Consiglio Nazionale delle Ricerche
Istituto di Calcolo e Reti ad Alte Prestazioni

A Self-Adaptive Approach for the Reconfiguration of Shipboard Power Systems

M. Cossentinoo, G. De Simone , S. Lopes, L. Sabatucci

Rapporto Tecnico N.:
RT-ICAR-PA-18-03

aprile 2018

Consiglio Nazionale delle Ricerche, Istituto di Calcolo e Reti ad Alte Prestazioni (ICAR) – Sede di Cosenza, Via P. Bucci 41C, 87036 Rende, Italy, URL: *www.icar.cnr.it*
– Sede di Napoli, Via P. Castellino 111, 80131 Napoli, URL: *www.na.icar.cnr.it*
– Sede di Palermo, Via Ugo La Malfa 153, 90146 Palermo, URL: *www.pa.icar.cnr.it*

Consiglio Nazionale delle Ricerche
Istituto di Calcolo e Reti ad Alte Prestazioni

A Self-Adaptive Approach for the Reconfiguration of Shipboard Power Systems

M. Cossentinoo, G. De Simone , S. Lopes, L. Sabatucci

Rapporto Tecnico N.:
RT-ICAR-PA-18-03

Data:
aprile 2018

---

[1] Istituto di Calcolo e Reti ad Alte Prestazioni, ICAR-CNR, Sede di Palermo, Via Ugo La Malfa 153, 90146 Palermo.

A Self-Adaptive Approach for the Reconfiguration of Shipboard Power Systems

Luca Sabatucci, Massimo Cossentino, Giada De Simone and Salvatore Lopes
ICAR-CNR
Palermo, Italy
{luca.sabatucci,massimo.cossentino,giada.desimone,salvatore.lopes}@icar.cnr.it

***Abstract*—The Shipboard Power System (SPS) is responsible for supplying energy to various services of a vessel. The proper functioning of the SPS is critical to the survival and safety of the ship. SPS reconfiguration consists in a variation of the electrical topology to successfully supply energy to critical services. SPS reconfiguration is a relevant problem because many accidents occurring during ship navigation are often due to electrical failures. The proposed reconfiguration procedure uses a distributed and mission-oriented approach, and it employs a generic-purpose self-adaptive middleware (MUSA). MUSA has been customized to dynamically reconfigure an SPS in case of failures or unexpected events. It allows obtaining a run-time solution that properly considers ship's mission and current scenario. We also implemented an experimental setup including a Matlab/Simulink simulation of a case study from literature, to validate the solution and to assess our approach.**

***Index Terms*—Shipboard power system, SPS reconfiguration, self-adaptive system**

I. Introduction

In recent years, the maritime sector is highlighting a high value of innovative and technological content (ICT), especially when faced with the need to respond to objectives such as safety, efficiency, and environmental impact. "EMSA's annual overview of 2015 marine casualties and incidents" reports that most of the accidents mentioned are due to loss of control or damage to ships or equipment. The ship power production and distribution failures play a relevant role in such incident scenarios. The Shipboard Power System (SPS) is the component responsible for granting energy to navigation, communication, and operational systems. It is consists of various electric and electronic equipment, such as generators, cables, switchboards, circuit breakers, fuses, buses, and many kinds of loads.

The electric and electronic equipment in modern ships is demanding higher performance from the power sources. Moreover, after the occurrence of faults and their subsequent isolation, there could be whole sections of the electric plant that remain without supply. A reliable SPS must be able to supply power even when loads demand for power variations or if critical events such as faults occur. The problem of fast and efficient restoration of the SPS service has been a topic of research for around three decades.

Modern ICT technologies can nowadays automatically accomplish real-time data acquisition, classification, assimilation, and correlation at a reasonable cost. Software-based reconfiguration systems consist of two different layers: the software layer encapsulates the logic for the monitor and the control of the underlying electrical layer. In practice, the software system manages onboard switchboards and circuit-breakers, to direct the power flow where it is necessary for restoring a fault situation.

The problematic of fault detection, isolation, and reconfiguration (FDIR) is under investigation in many others research fields. In literature, several authors study many approaches to monitor, control, and to reconfigure the electrical layer topology to survive when facing several kinds of scenarios.

In [1] authors survey FDIR methodologies, focusing the attention on reconfiguration techniques related to flight control systems. In particular, they classify the reconfiguration methodologies into two categories: multiple-model approach, and adaptive-control approach. In [2], authors compare reconfiguration techniques applied to the terrestrial and maritime domains. They include an analysis of the SPS characteristics, highlighting the need for integrated protection and power distribution.

In [3], authors surveyed several formulations of the reconfiguration problem and techniques used for the solution. They compare the SPS reconfiguration problem to that of large-scale systems, exploring the issue of optimal reconfiguration from a variety of perspectives. In [4], some of the most recent software-based reconfiguration methodologies have been analysed and classified by comparing the correlation between hardware and software properties, describing them from the electrical characteristics point of view. The present paper focuses on SPS reconfiguration in case of single or multiple failures. The proposed reconfiguration procedure uses a distributed and mission-oriented hierarchical approach, and it employs a self-adaptive middleware (MUSA). MUSA employ the Multi-Agent Systems technology, and it can configure itself and adapt to the end of dynamically reconfiguring an SPS in case of failures or unexpected events. It allows obtaining a run-time solution that adequately considers ships mission and current (fault) scenario thus including specific tasks, goals and non-functional requirements (e.g. quality aspects, QoS). We also implemented an experimental setup including a Matlab/Simulink simulation of a case study from literature[5], to validate the solution and to assess our approach.

This paper is organized as follows: Section II introduces the SPS domain and the reconfiguration problem; Section III illustrates the proposed solution architecture and algorithms. Section IV introduces two different fault scenarios that are used to demonstrate the adaptive ability of the system. Finally, some conclusions are drawn in Section V.

II. Shipboard Power Systems

The SPS is the electrical and electronic hearth of a ship, it is composed of a set of components such as power generators, buses, circuit breakers, heterogeneous loads, and others electric subsystems appointed to navigation, communication and so on. In the last decades, some ships are equipped with direct-current (DC) because of the following advantages if compared to the alternate-current (AC):

1) smaller components and compact power converters;
2) easier connections;
3) no reactive power and harmonic issues;
4) faults reduction and easier reconfiguration procedures.

The main disadvantage of DC systems is that voltage shifts are more difficult to be realised than in AC systems where transformers do that with minimal losses.

Loads often are distributed in zones and fed power from the main electric buses. It is usual to classify loads according to their importance into vital and non-vital categories, where vital loads are non-sheddable loads that directly affect the survivability of the ship, while the non-vital ones may be shed in order to prevent a total loss of ship's electrical power, or for protection purposes. Moreover, the loads can be categorised regarding QoS as un-interruptible, short-term interrupt, and long-term interrupt [6]:

1) un-interruptible load: loads that can not tolerate power interruptions on the order of two seconds;
2) short-term interrupt load: loads that can tolerate power interruption in the order of maximum one-five minutes;
3) long-term interrupt load: load that can tolerate service interruption longer than five minutes.

Reconfiguration in an electrical SPS is a critical operation requested in unexpected situations such as in the case of severe or major faults. The reconfiguration procedure is driven by the ship power and energy management control, that communicates with all the generators and loads to keep the continuity of service during reconfiguration operations. In this way, the reconfiguration of the electrical layer can isolate faults, restore/transfer power to vital loads, but also, more generally, it can optimise the management of electrical and electronic equipment to improve energy efficiency.

During normal navigation or after a specific event such as a weapon hit or a collision, there can be a series of multiple equipment damages. These can affect electrical layer and/or other systems such as the navigation one.

The strategy that enables restoration of the electrical power system is called *reconfiguration*. The number of steps and the adopted strategies (that can also involve humans) may vary. In particular, in a recent work [4], authors observed in literature exists several software-based reconfiguration techniques enabling smart and timely reconfiguration of the electrical layer due to a fault (or multiple faults). These systems need a specific environment perception and they enact reconfiguration strategies basing on several different levels of "smartness", allowing a sophisticated real-time perception of the situation and a ready management in case of emergencies.

Smart reconfiguration methodologies need complex coordination between electrical power and protective functions, and must deal with several electrical architectures (radial, ring, zonal, ...). Very frequently applied, zonal architectures are electrical configurations of the SPS where loads are ideally divided into zones. Such architectures are frequently used because they enable an easy sectioning of the ship electric level thus preventing that a single minor fault may spread in a systemic failure [4] or, conversely, that a damaged part of the system may be left apart from the functionality restoration procedure.

III. The Proposed Solution

This section illustrates the proposed solution, based on MUSA, a middleware for building self-adaptive systems, and on Matlab/Simulink for simulating the circuit.

A. MUSA: A Middleware for User-driven Service Adaptation

The Middleware for User-driven Self-Adaptation (MUSA) has arisen from a couple of pressing objectives in the research agenda of dynamic workflow execution: managing run-time business process evolution and adaptivity [8].

The key aspect is a clear separation of two points: 'what the system has to address' and 'how it will operate for addressing it'. The enablers of this vision are i) representing *what* and *how* as run-time artifacts the system may reason on (respectively goals and capabilities); ii) a reasoning system for connecting capabilities to goals; iii) finally a common grounding semantic, represented with some formalism.

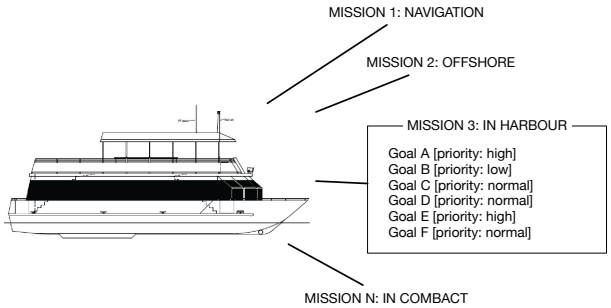

Figure 1: An example of vessel's Missions

The first aspect of MUSA is the ability to work with run-time requirements as a set of goals to be injected into the system [9]. A **goal** is a desired state an actor wants to achieve. In MUSA, a goal is provided to the system at run-time, exploiting the ability of the agent of being autonomous and proactive i.e. being able to explore a solution space, even when this space dynamically changes or contains uncertainty. For the specific context of the vessel, four goals represent the main system operations such as propulsion, rudder and stability, communication and ICT, and hotel. These are further decomposed in other sub-goals. For instance, propulsion is decomposed into main motors and maneuver gears. The hotel function is decomposed into air conditioning, lights, and other services.

MUSA tries to address the goals by finding suitable solutions using the concept of Capabilities as first-class entities for agent deliberation [10]. The concept of capability comes from planning actions [11] and it implements a service-oriented architecture. A **capability** describes a concrete operation the system may execute to change the current state of the world. Every agent knows its capabilities, their effects and the way these can be employed. In the specific context, capabilities coincide with the electrical actions (switchers) that allow to dynamically change the flow of power.

Consequently, self-adaptation is defined as a space search problem. The algorithm used in [10] is a symbolic planning algorithm, in which a set of distributed agents incrementally build a *computational graph* model by exploring different combinations of capabilities. The result is a set (possibly not empty) of solutions, in which each solution represents a sequence of actions to be executed to address the goal finally.

The agent-based, hierarchical and distributed nature of MUSA allows for managing multi-layer services as a single service, thus hiding the complexity of service composition. Moreover, agents are suitable for granting adaptation because they may change without affecting the whole structure.

B. A Mission-Oriented Solution

SPS reconfiguration problem embraces a series of possible scenarios, goals, and decisions based on functional and non-functional requirements. Functional requirements include prescriptive goals – related to onboard operations that must be granted without any degree of freedom – and soft goals which also can be satisfied partially, thus granting a minimal degree of functionality. The adoption of goals allows a seamless description of the expected behavior in terms of loads that must be powered.

Moreover, requirements in a vessel are not static: they change according to the operative context. Indeed, the operating scenario may change, and a series of reconfiguration sub-goals may be necessary to comply with specific requirements of the electrical layer. Some particular constraints are, for instance: providing energy to vital loads, protecting loads with different priorities, shedding non-damaged loads that may not be powered (possibles causes: insufficient electric power, no energy transportation route to that load). These sub-goals may strongly vary according to the kind of vessel (a warship vs. a cargo), the type of mission (approaching the harbor, offshore navigation, combat actions), and the current amount of power produced by generators and energy storage devices. The system must be flexible enough to switch its goals at run-time, for example when the ship's mission change.

To this aim, we introduce the concept of **Mission**. A mission is a description of the relation between the operating context and the degree of priority to be assigned to the system goals.

The solution we propose is based on a dynamic description of the vessel's missions. An example is shown in Figure 1. When the system power is under the value required for feeding all the vessel's loads, the SPS reconfiguration must consider not all the goals are equally important to be pursued. Indeed, some loads are mandatory for the vessel survivability [vital loads] while other ones are also important but not necessary [semi-vital loads]. Finally, other loads may be switched off without affecting ship mission accomplishing [non-vital loads]. Consequently, goals may be classified by different priority depending on the specific context. Thus, the reconfiguration system will always prefer to address a higher priority goal.

The architecture of the solution is based on the integration of MUSA and Matlab, as shown in Figure 2. MUSA provides a high-level reasoning infrastructure that is triggered when the monitoring sub-system discovers the standard electrical configuration is affected by a set of failures.

Algorithm 1 is the core of the proactive means-end reasoning procedure [10] that is responsible for generating a space of electrical configurations (namely WTS). Each configuration describes the state (open-closed) for each switcher of the electrical system. The initial configuration is W_I that describes a system affected by failures that require a reconfiguration.

Algorithm 1 $means\_end\_resoning(Goal, W_I, Assumptions, Cap)$

```
1: wts ← init_WTS(W_I)
2: while exit_condition do
3:    node ← get_most_promising_node(wts)
4:    w ← node.state
5:    for all c ← Cap do
6:       applies ← check_pre(w, Assumptions, c)
7:       if applies then
8:          w_exp ← generate_cap_evolution(w, c)
9:          compliance ← check_goal(w_exp, Assumptions, Goal)
10:         if compliance! = violation then
11:            score ← power_heuristic(w_exp)
12:            add(wts, w_exp, partial, score)
13:         end if
14:      end if
15:   end for
16: end while
17: sol_set ← search_solutions(wts)
18: return sol_set
```

The Cap argument is a set of possible actions over the circuit (capabilities). In this specific example, MUSA is provided with a set of capabilities for modifying the electrical topology by acting on the switchers. A MUSA solution is an ordered sequence of capabilities that from W_I leads the state towards a final state in which all the goals are fully satisfied.

The electrical topology is provided as a set of rules in the $Assumptions$ argument. It describes electrical dependencies of the network by using a simple ontology in which node, load, generator, and switcher are terms for describing electrical elements, up and down (powered/not powered) are the properties of a node and open/closed are the properties of a switcher. The connections between the nodes in the circuit are rendered as 'premises-conclusion' rules like:

$up(node1) \Leftarrow up(node2) \wedge closed(switcher2)$

indicating that node1 receives current from node2 via a switcher2.

A slice of the rules for describing the circuit of Figure 4 is reported below:

```
generator(mg1).
...
switcher(sw_1).
...
load(load1).
...
on(load1):- closed(sw_1), up(n10).
up(n10):- up(n1).
up(n10):- up(n11), closed(swp1).
...
up(n11):- closed(swp1), up(n10).
...
```

The first part lists all the elements (generators, switches and loads) of the circuit. Then it is reported the fact that load L1 is connected to node 10 via the switcher sw_1. It receives power when the node is up and the corresponding switch is closed. Node 10, in turn, receives energy from two nodes: node 1 and node 11 via the switcher swp1. Clearly each 'connection' rule has also the reciprocal rule. Therefore it is necessary to specify also that node 11 is connected to node 10 via the same switch swp1.

According to the current mission (described through the $Goal$ argument), the procedure aims at generating a wts, i.e. a directed graph in which arcs are capabilities and nodes are suitable electrical configurations(some of these are marked as 'exit' because they fully satisfy the mission). The main loop incrementally builds the wts, terminating when a given number of solutions are available or after a constant number of iterations.

The nodes in the wts are annotated with a score, indicating in which degree the configuration is 'close' to fully satisfy the mission. The loop starts by taking the most promising node as the base for expanding the graph. For each capability, if the preconditions apply in the selected node, then its postconditions are used to generate a state-evolution, i.e., a new state in which the effects of the capability are considered.

This new state is first checked against the goals. Three possible cases: 1) the state is not valid, i.e., it violates some goals, and consequently, it is discarded; 2) the state fully addresses the goals, and then it is added as an exit node; 3) it is an intermediate state. In these latter case, the configuration is evaluated with the domain-specific heuristics of Algorithm 2.

Algorithm 2 works by simulating a power balance. It calculates the power supplied by active generators (gen_pow) and the power absorbed from all the active loads ($load\_pow$). If the power is enough

Algorithm 2 $power\_heuristic(w_{exp})$

```
gen_pow ← calculate_gen_power(w_exp)
load_pow ← calculate_load_power(w_exp)
score = 0
for i = 1 : load_number do
    state ← up(load_i) : true|false
    if state then
        if gen_pow > load_pow_i then
            score = score + (state * priority_i)
            gen_pow = gen_pow − (state * priority_i)
        end if
    end if
end for
return score
```

Table I: Examples of usage of the heuristic

Example 1 (supplied pow > aborbed pow)					
	Generators		Loads		
	Main	Aux	Vital	Semi-Vital	Non-Vital
Power (MW)	6	2	0,5	1	0,5
State 0=down,1=up	11	00	1111111	11110	1101111000
Total Pow	12		11		
Score	4193144				

Example 1 (supplied pow < aborbed pow)					
	Generators		Loads		
	Main	Aux	Vital	Semi-Vital	Non-Vital
Power (MW)	6	2	0,5	1	0,5
State 0=down,1=up	10	10	1111111	11110	1101111000
Total Pow	10		11		
State after shedding	10	10	1111111	11110	1101100000
Total Pow after shedding	10		10		
Score	4193120				

to supply all the active loads ($up(load\_i)$), then the score is a weighted sum of active load powers. When the power is not enough, the score is calculated by shedding loads with the lowest priority that could not be fed. This lead to rewarding the action of switching on loads with the highest priority and penalizing situations in which the supplied power is less than necessary. An example of usage of the heuristic is shown in Table 2.

When a number of exit nodes are discovered, Algorithm 1 generates solutions by concatenating capabilities from W_I to each exit node.

In this process, MUSA makes a very limited use of physical values to elaborate the solutions. It calculates the available amount of power, and it penalizes configurations in which loads use more power than the available one. In the $search\_solution$ procedure, the role of Matlab becomes fundamental because it allows grounding the conceptual solution by employing Simulink to simulate physical parameters such as the effective current measured at the generators poles, identifying extra-voltage or unstable situations that a symbolic reasoning is not able to evaluate. The outcome of Matlab is to discard unfeasible solutions and to sort the remaining ones according to their quality. The whole adaptation cycle is summarized in Figure 2.

C. The Adaptation Cycle

Most of the modern approach to self-adaptation puts the feedback loop as the core of the architecture. The proposed solution adopts one of the most common models for realizing the feedback loop: the MAPE-K [12] structure, composed of data collection, data analysis, planning and acting. Figure 2 shows the architecture of the solution.

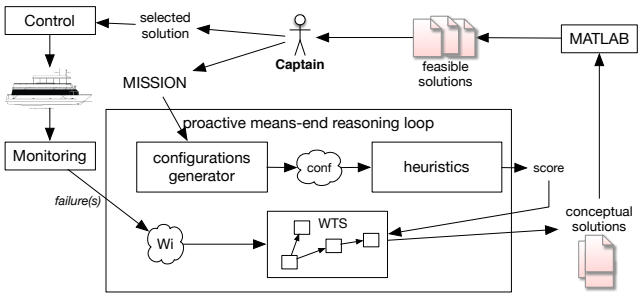

Figure 2: Architecture of the adaptive solution

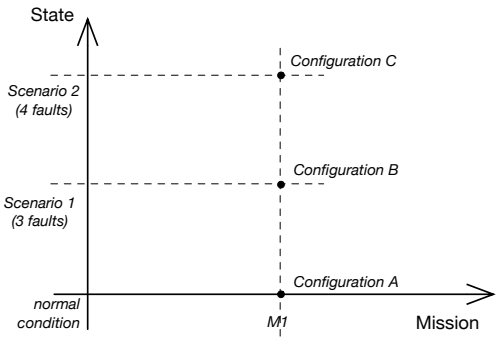

Figure 3: Faults vs Mission

The Monitor Module. The vessel is instrumented with a set of sensors for monitoring some physical variables. The monitor module shall control these sensors to collect raw data with the aim of detecting possible failures.

The Analysis Module. The system should be able of reasoning on raw data to estimate all the relevant vessel conditions (e.g., steady state, electrical failure, etc.) thus obtaining the necessary information to characterize and assess system performance fully. For instance, the analysis should infer the kind and the position of possible electrical failures when they occur.

The Planning. component is responsible for deciding the kind of recovery to enact. The Proactive Means-end Reasoning Module elaborates a configuration for maximizing the continuity-of-service of vital loads during the reconfiguration operations, avoiding instability or even system collapse. According to the current mission and the kind of maneuver, loads are dynamically dealt according to the three categories (vital, semi-vital and non-vital). The contribution of Matlab/Simulink allows selecting feasible solutions via simulation. The design of this module incorporates human factor to enable specialized operators (mainly the captain) to maintain situational awareness and take appropriate measures during normal and emergency conditions.

Execute. The main operations of the SPS reconfiguration are *connection/disconnection* of the loads and the generators. These actions are performed by controlling the automatic switches placed on electrical buses. Controller distribution and autonomy are fundamental features to allow each block may act independently from the rest of the system.

The proposed architecture allows the system to move in a bi-dimensional space generated by the product: Failure x Mission. This space is generically represented in a Cartesian graph like that shown in Figure 3.

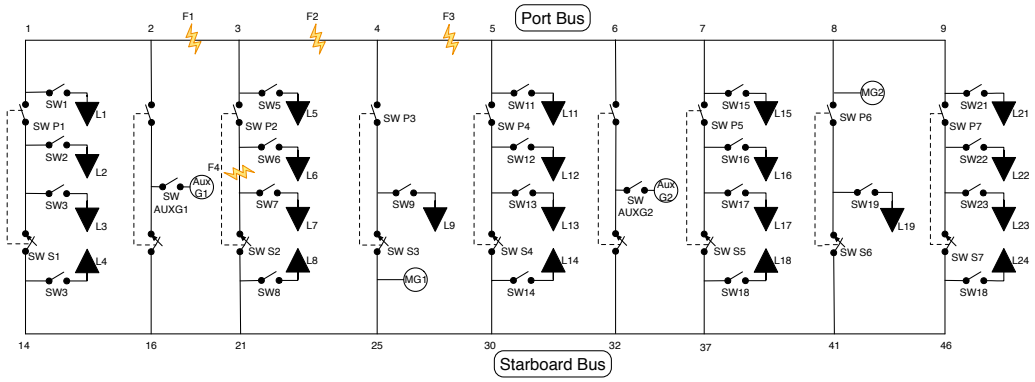

Figure 4: The adopted shipboard power system model.

Table II: Load classification and priority for the reference mission.

MISSION																						
Type	vital							semi-vital								non-vital						
Priority	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
Load	24	21	19	18	15	14	11	22	16	12	7	6	3	2	9	23	17	13	8	5	4	1

The ship captain selects the current mission of the vessel. The mission classifies the loads according to a typology (vital, semi-vital and non-vital) and finally, each of the loads is associated with a priority.

A monitoring module supervises the vessel's status and raises a new adaptation need when it discovers a failure scenario. The current state is described as W_I and it is provided to the proactive means-end reasoning loop. It explores a space of solution driven by the mission's goals and the heuristics. After a fixed number of iterations of Algorithm 1, the system produces a list of conceptual solutions. These are 'conceptual' because the algorithm works on a conceptual description of the electrical topology. It is up to the Matlab simulation to validate these solutions by verifying their feasibility in terms of physical aspects. Therefore, only feasible solutions will be presented to the vessel's captain.

The cycle concludes when the captain selects and makes operative the solution he prefers thus enabling the control sub-system to enact the solution in the real electrical circuit concretely.

The next section explains a couple of reconfiguration scenarios, generated by two different set of failures, producing a trajectory in Figure 3 from normal condition (configuration A) towards respectively configuration B or C.

The proposed architecture is able of operating a reconfiguration also when the vessel's conditions are the same, but the mission changes.

IV. Case Study

In this paper, we analyze a case study inspired by [5] to which we apply the proposed approach for reconfiguring the system when multiple failures occur. The formulation presented in [5] considers a new balanced hybrid (AC and DC) shipboard power system based on a high-performance medium-voltage DC-current (MVDC) ship power system. To allow an evaluation of the proposed approach, in this section we suppose the whole system is DC powered, and it is configured as reported in Figure 4.

This hypothesis does not impact the correctness of the approach evaluation since it regards the identification of alternative/optimal paths for powering as many loads as it is possible without exceeding the available amount of power (also considering the contribution of auxiliary generators if the case).

The proposed ship model comprises seven DC load zones that are powered by two primary generators (MG) and two auxiliary generators (AUXG). Each MG provides up to 6 MW while each AUXG provides up to 2 MW. It is assumed that nonvital loads can be shed to grant the power to the vital and semi-vital loads in case of emergencies. For the sake of simplicity, in the following experiments, we suppose a partial shading of loads is not possible. This simplification does not affect the evaluation of the reconfiguration system because the availability of a partial shading could be handled with the presence of multiple differently sized loads, each one with the only on/off shading capability. Just like in the cited Bose et al. work [5], during SPS reconfiguration, the status (ON/OFF) of switches is set so that needed power is delivered to loads after the occurrence of a fault. The reconfiguration is achieved by prioritizing power delivered to vital loads over semi-vital and nonvital loads.

To demonstrate the results provided by the proposed system, we will study the two different multiple-failures scenarios inspired by [5]: one involving three simultaneous faults and one involving four simultaneous faults.

A. Scenario 1: 3 faults

The first fault scenario (failures FS1+FS2+FS3 in Figure 5) occurs when multiple interruptions happen on the starboard bus. As a consequence of these multiple failures, loads L1, L5, L9 are no more powered. This has a serious impact on mission accomplishing since load L9 is a vital one. Loads L15, L18, L21, L24 are still unpowered because of the initial mission configuration.

The reconfiguration procedure performed by MUSA proposes several solutions. They respect the constraint coming from the maximum amount of available power (also considering auxiliary generators if switched on during the procedure). However, the MUSA module is not aware of the real behavior of the system at the most detailed level, including currents in each node, currents delivered to loads and currents dispatched by generators (that being real have a maximum amount of power they can provide). Indeed, the MUSA module operates at a symbolic level of abstraction. It computes which paths are enabled for current passing once a specific configuration of

Table III: Scenario 1. Results of the reconfiguration process (MUSA side).

config	c1	c2	c3	c4	c5	c6	c7	c8	gen state	load state	score
initial state									1100	11111111111111110000	4194288
fault cond									1100	10011111111010110000	2620784
1	x	x	x		x	x	x	x	1111	11111111111101111111	4194175
2	x		x	x	x	x	x	x	1110	11111111111101111111	4194175
3	x		x		x	x	x	x	1110	11111111111101111111	4194175
4	x		x		x	x	x		1110	11111111111101111110	4194174
5	x		x		x	x			1110	11111111111101111100	4194172
6	x		x		x				1110	11111111111101111000	4194168
7	x		x						1110	11111111111101110000	4194160
8			x						1100	11111111110101110000	4193648

Legend: *config* is the number of solution discovered by MUSA; *c1-c8* are the subset of all the capabilities used in this example (c1=switch_ON_aux1_generator_cap, c2= switch_ON_aux2_generator_cap, c3=open_switch_swp3_close_switch_sws3_cap, c4=open_switch_sw_5_cap, c5=close_switch_sw_15_cap, c6=close_switch_sw_18_cap, c7=close_switch_sw_21_cap, c8=close_switch_sw_24_cap); *gen state* is the state of the four generators (main1, main2, aux1, aux2); *load state* is the state of the loads according priorities (see Table II); *score* is the result of the heuristic of Algorithm 2.

Table IV: Scenario 1. Results of the simulation process (Matlab/Simulink side).

config	overloads	non-powered loads	wrongly non-powered	underused gen	redundant cap	solution size	feasible
1	MG1	L5				7	NO
2	MG1	L5			c4-open SW5	7	NO
3	MG1	L5				6	NO
4		L5-L24				5	YES
5		L5-L21-L24				4	YES
6		L5-L18-L21-L24				3	YES
7		L5-L15-L18-L21-L24				2	YES
8		L1-L5-L15-L18-L21-L24				1	YES

Legend: *config* is the number of solution discovered by MUSA; *overloads* are situations which the current at the ports of a generator is higher than a threshold; *not powered loads* are loads that are not supplied; *wrongly non-powered* are loads that could be supplied with energy but the configuration misses to do; *underused gen* are generators that are used below their possibility; *redundant cap* indicates the solution contains capabilities that could be removed because their effect is null; *solution size* is the number of capabilities that are used in the solution.

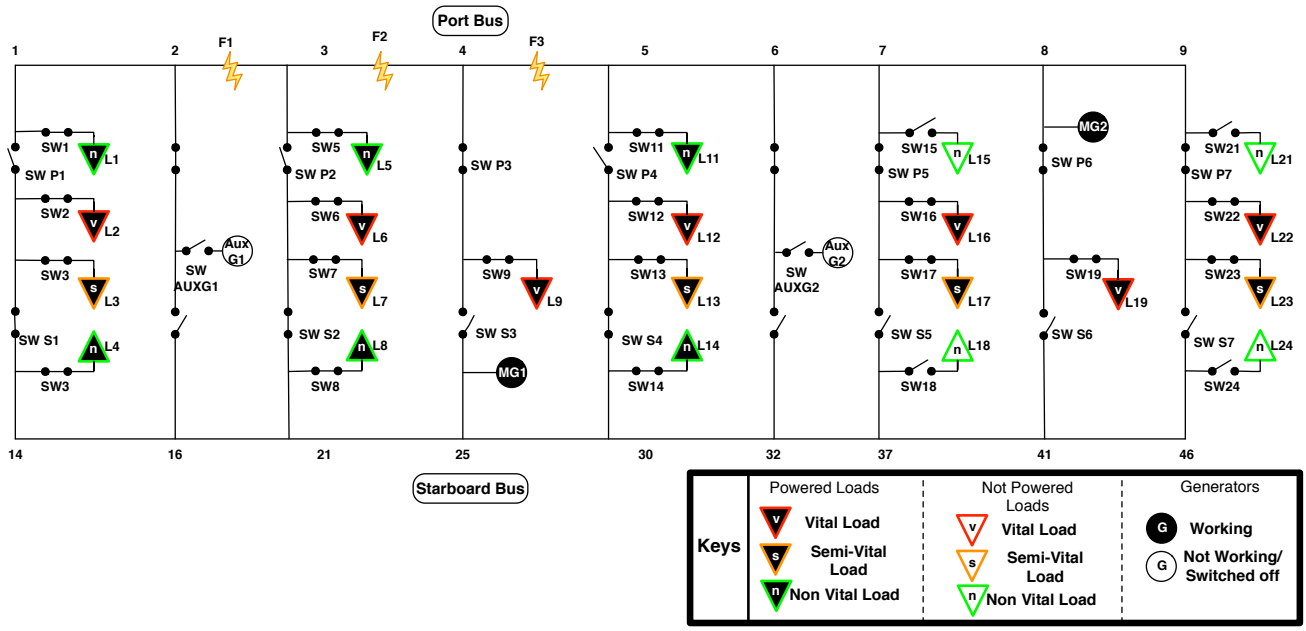

Figure 5: First scenario (3 faults): initial configuration of the system, and faults.

switches is selected and what total amount of current is demanded to generators by the current-reachable loads. By using Matlab/Simulink, our system simulates all the provided reconfiguration procedures and it removes those who violate physical specifications of the real system (for instance maximum amount of power for each generator). Results are reported in Table III. The first two rows of the table report the initial operating conditions selected by the captain according to the mission profile (see also II). It is worth to note that, although no faults are active, some loads are not powered (L15-L18-L21-L24). This descends from the limited power of the two main generators (not sufficient to power all the loads of the vessel) and the non-vital role of some loads for the mission. The quality of service (score) for this configuration is 4'194'288. After the three faults (Figure 5), the quality of service drops down to 2'620'784. This happens because loads L1-L5-L6-L9-L15-L18-L21-L24 are no more powered as a consequence of the faults. This is the initial condition the proposed reconfiguration approach has to cope with. The configurations generator proposes 8 different solutions to the problem as reported in Table III. Each configuration employs a different set of capabilities. As we can see looking at the score column, the first three proposed configurations achieve the same score result but they use a different set (and number) of capabilities to do that. Oddly, configuration 1 activates the auxiliary generator AUX2 without any evident advantage with regards to the following two configurations. Configuration 2 proposes to open switch sw5 (controlling load L5) but since this is not reachable anyway, the action has no effect on the result. From configuration 4 to 8, a growing number of loads is disconnected from power, this causes a decrease in the quality of service coming with a diminishing need for power (configuration 8 does not even need auxiliary generator AUX1) and the number of employed capabilities.

In order to better illustrate the proposed approach, we will study two configurations. The first one (configuration n.1 from Table III) prescribes the following operations:

```
cap: switch_ON_aux1_generator_cap
cap: close_switch_sw_15_cap
cap: close_switch_sw_18_cap
cap: close_switch_sw_21_cap
cap: close_switch_sw_24_cap
cap: switch_ON_aux2_generator_cap
cap: open_switch_swp3_close_switch_sws3_cap
```

The first step consists in switching on the generator AUXG1, then loads L15, L18, L21, and L24 are powered, the generator AUXG2 is switched on, and, finally, the transversal bus 3 configuration is changed (by opening switch SWP3 and closing SWS3). The reader will note that the prescribed operations do not follow a precise or logical order (for instance the two auxiliary generators are not switched on together). This is an obvious consequence of the configurations generator algorithm for solution space (WTS, see Figure 1) exploration and of the simplification implied by not studying transitory intermediate configuration states. The reconfiguration solution is supposed to be entirely applied at the same time (not a big issue when working in DC although some aspects will be further studied in the future).

The second reconfiguration solution we will study configuration n. 4 from Table III) prescribes the following operations:

```
cap: switch_ON_aux1_generator_cap
cap: close_switch_sw_15_cap
cap: close_switch_sw_18_cap
cap: close_switch_sw_21_cap
cap: open_switch_swp3_close_switch_sws3_cap
```

The procedure switches on auxiliary generator 1, together with loads L15,L18,L21. The configuration of transversal bus 3 is reversed as in the previous configuration.

Differences between these two configurations become evident after their simulation with the Matlab module. The overall results of the Matlab simulations are reported in Table IV). This summarizes the most relevant problems that can be found by using a physical-level simulation of the circuit. The first column reports the number of configurations, the second column reports the overloaded generators (if any). The first three configurations overload the generator MG1 thus becoming unacceptable (see the last column of the table, column 'feasible'). This condition may not be discovered at the symbolic level, since it only performs a global balance of power (demanded power vs available power). In reality, it may happen that power required to the available generators is not equally distributed and one of them may overload while the other remains well under its working limits. The third column lists loads that are not powered in the proposed configuration. This is directly linked to the quality of service score (from the previous table). Solutions with better scores are to be preferred if they satisfy the goal requirements (all vital loads are powered). The fourth column reports the list of loads that could be powered according to the circuit configuration, but they are switched off by the wrong use of a capability. This phenomenon does not happen in this scenario 1, but it will be present in the scenario 2 (see Table VI). Column 'underused gen' lists the generators that are switched on by the proposed configuration but their power is not effectively used according to the Matlab simulation (in other words they do not really provide any power). Again, this happens in scenario 2. Column 'redundant cap' lists the capabilities (better their scope) that are employed in the configuration but do not provide any effect (for instance the already discussed use of c4 in configuration 2). Column 'solution size' reports the number of employed capabilities. This is a sensitive metrics since we prefer shorter (and therefore intuitively simpler) solutions when they achieve the same score. Finally, column 'feasible' summarizes the previous results and it marks as acceptable solutions that do not violate physical limits of the circuit behavior (such as generator overloads).

Going back to the previously studied configurations n.1 and n.4, we can see that the Matlab simulation of the proposed solution n.1 reports that one generator (MG1) is overloaded and one load (L5) is not powered. This solution is therefore not feasible. Conversely, the simulation of configuration n.4 proves it abides the limits imposed by the electrical components, and it is therefore feasible. In this configuration, loads L5 and L24 are not powered but they are listed as non-vital in this mission; therefore this is not a problem. The two cases show the importance to clean the solutions provided by the configurations generator with the simulations done by a module that is well aware of the behavior of the physical layer of the system (Matlab in our case). Considering the results proposed in Table IV, we can see that the best solution is configuration n.4 that achieves a score of 4'194'174 and requires five capabilities. Following solutions (n.5-6-7-8), although feasible, achieve a lower score (in fact fewer loads are powered by these solutions) but also use a smaller number of capabilities, therefore may be useful in a real scenario when something could go wrong in applying the preferred solution n. 4.

B. Scenario 2: 4 faults

The second fault scenario studies the case when multiple interruptions occur on both the starboard and portboard buses; more

Table V: Scenario 2. Results of the reconfiguration process (MUSA side).

config	c1	c2	c3	c4	c5	c6	c7	c8	gen state	load state	score
initial state									1100	11111111111111110000	4194288
fault cond									1100	01011110111100011100000	1555568
1	x	x	x	x	x	x	x	x	1111	11111111111010111010	4193658
2	x		x	x	x	x	x	x	1110	11111111111010111010	4193658
3	x	x	x	x			x	x	1111	11111111111010111010	4193658
4	x		x	x			x	x	1110	11111111111010111010	4193658
5	x		x	x			x		1110	11111111111010111000	4193656
6	x		x	x					1110	11111111111010110000	4193648

Legend: *config* is the number of solution discovered by MUSA; *c1-c8* are the subset of all the capabilities used in this example (c1=switch_ON_aux1_generator_cap, c2= switch_ON_aux2_generator_cap, c3=open_switch_swp3_close_switch_sws3_cap, c4=open_switch_swaux1p_close_switch_swaux1s_cap, c5=open_switch_sw_2_cap, c6=close_switch_sw_2_cap, c7=close_switch_sw_15_cap, c8=close_switch_sw_21_cap); *gen state* is the state of the four generators (main1, main2, aux1, aux2); *load state* is the state of the loads according priorities (see Table II); *score* is the result of the heuristic of Algorithm 2.

Table VI: Scenario 2. Results of the simulation process (Matlab/Simulink side).

config	overloads	non-powered loads	wrongly non-powered	underused gen	redundant cap	solution size	feasible
1	N	L1-L5-L18-L24		AUX G2	c5-c6 open/close SW2	8	YES
2	N	L1-L5-L18-L24			c5-c6 open/close SW2	7	YES
3	N	L1-L5-L18-L24		AUX G2		6	YES
4	N	L1-L5-L18-L24				5	YES
5	N	L1-L5-L18-L21-L24	L21			4	YES
6	N	L1-L5-L15-L18-L21-L24	L15-L21			3	YES

Legend: *config* is the number of solution discovered by MUSA; *overloads* are situations which the current at the ports of a generator is higher than a threshold; *not powered loads* are loads that are not supplied; *wrongly non-powered* are loads that could be supplied with energy but the configuration misses to do; *underused gen* are generators that are used below their possibility; *redundant cap* indicates the solution contains capabilities that could be removed because their effect is null; *solution size* is the number of capabilities that are used in the solution.

precisely between nodes 3-4, 4-5 (these two faults are the same of the previous scenario), 16-21, 32-37 (see Figure 6). This is quite a disruptive scenario since it represents the situation where the aft part of the vessel is strongly damaged (faults F2, F3, F5 in Figure 6) and another fault hits the starboard bus near the bow (F6). The configurations generator module provides six solutions to cope with this scenario (see Table V). In this case, the fault conditions cause a drop in the quality of service score that is even worse than in the previous scenario. In fact, a large set of loads becomes inactive (L1, L2, L3, L4, L5, L9, L15, L18, L21, L24), some of them are vital (L2, L9) or semi-vital (L3). The proposed set of configurations (see Table V) presents a peculiarity: some capabilities have a mutually exclusive effect, this happens for c5 and c6 that respectively open and close switch SW2. Moreover, configurations n.1 and n.3 switch on the auxiliary generator AUX2, but Matlab simulation proves this is totally ineffective in the current situation. Configurations n.5 and n. 6 do not close switches controlling loads L15 and L21 thus missing the opportunity to power them. The best solution is to be found in the first 4 configurations since they achieve the best score. Among them, configurations n.4 would be the preferable ones, indeed, it employes the lowest number of capabilities. In fact, solution 1 wrongly attempts to use AUX2 (no power taken from that by the circuit) and to open/close SW2 (the two operations cancel one the other), solution 2 wrongly opens/closes SW2, solution 3 wrongly switches on AUX2.

The two previous experiments show the ability of the proposed system to move on the vertical axis of Figure 3 thus proving it can respond in a not a-priori configured way to a change in the environment (different fault scenarios) by proposing more than one reconfiguration solutions. It is worth to note that the proposed system could easily automatically identify and enact the best solution but we decided not to implement that because in real scenarios, the final responsibility for the adoption of a reconfiguration strategy should always be on the person in charge.

V. Conclusions

This paper presented an adaptive architecture for dealing with the reconfiguration of Shipboard Power Systems (SPSs) that is the component responsible for supplying energy to various services of a vessel. The proposed solution adopts MUSA, a generic-purpose self-adaptive middleware, as the base for engineering the reconfiguration system. We have extended the main concepts of MUSA by introducing the new concept of Mission, a dynamic container of goals, associated with their priorities, to be considered when reconfiguring the system. Whereas the advantage of MUSA is the ability to reason at the symbolic level, we have also added a physical simulator built with Matlab to validate the solutions. We finally proposed a case study in which we discuss two different failure scenarios, and we demonstrate how the system behaves in different circumstances.

References

[1] I. Hwang, S. Kim, Y. Kim, C. E. Seah, A survey of fault detection, isolation, and reconfiguration methods, IEEE Transactions on Control Systems Technology 18 (3) (2010) 636–653. doi:10.1109/TCST.2009.2026285.

[2] W. M. Dahalan, H. Mokhlis, Techniques of network reconfiguration for service restoration in shipboard power system: A review, Australian Journal of Basic Applied Science 4 (11) (2010) 55565563.

[3] K. C. Nagaraj, J. Carroll, T. Rosenwinkel, A. Arapostathis, M. Grady, E. J. Powers, Perspectives on power system reconfiguration for shipboard applications, in: 2007 IEEE Electric Ship Technologies Symposium, IEEE, 2007, pp. 188–195.

[4] L. Agnello, M. Cossentino, G. De Simone, L. Sabatucci, Shipboard power systems reconfiguration: a compared analysis of state-of-the-art approaches, in: Smart Ships Technology 2017, Royal Institution of Naval Architects (RINA), 2017, pp. 1–9.

[5] S. Bose, S. Pal, B. Natarajan, C. M. Scoglio, S. Das, N. N. Schulz, Analysis of optimal reconfiguration of shipboard power systems, IEEE Transactions on Power Systems 27 (1) (2012) 189–197.

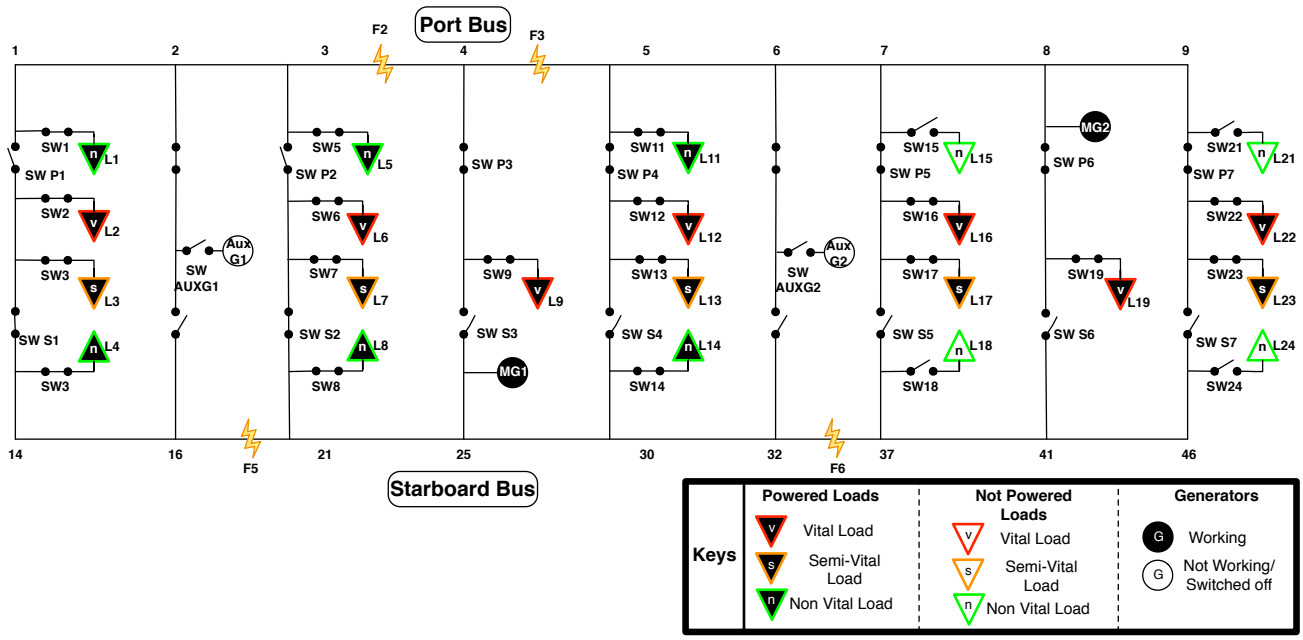

Figure 6: Second scenario (4 faults, both buses involved): initial configuration of the system, and faults.

[6] IEEE, Recommended practice for shipboard electrical installations – systems engineering, IEEE Std 45.3-2015 (2015) 1–74doi:10.1109/IEEESTD.2015.7172975.

[7] J. O. Kephart, D. M. Chess, The vision of autonomic computing, Computer 36 (1) (2003) 41–50.

[8] L. Sabatucci, C. Lodato, S. Lopes, M. Cossentino, Towards self-adaptation and evolution in business process., in: AIBP@ AI* IA, Citeseer, 2013, pp. 1–10.

[9] L. Sabatucci, P. Ribino, C. Lodato, S. Lopes, M. Cossentino, Goalspec: A goal specification language supporting adaptivity and evolution, in: International Workshop on Engineering Multi-Agent Systems, Springer, 2013, pp. 235–254.

[10] L. Sabatucci, M. Cossentino, From Means-End Analysis to Proactive Means-End Reasoning, in: Proceedings of 10th International Symposium on Software Engineering for Adaptive and Self-Managing Systems, Florence, Italy, 2015.

[11] M. Gelfond, V. Lifschitz, Action languages, Computer and Information Science 3 (16).

[12] P. Vromant, D. Weyns, S. Malek, J. Andersson, On interacting control loops in self-adaptive systems, in: Proceedings of the 6th International Symposium on Software Engineering for Adaptive and Self-Managing Systems, ACM, 2011, pp. 202–207.