

Designing operational control architectures of critical systems by reachability analysis

Thibault Lemattre, Bruno Denis, Jean-Marc Faure, Jean-François Pétin and Patrick Salaün

Abstract—This paper presents a method that eases the design of the operational architecture of a control system by providing, from the knowledge of some characteristics of the functions that the control system must ensure and a generic model of controller, an assignment solution of these functions that satisfies capabilities and distribution constraints. This method relies on the verification of a reachability property on a network of communicating automata that models the assignment process. The benefit of this proposal is illustrated by a non-trivial case study from industry.

I. INTRODUCTION

Whatever the process, the operational control architecture of an industrial process is composed of several controllers (PLCs or real-time industrial computers) which gather information from the process through sensors, execute the functions that are required to control correctly the process (physical parameters regulation, tasks synchronization, operation modes management,) and generate signals towards the process. A controller is then a physical device that permits to implement one or several control functions by providing the necessary hardware resources and in particular logic and analogic interfaces from/to the process.

Moreover, for critical processes like for instance power production and distribution, several kinds of controllers with different integrity levels are to be used to build the operational architecture. Safety-related functions must indeed be implemented in controllers with a high integrity level, with several internal redundancies, while implementation of less critical functions requires controllers with lower integrity levels.

Design of the operational control architecture of a critical process requires then to assign all the necessary control functions to controllers, while satisfying constraints that are related to:

- the capabilities (numbers of the different kinds of resources) of the controllers;
- the distribution of functions, because two functions may or may not be combined into a single controller according to their safety levels.

In the current industrial practice, the assignment of functions to controllers is performed in a non-automated way,

T. Lemattre, B. Denis, J.-M. Faure are with LURPA, ENS Cachan, 61 av. du President Wilson, 94235 Cachan Cedex, France {lemattre,denis,faure}@lurpa.ens-cachan.fr

T. Lemattre, P. Salaun are with Electricite de France, Recherche & Developpement, 6 quai Watier, 78400 Chatou, France {thibault.lemattre,patrick.salaun}@edf.fr

J.-F. Petin is with CRAN, UMR 7039 CNRS,Nancy Université, BP 70239, 54506 Vandoeuvre les Nancy, France jean-francois.petin@cran.uhp-nancy.fr

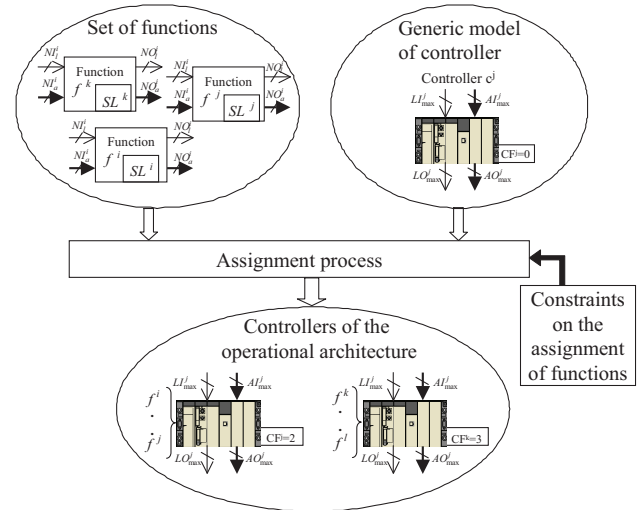


Fig. 1. Aim of the study

based on the expertise of designers, and is a tedious and time-consuming task. This paper aims to address this problem by proposing a method (Figure 1) which provides, from the knowledge of the characteristics of the control functions to implement and a generic model of controller, an assignment solution respecting the capabilities and distribution constraints.

This proposal is based on the verification of a reachability property in a discrete state space. Such an approach has been already used successfully in [1] and [2]. This contribution differs from these two references because time is not considered in an assignment problem. It should be also emphasized that this work constitutes a prerequisite for the evaluation of the time performances of the operational architecture. The results obtained in this field by using simulation techniques ([3], [4], [5]), formal verification [6], or algebraic approaches [7], assume indeed that the processing times of the controllers are known, which of course requires that all functions have been previously assigned.

The paper is organized as follows. The elements of the problem (control functions, controllers, and assignment constraints) are formally defined in Section II. Section III describes the way to obtain automatically an assignment solution by reachability analysis, while the implementation of this principle by using a formal verification tool is detailed in Section IV. The value of this contribution is shown in Section V thanks to two cases studies: a simple case with an

illustrative purpose, and a larger one to emphasize scalability. Prospects for further work are given in Section VI.

II. MODELLING OF THE ELEMENTS OF THE PROBLEM

This section aims at presenting all the notations used and at illustrating the assignment process on a toy example. The following assumptions are retained for this study:

- a controller can host between 0 to n functions, as long as the assignment constraints are met;
- a function must be assigned to one and only one controller;
- a function cannot be decomposed into smaller ones;
- there is no hierarchy of functions.

A. Notations used

Let F be the set of functions f^i , with $i \in \{1, \dots, L\}$ and C the set of controllers c^j , with $j \in \{1, \dots, M\}$, L and $M \in \mathbb{N}^*$. The assignment of the function f^i to the controller c^j will be denoted by $c^j \leftarrow f^i$.

1) *Formal definition of a function:* A function $f^i \in F$ is defined as a 5-tuple $(SL^i, NI_l^i, NI_a^i, NO_l^i, NO_a^i)$ with:

- $SL^i \in SL$ (Safety Level); the lower the safety level is, the more critical the function is. In the sequel of this paper, the functions are ranked in four levels; hence, the set SL is $SL = \{1, 2, 3, 4\}$;
- $NI_l^i, NI_a^i \in \mathbb{N}$: numbers respectively of logic and analogic input data of the function;
- $NO_l^i, NO_a^i \in \mathbb{N}$: numbers respectively of logic and analogic output data of the function.

2) *Formal definition of a controller:* A controller $c^j \in C$ is defined as a 5-tuple $(CF^j, LI_{max}^j, AI_{max}^j, LO_{max}^j, AO_{max}^j)$ with:

- $CF^j \in CF$ (Criticality Factor)¹, the lower the criticality factor is, the more reliable the controller is; the initial value (when no function is assigned to controller c^j) of CF^j is 0, $\forall j$. CF^j is changed during the assignment process while respecting the distribution constraints given in 3;
- $LI_{max}^j, AI_{max}^j \in \mathbb{N}$ maximum numbers of respectively logic and analogic input interfaces of the controller;
- $LO_{max}^j, AO_{max}^j \in \mathbb{N}$ maximum numbers of respectively logic and analogic output interfaces of the controller.

3) *Assignment constraints:* Two types of constraints determine the assignment of functions in this study:

- 1) capabilities constraints in terms of numbers of inputs/outputs of the controllers,
- 2) constraints on distribution of functions.

- capabilities constraints ;

Let:

- $F_j = \{f^i \in F | c^j \leftarrow f^i\}$ be the set of functions f^i which are assigned to c^j ;
- $I_j = \{i \in \{1, \dots, L\} | c^j \leftarrow f^i\}$.

¹This feature is also termed Integrity Level.

Then the following four conditions must hold:

$$\forall j \in \{1, \dots, M\}$$

$$\sum_{i \in I_j} NI_l^i \leq LI_{max}^j \quad (1)$$

$$\sum_{i \in I_j} NI_a^i \leq AI_{max}^j \quad (2)$$

$$\sum_{i \in I_j} NO_l^i \leq LO_{max}^j \quad (3)$$

$$\sum_{i \in I_j} NO_a^i \leq AO_{max}^j \quad (4)$$

- distribution constraints ;

A relation \mathfrak{R} from SL to CF then defines the possible associations between the safety level of a function and the criticality factor of a controller :

$$\forall j \in \{1, \dots, M\} \text{ such that } Card(F_j) > 0, SL^i \mathfrak{R} CF^j.$$

For safety reasons, the functions must be assigned to the controllers according to their safety levels. The most critical functions ($SL^i = 1$) must be gathered in controllers which do not host any less critical function. The other functions may be gathered in a single controller as follows:

- safety level 2 functions with safety level 3 functions;
- safety level 3 with safety level 4 functions.

Hence, a controller c^j must host, when all functions have been assigned:

- only safety level 1 functions, its criticality factor is then equal to 1;
- safety level 2 and 3 functions only, its criticality factor is then equal to 2;
- safety level 3 and 4 functions only, its criticality factor is then equal to 3.

The relation \mathfrak{R} is then defined as follows:

$$\mathfrak{R} = \{(1, 1), (2, 2), (3, 2), (3, 3), (4, 3)\} \subset SL \times CF \quad (5)$$

This relation is depicted graphically in figure 2.

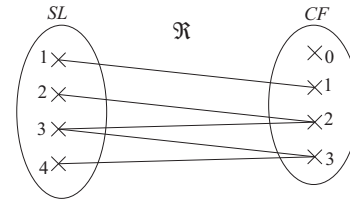


Fig. 2. Relation between Safety Level and Criticality Factor

B. Illustration of the assignment of functions

This toy example is based on a set of five functions f^1, f^2, f^3, f^4, f^5 , whose safety level is in $\{1, 2\} \subset SL$, and which are defined as follows:

- $f^1 = (2, 5, 4, 1, 3)$
- $f^2 = (1, 5, 6, 2, 4)$
- $f^3 = (2, 1, 3, 6, 4)$

- $f^4 = (1, 6, 8, 5, 2)$
- $f^5 = (1, 3, 4, 5, 2)$

These functions have to be assigned on a set of three controllers c^1, c^2, c^3 , whose capabilities are the same:

$$\forall j \in \{1, 2, 3\}, LI_{max}^j = AI_{max}^j = LO_{max}^j = AO_{max}^j = 10$$

One possible assignment of these functions to the three controllers is described in Figure 3. This solution was obtained by first assigning the function f^1 to the controller c^1 , thus fixing the value of its criticality factor to $CF^1 = 2$. The function f^2 was then assigned to the controller c^2 , thus fixing the value of its criticality factor to $CF^2 = 1$. Then the function f^3 was assigned to the controller c^1 because its safety level is consistent with $CF^1 = 2$ and the sums of the numbers of inputs/outputs of the two functions do not exceed the capabilities of the controller. The function f^4 was then assigned to the controller c^3 because the sums of logic and analogic inputs of functions f^2 and f^4 are beyond the capabilities of the controller c^2 . The function f^5 was finally assigned to controller c^2 , because the remaining capabilities of the controller c^3 were too small for f^5 be assigned to it. It should be noted that this solution is not unique. As the controllers have the same numbers of inputs/outputs, other satisfactory solutions can be obtained through a circular permutation of controllers or simply by swapping two controllers. These solutions are equivalent to the one detailed above, provided no other constraint is introduced.

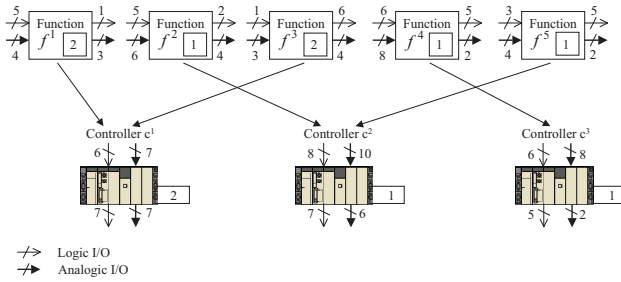


Fig. 3. Example of assignment of five functions to three controllers

III. METHOD PROPOSED

The method proposed for the automatic assignment of functions is based on two principles:

- modelling the assignment problem as a set of competing question-response mechanisms between models, in the form of communicating automata, of assignment requests and of requests acceptances;
- investigating whether the execution of this set, which is a network of communicating automata, can lead to a state reachable from the initial state where all functions are assigned.

This section first presents the formalism used to construct the request and acceptance models, which are then detailed. The state of the network of automata characterizing the assignment of all functions is then defined, which allows to state the reachability property searched.

A. Definition of the formalism used

The formalism used is a network of automata communicating through shared variables and synchronized by transition labels.

An automaton A is a N -tuple $A = (S, X, L, T, S_m, s_0, v_0)$, where:

- S is a finite set of locations;
- X is a set of n integer variables;
- L is a set of labels which can be decomposed into three disjoint sets L_i, L_o, L_l , where
 - L_i is the set of reception labels;
 - L_o is the set of emission labels;
 - L_l is the set of local labels (internal labels of an automaton).
- T is a set of transitions $(s, l, g, m, s') \in S \times L \times G \times M \times S$, where G is the set of guards (conditions on the variables of X) and M is the set of updates on the valuations of variables;
- S_m is a set of marked locations;
- s_0 is the initial location;
- v_0 is the initial valuation of variables.

A trace (or execution) of A is a succession of evolutions from the initial state: $(s_0, v_0) \xrightarrow{t_1} (s_1, v_1) \xrightarrow{t_2} (s_2, v_2) \dots \xrightarrow{t_n} (s_n, v_n)$.

A network of automata $NA = A^1 || A^2 || \dots || A^n$ is defined as $NA = (S, X, L, T, S_m, s_0, v_0)$, with:

- $S \subseteq S^1 \times S^2 \times \dots \times S^n$
- $X = X^1 \cup X^2 \cup \dots \cup X^n$
- $L = L^1 \cup L^2 \cup \dots \cup L^n$
- $T \subseteq S \times L \times G \times M \times S$, where G is the set of guards (conditions on the variables of X) and M is the set of updates on the valuations of variables
- $S_m = S_m^1 \times S_m^2 \times \dots \times S_m^n$
- $s_0 = s_0^1 \times s_0^2 \times \dots \times s_0^n$
- $v_0 : X \leftarrow \mathbb{N}$ is the initial valuation of variables.

An evolution of $NA(s, v) \xrightarrow{t} (s', v')$ is possible if:

- an evolution occurs in one of the automata by firing a transition t containing a local label, the guard being satisfied;
- two transitions t_k^α, t_m^β of a pair of automata (A^α, A^β) with t_k^α containing the label $l_k^\alpha \in L^\alpha$ and t_m^β containing the label $l_m^\beta \in L^\beta$ such that $l_k^\alpha = l_m^\beta$ are fired simultaneously, the guards of these transitions being satisfied.

B. Generic models of assignment request and of requests acceptance

Figures 4 and 5 present respectively the generic model of an assignment request sent by a function and of the acceptance of requests by a controller; these models are denoted δ and α . The following conventions are used in these models:

- the initial locations are indicated by a source arc;
- the marked locations are indicated by two concentric circles;
- the location names are in bold;

- the label names are in italics and followed by an ! (resp. ?) for emission (resp. reception) labels;
- the variables updates are underlined.

Moreover, the guards which are always true and the internal labels are not shown for brevity.

1) *Assignment request model*: The initial location of the model is 'Function not assigned'. Only one transition, which corresponds to the emission of an assignment request can be fired from this location. Once this request has been emitted, the model waits (in the location 'Assignment Possible?') for the response from an acceptance model, which can be:

- *Refusal*, then the model returns to the initial location;
- *Ok*, then the model evolves to the location "Function assigned" which is a terminal marked location.

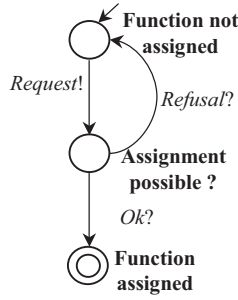


Fig. 4. Generic model of assignment request (δ)

2) *Requests acceptance model*: From the initial location, which is also marked, this model can evolve to the location 'Checking constraints' only upon reception of a request from an instance of a model δ for the function ' $f^k = (SL^k, NI_l^k, NI_a^k, NO_l^k, NO_a^k)$ '. From this location, three transitions may be fired:

- The first transition whose guard is termed 'Violation of one of the assignment constraints' represents the violation of one of the assignment constraints of distribution or capabilities, and the label *Refusal* is then emitted. The guard 'Violation of one of the assignment constraints' is then true iff at least one of the following fifth conditions is not satisfied:

$$\sum_{i \in I_j} NI_l^i + NI_l^k \leq LI_{max}^j \quad (6)$$

$$\sum_{i \in I_j} NI_a^i + NI_a^k \leq AI_{max}^j \quad (7)$$

$$\sum_{i \in I_j} NO_l^i + NO_l^k \leq LO_{max}^j \quad (8)$$

$$\sum_{i \in I_j} NO_a^i + NO_a^k \leq AO_{max}^j \quad (9)$$

$$\begin{aligned} & ((CF^j = 1 \wedge SL^k = 1) \vee \\ & (CF^j = 2 \wedge (SL^k = 2 \vee SL^k = 3)) \vee \\ & (CF^j = 3 \wedge (SL^k = 3 \vee SL^k = 4))) \quad (10) \end{aligned}$$

- The second transition whose guard is termed 'First assignment' represents the acceptance of a request whereas no other function has been previously assigned to the controller (guard 'First assignment' true). This guard is true iff conditions (6) to (9) are satisfied and:

$$CF^j = 0 \quad (11)$$

The variables $\sum_{i \in I_j} NI_l^i$; $\sum_{i \in I_j} NI_a^i$; $\sum_{i \in I_j} NO_l^i$; $\sum_{i \in I_j} NO_a^i$, are then updated and the criticality factor CF^j is set to the value of the safety level of the function if this level is smaller than 4 for $SL^k = 4$ then CF^j is set to 3.

- The third transition whose guard is termed 'Additional assignment' represents the acceptance of a request whereas at least one other function has been previously assigned (guard 'Additional assignment' true). This guard is true if all conditions (6) to (10) are satisfied. Only the variables: $\sum_{i \in I_j} NI_l^i$; $\sum_{i \in I_j} NI_a^i$; $\sum_{i \in I_j} NO_l^i$; $\sum_{i \in I_j} NO_a^i$, are updated. The criticality factor CF^j remains unchanged. In the latter two cases, the label *Ok* is emitted.

From the location 'Analysis of the state of the controller', two evolutions are possible which correspond to:

- the fact that all capabilities of the controller have been reached : $\sum_{i \in I_j} NI_l^i = LI_{max}^j$; $\sum_{i \in I_j} NI_a^i = AI_{max}^j$; $\sum_{i \in I_j} NO_l^i = LO_{max}^j$; $\sum_{i \in I_j} NO_a^i = AO_{max}^j$; the guard 'No other possible assignment' is then true and the model evolves to the marked location 'Controller saturated';
- the fact that at least one capability of the controller is not reached (guard 'other possible assignment' true), then the model evolves to the initial location 'Controller waiting'.

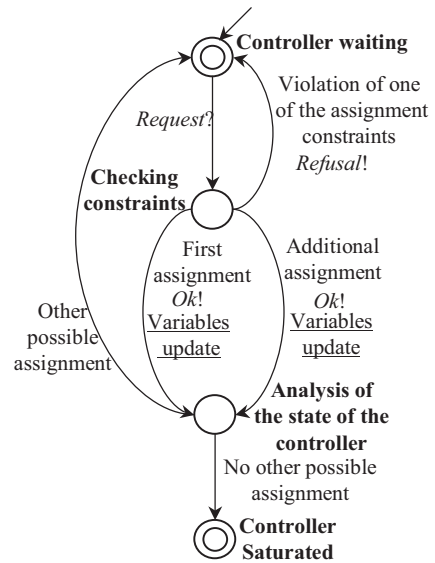


Fig. 5. Generic model (α) of requests acceptance

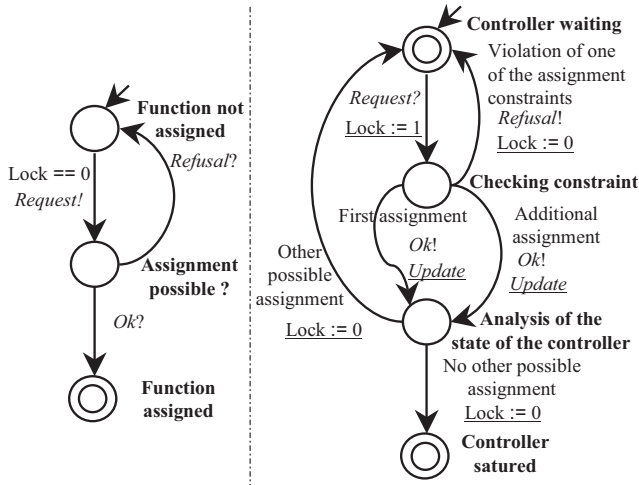
C. Instantiated model for a given functional architecture

This model is a network of communicating automata $NA = \delta^1 \parallel \delta^2 \parallel \dots \parallel \delta^L \parallel \alpha^1 \parallel \alpha^2 \parallel \dots \parallel \alpha^M$ that includes:

- as many instances ($\delta^1, \delta^2, \dots, \delta^L$) of the model in Figure 4 as there are functions,
- M instances ($\alpha^1, \alpha^2, \dots, \alpha^M$) of the model in Figure 5; the choice of M will be discussed in Section IV.

A synchronous evolution of two automata is possible only if these two automata emit and receive one of the following label pairs:

- *Request!* and *Request?*;
- *Ok!* and *Ok?*;
- *Refusal!* and *Refusal?*.



Lock := 1 (or 0) represent updates of the valuation of the variable Lock (the variable is set (reset) when the transition is fired)

Lock == 0 represents a guard that is true when the valuation of Lock is equal to 0

Fig. 6. Semaphore use

To avoid inconsistencies such as the fact that an instance α^i emits a reply to an instance δ^j which is not the one having emitted the assignment request, the question-response mechanism must be designed as a critical section protected by a semaphore. The achievement of this critical section is described in figure 6 with the semaphore 'Lock'.

An instance δ^j can emit a request only when Lock is equal to zero. As this Boolean variable is set by an instance α^i when it receives a request and reset when it emits the response (Refusal or Ok) to this request, the question-response mechanism between an instance δ^k and an instance α^i cannot be interrupted, i.e. only these two instances may be involved in the assignment process from the emission of the request to the response.

D. Definition of the reachability property searched

All the functions are assigned when the marked location is reached in all the instances of δ . In this case, the active location of the instances of α may be the terminal location

or the initial location, which are both marked. Hence, the reachability property to check can be informally stated as follows:

From the initial state, is it possible to reach a state of the network of automata such that the active location is a marked location in all the automata of the network?

IV. IMPLEMENTATION WITH A FORMAL VERIFICATION TOOL

A. Finding a solution

The techniques of formal verification by model checking [8] aim to prove that a model satisfies or does not satisfy a formal property, which may be a reachability property. It is hence natural to consider the implementation of the method proposed by using such a technique. This requires first to formally state the property searched, given in textual way in the previous section. Using the quantifiers of the CTL temporal logic, this property, noted P, can be written:

$$P: EF \text{ Full assignment}$$

Full assignment designating the state of the network such that the active location is a marked location for all automata. This property is verified if there exists at least one trace from the initial state of the network which reaches the state *Full assignment*.

The search for an assignment solution can be performed by proving that the network of automata NA satisfies the above property, which will be noted: $NA \models P$. This proof obviously depends on the number M of instances of α . For example, it is pointless to try to assign two functions ($L = 2$) to a single controller ($M = 1$) if these functions have inconsistent safety levels. However, we can note that if: $\forall i, \forall j, NI_l^i \leq LI_{max}^j, NI_a^i \leq AI_{max}^j, NO_l^i \leq LO_{max}^j, NO_a^i \leq AO_{max}^j, M = L$ surely leads to a positive proof. An examination of the trace leading to the state *Full assignment* generally shows in this case that some controllers are not used; only a number $N < M$ of controllers are hosting one or more functions.

B. Lessening the number of necessary controllers

The iterative proof process of algorithm 1 has been then developed to reduce the number of controllers used in the operational architecture. The first proof is performed with $M = L$. If $Card(I)$ controllers do not hold any function at the end of the assignment process, then another proof is attempted with a number of instances of α equal to $M - Card(I) - 1$, and so on till the proof is negative.

V. CASE STUDIES

A. Choice of the formal verification tool

Several model-checking-based formal verification tools, such as NuSMV, SPIN, UPPAAL, may be considered for achieving the reachability analysis on which the search for an assignment solution relies. The UPPAAL tool [9] was selected because it has a very ergonomic graphical interface and can provide an execution trace even in case of positive proof. It is important to note that only these features have motivated this choice; the ability of this tool to check

Algorithm 1 Search of an assignment solution lessening the number of controllers

INPUTS: Features of the L functions
 $f^i \in F = (SL^i, NI_l^i, NI_a^i, NO_l^i, NO_a^i), \forall i \in \{1, \dots, L\}$
 Features of a generic controller
 $LI_{max}, AI_{max}, LO_{max}, AO_{max}$

OUTPUTS: Minimum number N of controllers for an operational architecture.
 List of the functions assigned to each controller
 $F_j = \{f^i \in F | c^j \leftarrow f^i\}, \forall j \in \{1, \dots, N\}$
 /* Initialization step: */
 $M \leftarrow L; N \leftarrow M$
 $NA = \delta^1 \|\delta^2\| \dots \|\delta^L\| \alpha^1 \|\alpha^2\| \dots \|\alpha^M$
 /* Iterative construction: */
while $NA \models P$ **do**
 if $\exists F_j = \emptyset | j \in \{1, \dots, N\}$ **then**
 $N \leftarrow M - Card(I) - 1$ with
 $I = \{I_j, j \in \{1, \dots, M\} | I_j = \emptyset\}$
 else
 $N \leftarrow N - 1$
 end if
end while
 $N \leftarrow N + 1$
 /* Display step: */
 • N ;
 • $F_j, j \in \{1, \dots, N\}$.

properties on timed models does not constitute a selection criterion, as the communicating automata considered in this work are not timed.

In the following case studies, the UPPAAL parameters have been set so that reachability analysis be done depth first to fasten analysis.

B. First case study

This case aims to illustrate the approach. The functional architecture consists of twenty functions which are defined in Table 1, and the controllers features are as follows:

$$\forall j \in \{1, \dots, M\}, LI_{max}^j = AI_{max}^j = LO_{max}^j = AO_{max}^j = 32$$

A first reachability analysis with an initial number of controllers $M = 20$ provides a solution in which only $N = 5$ controllers are hosting at least one function (15 controllers are not used). By performing a new analysis with an initial number of controllers $M = 4$, a more compact solution can be found, in which 4 controllers are actually used. It is not possible to further reduce the number of controllers, as an analysis with $M = 3$ does not provide any solution. The final assignment solution for $N = 4$ controllers is detailed in Table II.

The durations of the various analyses required to obtain this solution are given in Table III.

C. Second case study

To assess scalability of the proposal, a study based on 200 functions was subsequently undertaken. These 200 functions are all different from each other, and their characteristics are

TABLE I
FUNCTIONS DESCRIPTION WITH $L=20$

Functions	SL^i	NI_l^i	NI_a^i	NO_l^i	NO_a^i
1	1	3	5	2	4
2	1	4	6	1	3
3	2	3	7	2	1
4	2	3	4	4	3
5	2	7	5	6	2
6	2	7	2	8	6
7	1	4	5	2	4
8	2	3	6	4	5
9	1	3	7	4	2
10	1	7	2	6	4
11	3	3	5	2	4
12	4	4	6	1	3
13	3	3	7	2	1
14	4	3	4	4	3
15	3	7	5	6	2
16	2	7	2	8	6
17	4	4	5	2	4
18	3	3	6	4	5
19	4	3	7	4	2
20	1	7	2	6	4

TABLE II
CONTROLLERS FEATURES AND FUNCTIONS DISTRIBUTION FOR THE SET OF FUNCTIONS OF TABLE I

c^j	CF^j	F_j	$\sum_{i \in I_j} NI_l^i$	$\sum_{i \in I_j} NI_a^i$	$\sum_{i \in I_j} NO_l^i$	$\sum_{i \in I_j} NO_a^i$
c^1	3	$\{f^{12}, f^{13}\}$	7	13	3	4
c^2	2	$\{f^3, f^4, f^5, f^6, f^8, f^{16}\}$	30	26	32	23
c^3	3	$\{f^{11}, f^{14}, f^{15}, f^{17}, f^{18}, f^{19}\}$	23	32	22	20
c^4	1	$\{f^1, f^2, f^7, f^9, f^{10}, f^{20}\}$	28	27	21	21

TABLE III
DURATION OF THE REACHABILITY ANALYSIS FOR THE SET OF FUNCTIONS OF TABLE I

Initial number of controllers	20	5	4
Duration	Unmeasurable	Unmeasurable	6 s

Unmeasurable: Too small to be measured by simple means

randomly distributed as follows: $\forall i \in \{0, \dots, L\}, SL^i \in \{1, 2, 3, 4\}, \{NI_l^i \in \{0, \dots, 9\}, NI_a^i \in \{0, \dots, 9\}, NO_l^i \in \{0, \dots, 9\}, NO_a^i \in \{0, \dots, 9\}$.

The controllers used are all the same and their characteristics are: $\forall j \in \{1, \dots, M\}, LI_{max}^j = AI_{max}^j = LO_{max}^j = AO_{max}^j = 32$. A first reachability analysis was conducted, providing an assignment solution in which some controllers are not hosting any function; the number of really useful controllers is $N = 37$. The analysis performed with $M = 36$ also provides a solution. The number of controllers required to achieve the operational architecture cannot however be reduced further, as the analysis performed with $M = 35$ controllers does not provide any solution. The assignment solution hence uses at least $N = 36$ controllers.

Table IV shows the durations of the different reachability analyses conducted in this case study. These values show that the approach proposed is quite feasible in the industrial context of operational control architectures design.

TABLE IV
DURATION OF THE REACHABILITY ANALYSIS WITH L=200

Initial numbers of controllers	200	37	36
Duration	110 s	46 s	50 s

VI. CONCLUSIONS AND FUTURE WORKS

This paper has first shown that assignment of control functions with different safety levels on controllers can be formalized as a constraint satisfaction problem with arithmetic constraints on integers that represent capabilities constraints and logic constraints that model constraints on distribution of functions. A novel approach to solve this issue has then been proposed. This contribution is based on reachability analysis in a network of communicating automata which models the assignment process. The treatment of a non-trivial example, by using a common verification tool has shown scalability of this proposal.

Ongoing works are aiming at introducing new capabilities and distribution constraints and to propose not only one assignment solution but a set of solutions. Comparison to other methods, like integer linear programming, is also planned on the basis of several case studies.

REFERENCES

- [1] Gerd Behrmann, Ed Brinksma, Martijn Hendriks, and Angelika Mader. Production scheduling by reachability analysis - a case study. *Parallel and Distributed Processing Symposium, International*, 3:140–147, 2005.
- [2] S. Subbiah and S. Engell. Short-Term Scheduling of Multi-Product Batch Plants with Sequence-Dependent Changeovers Using Timed Automata Models. In *20th European Symposium on Computer Aided Process Engineering*, volume 28, pages 1201–1206, 2010.
- [3] G. Marsal, B. Denis, J-M. Faure, and G. Frey. Evaluation of Response Time in Ethernet-based Automation Systems. In *Proceedings of the 11th IEEE International Conference on Emerging Technologies and Factory Automation, ETFA'06*, Prague, Czech Republic, September 2006.
- [4] P. Meunier, B. Denis, and J-J. Lesage. Temporal performance evaluation of control architecture in automation systems. In *Proceedings of 6th EUROSIM Congress on Modelling and Simulation*, Ljubljana, Slovénia, September 2007.
- [5] Dmitry A. Zaitsev. An Evaluation of Network Response Time using a Coloured Petri Net Model of Switched LAN. In *Proceedings of 5th Workshop and Tutorial on Practical Use of Coloured Petri Nets and the CPN Tools*, pages 157–167, Aarhus, Denmark, October 2004.
- [6] S. Ruel, O. De Smet, and J-M. Faure. Finding the bounds of response time of networked automation systems by iterative proofs. In *Proceedings of the 13th IFAC Symposium on Information Control Problems in Manufacturing (INCOM 2009)*, pages 1365–1370, Moscow, Russia, June 2009.
- [7] B. Addad and S. Amari. Modeling and Response Time Evaluation of Ethernet-based control Architectures using Timed Event Graphs and Max-Plus Algebra. In *IEEE Conference on Automation Science and Engineering*, pages 418–423, United States, 2008.
- [8] B. Bérard, M. Bidoit, A. Finkel, F. Laroussinie, A. Petit, L. Petrucci, and P. Schnoebelen. *Systems and Software Verification*. Springer, 2001.
- [9] G. Behrmann, J. Bengtsson, A. David, K. Larsen, P. Pettersson, and W. Yi. UPPAAL implementation secrets. In *Proc. of 7th International Symposium on Formal Techniques in Real-Time and Fault Tolerant Systems (FTRTFT)*, Oldenburg, Germany, September 2002.