Methodological Approach for Developing Reconfigurable Automation Systems

Isabel Sarachaga, Arantza Burgos, María Luz Alvarez, Nagore Iriondo, and Marga Marcos, *Senior Member, IEEE*

***Abstract*—One challenge for the designers of the current industrial control systems consists of assuring the system reconfiguration at runtime, envisaged at system design time. This article focuses specifically on the fault tolerance to controller failures (as it is the most complex challenge) aiming at redistributing the responsibilities at runtime in order to finish the current production plan. Its applicability is illustrated by means of a case study that consists of a reconfigurable control application based on the IEC 61131-3 for a flexible assembly cell. A prototype software tool guides the developer during the development process. This approach contributes to develop more flexible control systems because they can react in case of a controller failure, to increase the system availability during its operation and to reduce the production downtimes to perform the maintenance tasks.**

***Index Terms*—Industrial automation, IEC 61131-3, methodology for industrial automation systems (MeiA$_\bullet$), reconfigurable control applications.**

I. Introduction

INDUSTRY 4.0 focuses on creating smart factories that lead to customized production with zero defects. To achieve this, flexibility, availability, and robustness in manufacturing systems are fundamental [1].

In this context, an efficient operation of the control system requires advanced software engineering solutions as they have a great impact on the efficiency of automation systems [2]. Vogel-Heuser *et al.* [3] introduce the essential challenges of software engineering and the requirements that software has to meet in the specific domain of automation. Vogel-Heuser *et al.* in [4] discussed the challenges, state of the art, and research road map of the automated production systems. One important conclusion is that the ability of a system to operate correctly despite the presence of faults is a big challenge. Thus, dynamic adaptation becomes one of the most important requirements in order to increase the efficiency of the production processes [5]. *Schutz et al.* [6] also remark that dynamic reconfiguration of the production system at runtime is a suitable approach for achieving such adaptation.

In the field of automation, dynamic reconfiguration at runtime (Requirement 1) aims at optimizing the production process avoiding system stops in order to finish the current production plan. Notwithstanding this, the problem is difficult to solve, even more if redundant equipment is not available. Fault tolerance to controller failure requires providing the automation system with mechanisms to detect the failure and to decide how to recover the production process. Additionally, in the absence of redundant equipment, the software responsible for controlling the physical modules must coexist in the controller that will assume the control responsibilities of the failed controller. As the software architecture become complex, it would be desirable to introduce methodologies and support tools (Requirement 2). This would allow identifying physical modules and their corresponding self-contained software that might be designed and tested using widely established methods and techniques in the automation field (Requirement 3).

On the other hand, it is advisable to take advantage of the maturity reached by software engineering disciplines. In [7], a comparative analysis of different software engineering methods suitable for intelligent manufacturing is done. One of the main conclusions of the authors is the strong need of addressing methodological aspects in the design of complex intelligent manufacturing systems, and furthermore, if advanced technologies from the software engineering field are to be applied.

Another important aspect when developing industrial solutions is to involve the technical personnel in the development (Requirement 4). They know the production process in depth, and thus, the conditions that may lead to failure situations, and even more important, when and how a failure can be recovered.

There are several approaches in the literature, analyzed in Section II, that cover a subset of these aspects. But, as far as authors know, the coverage of all requirements is not completely meet.

In this context, this article contributes a methodological approach to develop reconfigurable control applications using the industrial standards and where the production process experts participate. The approach is based on the redistribu-

Manuscript received May 3, 2018; revised October 24, 2018 and March 13, 2019; accepted June 15, 2019. Date of publication July 18, 2019; date of current version January 16, 2020. This work was supported in part by the University of the Basque Country (UPV/EHU) under Project GIU18/162 and in part by the Ministerio de Ciencia, Innovación y Universidades RTI2018-096116-B-I00. Paper no. TII-18-1119. *(Corresponding author: María Luz Alvarez.)*

The authors are with the Department of Automatic Control and Systems Engineering, Escuela de Ingeniería de Bilbao, University of the Basque Country, 48013 Bilbao, Spain (e-mail: isabel.sarachaga@ehu.eus; arantzazu.burgos@ehu.eus; marialuz.alvarez@ehu.eus; nagore.iriondo@ehu.eus; marga.marcos@ehu.eus).

Color versions of one or more of the figures in this article are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TII.2019.2925837

tion of control responsibilities at runtime under failures by providing dynamic allocation of control resources, without using redundant (hot standby) controllers.

A previous work of authors [8] proposed a methodology and support tool that guide the development of control applications based on the IEC 61131-3 standard. This article goes a step further and allows the design and development of the fault-tolerant control systems to controller failure at run time. It contributes new steps of the methodology for industrial automation systems (**MeiA$_\bullet$**) and supporting tool through the analysis and design phases. Furthermore, this approach involves the designer in the implementation making use of the techniques industrialists are used to. It guides the developer to define the information needed to recover from faults as well as to implement the failure recovery.

This article is organized as follows. Section II reviews the works in manufacturing systems that address, even partially, the identified requirements, assessing their pros and cons of the approaches. Section III describes the scenario to deal with full support to controller failure and recovery. It also provides a brief overview of the **MeiA$_\bullet$** methodology, detailing the seven steps to perform the analysis and design required to address the dynamic reallocation of control resources at run time. The section ends discussing the implementation of reconfigurable control applications based on the IEC 61131-3 standard. In Section IV, the validation of the proposal is performed using an industrial platform through a case study, validating that the information obtained through the seven steps allows reconfiguring the control system and resuming the production process. This section also presents, as a proof of concept, a prototype software tool that guides the developer during the development process. Finally, Section V concludes this article.

II. Related Work

The closest works related to control infrastructure reconfiguration deal with the adaptation of control logic in operation and the redistribution of control software at runtime to avoid hardware redundancy. The related works are analyzed from the following requirements point of view: R1) support for the system reconfiguration; R2) methodological design approach; R3) use of industrial standards; and R4) expert involvement.

Self-configuration and restart after a fault to become operable again and increase the overall equipment effectiveness is the challenge addressed in [9]. The authors propose to isolate single process faults and to implement automatic reactions to them, avoiding machine or plant shutdowns and operator intervention. Thus, it is a monolithic solution that integrates the detection and recovery within the control software. Besides, this article does not address how to deal with the control system design.

Most of the research on the control system reconfiguration addresses reconfiguration at the highest control layers, focusing on self-reconfiguration intelligence of the whole control system. But industry practice places greater emphasis on reducing the control design cost and provision for more moderate levels of reconfiguration [10]. Following, related work making use of novel technologies in the manufacturing field, such as multiagent systems (MAS), ontologies, model-driven engineering (MDE), or service-oriented architecture (SOA), are analyzed.

Leitao *et al.* [11] summarize long-term research and development of multi-agent industrial automation systems. This technology was used to implement the adaptive holonic control architecture ADACOR2 that combines the holonic design principles with the agent-based technology and empowers this combination with bioinspired mechanisms (namely self-organization principles) [12]. This article is an interesting proposal for reconfigurable architectures but it is not clear how the industrial automation designer is involved and it is not based on industrial standards. Barbosa *et al.* [12] proposes a generic and customizable multiagent architecture that monitors the quality of service of an automation system, and if needed, triggers a reconfiguration of the control system to recover it. The basic mechanism is to reconfigure the system architecture by replicating the control code of a part of the process (a mechatronic component) in different controllers. Again, the approach is based on the use of the MAS technology, which is hardly used in industry.

Agent technology and MDE have proven potential in various prototypical implementations and academic environments. Lepuschitz *et al.* [14] present a survey on standards and ontologies for the process automation domain. Lepuschitz *et al.* [14] provide a model-based design that collects all the information necessary to characterize the availability requirements of the automation systems and a tool that uses the automation plant design to automatically update/extend the system implementation with the appropriate code to support the availability requirements. The control system is divided into several mechatronic components (MCs) that could be designed using the **MeiA$_\bullet$** methodology. The goal is to generate the code for the MAS approach described in [12]. Wehrmeister *et al.* [16] combine aspects and object-orientation in the MDE to design distributed industrial mechatronic systems that provide the reconfiguration of active objects. This article also assesses both technologies, comparing their strengths and weaknesses. This proposal lacks a methodology and it is based on technologies coming from the software engineering domain.

An overview about how the SOA concept has been implemented at the “Digital Factory” is provided in [17]. The combination of holonic and MAS approaches and SOA techniques can also be effective to design the reconfigurable and distributed manufacturing control system. da Silva *et al.* [18] present a systematized procedure that describes the required data and techniques to design a service-oriented active holonic control system. Although it achieves manufacturing reconfiguration, it is not clear who and based on what information decides when a failure situation is recoverable. Besides, the level of knowledge on IT technologies demanded from the domain experts is very high.

The IEC 61499 addresses distributed automation systems and there are many works focusing on the control reconfiguration based on it. da Silva *et al.* [18] gives a good review on the control system design and supporting tools for IEC61499 applications, including the promising application area of flexible and reconfigurable automation systems.

TABLE I
ASSESSMENT OF RELATED WORKS

	R1	R2	R3	R4
[9]	-	-	-	√
[12]	√	-	-	-
[13][15][21]	√	√	-	-
[16]	√	-	-	-
[18]	√	√	-	√
[20]	√	-	√	-
[22]	√	-	√	-
[23]	√	-	√	-

A comparison between the MAS and IEC 61499 FBs is presented in [20]. It concludes the following:

1) both control strategies can be used for implementation in the reconfigurable systems;
2) the MAS platforms are more mature than the currently FB development environments.

This article focusses on implementation, but it neither presents a methodology nor the domain expert involvement in the development process.

The combination of the MAS and the IEC 61131-3 is addressed in [21]. The authors extend previous works with an architecture, which provides reconfiguration mechanisms intended to reconfigure the IEC 61131-3 control software using a multiagent system. As the previous work, this approach uses the MAS technology, which is hardly used in the industry. Hoffman and Basson [22] present an IEC 61131-3-based holonic control according to the ADACOR architecture. When comparing this approach to the MAS, the authors observe that the reconfiguration times and effort required should be similar for both approaches, but the skill levels required are quite different because experience with the MAS is rare among industrialists. They also notice that IEC 61131-3-based implementations are more suitable to lower the control levels, while MAS advantages are more effective at higher control levels. Again, it lacks a methodology to use the approach. Besides, it is not clear who decides if the failure is recoverable and how, as the domain expert is not involved. Merz *et al.* [23] present a method for dynamic reconfiguration by means of runtime deployment. Reconfiguration times are acceptable although they are highly dependent on the control software, the underlying hardware, and the deployment algorithm. Methodological aspects and the domain expert role are not addressed.

Table I summarizes the analysis of the commented approaches.

In summary, these works, although addressing reconfiguration or fault recovery based on different approaches, do not cover all the aspects identified as desirable to make reconfigurable production systems a reality. One of the reasons might be the penetration of novel technologies, which are unknown for industrialists that uses secure, enough proven, and reliable technologies (the PLC is still the controller per excellence). Besides, the designer neither understands nor participates on the implementation. Consequently, abstraction technologies are hardly accepted probably due to the supervision is performed automatically and even decisions are made at runtime without intervention of the operator. One solution to this could be to use them as hidden technologies in order to give support through methodologies and tools. This is one of the goals of this article.

III. Full Support to Controller Failure Detection and Recovery

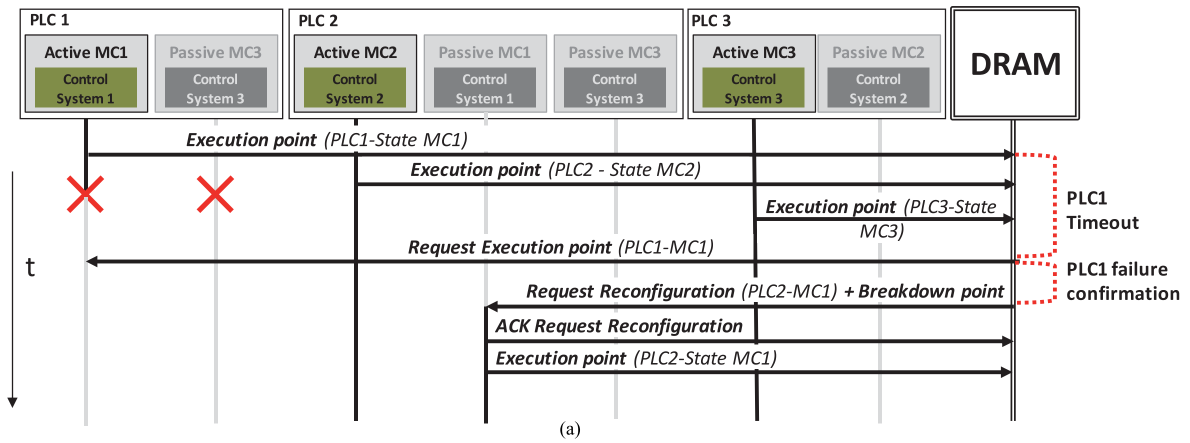
The dynamic reallocation of control resources at run time requires mechanisms to detect controller failure, diagnose the cause, and decide the recovery actions. The controller failure is detected by the dynamic resource allocation manager (DRAM) when the control system state is not received within a timeout. This manager has the configuration information about the resources (PLC i) with their modules (self-contained software for control a physical equipment). Each PLC contains actives modules that control different physical equipment and passive modules that will become actives when the DRAM decides that they must assume the control responsibilities of a failed controller. So, the DRAM has also the strategies for control resource reallocation when a PLC failure occurs.

These strategies use the control system states (called *control system points*) related to each reconfiguration situation, which allows identifying the execution point of the process with different precision degree depending on the process information. Therefore, a control system point can be related to different process points. Different types of control system points have been identified: *execution point* (current state of the control system), *breakdown point* (last recorded state of the control system before controller failure), and *restart point* (state of the control system that must be launched in a backup controller). A breakdown point can be a *direct point* if it allows resuming the control system directly (without recovery actions), or a *critical point* if it does not allow resuming the control system directly. A critical point can be a *recovery point*, which requires performing recovery actions or resuming the control system from another state (without recovery actions), or a *nonrecovery point*, which does not allow resuming the control system.

Likewise, a process state (called *process point*) can be related to different control system points. The previous classification also applies to process points taking into account the process instead of the control system.

Fig. 1 shows a possible reconfiguration scenario. At the top [see Fig. 1(a)], the sequence diagram illustrates the interaction among the entities of the system (PLCs and the DRAM) from normal operation to a failure detection and actions to recover it. Fig. 1(b) illustrates the system after the recovery. The automation system comprises three PLCs. In the example, every PLC has one active control module, and one or more passive modules. During the design phase, it is assured that several control modules can coexist in the same PLC.

When the DRAM does not receive the execution point of MC1 before timeout expires, it confirms the PLC1 failure and applies the reconfiguration strategies. In this case, the DRAM requests the activation of the MC1 module to the PLC2, sends the corresponding breakdown point, and waits for acknowledge.


PLC 1
Active MC1
Control System 1
Passive MC3
Control System 3
PLC 2
Active MC2
Control System 2
Passive MC1
Control System 1
Passive MC3
Control System 3
PLC 3
Active MC3
Control System 3
Passive MC2
Control System 2
DRAM
Execution point (PLC1-State MC1)
Execution point (PLC2 - State MC2)
Execution point (PLC3-State MC3)
Request Execution point (PLC1-MC1)
Request Reconfiguration (PLC2-MC1) + Breakdown point
ACK Request Reconfiguration
Execution point (PLC2-State MC1)
PLC1 Timeout
PLC1 failure confirmation
t
(a)

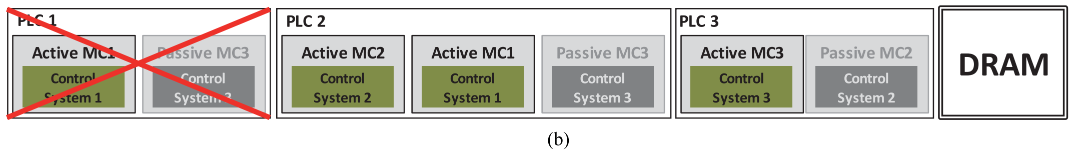


Fig. 1. Reconfiguration scenario.

Once MC1 will be active in the PLC2, it will send the execution point to the DRAM.

A. Dynamic Reallocation of Control Resources

The dynamic reallocation of control resources is addressed through seven steps that use the detailed design of a control system developed applying the **MeiA•** methodology.

MeiA• methodology combines the maturity of software engineering disciplines with methods and standards widely used in the automation field. Six phases manage the analysis and design of the system from different perspectives. Each phase is related to one operation mode: the first phase (main sequence), the second phase (manual operation mode), and the third phase (test operation mode) organize the start-up and the safe stop of the control system in automatic, manual, and test operation modes, respectively; the fourth phase (failures) identifies, analyses, and evaluates the potential process failures; the fifth phase (emergency) provides the start-up and the safe stop of the control system for emergency treatment; and the sixth phase (normal production) organizes the production cycle.

Each phase comprises a set of steps that guides the analysis and design of an industrial control system to obtain the GEMMA (Guide des Modes d'Etude et d'Arrêts Marches) [24] and use Case diagrams, the GRAFCETs (Graphe Fonctionnel de Commande, Etapes, Transitions) [25], [26] in charge of organizing and coordinating the possible system states, and the control signals to be used in the GRAFCETs related to the actions of the production cycle. Moreover, these steps allow identifying the control and monitoring requirements, as well as the configuration of the operator panel and other required auxiliary panels.

The detailed design obtained when applying **MeiA•** methodology (Phases I–VI) must be inspected in order to analyze the process points that allow identifying the control system points related to each reconfiguration situation. Then, the following seven steps perform the analysis and design required to address the reconfiguration of the control resource allocation.

Step 1. Definition of the Control System Point: The internal signals of every procedure are analyzed in order to identify those required to define the control system point: the value of the steps (active/inactive), the signals related to the main states (start, initial conditions, start-up process, stop at the end of the cycle, etc.), coordination and information signals, counters, timers, etc., and other signals that must be considered to perform the reconfiguration diagnosis and to resume the control system after the control resource reallocation.

Step 2. Communication of the Execution Point: The following aspects related to the execution point of the control system are addressed: how to obtain it, when and how it is updated, and when and who receives it. For example, it can be obtained at each PLC cycle, periodically, or it can be asked by the DRAM.

Step 3. Diagnosis of the Process Points: The critical points of the process are identified and analyzed by the designer (who knows the production process) aiming at finding the direct points and the critical points of the control system. It requires the previous identification of the critical elements in the process (suction cups, magnets, operations with time restrictions, etc.) related to control system variables and the procedures that manage these elements. This analysis could conclude that more sensors are needed for providing information about the critical points.

Step 4. Characterization of Critical Points: The control system point (or points) related to every process critical point must be characterized according to the format defined in the step 1. This characterization requires the identification of procedural steps and other needed signals (with their values).

Step 5. Recovery Points: The way to resume the control system and the process must be established for each process recovery point.

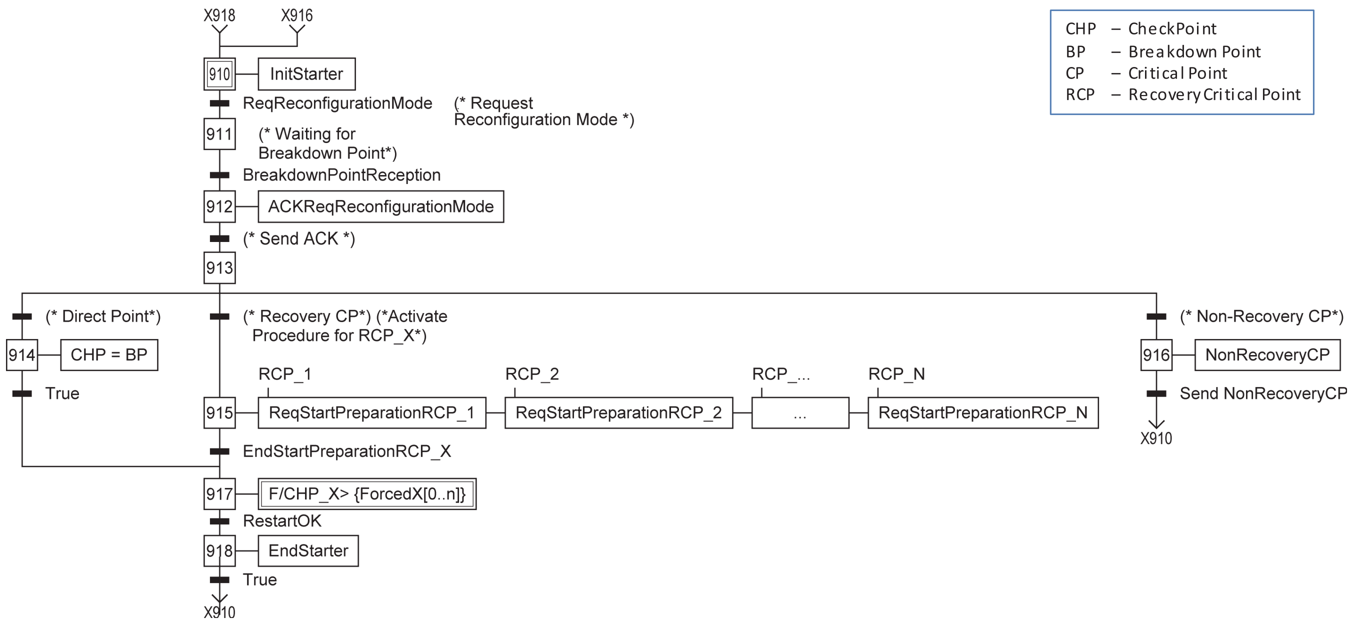


Fig. 2. GRAFCET template for starter.

Step 5.1. Actions: The need to complete certain tasks before resuming the control system in another controller must be established (for instance, remove some part, take some actuator to a certain position, visualize some message, unlock mechanical actuators, etc.).

Step 5.2 Restart point (CheckPoint): The restart point of the control system must be established.

Step 6. Request of Reconfiguration Mode: The activation procedure of the reconfiguration mode must be defined as well as the way to give the warning.

Step 7. Confirmation of Resumption: The signals indicating the correct resumption of the control system must be identified.

During the analysis, these steps identify the GEMMA states associated to the reconfiguration, the transactions between the states and the evolution conditions. When the control system is resumed in another controller, if the controller failure happened in a recovery point and previous tasks are required, the system will evolve to "preparation to restart after failure" state (identified as A5 in GEMMA), before reaching the state induced by the restart point.

These steps also identify the use cases that describe the functionality, the actors that participate in them and the preconditions that must be met to implement these use cases. The "request restart in another controller" use case may include the "send reconfiguration warning" and "prepare restart after controller breakdown" use cases.

After the analysis, the elements that support the dynamic reallocation of control resources at run time must be designed and coded. These elements are the DRAM (that could be implemented using different technologies), "state reporter," and "starter."

The first design step consists of customizing the GRAFCET templates called "state reporter" and "starter." The former provides the state of the control system (execution point), while the later performs the required actions and activates the control system in another controller using the breakdown point of the control system after the controller failure. The external signals, the signals from sensors and to actuators, the signals from the operator panel, and the signals from the supervisor will be reassigned to the new controller.

The "starter" sends the reconfiguration acknowledge and evaluates the breakdown point: if it is a direct point, the restart point will be the breakdown point; if it is a nonrecovery point, a notification will be sent; and if it is a recovery point, the "restart preparation" procedure associated to the critical point will be activated. This procedure performs the recovery actions before resuming the control system in another controller and determines the restart point. As an example, Fig. 2 illustrates the template used to generate the code of the starter module that the PLC must execute after receiving a reconfiguration request from the DRAM (see Fig. 1).

Fig. 3 illustrates the dynamic reallocation of the control resources at run time. When PLC1 failure occurs, the DRAM applies the reconfiguration strategies and sends the breakdown point and the request to activate the MC1 module to the PLC2. The "starter" of MC1 evaluates the breakdown point, determines the restart point, and sends the reconfiguration acknowledge.

B. Reconfigurable Control Applications Based on the IEC 61131-3 Standard

The design model involves the minimal units of design (called design organization units—DOUs) expressed as a GRAFCET and needed to generate the program organization units (POUs) in one to one correspondence.

In order to transform the design (technological GRAFCETs) into code, two different aspects must be considered: the sequential part that comprises the activation and deactivation

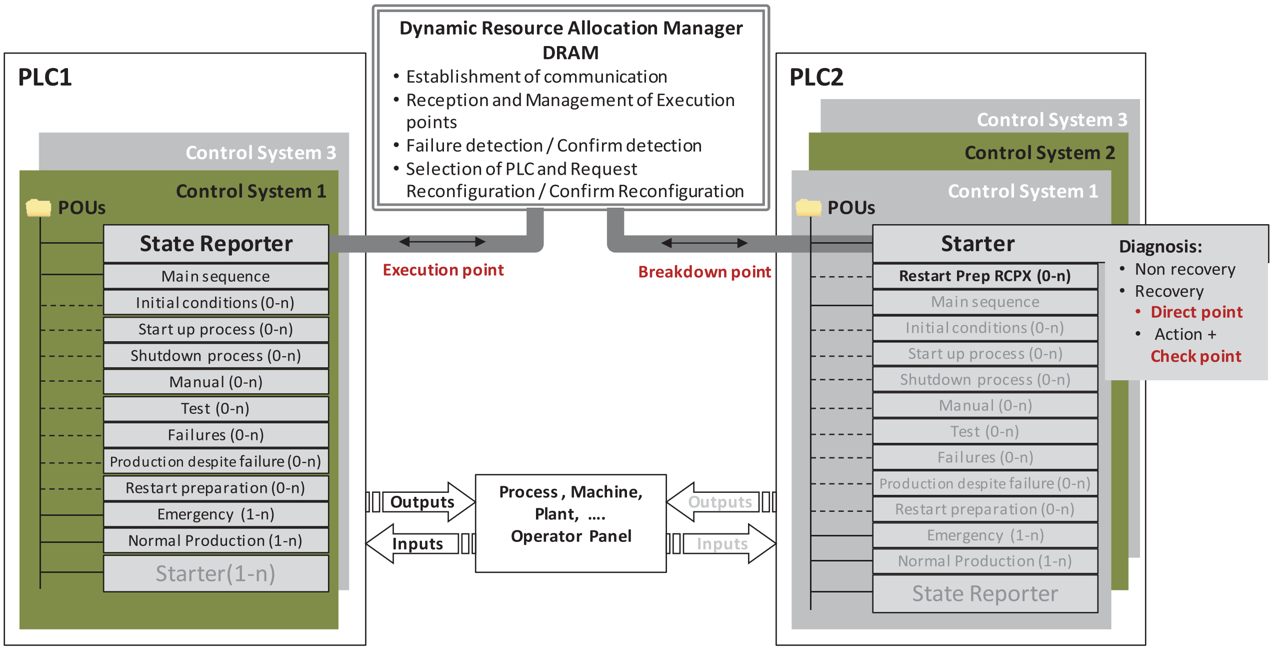


Fig. 3. PLC running Module 1 failure and recovery process.

sequence of the GRAFCET steps, and the combinational part that generates the system outputs based on the current active steps. Note that each output is generated once in the automation project according to the design requirements for the industrial automation systems. The definition of each variable must also be taken into account.

The sequential part is implemented by function blocks (a POU type) that include the declaration of variables, and the activation and deactivation of each step. A step is activated when the previous step is activated (or steps) and the transition condition is fulfilled, whereas a step is deactivated when the successor step is activated (or steps). In addition, two input variables have been added to each POU: *Init* for activating the initial steps of the GRAFCETs when a reset to initial state is required, and *Reset* for deactivating all steps when an emergency stop occurs. Furthermore, other two input variables are necessary to implement the control infrastructure reconfiguration: *Forced* for activating the forcing of the GRAFCETs, and *ForcedX[0..n]* with the values obtained from the CheckPoint to force each step (step number 917 in Fig. 2).

The combinational part is implemented by a program (a POU type) that includes the declaration of variables, instances to each POU of the sequential part, and the generation of the system outputs. The codification of the signals related to the process actuators (outputs) requires a search in all GRAFCET actions in order to identify all the steps in which they appear, taking into account the action type (continuous, conditional, or assignment). The control signals that coordinate and synchronize the DOUs are programmed in a similar way. Counters, timers, pulses, etc., are implemented by instantiating the functional blocks provided by the IEC 61131-3 standard, but their initialization and updating are programmed in a similar way to the outputs.

IV. Assessment

This section presents an assessment through a case study. The applicability of the proposed approach has been evaluated in the flexible assembly cell DISA FMS-200, located in the Department of Automatic Control and Systems Engineering, University of the Basque Country. The cell is divided in five functional areas or stations whose aim is to mount a set of four pieces (base, bearing, shaft, and cap) on a pallet, which moves through a modular conveyor system (transfer). The cell can assemble six sets combining different materials, sizes, and colors of the pieces. The pieces are distributed in small warehouses accessible by the stations, and the pallets have a code that indicates the type of piece to be mounted.

The stations are in charge of a number of tasks: in the first station, the base is placed correctly on the pallet located on the transport system; in the second station, the bearing and shaft are placed on the base; the third station is in charge of putting the cap; in the fourth station, the set mounted on the pallet is withdrawn from the system and stored in the warehouse; the fifth station takes care of introducing pallets on the transport system for assembling new sets or collecting assembled sets from the warehouse; and the transport system manages the movement of the pallets between the stations.

For simplicity, the control infrastructure reconfiguration has been evaluated using the station 3 (S3) that is in charge of putting the correct cap on the assembled set on the pallet. There are aluminum caps, white nylon caps, and black nylon caps with two different heights.

The operations performed at this station are distributed around an eight-position turntable that produces the exchange of caps between successive positions. The operations involve the following:

1) feeding the cap from the station warehouse and placing it on the turntable;
2) detection of the material and the color of the cap by means of three sensors located in consecutive positions;
3) measurement of the cap height;
4) removal of the incorrect cap;
5) inserting the correct cap on the assembled set on the pallet.

The developed implementation consists of two Schneider Electric PLCs connected in a local Fipway industrial network. The control system of the station 3 is executed in PLC1, but PLC2 takes its control when PLC1 breakdown occurs (see Fig. 1).

As storage size for control software is typically not the limiting factor in the area of industrial automation [23] and usually the source code of the controller cannot be changed at run time, the full set of functions exists on each controller. Communication is done by the production/consumer function provided by Fipway for shared tables.

The detailed design of the control system for the station 3 developed applying **MeiA•** methodology has been generated by a prototype software tool. The **MeiA** framework [27] is an MDE-based approach that makes use of both, automation and software engineering methods and techniques giving fully support to the developer of automation control systems. It has been developed using Eclipse IDE, and it is based on Java and XML technologies, W3C XML schema for defining the domain ontologies, and XSLT stylesheet technology for implementing model to model and model to text transformations.

Once the process points and the control system points have been identified in the detailed design, the following seven steps perform the analysis and design for the reconfigurable control application based on the IEC 61131-3 standard.

A. Step 1. Definition of the Control System Point

The control system point is defined by the value of the steps (active/inactive) of every procedure, the coordination and information signals, the counter of caps at the station warehouse, the vector with the occupation of the turntable positions, and the vector with the characteristics of the cap located on each turntable position.

B. Step 2. Communication of the Execution Point

The value of the execution point is collected in each PLC cycle as working with shared memory tables. Each PLC sets a bit to indicate that the execution point has been updated. The DRAM obtains the execution point from each PLC, resets the corresponding bit, and starts a timer with the longest cycle time for each PLC. If the timer expires, the manager confirms the PLC failure and applies the reconfiguration strategies.

C. Step 3. Diagnosis of the Process Points

Taking into account the critical elements in the process, the following critical operations have been considered:

1) the introduction of the cap on the turntable because a handling device with 180° rotation and parallel opening gripper is in charge of this operation;
2) the measurement of the cap height since a pneumatic cylinder moves the plunger;
3) the removal of the incorrect cap that is performed by a two-axis handling device whose terminal element consists of a vacuum clamping platform with three suction cups and a vacuum ejector that takes care of the suction;
4) the insertion of the correct cap on the assembled set that is carried out by a pneumatic handling device with parallel opening gripper;
5) the turntable turn that is actuated by a pusher cylinder and fitted with retaining cylinders.

The analysis of these operations (that of course depend on the particular process) leads to the identification of the critical points of the control system, in this case, seven nonrecovery points and three recovery points.

For example, considering the removal of the incorrect cap, if the controller breakdown occurs before suction, the control system points are direct points. If it occurs in the moment of the suction, the control system point is a recovery point since the control system must be resumed from the previous state (the single-acting cylinder must be expanded again to collect the cap). If it occurs after the suction, the control system points are nonrecovery points because the state of the handling device is unknown, but if the handling device is over the evacuation ramp, the control system point is a direct point.

D. Step 4. Characterization of Critical Points

For simplicity, only the recovery point identified in the removal of the incorrect cap will be characterized. The procedure in charge of this operation is *S3_RemoveIncorrectCap*.

A graphical user interface has been integrated within the **MeiA** framework aiming at supporting the analysis and design of dynamic reallocation of control resources. It shows a section with the flow diagram corresponding to the analysis steps, as well as the input data section associated to each step. In the case of the fourth step, the corresponding input data section provides a tree with all the procedures related to the control system and allows creating, saving, editing, or deleting a critical point. For the example, from the tree with all the procedures related to the control of the station 3, the procedure *S3_RemoveIncorrect Cap* must be expanded to display its steps and signals (the corresponding GRAFCET is showed in a parallel window).

In order to characterize the recovery critical point of the control system, the step that controls the vacuum ejector activation must be identified; concretely, it is the step 94. Moreover, the name, type (recovery or nonrecovery point), and description will be added. This information is shown in a list of critical points at the user interface.

E. Step 5. Recovery Points

The way to resume the control system and the process must be established for the three process recovery points identified, but, as an example, only the recovery point that has been characterized in the previous step is considered. Since the resuming of the control system does not require previous actions, only the restart point must be established.

The restart point is obtained from the breakdown point. As the control system must be resumed from the previous state, the step 94 must be deactivated and the step 93 must be activated to construct the variable ForcedX[0..n] with the values to force the GRAFCETs.

F. Step 6. Request of Reconfiguration Mode

The DRAM will request the reconfiguration mode to PLC2 when the PLC1 fails. The "starter" of the PLC2 control system will receive the *ReqConfigurationMode* signal and the breakdown point (see Fig. 3).

G. Step 7. Confirmation of Resumption

The correct resumption of the new controller requires the activation of the main sequence. Therefore, if one of its steps is active, it will be enough to verify the correct resumption of the control system.

The implementation has allowed verifying that the information obtained through these seven steps is enough to reconfigure the control system and resume the production. But the control infrastructure reconfiguration suffers from the rigidity of the controllers and communication systems at the device level: the failure of the PLC that controls the station 3 requires not only that another PLC assumes its functions, but also the inputs and outputs must be redirected. The assignment of the inputs and outputs to each controller is very rigid and univocal, so it is very difficult to act directly on the bus configuration. Other mechanisms are needed that allow the flexible allocation of inputs and outputs to each controller depending on the selected configuration. In this case, the inputs and the outputs are connected to a third PLC that writes the sensor values in the shared tables and reads the value of the outputs written by the other PLCs for assigning them to the outputs. However, as new open systems and programming standards are introduced in the automation field, it is expected that this type of problem will be reduced. Note that the DRAM has access to these shared tables to achieve communication with the PLCs in this system. Thus, one of the requirements for implementing the DRAM is to be connected to all PLCs in the system.

From the point of view of the control system, the reconfiguration implementation can lead to the following:

1) generation and download of automation projects according to the particular situation;
2) switching over the code already loaded in the controllers without the need for code generation.

The latter solution uses more resources, but the reconfiguration time may be faster. So, this alternative has been used in the case study (i.e., the control code exists on each controller) and the reconfiguration time has been enough for industrial processes without hard real-time constraints.

V. Conclusion

This article presented an integrated design flow for modeling of reconfigurable control applications based on the IEC 61131-3 standard and the necessary toolset support in order to assure the system reconfiguration at runtime, envisaged at system design time.

The analysis and design required to address the dynamic reallocation of control resources at run time was performed through the study of critical process points. This article allowed the identification of the direct points and the critical points of the control system so that the restart points of the control system could be established, as well as the required actions before resuming the control system in another controller. This redistribution of responsibilities at runtime seeks to ensure the completion of the current production plan.

The reconfigurable control applications exhibit more flexibility because they can react in case of the controller failure, increase system availability during its operation, and reduce production downtimes to perform the maintenance tasks.

In the case study, special care was taken to use technology dominantly used by industry for both control hardware and software platforms. The use of methods and standards widely used in the automation field allowed the maintenance crews to be familiar with the implementations.

Future work focuses on applying the proposed approach for modeling reconfiguration control applications under other design paradigms (under Industry 4.0 paradigm) for verifying that the obtained information is enough to reconfigure the control system and resume the production.

References

[1] H. Kagermann, W. Wolfgang, and J. Helbi, *Recommendations For Implementing the Strategic Initiative INDUSTRIE 4.0: Final Report of the Industrie 4.0 Working Group*, *Forschungsunion, Acatec*, Munich, Germany, 2013.

[2] V. Vyatkin and A. Zoitl, "Editorial: Advanced software engineering in industrial automation," *Control Eng. Pract.*, vol. 21, no. 11, pp. 1606–1607, Nov. 2013.

[3] B. Vogel-Heuser *et al.*, "Challenges for software engineering in automation," *J. Softw. Eng. Appl.*, vol. 07, no. 05, pp. 440–451, 2014.

[4] B. Vogel-Heuser, A. Fay, I. Schäfer, and M. Tichy, "Evolution of software in automated production systems: Challenges and research directions," *J. Syst. Softw.*, vol. 110, pp. 54–84, Dec. 2015.

[5] C. Timurhan Sungur, U. Breitenbücher, F. Leymann, and M. Wieland, "Context-sensitive adaptive production processes," *Procedia CIRP*, vol. 41, pp. 147–152, 2016.

[6] D. Schutz, A. Wannagat, C. Legat, and B. V.-Heuser, "Development of PLC-based software for increasing the dependability of production automation systems," *IEEE Trans. Ind. Inform.*, vol. 9, no. 4, pp. 2397–2406, Nov. 2013.

[7] A. Giret and D. Trentesaux, "Software engineering methods for intelligent manufacturing systems: A comparative survey," in *Industrial Applications of Holonic and Multi-Agent Systems (Lecture Notes in Computer Science)*, vol. 9266. Cham, Switzerland: Springer, 2015, ch. 2.

[8] M. L. Alvarez, E. Estévez, I. Sarachaga, A. Burgos, and M. Marcos, "A novel approach for supporting the development cycle of automation systems," *Int. J. Adv. Manuf. Technol.*, vol. 68, no. 1–4, pp. 711–725, Sep. 2013.

[9] B. V.-Heuser, S. Rösch, J. Fischer, T. Simon, S. Ulewicz, and J. Folmer, "Fault handling in PLC-based industry 4.0 automated production systems as a basis for restart and self-configuration and its evaluation," *J. Softw. Eng. Appl.*, vol. 9, pp. 1–43, Jan. 2016.

[10] K. Hoffman, A. H. Basson, and A. Roux, "Towards alternatives for agent based control in reconfigurable manufacturing systems," in *Enabling Manufacturing Competitiveness and Economic Sustainability*. Cham, Switzerland: Springer, 2014, pp. 237–242, ch. 40.

[11] P. Leitão, V. Mařik, and P. Vrba, "Past, present, and future of industrial agent applications," *IEEE Trans. Ind. Inform.*, vol. 9, no. 4, pp. 2360–2372, Nov. 2013.

[12] J. Barbosa, P. Leitão, E. Adam, and D. Trentesaux, "Behavioural validation of the ADACOR2 self-organized holonic multi-agent manufacturing system," in *Industrial Applications of Holonic and Multi-Agent Systems (Lecture Notes in Computer Science)*, vol. 9266. Cham, Switzerland: Springer, 2015, ch. 6.

[13] R. Priego, N. Iriondo, U. Gangoiti, and M. Marcos, "Agent-based middleware architecture for reconfigurable manufacturing systems," *Int. J. Adv. Manuf. Technol.*, vol. 92, no. 5–8, pp. 1579–159, Sep. 2017.

[14] W. Lepuschitz, A. Lobato-Jimenez, E. Axinia, and M. Merdan, "A survey on standards and ontologies for process automation," in *Industrial Applications of Holonic and Multi-Agent Systems (Lecture Notes in Computer Science)*, vol. 9266. Cham, Switzerland: Springer, 2015, ch. 3.

[15] R. Priego, A. Armentia, E. Estévez, and M. Marcos, "On applying MDE for generating reconfigurable automation systems," in *Proc. IEEE 13th Int. Conf. Ind. Inform.*, Cambridge, U.K., 2015, pp. 1233–1238.

[16] M. A. Wehrmeister, E. P. de Freitas, A. P. Delazari, and C. E. Pereira, "Combining aspects and object-orientation in model-driven engineering for distributed industrial mechatronics systems," *Mechatronics*, vol. 24, no. 7, pp. 844–865, Oct. 2014.

[17] J. Wermann, E. C. Moraes, and A. W. Colombo, "A service-oriented architecture implementation in the digital factory of the university," in *Industrial Applications of Holonic and Multi-Agent Systems (Lecture Notes in Computer Science)*, vol. 9266. Cham, Switzerland: Springer, 2015, ch. 7.

[18] R. M. da Silva, D. J. Filho, and P. E. Miyagi, "From conception to implementation of reconfigurable and distributed manufacturing control system," in *Industrial Applications of Holonic and Multi-Agent Systems (Lecture Notes in Computer Science)*, vol. 9266. Cham, Switzerland: Springer, 2015, ch. 4.

[19] V. Vyatkin, "IEC 61499 as enabler of distributed and intelligent automation: State-of-the-art review," *IEEE Trans. Ind. Inform.*, vol. 7, no. 4, pp. 768–781, Nov. 2011.

[20] K. Kruger and A. H. Basson, "Multi-agent systems vs IEC 61499 for holonic resource control in reconfigurable systems," *Procedia CIRP*, vol. 7, pp. 503–508, Jan. 2013.

[21] R. Priego, D. Schütz, B. Vogel-Heuser, and M. Marcos, "Reconfiguration architecture for updates of automation systems during operation," presented at the IEEE 20th Conf. Emerg. Technol. Factory Autom., Luxembourg, Luxembourg, pp. 1–8, 2015.

[22] A. J. Hoffman and A. H. Basson, "IEC 61131-3-based holonic control of a reconfigurable manufacturing subsystem," *Int. J. Comput. Integr. Manuf.*, vol. 29, no. 5, pp. 520–534, May. 2016.

[23] M. Merz, T. Frank, and B. V.-Heuser, "Dynamic redeployment of control software in distributed industrial automation systems during runtime," in *Proc. IEEE Int. Conf. Autom. Sci. Eng.*, Seoul, Korea, 2012, pp. 863–868.

[24] *Guide d'Étude des Modes de Marches et d'Arrêts (GEMMA)*, Agence nationale pour le Developpment de la Production Automatisée (ADEPA), Montrouge , France, 1981.

[25] Association Française pour la Cybernétique Economique et Technique Commission, "Normalisation de la Représentation du Cahier des Charges d'un Automatisme Logique," Association Française pour la Cybernétique Economique et Technique (AFCET) Commission, Tech. Rep. 61-62, 1977.

[26] J. M. Diez, R. Montoya, and P. A. Blanco, "Metodología para la elaboración de los programas a implementar en autómatas programables. MEPUS," (in spanish), *Revista Iberoamericana de Automática e Informática Ind. RIAI*, vol. 13, no. 3, pp. 322–329, Jul. 2016.

[27] M. L. Alvarez, I. Sarachaga, A. Burgos, E. Estévez, and M. Marcos, "A methodological approach to model-driven design and development of automation systems," *IEEE Trans. Autom. Sci. Eng.*, vol. 15, no. 1, pp. 67–79, Jan. 2018.

Isabel Sarachaga received the B.Sc. and Ph.D. degrees in computer science from the University of Deusto, Bilbao, Spain, in 1990 and 1999, respectively.

She is a Permanent Associate Professor in Automatic Control and Systems Engineering with the University of the Basque Country (UPV/EHU), Bilbao, Spain. Since 2001, she has been a Member of the Systems Control and Integration Research Group, UPV/EHU. She was the Vice-Dean with the Faculty of Engineering from 2012 to 2017. She is a coauthor of more than 35 technical papers in international journals and conference proceedings, and a Researcher of more than 30 research projects funded by the National and European R&D Programs. Her main research interests include the application of the model-driven engineering paradigm to industrial control systems and reconfigurable distributed applications in the context of Industry 4.0.

Dr. Sarachaga was a Program Cochair of the IEEE Conference on Emerging Technologies and Factory Automation 2010.

Arantza Burgos received the B.Sc. degree in computer science from the University of Deusto, Bilbao, Spain, in 1991, and the master's degree in robotic and automation and the Ph.D. degree in telecommunication engineering from the University of the Basque Country, Bilbao, in 1992 and 1997, respectively.

She is currently a Permanent Associate Professor of Automatic Control and Systems Engineering with the University of the Basque Country, where between 1992 and 1998, she worked with the Faculty of Engineering as a Coordinator of the Advanced Manufacturing Technology Master, and was the Vice-Dean with the Technical Engineering School from 1999 to 2004. Since 2006, she has also been a Member of the Systems Control and Integration Research Group. Since 1992, she has been working as a Researcher on different projects related to communications, industrial automation systems, software engineering, and bioelectronics. In these fields, she has authored and coauthored more than 60 technical papers in international journals, conference proceedings, book chapters, and monographs.

María Luz Alvarez received the B.Sc. degree in physics, the master's degree in advanced manufacturing technology, and the Ph.D. degree in control engineering, robotic and automatic from the University of the Basque Country, Bilbao, Spain, in 1990, 1994, and 2015, respectively.

She is a Permanent Professor in Automatic Control and Systems Engineering with the University of the Basque Country, where she has worked as a Lecturer and a Researcher with the Automatic Control and Systems Engineering Department, Faculty of Engineering in Bilbao. Since 2011, she has been a Member of the Systems Control and Integration Research Group and has taken part as a Researcher in different projects related to communications, control systems for flexible manufacturing systems, and software engineering. Her main current research interests include industrial control systems in the context of Industry 4.0.

Nagore Iriondo received the degree in industrial engineering and the Ph.D. degree in automatic control from the University of the Basque Country (UPV/EHU), Bilbao, Spain, in 1989 and 2013, respectively.

She is a Lecturer in Automatic Control and Systems Engineering with the UPV/EHU. She has coauthored more than 25 technical papers in international journals and conference proceedings in the field of distributed industrial control systems. She has also been involved in several research projects funded by the National and European R&D Programs. Her main expertise deals with applying control engineering concepts to the industrial process control field. She is currently also working in the line of distributed industrial control systems, in the application of model-driven engineering techniques that support the development of these systems.

Marga Marcos (SM'88) received the B.Sc., M.Sc., and Ph.D. degrees in control engineering from the University of the Basque Country, Bilbao, Spain, in 1983, 1984, and 1988, respectively.

She is a Professor in Control Engineering with the University of the Basque Country, where she was the Vice-Dean with the Faculty of Engineering from 1990 to 1993 and the Chairman with the Control Department from 1995 to 2005. She has authored and coauthored more than 200 technical papers in international journals and conference proceedings. She has acted as the main Researcher for more than 80 research projects funded by the National and European R&D Programs. Her main research interests include the application of model-driven engineering to automation systems and the application of multiagent systems in manufacturing.

Prof. Marcos has served and has been serving Technical Committees (TCs) of the International Federation of Automatic Control (IFAC) (AARTC, WRTP, and CC) and IEEE (NBCS, ICPS, and IA). She was the Chair of the IFAC TC Computer for Control from 2014 to 2017, a Member of the EUCA, NOC of IFAC Spain, Publication Cochair for the IEEE Conference on Decision and Control in 2005, and the General Cochair for the IEEE Conference on Emerging Technologies and Factory Automation in 2010. She is currently an Associate Editor for the IEEE TRANSACTIONS ON AUTOMATION SCIENCE AND ENGINEERING.