Model-based Self-adaptive Management in a Smart Grid Substation

Salim Chehida, Karim Fellah, Eric Rutten, Guillame Giraud, Stéphane Mocanu

Model-based Self-adaptive Management in a Smart Grid Substation

Salim Chehida*, Karim Fellah*, Eric Rutten*, Guillaume Giraud†, and Stéphane Mocanu*

* *Univ. Grenoble Alpes, Inria, CNRS, Grenoble INP, LIG* – Grenoble, France
{Salim.Chehida, Karim.Fellah, Eric.Rutten, Stephane.Mocanu}@inria.fr

† *RTE* – Paris, France – guillaume-np.giraud@rte-france.com

***Abstract*—The design of Cyber Physical Systems (CPS) is becoming increasingly complex due to the dynamic changes in their environments and infrastructures, requiring them to be self-adaptive. An important class of CPS is Industrial Control Systems (ICS), where a major trend is to upgrade from historically specific hardware and technologies towards more software-defined, virtual approaches involving the Could-Fog-Edge continuum. In this work, we propose a model-based approach for the design of the self-adaptation in ICS, inspired by, and applied to an industrial case study in Smart Grids, more particularly an electrical substation from RTE (the French Energy Transmission company). The problem is to allocate and reallocate dynamically a set of control functions upon a distributed computing infrastructure, with self-adaptation to variations and perturbations. We define and implement the model-based autonomic management feedback loop using constraint programming, to describe the space of possible configurations, as well as the constraints and objectives formalizing the operators strategies. This model is used in simulation, calling the constraints solver at each cycle of the loop.**

***Index Terms*—Self-adaptive Systems, CPS, Electrical Substation, Constraint Programming, Autonomic Manager, Energy, Smart Grid.**

I. Introduction

Industrial Control Systems (ICS) are a class of Cyber Physical Systems (CPS) which is taking importance as an application domain of distributed computing systems, especially as there is a major trend w.r.t. its Information Technology (IT) upgrading. Indeed, historically ICS were relying on specific hardware and technologies like Programmable Logic Controllers (PLC), field buses and network protocols. Recently, in relation with the notion of Industry4.0, more modern numeric technologies and infrastructures are introduced in ICS, leveraging more software-defined, virtual approaches and adopting the Cloud-Fog-Edge continuum paradigm, of which it is a significant instance. CPS systems are exposed to changes that can occur in their environment (e.g., physical processes, weather, users demands and behaviors), their applicative function and requirements (e.g., nominal, emergency or degraded modes) and their computing infrastructure (variations in e.g., computing resources available, quality of connection). These can happen at any time, affecting the correct operation of the system and its Quality of Service (QoS). Self-adaption approaches can handle this challenge by introducing autonomic control loops that allow the system to evolve and reconfigure itself at runtime in response to environment and requirements changes.

In this work, we propose a model-based approach for the design of the autonomic feedback loop manager of self-adaptation in a class of Industrial Control Systems. We consider an industrial case study in the domain of Smart Grids, where the IT infrastructures are currently being redefined. The particular target concerns a Virtualized Electrical Substation Automation System from RTE[1] (the French Energy Transmission company). This use case is used as an inspiration for our approach in modelling and managing self-adaptations. This more general proposal is then applied to the specific use case. In the substation, several computing resources (servers) are available to run protection and automation functions. We consider the problem of allocating and reallocating dynamically the set of functions, available in different versions (e.g. nominal, back-up), upon a heterogeneous distributed computing infrastructure. The compute servers based in a substation are susceptible to undesirable events that can change their operational states (e.g., loss of computing node: computer shutting down, software errors and system bug, cyber-attacks), and impact on the availability of running protection functions. Therefore, a self-adaptive control system is needed to monitor the states of compute servers and take into consideration the arrival of external incidents (e.g., high temperature, flooding, cyber-attack) in order to generate new configurations defined by the activation or not of protection functions depending on the requirements

This work was carried out within the CPS4EU project, which has received funding from the ECSEL Joint Undertaking (JU) under grant agreement No 826276. The JU receives support from the European Union's Horizon 2020 research and innovation program and France, Spain, Hungary, Italy, Germany. The proposed results reflect only the authors' view. The JU is not responsible for any use that may be made of the information the present work contains.

[1] https://www.services-rte.com/en/home.html?profil=32

of power system components to be protected and the strategy adopted by the substation operator.

In our approach, we design an autonomic feedback loop manager of the execution of the functions on compute nodes with respect to resource and coherence constraints. We build a decision model using the constraint programming technique. The model implements a set of equations and constraints specifying the system requirements and the objectives of the RTE operators. Then, a constraint solver is called at each cycle of the self-adaptive control loop to compute the adequate configuration of protection functions allocations on the compute servers. Our contribution is to apply a rather classical model in a novel framework by integrating it into a self-adaptation feedback loop, and working on an industrial system use case.

We introduce our industrial case study and its requirements in Section II. We then present the self-adaptive control model generalizing the substation problem, and its formulation in constraint programming in Section III. In Section IV, we define different optimization objectives corresponding to applicative strategies, and give the experimentation results provided by our tool. In Section V, we present related works. Finally, Section VI draws the conclusion and the perspectives of this work.

II. Industrial use case and motivation

A substation automation system is a collection of hardware and software components that are used to monitor and control an electrical substation, both locally and remotely. It is in charge of fast protection functions ensuring the safety of equipment and people as well as slower automation functions like automatic reclosing of the circuit breaker after a fault or disturbance recording. Even if most functions can be mutualized on the same processor, protection functions, which are critical for safety as they are in charge of electric fault elimination, are processed by dedicated Intelligent Electronic Devices (IED). These IED can collect and record information on many different parameters of a system, process them based on complex logic in a few dozen milliseconds and make decisions on abnormal situations to send control commands to switches and breakers to clear the fault.

RTE is currently working on an innovative approach to virtualize the protection functions. This approach consists in replacing the classical protection and automation relays (i.e. IEDs) by a new distributed solution, where all these functions are virtualized on several servers (one for redundancy) for the whole substation (See Figure 1). This minimizes the complexity of the field equipment of an electrical substation, and relocates the treatments ("the intelligence") in servers at the Edge level (substation) instead of being spread through the different feeders. Typically, in an eight-feeders substation, the number of equipment could fall from at least 16 IED to 5 servers. These compute servers at the edge level receive the measurements and the status information sent by field equipment (current and voltage sensors, and circuit breaker position relays), support virtual machines ensuring protection functions, and send commands to actuate the field switch gear. As shown in Figure 1, in addition to the 3 compute servers, 2 control/storage servers (one for redundancy) monitor the status of compute servers, and store the context data. It is worth mentioning that the failure of control servers cannot affect compute servers' treatments. The gateway communicates with the area network and acts as a processor only in global network level (not substation level). The three compute servers in the substation have different processing power capacities, within which the protection functions should run. Furthermore, the functions consume various percentage of CPU (Central Processing Unit) and could be migrated from one compute server to another.

The main challenge in our system is to build self-adaptive mechanisms that allow for, (i) on the one hand, providing a more efficient use of the computing resources coupled with a more secure operation, and (ii) on the other hand, reacting to abnormal situations in order to ensure fast protection of the electric grid components, namely the power lines that transmit electricity between substations and the transformers that change the voltage level. As with IED, the mission-critical protections functions are doubled with a main and a back-up versions, whereas auxiliary functions are not duplicated. Back-up functions provide a minimal but sufficient protection whereas main functions usually provide better performance but consume more CPU.

From this concrete industrial use case, we will derive in the next section a generalized formulation of the problem, in terms of functions, resources and constraints.

III. Self-adaptation in a feedback control loop

The problem to be solved is to allocate and reallocate dynamically a set of control functions upon a distributed computing infrastructure, with self-adaptation to variations and perturbations, while following high-level constraints and objectives. In Figure 2, we design the feedback loop scheme that supports self-adaptation of the virtualized electrical substation, characterized by its capability to reconfigure the execution of protection functions while respecting strategies which are specific to the application domain, here defined by RTE. Those reconfigurations make the substation control station more robust w.r.t. changes in external system context such as servers dropping off, or an impairment of computation power due to increased temperatures or other external incidents such as flooding.

A. Decision model

As shown in Figure 2, we consider a set of *compute servers* in the substation. The variable N_c represents the number of computers and the variable k specifies the compute server index. We define the state variable Act_C_k that represents the state of a computer C_k.

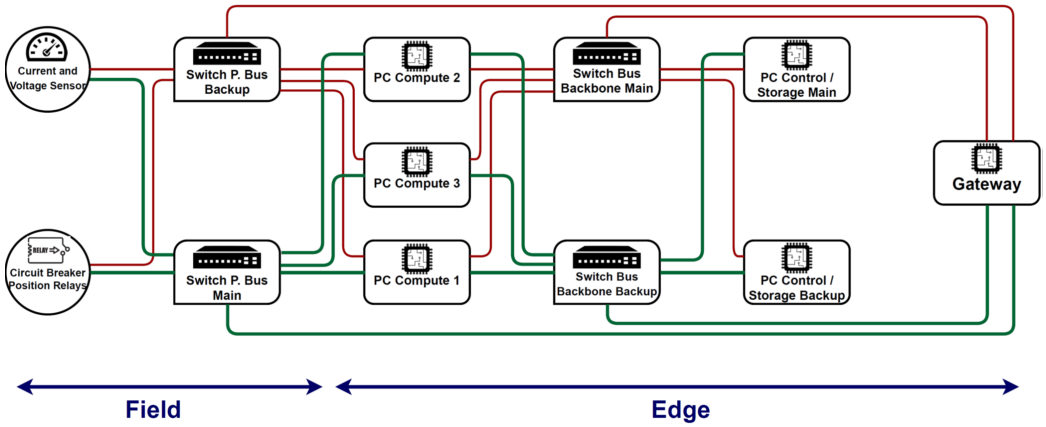

Fig. 1. RTE substation processing and storage infrastructure.

- $Act\_C_k = 1$ if the computer C_k is ON
- $Act\_C_k = 0$ if the computer C_k is OFF (by loss of computing node: computer shutting down (HW), software errors and system bug, cyber-attacks, etc.).

In our model, 3 types of protection functions are considered: *main*, *back-up* and *auxiliary*, as introduced in Section II. For simplification purposes, we consider that the number of main functions is equal to the number of back-up functions. In practice, there could be cases where some functions do not have a back-up version: then we could adapt the equations quite simply. We define the following variables:

- N_M is the number of main or back-up functions
- N_A is the number of auxiliary functions
- i represents the index of the corresponding main or back-up protection function, spanning between 1 and N_M.
- z represents the index of the corresponding auxiliary protection function, spanning between 1 and N_A.

We define the following state variables:

- $Act\_FM_{i,k}$ that represents the activation state of the main function i on computer C_k.
- $Act\_FB_{i,k}$ for the activation state of the back-up function i on computer C_k.
- $Act\_A_{z,k}$ that represents the activation state of the auxiliary function z on computer C_k.

These are the decision variables in our problem: we want to decide upon which functions to activate where.

The system controller receives the states of the compute servers (Act_C_k) from the monitor and, depending on the reconfiguration strategy (application-related, defined by RTE) and external incidents, a new configuration will be generated and applied to the system, representing the new states of protection functions (active/non-active) on compute servers ($Act\_FM_{i,k}$, $Act\_FB_{i,k}$, $Act\_A_{z,k}$).

B. The system controller as constraint equations

We define and implement the decision model using constraint programming, to describe the space of possible

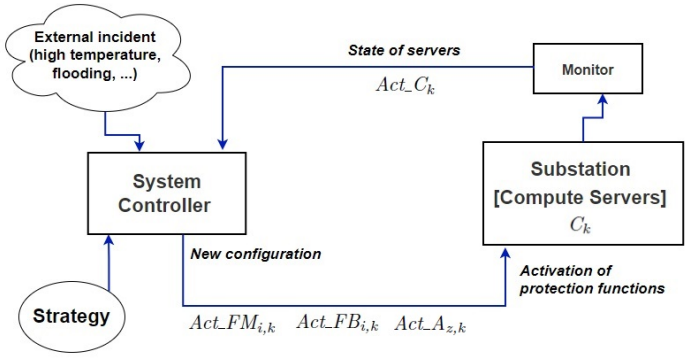

Fig. 2. RTE substation self-adaptive control system.

configurations ensuring a set of constraints. The objectives formalizing the operators strategies will be explained in the next Section.

A configuration ***Config*** is specified by a Boolean vector of size $(N_c + (2 \times N_M) + N_A)$ that represents the states of the compute servers and protection functions i.e., the global configuration generated by the self-adaptive controller.

$$\textbf{\textit{Config}} = (Act\_C_k; Act\_FM_{i,k}; Act\_FB_{i,k}; Act\_A_{z,k}).$$

The first column represents the states of compute servers, the other columns represent respectively the activation states of main, back-up and auxiliary functions on three compute servers. In the following configuration, we have example of 3 compute servers, 3 main functions, 3 back-up functions and 2 auxiliary functions:

(1.1.0 ; 1.0.0,0.1.0,0.1.0 ; 0.1.0,1.0.0,1.0.0 ; 1.0.0,0.1.0)

In our system, each protection function consumes a percentage of CPU capacity of the compute server (processor load) which has to be between 0 and 1. Such processing load for each function is different from one compute server to another. The functions processing loads are represented as following:

- $FM_{i,k}$ represents the processing load of the main function i on compute server C_k.
- $FB_{i,k}$ represents the processing load of the back-up function i on compute server C_k.
- $A_{z,k}$ represents the processing load of the auxiliary function z on compute server C_k.

We define the variables PM_k, PB_k, PA_k that represent respectively the processing load of the main, back-up, and auxiliary functions, on compute server C_k.

$\forall\ k\ (k = 1..N_c),\ PM_k = \sum_{i=1}^{N_M} (FM_{i,k} * Act\_FM_{i,k})$.
$\forall\ k\ (k = 1..N_c),\ PB_k = \sum_{i=1}^{N_M} (FB_{i,k} * Act\_FB_{i,k})$.
$\forall\ k\ (k = 1..N_c),\ PA_k = \sum_{z=1}^{N_A} (A_{z,k} * Act\_A_{z,k})$.

The total sum of processor loads P_k of a compute server C_k, is the sum of different processing loads related to different version of protection functions allocated in this server.

$\forall\ k\ (k = 1..N_c),\ P_k = PM_k + PB_k + PA_k$

We specify 5 constraints to build our controller. The two first constraints C1 and C2 are classical capacity constraints. More interestingly, and more specifically to our self-adaptive system are the constraints C3, C4, and C5.

Constraint C1: it ensures that protection function can be active only on an operational compute server. If the compute server is OFF *($Act\_C_k = 0$)*, it cannot execute or allocate any protection function.

$\forall\ i\ (i = 1..N_M), \forall\ k\ (k = 1..N_c),\ Act\_FM_{i,k} \leq Act\_C_k$
$\forall\ i\ (i = 1..N_M), \forall\ k\ (k = 1..N_c),\ Act\_FB_{i,k} \leq Act\_C_k$
$\forall\ z\ (z = 1..N_A), \forall\ k\ (k = 1..N_c),\ Act\_A_{z,k} \leq Act\_C_k$

Constraint C2: it guarantees that processor load P_k on a compute server C_k has to be not more than the maximum processor load capacity MP_k of this server.

$\forall\ k\ (k = 1..N_c),\ P_k = PM_k + PB_k + PA_k \leq MP_k$

Constraint C3: It ensures that a main function should not be executed with its corresponding back-up function on the same compute server, in order to avoid a common mode of failure.

$\forall\ i\ (i = 1..N_M), \forall\ k\ (k = 1..N_c),$
$Act\_FM_{i,k} + Act\_FB_{i,k} \leq 1$

Constraint C4: It ensures that for all protection functions, if a main or back-up function is active, each should be executed in one compute server and only one.

$\forall\ i\ (i = 1..N_M),\ 0 \leq \sum_{k=1}^{Nc} (Act\_FM_{i,k}) \leq 1$
$\forall\ i\ (i = 1..N_M),\ 0 \leq \sum_{k=1}^{Nc} (Act\_FB_{i,k}) \leq 1$

Constraint C5: It guarantees that for all functions, at least a main protection function or its corresponding back-up function should be executed somewhere on a compute server.

$\forall\ i\ (i = 1..N_M),\ 0 < \sum_{k=1}^{Nc} (Act\_FM_{i,k} + Act\_FB_{i,k})$

While the previous constraints filter out incorrect configurations, and produce a set of potentially multiple remaining possible solutions, we need additional strategies to choose between them. They will define optimization objectives, defining particular solutions which can be used to drive actions on the system. We describe such strategies in the next Section IV-A, because they correspond to applicative strategies that are specific to the use case.

IV. Application to the industrial use case

The model from previous section is implemented using OPL (Optimization Programming Language) language[2], and problems were solved by the IBM ILOG CPLEX Optimization Studio version 12.8.0[3]. The CPLEX tool allows to create a model file that contains the model to solve, a data file that contains data for a model, a parameter file that allows you to configure the CPLEX solver, and a runtime configuration file which defines the model to be solved as well as the parameters and data when the user requests the execution of the project. We ran all the experiments on a Corei7 2.7GHz computer with 16Gb of RAM (Random Access Memory) under Windows 10 pro 64-bits.

To demonstrate the self-adaptive mechanism, we use an example of the substation 225 kV/63 kV from RTE. The substation has 21 functions structured in groups of lines and transformers. Two 225 kV lines (L225), two 63 kV lines (L63), two 225/63 kV transformers (T225) and two 63/20 kV transformers (T63). Table I shows the numerous protection functions with their CPU cost. Each group is composed of main and back-up functions and also a set of auxiliary functions.

A. Reconfiguration strategies

The strategies are formalizing options available to the Smart Grid operators, corresponding to different applicative or technical criteria from the point of view of, in the case of our case study, the management of electrical networks. There can be several such strategies, which can have various advantages and costs or drawbacks: one of them is used at a given time, but it is envisageable to switch between them according to applicative scenarios. This would constitute an additional feedback loop deciding upon the choice of strategy.

In the current work, after applying the 5 constraints, we define two different criteria ST1 and ST2 to select between multiple optimal configurations. To define them, we consider the following variables related to lines and transformers introduced in Section II:

- N_c for the number of compute servers
- N_A for the number of auxiliary functions
- N_L for the number of power lines
- N_T for the number of transformers
- i represents the index of the corresponding line (spanning between 1 and N_L) or transformer (spanning between 1 and N_T)

[2]https://www.ibm.com/docs/en/icos/12.8.0.0?topic=opl-optimization-programming-language

[3]https://www.ibm.com/products/ilog-cplex-optimization-studio

Group	Function	Type	Name	CPU cost
Line 225 kV	225kV_Li_FM Main function	FM	Relay protection (PX)	0.083
	225kV_Li_FB Back-up function	FB	Back-up protection (PS)	0.067
	225kV_Li_A1 Auxiliary function 1	A	Circuit breaker failure (ADD)	0.033
	225kV_Li_A2 Auxiliary function 2	A	Earth wattmetric protection (PW)	0.067
	225kV_Li_A3 Auxiliary function 3	A	Auto-recloser system (ARS)	0.042
	225kV_Li_A4 Auxiliary function 4	A	Disturbance recorder (EP)	0.017
Transformer 225/63 kV	225kV_Ti_FM Main function	FM	Transformer protection (PTP)	0.108
	225kV_Ti_FB Back-up function	FB	Back-up transformer protection (PTP)	0.108
	225kV_Ti_A1 Auxiliary function 1	A	Circuit breaker failure (ADD)	0.033
	225kV_Ti_A2 Auxiliary function 2	A	Disturbance recorder (EP)	0.017
Line 63 kV	63kV_Li_FM Main function	FM	Relay protection (PX)	0.083
	63kV_Li_FB Back-up function	FB	Back-up protection (PS)	0.067
	63kV_Li_A1 Auxiliary function 1	A	Circuit breaker failure (ADD)	0.033
	63kV_Li_A2 Auxiliary function 2	A	Earth wattmetric protection (PW)	0.067
	63kV_Li_A3 Auxiliary function 3	A	Auto-recloser system (ARS)	0.042
	63kV_Li_A4 Auxiliary function 4	A	Disturbance recorder (EP)	0.017
Transformer 63/20 kV	63kV_Ti_FM Main function	FM	Relay protection (PSPT)	0.117
	63kV_Ti_FB Back-up function	FB	back-up protection (PX BARRE)	0.067
	63kV_Ti_A1 Auxiliary function 1	A	Circuit breaker failure (ADD)	0.033
	63kV_Ti_A2 Auxiliary function 2	A	Auto-recloser system (ARS)	0.042
	63kV_Ti_A3 Auxiliary function 3	A	Disturbance recorder (EP)	0.017

TABLE I
PROTECTION FUNCTIONS.

- z represents the index of the corresponding auxiliary protection function (spanning between 1 and N_A)
- $Act\_Li\_FM_k$ represents the activation state of the main function of line Li on the compute server C_k
- $Act\_Ti\_FM_k$ for the activation state of the main function of transformer Ti on the compute server C_k
- $Act\_Li\_FB_k$ represents the activation state of the back-up function of line Li on the compute server C_k
- $Act\_Ti\_FB_k$ for the activation state of the back-up function of transformer Ti on the compute server C_k
- $Act\_Li\_A_{z,k}$ represents the activation state of the z^{th} auxiliary function of line Li on the compute server C_k
- $Act\_Ti\_A_{z,k}$ for the activation state of the z^{th} auxiliary function of transformer Ti on compute server C_k

1) Strategy ST1: complete groups of auxiliary functions:

We choose to have complete groups of auxiliary functions running, i.e. groups with full functionalities, instead of random auxiliary functions missing in different groups. This strategy allows to maximize the number of totally operational line or transformer automations.

The objective function is expressed as follows:

$$Max \quad \{\sum_{k=1}^{N_c} \left(\prod_{z=1}^{z=N_A} \sum_{i=1}^{N_L} Act\_Li\_A_{z,k} + \prod_{z=1}^{z=N_A} \sum_{i=1}^{N_T} Act\_Ti\_A_{z,k} \right) \}$$

The products of $Act\_Li\_A_{z,k}$ and $Act\_Ti\_A_{z,k}$ ensure the activation or not of all the auxiliary functions in the lines and the transformers. If the product is equal to 1, this means that all the auxiliary functions are active. We maximize then the number of lines and transformers with complete auxiliary functions.

2) Strategy ST2: order preference of lines and transformers:

We consider the following order of preference: lines then transformers, in decreasing order of the value of their voltage level. Indeed, the lines are much more prone to faults, so it is interesting to have all the auxiliary functions, e.g. to automatically close the line after a fault has been cleared. We also favour the higher voltage levels as their loss leads to a higher risk of load shedding. It means that we need to ensure the availability of protection functions for lines in priority, then the availability of protection functions for the transformers.

The objective function is given below:

$$Max \{ \sum_{k=1}^{N_c} (\sum_{i=1}^{N_L} \sum_{z=1}^{N_A} (Act\_Li\_FM_k + Act\_Li\_FB_k + Act\_Li\_A_{z,k}); \boxed{Obj1} \quad \sum_{i=1}^{N_T} \sum_{z=1}^{N_A} (Act\_Ti\_FM_k + Act\_Ti\_FB_k + Act\_Ti\_A_{z,k}) \boxed{Obj2}) \}$$

We consider that the lines Li and transformers Ti are indexed by a decreasing order of the value of their voltage level. In our use case, the lines L225 then L63, the transformers T225 then T63. We use a multi-objective optimization by sorting the objectives in decreasing priority. For each compute server C_k, the first objective *Obj1*

maximizes the number of functions activated in lines in decreasing order of the value of their voltage level. The second objective *Obj2* maximizes the number of functions activated in transformers in decreasing order of the value of their voltage level.

B. Simulation scenarios

Referring to the constraints and strategies mentioned above, in this section we illustrate how the self-adaptive control system reacts to change in compute servers states (one or more computers are not operational/shutting-down). When exploiting the available computing resources to execute a maximum number of protection functions, an external factor such as temperature can impact on processor capacity, which can be reduced to less than 1 (less than 100%), thus, the number of running protection functions will also be limited. As the substation is susceptible to several unexpected incidents, many scenarios can happen in terms of compute servers' states modification, which can impact on the available computing resource and the capacity of their CPU. Though the use case has 2 power lines and 2 transformers for each voltage level, the simulations only include 1 element from each type for a better readability of the results. The computing power of each server is also set to 50%.

Figure 3 shows an example of reconfiguration scenario. We consider that the loss of computer servers is due to the arrival of external incidents such as floods and cyber-attacks. We will explain the response of our adaptive control mechanism when servers are lost or their capacity is reduced.

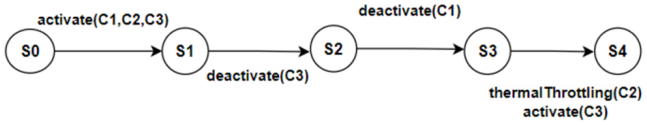

Fig. 3. Reconfiguration scenarios of substation.

At state S1, all compute servers are active. At state S2, we lose C3 and at state S3 the server C1 is lost. A state S4, the server C3 is recovered and the server C2 is subjected to thermal throttling which consists in adjusting the clock speed of the CPU based on the amount of heat it is currently generating to maintain safe core operating temperatures (usually below 90°c). This reduction of clock rate leads to reduction of the processor performance. Fanless devices like the ones used in substations are more likely to experience this phenomenon, especially during hot weather. In our example, the external temperature leads to a thermal throttling with a 50% reduction of clock rate and so a 50% decrease of the performance of server C2. This simplified taking into account of thermal throttling opens perspectives of considering the composition, in a multiple loops approach, of our autonomic manager with another feedbak loop performing finer regulation of clock rate [4].

Figure 4 presents the selected configurations when considering strategy ST2 (order of preference of groups). A configuration represents respectively the activation states of main (F1M), back-up (F1B) and auxiliary (A1, A2, A3 and A4) functions on three compute servers (sub-columns represent respectively servers C1, C2 and C3). For each configuration, we show the number of activated functions (NB FCT) and also the number of complete groups (CMP GR) where all functions of the group are activated. At state S1 with all compute servers active, we can run all protection functions and complete the 4 groups (L225, L63, T225, T63). When losing server C3 at state S2, 4 functions cannot be executed and the group T63 cannot be completed. When only server C2 is active at state S3, we can only run 9 protection functions and none of the groups can be completed. Finally, at state S4, 13 functions can be run with one group completed (L225).

Figure 5 shows the selected configurations regarding strategy ST1 (complete groups of auxiliary functions). The "CMP GR" column in Figure 5 represents the number of complete groups of auxiliary functions. When losing server C3 in S2, we reach the same objective as for state S1 when all servers are active with 4 complete groups of auxiliary functions. When losing servers C3 and C1, the best configuration can provide only 2 complete groups of auxiliary functions. At S4 when recovering the server C3 and the server C2 is subjected to thermal throttling, 3 groups of auxiliary functions can be completed with 15 functions activated.

C. Evaluation

Following the behavior scenario shown Figure 3, the transitions triggered by self-adaptation controller between the different states can be explained as follows from the applicative point of view of the use case:

State S0 to S1 - all servers are started. With order of preference strategy (ST2) so all functions are available as we are in the optimal configuration. Strategy ST1 (complete groups of auxiliary functions) leads to a different situation where only one of main or backup functions is active for each element, which is in line with the criteria but less relevant from an operational point of view.

State S1 to S2 - server 3 is lost. With ST2, all main and back-up functions are available, but we only have 3 complete groups and 17 functions. ST1 allows to have all (4) complete auxiliary function groups and 19 functions, but 225 line has only main protection and 63 transformer only back-up protection. In this situation it is not straightforward to choose the best operational option.

State S2 to S3 - servers 1 is also lost (only server 2 is available). With these scare resources, ST2 leads to all elements with only back-up protection. 225 line has all its auxiliary functions, 63 line has one and other elements have none. ST1 leads to a quite similar solution for protection functions (only 225 line is on main

X Function activated

	COMPUTERS			GR	F1M			F1B			A1			A2			A3			A4			CMP GR	NB FCT
S1	1 (0.5)	1 (0.5)	1 (0.5)	L225	-	X	-	-	-	X	-	-	X	-	X	-	-	X	-	-	X	-	4	21
				L63	-	-	X	-	X	-	-	X	-	X	-	-	X	-	-	-	X	-		
				T225	-	-	X	-	X	-	X	-	-	X	-	-								
				T63	X	-	-	-	X	X	-	-	-	-	X	-	-	X						
S2	1 (0.5)	1 (0.5)	0	L225	X	-	-	-	X	-	X	-	-	X	-	-	-	X	-	X	-	-	3	17
				L63	-	X	-	X	-	-	X	-	-	-	X	-	-	X	-	X	-	-		
				T225	-	X	-	X	-	-	X	-	-	-	X	-								
				T63	-	-	-	-	X	-	-	-	-	-	-	-	-	-						
S3	0	1 (0.5)	0	L225	-	-	-	-	X	-	-	X	-	-	X	-	-	X	-	-	X	-	0	9
				L63	-	-	-	-	X	-	-	-	-	-	-	-	-	-	-	-	X	-		
				T225	-	-	-	-	X	-	-	-	-	-	-	-								
				T63	-	-	-	-	X	-	-	-	-	-	-	-	-	-						
S4	0	**1 (0.25)**	**1 (0.5)**	L225	-	X	-	-	-	X	-	X	-	-	-	X	-	X	-	-	X	-	1	13
				L63	-	-	-	-	-	X	-	-	X	-	-	X	-	-	X	-	-	X		
				T225	-	-	-	-	X	-	-	-	-	-	-	-								
				T63	-	-	-	-	X	-	-	-	-	-	-	-	-	-						

Fig. 4. Reconfiguration by order of preference (ST2).

X Function activated

	COMPUTER			GR	F1M			F1B			A1			A2			A3			A4			CMP GR	NB FCT
S1	1 (0.5)	1 (0.5)	1 (0.5)	L225	X	-	-	-	-	-	X	-	-	X	-	-	X	-	-	-	X		4	17
				L63	-	-	X	-	-	-	-	X	-	-	-	X	-	X	-	-	-	X		
				T225	-	-	-	-	X	-	X	-	-	X	-									
				T63	X	-	-	-	-	-	-	X	-	-	X	-	-	X						
S2	1 (0.5)	1 (0.5)	0	L225	X	-	-	-	-	-	X	-	-	X	-	-	X	-	-	-	X	-	4	19
				L63	-	X	-	X	-	-	-	X	-	X	-	-	-	X	-	-	X	-		
				T225	-	X	-	X	-	-	-	X	-	-	X	-								
				T63	-	-	-	-	X	-	X	-	-	-	X	-	-	X	-					
S3	0	1 (0.5)	0	L225	-	X	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	2	10
				L63	-	-	-	-	X	-	-	-	-	-	-	-	-	-	-	-	X	-		
				T225	-	-	-	-	X	-	-	X	-	-	X	-								
				T63	-	-	-	-	X	-	-	X	-	-	X	-	-	X	-					
S4	0	**1 (0.25)**	**1 (0.5)**	L225	-	-	-	-	-	X	-	X	-	-	-	-	-	-	X	-	-	-	3	15
				L63	-	-	-	-	-	X	-	-	X	-	-	X	-	-	X	-	X	-		
				T225	-	-	-	-	X	-	-	-	X	-	-	X								
				T63	-	-	-	-	-	X	-	-	X	-	X	-	-	X						

Fig. 5. Reconfiguration by complete groups of auxiliary functions (ST1).

protection), but both transformers have all their auxiliary functions. ST1 seems to have a slight advantage here.

State S3 to S4 - server 3 gets back and server 2 experiences thermal throttling and reduction of capacity by half. ST2 manages to have all functions available from 225 line but only 13 functions and 1 complete group. ST1 does better with 15 functions and 3 complete groups.

As a conclusion, both strategies get the expected results. If the assumption that main and back-up protections are equivalent and that only one is needed at the time, strategy ST1 offers better results. Such scenario analysis illustrates the potential of our model to offer assistance to automation designers, to explore choices of formulations in requirements and strategies.

V. Related work

Some related work features several papers such as [15] and [9], which present the state-of-art approaches for self-adaptation in CPS through feedback loops. According to [9], the performance, flexibility, and reliability are the main concerns and energy is the principal application domain. Thus, different adaptation mechanisms have been used as for example: MAPE-K (Monitor-Analyze-Plan-Execute-Knowledge) [6]. Such self-adaptation is supported by reconfiguration mechanisms at the level of the middleware, as for example in component-based approaches [11]. Self-adaptation can be used to address and solve problems of resource management e.g., load balancing [10], or service placement [1], but it is done at runtime. It can perform interventions at different levels of the software stack, from HPC to IoT and in between in the Cloud-Fog-Edge continuum [5]. The modelling and decision ap-

proaches involved in the design of the feedback loops are various and can range from Machine Learning to Control Theory [12]. Finally, it is to be noted that the notions of co-existing multiple loops, managing different aspects of complex systems, can be approached in a variety of compositions and decentralization patterns [14].

Regarding the use of Constraint programming for the modeling of heterogeneous distributed systems, there is related work in solving of service placement problems in Fog Computing [2], essentially at deployment time, rather than such as in our approach dynamically at runtime across the duration of the application. Other work involving constraint programming integrated in autonomic loop feature [8] where it is applied to the different domain of Device Management in the IoT, and [3] where it is used in different context of Cloud computing, to automate the management of varying-demand services, while still enforcing Service-Level Agreements (SLAs). The difference with our approach is that we model a different kind of system, and of constraints between functions, in our CPS context. A closer work [13] also consider smart grid CPS, but at a higher level in the architecture, whereas here we concentrate on the substation level.

VI. Conclusion

In summary, our results are in:

- a model-based approach for the design of the automation of real-time self-adaptation in Industrial Control Systems;
- the application to an industrial case study in Smart Grids, more particularly an electrical substation;
- the definition and implementation of the decision model using constraint programming, to describe the space of possible configurations, as well as the constraints and objectives formalizing the operators strategies;
- the use of the model in simulation, calling the solver at each cycle of the self-adaptation control loop, offering design assistance and rapid prototyping to automation designers, to explore choices of solutions in requirements and strategies.

Amongst perspectives of this work, we can mention:

- the integration of the controller with the existing middleware infrastructures in the use case, for a concrete implementation and experimentation;
- the model enhancement, e.g. to take into account more of the dynamics of the environment, in relation with Control Theory [7]: for example in case of heat wave or flooding in order to react in one step while taking into account the speed of rapid phenomena;
- the integration of this level of autonomic management in a more global architecture, where it would contribute one of the loops in a multiple loops system, with a coordination to avoid interferences and improve global management efficicency.

Finally, this industrial use case analysis lead to the proposal of a model which is more generic that the particular case, and shows potential of generalization to wider classes of systems.

References

[1] Farah Ait Salaht, Frédéric Desprez, and Adrien Lebre. An overview of service placement problem in Fog and Edge Computing. *ACM Computing Surveys*, 53(3), 2020.

[2] Farah Ait Salaht, Frédéric Desprez, Adrien Lebre, Charles Prud'Homme, and Mohamed Abderrahim. Service Placement in Fog Computing Using Constraint Programming. In *SCC 2019 : IEEE International Conference on Services Computing*, pages 19–27, Milan, Italy, July 2019. IEEE.

[3] Zakarea Al-Shara, Frederico Alvares, Hugo Bruneliere, Jonathan Lejeune, Charles Prud'Homme, and Thomas Ledoux. CoMe4ACloud: An End-to-End Framework for Autonomic Cloud Systems. *Future Generation Computer Systems*, 86, 2018.

[4] Sophie Cerf, Raphaël Bleuse, Valentin Reis, Swann Perarnau, and Eric Rutten. Sustaining Performance While Reducing Energy Consumption: A Control Theory Approach. In *EURO-PAR 2021 - 27th International European Conference on Parallel and Distributed Computing*, volume 12820 of *Euro-Par*, pages 334–349, Lisbon, Portugal, August 2021. Springer.

[5] L. de Souza Cimino, J. E. E. de Resende, L. H. M. Silva, S. Q. S. Rocha, M. de Oliveira Correia, G. S. Monteiro, G. N. de Souza Fernandes, R. da Silva Moreira, J. G. de Silva, M. I. B. Santos, and et al. A middleware solution for integrating and exploring IoT and HPC capabilities. *Software: Practice and Experience*, 49(4):584–616, 2019.

[6] Jeffrey O Kephart and David M Chess. The vision of autonomic computing. *Computer*, 36(1):41–50, 2003.

[7] Marin Litoiu, Mary Shaw, Gabriel Tamura, Norha M. Villegas, Hausi Müller, Holger Giese, Romain Rouvoy, and Eric Rutten. What Can Control Theory Teach Us About Assurances in Self-Adaptive Software Systems? In R. de Lemos, D. Garlan, C. Ghezzi, and H. Giese, editors, *Software Engineering for Self-Adaptive Systems 3: Assurances*, volume 9640 of *LNCS*. 2017.

[8] Ghada Moualla, Sébastien Bolle, Marc Douet, and Eric Rutten. Self-adaptive Device Management for the IoT Using Constraint Solving. In *17th Conference on Computer Science and Intelligence System*, Sofia, Bulgaria, September 2022.

[9] Henry Muccini, Mohammad Sharaf, and Danny Weyns. Self-adaptation for cyber-physical systems: A systematic literature review. In *Proceedings of the 11th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, SEAMS '16, page 75–81, New York, NY, USA, 2016.

[10] B. Pourghebleh and V. Hayyolalam. A comprehensive and systematic review of the load balancing mechanisms in the internet of things. *Cluster Computing*, 2019.

[11] Lionel Seinturier, Philippe Merle, Romain Rouvoy, Daniel Romero, Valerio Schiavoni, and Jean-Bernard Stefani. A Component-Based Middleware Platform for Reconfigurable Service-Oriented Architectures. *Software: Practice and Experience*, 42(5):559–583, May 2012.

[12] Stepan Shevtsov, Mihaly Berekmeri, Danny Weyns, and Martina Maggio. Control-theoretical software adaptation: A systematic literature review. *IEEE Transactions on Software Engineering*, 44(8):784–810, 2018.

[13] Mahyar Tourchi Moghaddam, Eric Rutten, and Guillaume Giraud. Hierarchical Control for Self-adaptive IoT Systems A Constraint Programming-Based Adaptation Approach. In *HICSS 2022 - Hawaii International Conference on System Sciences*, pages 1–10, Hawaii, United States, December 2022.

[14] D. Weyns, B. Schmerl, V. Grassi, S. Malek, R. Mirandola, C. Prehofer, J. Wuttke, J. Andersson, H. Giese, and K. M. Göschka. On patterns for decentralized control in self-adaptive systems. In *Software Engineering for Self-Adaptive Systems II*, LNCS, page 76–107. Springer, 2013.

[15] Sherali Zeadally, Teodora Sanislav, and George Dan Mois. Self-adaptation techniques in cyber-physical systems (cpss). *IEEE Access*, 7:171126–171139, 2019.