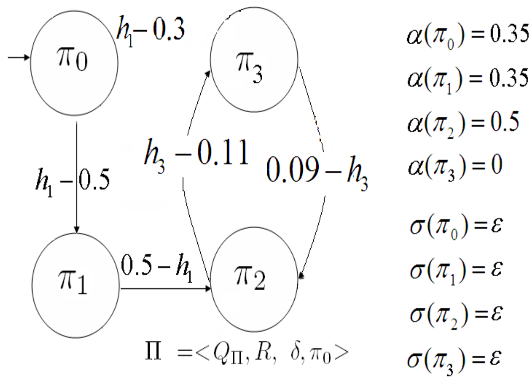

Fig. 7. The updated controller and Injector

that could be treated like an elementary component by the reconfiguration module. This would allow the construction of fault tolerant and hierarchical large scale hybrid systems.

REFERENCES

Rajeev Alur. Discrete abstractions of hybrid systems. *Proceedings of the IEEE*, 88(7):971–984, 2000.
Karl Aström, Pedro Albertos, Morgens Blanke, Alberto Isidori, Walter Schaulelberger, and Ricardo Sanz. *Control of Complex systems*. Springer - Verlag, 2001.
X.D. Koutsoukos, P. J. Antsaklis, J.A. Stiver, and M.D. Lemmon. Supervisory control of hybrid systems. *Proceedings of the IEEE*, 88:1026–1049, July 2000.
Lionel Lorimier. *La caractérisation dynamique des défaillances, Une nouvelle approche pour la gestion active des défaillances au sein des systèmes physiques industriels complexes*. PhD thesis, Ecole Centrale de Lille et Université des Sciences et Technologies de Lille, 2005.
Suzanne Manz and Peter Gohner. *Development of Hybrid Component Models for Online Monitoring of Complex Dynamic Systems*, volume 279/2002, chapter 22, pages 391–418. Springer Berlin/Heidelberg, 2002.
James A. Stiver, Panos J. Antsaklis, and Michael D. Lemmon. Interface and controller design for hybrid control systems. In *Hybrid Systems II*, pages 462–492, London, UK, 1995. Springer-Verlag. ISBN 3-540-60472-3.
Rong Su, Sherif Abdelwahed, Gabor Karsai, and Gautam Biswas. Discrete abstractions and supervisory control of switching systems. *IEEE International Conference on Systems, Man and Cybernetics*, 1:415–421, October 2003.