Int J Adv Manuf Technol (2013) 68:711–725
DOI 10.1007/s00170-013-4793-4

ORIGINAL ARTICLE

A novel approach for supporting the development cycle of automation systems

M. L. Alvarez · E. Estévez · I. Sarachaga · A. Burgos · M. Marcos

Received: 3 August 2012 / Accepted: 21 January 2013 / Published online: 12 February 2013

Abstract The growing complexity of current automation applications demand clear methodologies and procedures in order to assure that the system under design meets the requirements. Within the industrial automation field there are several methods, techniques, tools and standards that have captured the experience of designers through years of practice. On the other hand, the software engineering discipline offers different methodologies covering the different phases of the life cycle (analysis, design, test and maintenance). This work pretends to take advantage of both: the maturity of the software engineering discipline and the well-spread methods and standards of the industrial automation field. In particular, the proposed methodological approach uses model based engineering principles for developing automation control systems combining modeling methods of automation field. Specifically, it combines GEMMA (Guide d'Étude des Modes de Marches et d'Arrêts), UML (Unified Modeling Language) use case diagrams and GRAFCET (GRAphe Fonctionnel de Commande, Etapes, Transitions) for assisting the designer during the analysis, design and coding phases. Using GEMMA, the states of the automated process are identified. Use case diagrams allow identifying the actors that participate in the operation modes. Finally, a set of GRAFCET templates assists in the design of use cases. Thus, the user is guided through the development phases in which the concepts used are familiar to him/her. The final step includes the automatic generation of the Program Organization Units (POUs) in PLCopen XML interface. The proposed methodology is illustrated by means of a simple but illustrative mechatronic case study.

Keywords Industrial automation systems · Design methodology · GEMMA · GRAFCET · PLCopen

1 Introduction

In recent years, a great effort has been done by international organizations for promoting the use of standards within the industrial automation field. The International Electrotechnical Commission (IEC) promotes the use of standards in this field. In particular, the IEC 61131–3 standard [1] and John and Tiegelkamp [2] propose a software model and programming languages for industrial process measurement and control systems. Currently, most of the PLC software vendors are becoming IEC 61131–3 standard compliant.

On the other hand, the international organization PLCopen is an international association that promotes the use of this standard resolving topics related to control programming [3]. It is organized in technical committees (TC) and TC6 for XML (eXtended Markup Language) has defined an open interface for achieving interoperability between PLC programming tools.

M. L. Alvarez (✉) · I. Sarachaga · A. Burgos (✉) · M. Marcos
Departamento de Ingeniería de Sistemas y Automática,
ETSI de Bilbao, UPV/EHU, Bilbao, Spain
e-mail: marialuz.alvarez@ehu.es
e-mail: arantzazu.burgos@ehu.es

I. Sarachaga
e-mail: isabel.sarachaga@ehu.es

M. Marcos
e-mail: marga.marcos@ehu.es

E. Estévez
Departamento de Ingeniería Electrónica y Automática,
EPS de Jaén, Jaén, Spain
e-mail: eestevez@ujaen.es

Notwithstanding this great effort for promoting the use of standards, methodological approaches are still necessary to develop current complex automation systems, but using domain specific modeling languages and tools as proposed by Kandare et al. [4] for the field of procedural control software development. The Model Driven Engineering (MDE) [5] seems to fit perfectly with this goal. MDE centers model concept to software development. The aim of MDE is to use the concept of *model* within the software development process. Thus, it is focused on creating and exploiting domain models which contain the domain knowledge and rules by means of abstract representations. Model Driven Development (MDD) [6] also relies on the use of models to represent the system elements of the domain and their relationships. However, the models are used in all system development activities, until the final system is itself generated. [7] presents the challenges and benefits when introducing MDD practices in the automation domain. The most extended MDD initiative is the Model Driven Architecture (MDA) [8] by OMG (Object Management Group). This software design approach separates the system specification (Platform Independent Model [PIM]) from the implementation issues (Platform Specific Model [PSM]).

Any development process starts by specifying the functional and non-functional requirements to represent the desired behaviour. For industrial control systems, these requirements are described by means of operation modes. The work of Panjaitan and Frey [9] analyses different guidelines for the definition and handling of operation modes: GEMMA (Guide des Modes d'Etude et d'Arrêts Marches), S88 standard from ISA, PackML proposed by OMAC (Open Modular Architecture Controls) users group, IEC 61804 and IEC 61499–1. They propose an operation mode guidance compatible with distributed applications and they use UML (Unified Modeling Language) state chart [10] for modeling the behaviour of modes and their internal states. Furthermore, they present a development process approach based on UML and IEC 61499 in order to provide a guideline and modeling support in the early development phases [11].

In the automation field, GRAFCET (GRAphe Fonctionnel de Commande, Etapes, Transitions) [12] and GEMMA [13] are widely spread as modeling language, and as natural extension for defining operation modes, respectively. Several authors have developed different methodologies combining GEMMA and GRAFCET. For instance, Ponsa and Vilanova [14] present a methodology with five phases (automation, supervision, interaction, implementation and proofs). It starts with the GRAFCET that implements the production cycle that is the starting point for the application of GEMMA. One important aspect is the interaction with the operator [15, 16].

González et al. [17] describe an object-oriented methodology that combines UML, GEMMA and GRAFCET to systematise the analysis and modeling process for discrete event sequential systems. This methodology provides a tool with editing facilities to generate the hierarchy of the object-oriented models. The resulting PLC projects are bigger and more complex due to the conversion from object-oriented models to current PLC programming languages.

Machado and Seabra [18] propose a methodology based on GEMMA and GRAFCET but with a special emphasis on the synchronization aspects. The GEMMA associated to system behaviour is translated into a high-level SFC (Sequential Function Chart) [19] and each GEMMA state into low level SFCs. The synchronization is obtained by means of the vertical coordination method.

These previous works present different methodologies combining GEMMA and GRAFCET, but none of them profit from the advantages of MDE. The novel methodology proposed in this paper offers: time saving by means of model reuse, templates and code generation; quality improvement by employing design guidelines, validated models and tested transformations; documentation improvement by using models with well-defined semantics; and communication improvement by employing the domain terminology.

On the other hand, several research works can be found in the literature offering integrated development environments to guide the control system construction based on model-driven development. Most of them use UML for describing the domain concepts. Some focus on the IEC 61131 standard [20–22] and others in the IEC 61499 standard [23–25]. Other research studies propose SysML for modeling automated systems [26, 27]. In the work of Beremiz [22], a deep analysis of the current state in development frameworks for automation systems can be found. To some extent, all of them use the concept of domain model that allows describing and validating of designs [28].

Most of the aforementioned development environments support the development cycle of automation applications. A common problem that arises is that they require a high level of knowledge on software engineering techniques and tools, or expertise in UML or SysML. Furthermore, the initial phase of requirements analysis and high level design associated to requirements is hardly covered. This work pretends to fill this gap guiding the user through the analysis and design phases. In this sense, it is complementary to the research works cited.

Furthermore, the new methodological approach relies on domain specific modeling methods using the syntax and

lexicon that the final users commonly use. It is based on well-known standards and formalisms of the field: GEMMA is used to identify the operation modes, use case diagrams are used to discover the participants that take part in operation modes, and a set of GRAFCET templates assists the user in the design of the use cases. It also takes into account the best practices of the methodologies commented above [14, 17, 18].

The MEthodology for Industrial Automation (MeiA**.**) systems proposed here offers design guidelines and templates in order to methodologically define industrial control systems. The former is achieved using GEMMA and use cases and the latter by means of GRAFCET templates. Thereby, the platform independent model is generated automatically making particular emphasis on operation modes.

Furthermore, using model based techniques [29, 30], the source code of the Program Organization Units (POUs) is obtained. POUs are derived from the GRAFCETs templates adapted by the user, once they are enriched with implementation details. This is achieved using an SFC IEC 61131–3 code generator (a plug-in of a pipeline based framework defined in the study of Lüder et al. [31]). Additionally, in order to provide vendor tool independence, the POUs are expressed in PLCopen XML format, assuring, thus, portability.

The layout of the paper is as follows. Section 2 presents the "MeiA**.** methodology". Section 3 details a novel approach to give support to MeiA**.**. This is based on GEMMA, use case diagrams and GRAFCET as domain specific modeling methods, and it uses the syntax and lexicon that the final users commonly use. Section 3 concludes with the automatic generation of the automation project in PLCopen XML format. In Section 4, the proposed methodology is applied to the development of the control system of a warehouse mechatronic unit. Finally, conclusions are summarized in Section 5.

2 MeiA**.** methodology

As discussed above, based on the user requirements, MeiA**.** guides the analysis and design phases when developing industrial automation system. Through six phases with its corresponding steps, the methodology helps in the identification of the operation modes, as well as the control and monitoring needs, including the operator panel and other required auxiliary panels. MeiA**.** offers design guidelines and templates for each phase in order to methodologically define industrial control systems.

The six phases are briefly described below, while the main steps in order to complete every phase are summarized in Table 1.

Phase 1: main sequence

This phase consists of establishing the main sequence. It defines the control system start-up and stop, generating the control signals that report to the production procedures about the system state: ready to start production, end of cycle request, end of operation, initial state stop reached.

Phase 2: manual operation mode

In this phase the system requirements to operate in manual mode are evaluated. The specific needs different from the common cycle are analyzed. Generally these operations will be performed under the supervision of maintenance operators; for instance, sensor or actuator calibration, and execution of preventive maintenance operations. These aspects greatly influence the selection of the control strategies to be included in the SCADA system, as well as the design of the operation and auxiliary panels.

Phase 3: semi-automatic operation mode

The system requirements to operate in semi-automatic mode are evaluated in this phase. Therefore, the needs for verifying certain movements or process parts step by step or continuously are analyzed. Generally these operations will be performed according to the usual cycle sequence under the control of the operator in charge of the task. It affects the design of the SCADA system, as well as the operation and auxiliary panels.

Phase 4: production cycle

The production cycle is defined taking into account the operation modes identified in the previous phases. Besides, from the analysis of the process actions, the requested stops are derived; for instance, raw material is needed, parts have to be removed or tool maintenance. These situations lead to a temporal process stop until the corresponding action is carried out. If not, they could lead to a failure situation.

Phase 5: process failures

The process failures are identified, analysed and evaluated in order to classify them as failures that run a reconfiguration (even accepting product quality degradation) or failures that force a controlled stop. Once failures have been analyzed, situations with similar treatments could be grouped.

Phase 6: emergencies

The emergencies are evaluated determining the actions to lead the system to a safe situation. In production processes is common to implement emergencies in independent modules due to safety reasons. However, if the controller receives the signal that activates the emergency, a controlled stop must be triggered.

Through these six phases, the platform independent model is obtained. In fact, it represents the control

Table 1 Methodology overview

Phase 1: Main Sequence		
Steps	1	Establishment of start-up procedure
	2	Identification of the starting safe state required for automatic operation mode as well as the steps to reach it
	3	Analysis of the tasks to be performed before starting production
	4	Establishment of the procedure to request system stop during production
	5	Identification of the signals indicating the end of the last production cycle
	6	Analysis of the tasks to perform after a production stop
Phase 2: Manual Operation Mode		
Steps	1	Definition of the activation procedure
	2	Identification of available control elements to perform the process actions
	3	Definition of the de-activation procedure
Phase 3: Semi-automatic Operation Mode		
Steps	1	Definition of the type of sequenced operation (step by step or block)
	2	Definition of the activation procedure
	4	Definition of the de-activation procedure
Phase 4: Production Cycle		
Steps for each stop request	1	Definition of process stop detection
	2	Establishment of the protocol
	3	Definition of the procedure to resume the process (once the cause overcome)
Phase 5: Process Failures		
Steps for each failure	1	Identification of process failure detection
	2	Definition of the failure diagnosis
	3	Definition of the failure protocol
	4	Definition of production-with-failure operation mode (if supported)
Phase 6: Emergencies		
Steps	1	Identification of emergency detection and emergency mode activation
	2	Identification of the need of actuators remaining activated in this mode
	3	Identification of the procedures that must be de-activated
	4	Analysis of the need for diagnosing the emergency source
	5	Analysis of the viability to solve emergencies (in manual mode, etc.)
	6	Definition of the procedure to deactivate this mode
	7	Analysis of the procedure to resume the operation
	8	Analysis of the need to reach an intermediate stop or to execute a shutdown process
	9	Definition of the procedures to be executed after emergency de-activation

engineering domain view of the automation system. It defines "what" the control system has to do in order to meet the functional and non-functional requirements. It represents a high level design defined by a hierarchical component-based architecture described in one study [29] which has been extended to distinguish the process and control data in order to make emphasis on operation modes. The functional model contains components, connectors, channels and ports.

A component groups part of the functionality of the control system. There are two types of components: the *control components* related to the control functions, and *production components* corresponding with the production functions.

The connectors communicate components. There are three types of connectors: the *field* connectors are the inputs and outputs of the control system, i.e., field signals coming from or going to the process (sensors and actuators), human machine interface and/or operator panels; the *internal* connectors communicate functional components at the same hierarchical level; and the *control* connectors are the control signals coming from the higher level.

The channels group the same type connectors that come from/go to the same component, while the ports represent the point in which connectors grouped in channels are related to components. Two types of ports can be distinguished: the *data* ports related to field

connectors and internal connectors; and the *control* ports for control connectors. They can be vertical ports with channels that contain inherited connectors, or horizontal ports to communicate functional components at the same hierarchical level.

3 A novel approach for supporting MeiA. methodology

The development of automation application starts from collecting the system requirements. They provide the information about the process operation as well as the decomposition in sub-systems and the relationships among them.

Specifications are the "raw material" from which a project is built; incorrect or incomplete specifications may lead to fault or at least unexpected results. Thus, the tools used during the analysis phase must help in the collection of correct specifications, ensuring that inconsistencies and unexpected situations are avoided. For supporting this phase, MeiA**.** relies on GEMMA and UML use case diagrams. During the design phase, MeiA**.** uses GRAFCET as a method for describing behaviour.

The following sub-sections present a brief overview of GEMMA, UML use ase diagrams and GRAFCET; then the generation of the functional model in terms of components and connectors; and finally, the automatic generation of the POUs in a PLCopen XML format, from the functional model commented above and GRAFCETs.

3.1 Base analysis and design methods of MeiA**.**

GEMMA has been developed by ADEPA (Agence pour le Developpement of Appliquée Productique). It is a guide for performing a systematic search of the possible states of an automated process from a control perspective, distinguishing between energized and not energized control. When the application is energized, the possible states are grouped into three families of start and stop modes: *F* family for operation procedures, *A* family related to stop procedures, and *D* family concerning failure procedures. Table 2 summarizes the possible states.

GEMMA offers a graphical notation (Fig. 1): rectangles for the typified states for automation systems, oriented lines to represent transitions between states and conditions for the evolution between states.

The guide recommends the following steps: determination of possible process states, identification of transitions between states and conditions for such transitions. This analysis provides valuable information about the behaviour of the process, the control system boundaries and the interfaces to the environment.

On the other hand, UML use case diagrams allow defining static behaviour diagrams used to describe a set of actions (*use cases*) that a system or some systems (*subject*) can perform in collaboration with one or more external users of the system (*actors or stakeholders*). Each use case should provide some observable and valuable result to the actors or other stakeholders of the system. Figure 2 illustrates an example.

Table 2 GEMMA states grouped in families

F Family: all states for operation procedures needed to obtain the desired result of the process	
F1 — Normal production F2 — Start up process F3 — Shutdown process	States associated to normal operation
F4 — Unsequenced test mode F5 — Sequenced test mode F6 — Test mode	States related to testing and verification
A Family: all states for stop procedures that stop the process operation	
A1 — Initial state stop A4 — Determined state stop	Stop states
A2 — Requested stop at the end of cycle A3 — Requested stop in a determined state	States which lead to stop the system
A5 — Preparation to restart after failure A6 — Production reset to the initial state A7 — Production reset to a determined state	States that guide the system from a failure state to a stop state
D Family: all states for failure procedures	
D1— Emergency stop D2 — Diagnosis and/or treatment of failure D3 — Production despite failure	States related to system failure

GRAFCET was developed by AFCET (Association Française pour la Cybernétique Economique et Technique). In 1982, it became a French standard (NF C 03–190 UTE) proposed by ADEPA and, later, an international standard (IEC-848) [32]. Furthermore, it has been the basis for developing a new PLC programming language called SFC, which is part of the IEC 61131–3 standard. It allows describing sequences.

GRAFCET is a graphical method that describes the behaviour of sequential systems by means of steps with associated actions, transitions with associated conditions and oriented lines. The steps represent the system states, the transitions indicate the possibility of evolution between states, and the oriented lines establish the evolution by linking steps to transitions and transitions to steps. In order to evolve, every step previous to a transition must be active and the corresponding conditions must be fulfilled, thereby the following steps become active and all preceding steps are deactivated. Only when a step is active, its actions will be executed.

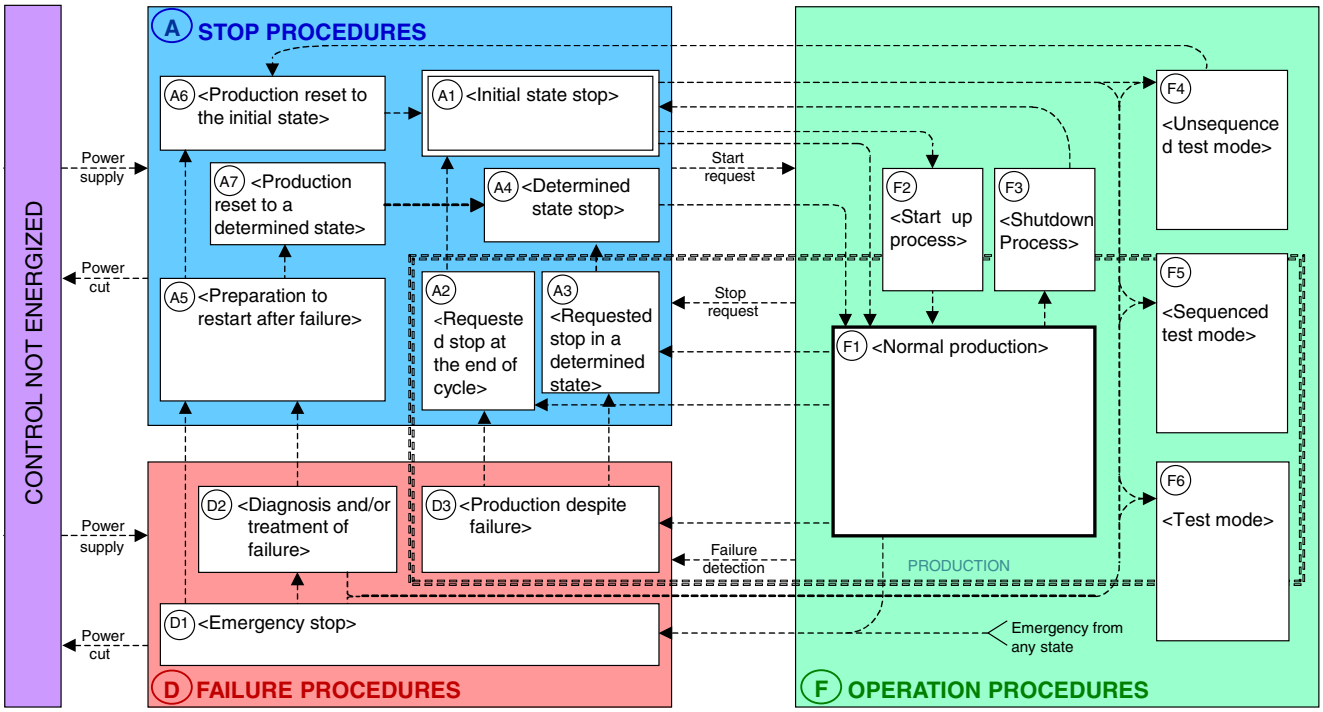


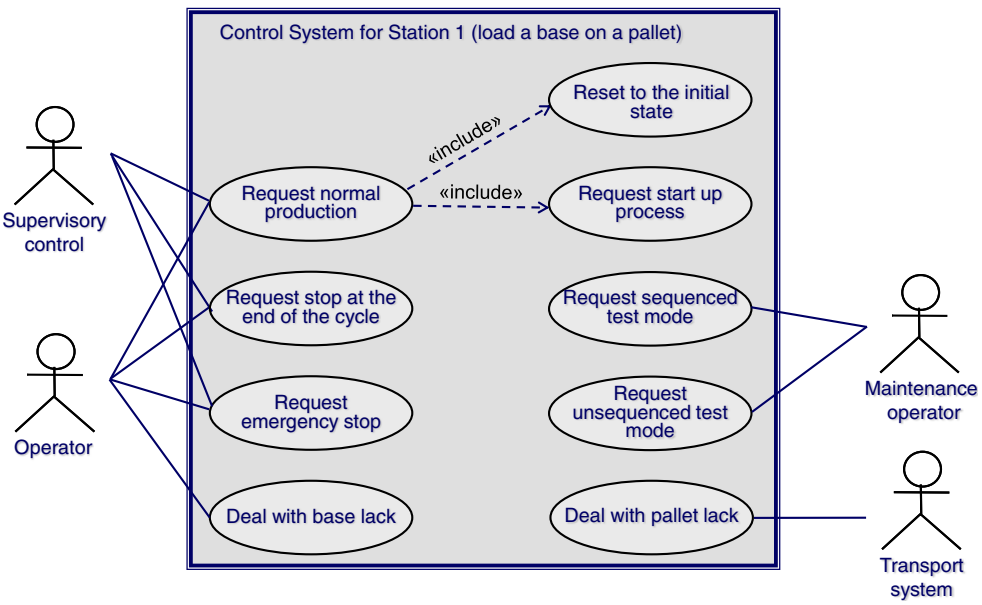
Fig. 1 Graphical notation of GEMMA

Nevertheless, a sequence can be combined with other structures to represent complex behaviours; for instance, alternative tasks can be represented by means of convergences and divergences in "or", while concurrent tasks can be represented by means of convergences and divergences in "and". Loops and conditional jumps can also be represented.

Furthermore, partial diagrams can be enclosed into macro-steps encouraging structured designs for complex processes. In this sense, as a design can consists of different GRAFCET diagrams, diagram hierarchies can be used to coordinate them.

GRAFCET allows specifying an automation system at three different levels: functional, technological and operational description. At the first level, the objective is a functional overview of the system without many details and, of course, it must not contain any reference to technologies. At the second level, technologies are introduced in the design in order to complete the tasks identified from the functional description with technical

Fig. 2 Example of use case diagram



information about sensors and actuators. Finally, at the third level, the task sequence is established from the controller point of view, describing the system evolution by means of the activated outputs depending on the input values.

Currently, GRAFCET has a dominant position as a modeling language for automation systems because its intuitive graphical representation, simplicity and clarity in the representation.

3.2 Implementation of MeiA. steps

The MeiA. methodology uses domain specific modeling methods using the syntax and lexicon that the final users commonly use. The operation modes are identified by means of GEMMA, use case diagrams are used to identify the actors participating in the operation modes. From these latter, the evolution conditions among GEMMA states can be derived. Finally, a set of pre-defined GRAFCET templates assist during the design of the use cases.

Two types of GRAFCET diagrams have been defined: the so-called *decision GRAFCETs* and *production GRAFCETs*. The former organizes and coordinates the possible system states, such as system start-up, manual operation, automatic or semi-automatic operation, or emergency procedures. The latter corresponds to the actions related to the production cycle. Thus, coordination requires special attention. In this sense, a guideline has been established, combining vertical or hierarchical coordination for decision GRAFCETs, as well as, horizontal and vertical coordination for production GRAFCETs.

Guidelines for the user have been defined for each methodology phase and the corresponding GEMMA diagram, use case diagram and GRAFCETs can be obtained. Once the design is finished, the functional model is generated automatically.

Due to the space limitations, the implementation of the-MeiA. methodology only the first phase is detailed (Fig. 3).

Figure 4 illustrates the combination of GEMMA, use case diagrams and GRAFCET, as well as the relationship among them for the first phase, i.e., the *Main Sequence*. The GEMMA states associated to the main sequence and the transactions between the states that compose such sequence are illustrated. When the system is in "Initial state stop [A1]" and automatic operation mode is requested, the system evolves to "Production reset to the initial state [A6]" to verify the initial and safety conditions. If certain tasks are required to prepare the system before starting production,

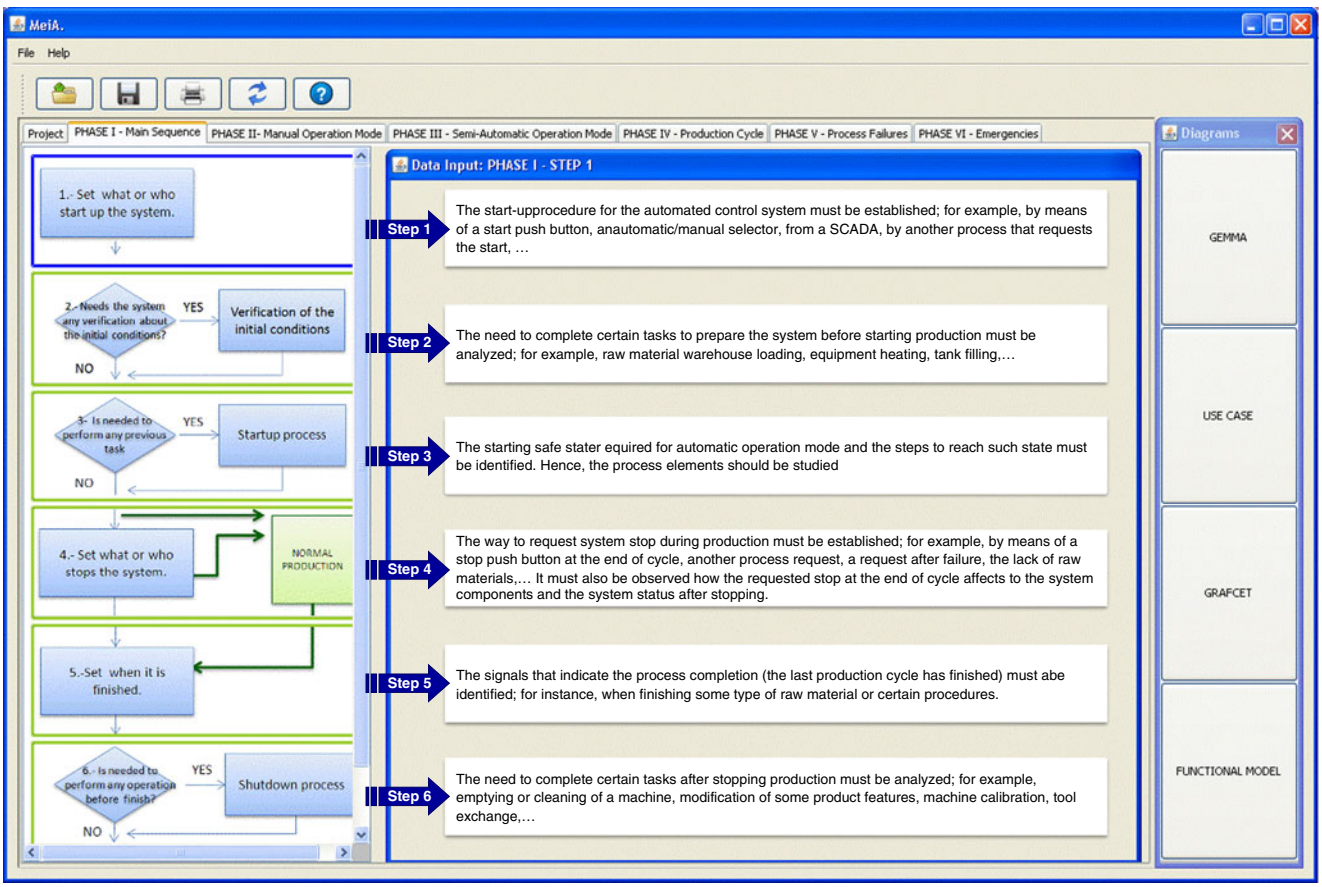


Fig. 3 User guidelines for the first phase of the methodology

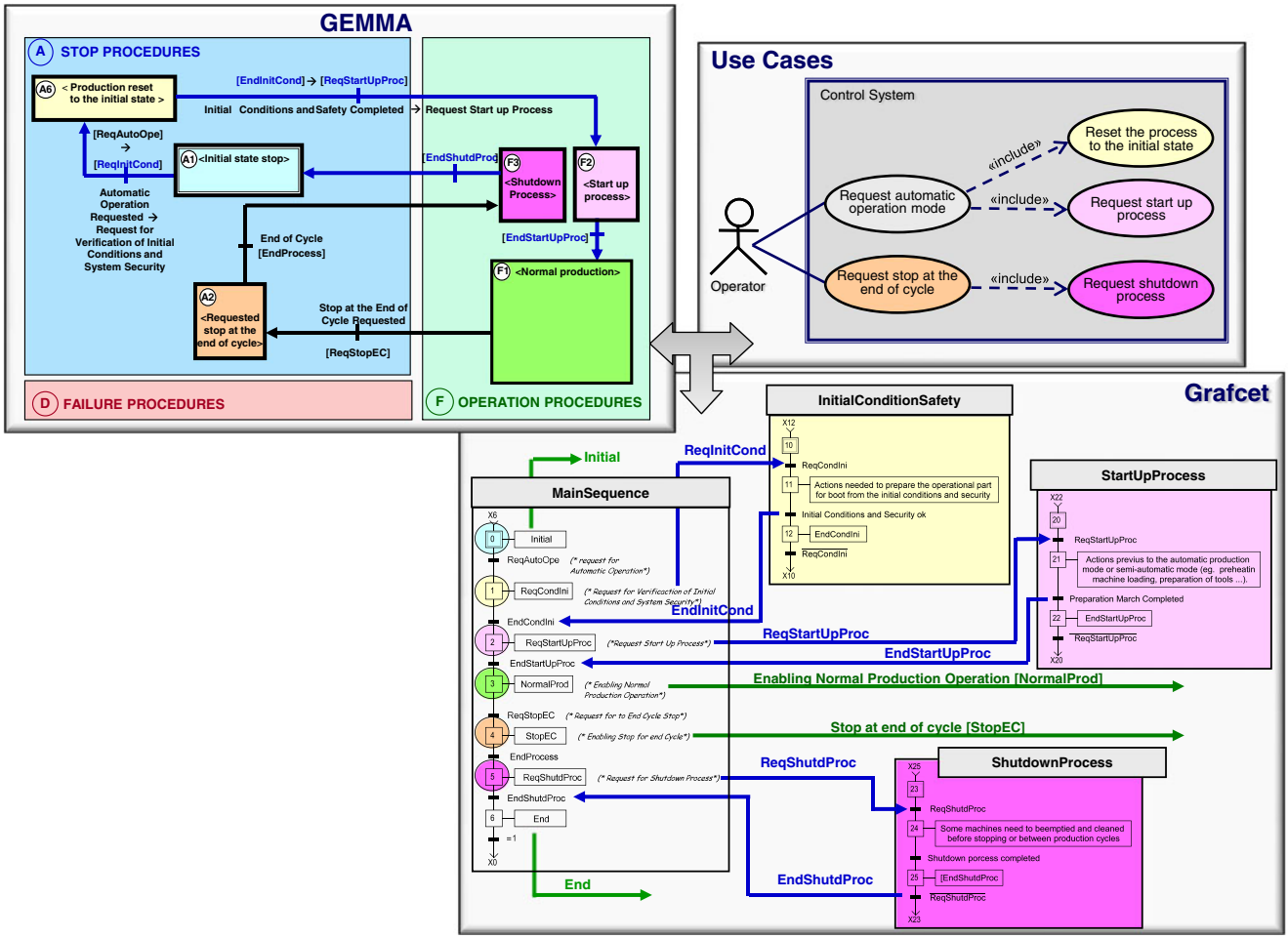


Fig. 4 Analysis and design tools or establishing the main sequence (first phase)

the system will pass from "Production reset to the initial state [A6]" to "Start up process [F2]" before reaching "Normal production [F1]" state. During production, when a stop at the end of cycle is requested, the system will evolve to "Requested stop at the end of the cycle [A2]" until the end of cycle. After stopping production, if certain tasks are required, the system will evolve to "Shutdown process [F3]" before stopping in "Initial state stop [A1]".

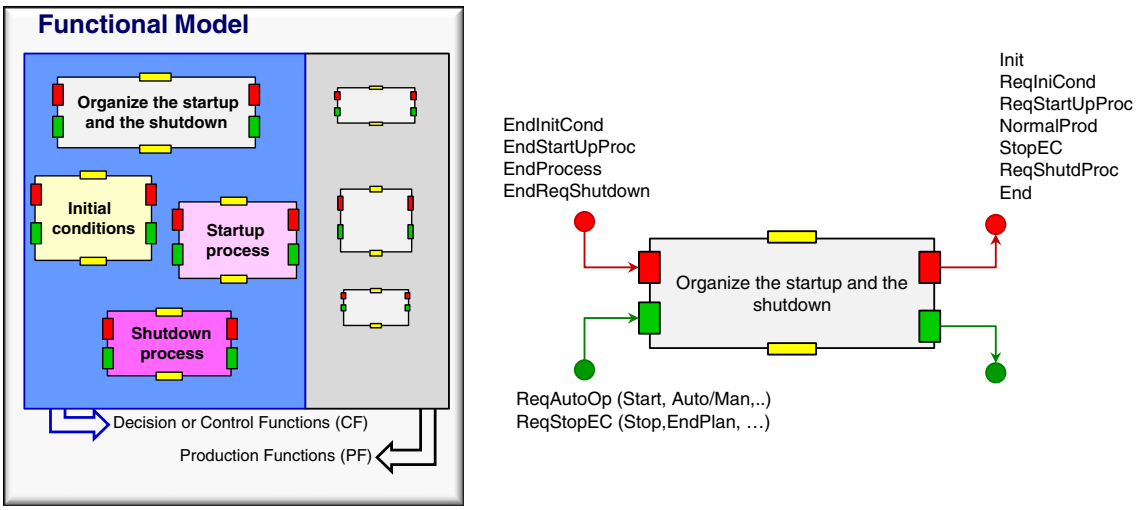


Fig. 5 Functional model for Phase 1

The evolution conditions between GEMMA states can be identified using the use case diagram. They are derived from the actors involved in "Request automatic operation mode" and "Request stop at the end of cycle" use cases, as well as from the preconditions that must be met to implement these use cases. Furthermore, depending on the application, the "Request automatic operation mode" use case could include "Reset the process" and/or "Request start up process" use cases. In the same way, "Request stop at the end of cycle" use case could include "Request shutdown process".

Once the GEMMA states, the transitions among states and the evolution conditions have been identified for the main sequence, the next step consists of adapting the GRAFCET template called "Main Sequence". This GRAFCET could lead to other three GRAFCET: "Initial conditions", "Start up process" and "Shutdown process". When the verification of initial and safety conditions or the start-up/shutdown processes are not required, the steps that activate the additional GRAFCET can be eliminated, i.e., the "Main Sequence" GRAFCET is adapted to the application.

Finally, related to the functional model, up to four control components could be obtained. The control component that organizes the start-up and the shutdown is mandatory, but the other three components may not be required; it depends on the particular project. The production components will be identified in the fourth phase.

These components are characterized by their interface consisting of by three types of ports: data ports (green ports in Fig. 5), control ports (red) and internal ports (yellow). Besides, the functional model also contains connectors to link those components.

3.3 Automatic generation of the POU's code

In order to transform the design into code, two different aspects must be considered: the sequential part that comprises the activation and deactivation 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 industrial automation systems.

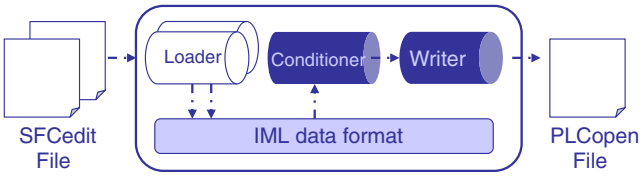


Fig. 7 General scenario of PLCopen converter framework

There are several GRAFCET editors, and one of the most spread is SFCEdit© [33]. SFCEdit provides a graphical set to produce GRAFCET diagrams 100 % compliant to the IEC 60848 standard. Besides, this tool allows exporting GRAFCETs to XML format, WMF (Windows MetaFile) and EMF (Enhanced MetaFile) graphical formats. In order to automatically generate the code, and being necessary to manipulate the information, the exported XML file is the starting point for the generation of the corresponding control code following SFC programming language.

Figure 6 illustrates the grammar used by *SFCEdit* for exporting projects in XML. Every project is composed at least by one GRAFCET defined with one or more branch(s). When there are divergences and convergences, it is composed by two or more branches. In this situation, it is stored in *hlink* concept that is characterized by two properties: its identifier (*seqid*) and the nature of the branch (*type*), i.e., simultaneous divergence and convergence and selector divergence and convergence.

Every sequence is defined by a set of steps and transitions. The characterization of these elements has been done following the IEC 60848 standard.

In order to generate the code from XML files following this grammar, the project defined by Lüder [31] has been used. It defines a pipeline-based framework which is plug-in-based. It allows the user to transform different source models (Gantt Chart, PERT Chart, Impulse Diagram, etc.) defined during the firsts phases of the application development cycle to an IML (Intermediate Model Layer) model expressed in the SFC programming language. In particular,

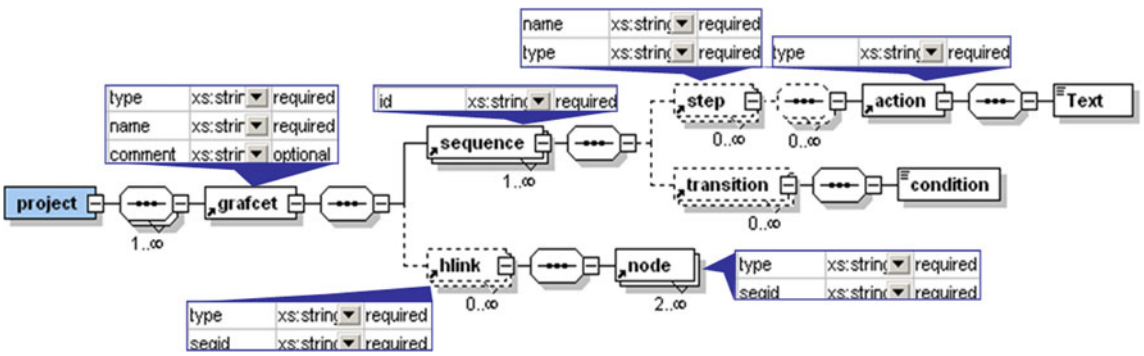


Fig. 6 Lexicon and syntax of SFCedit GRAFCET editor

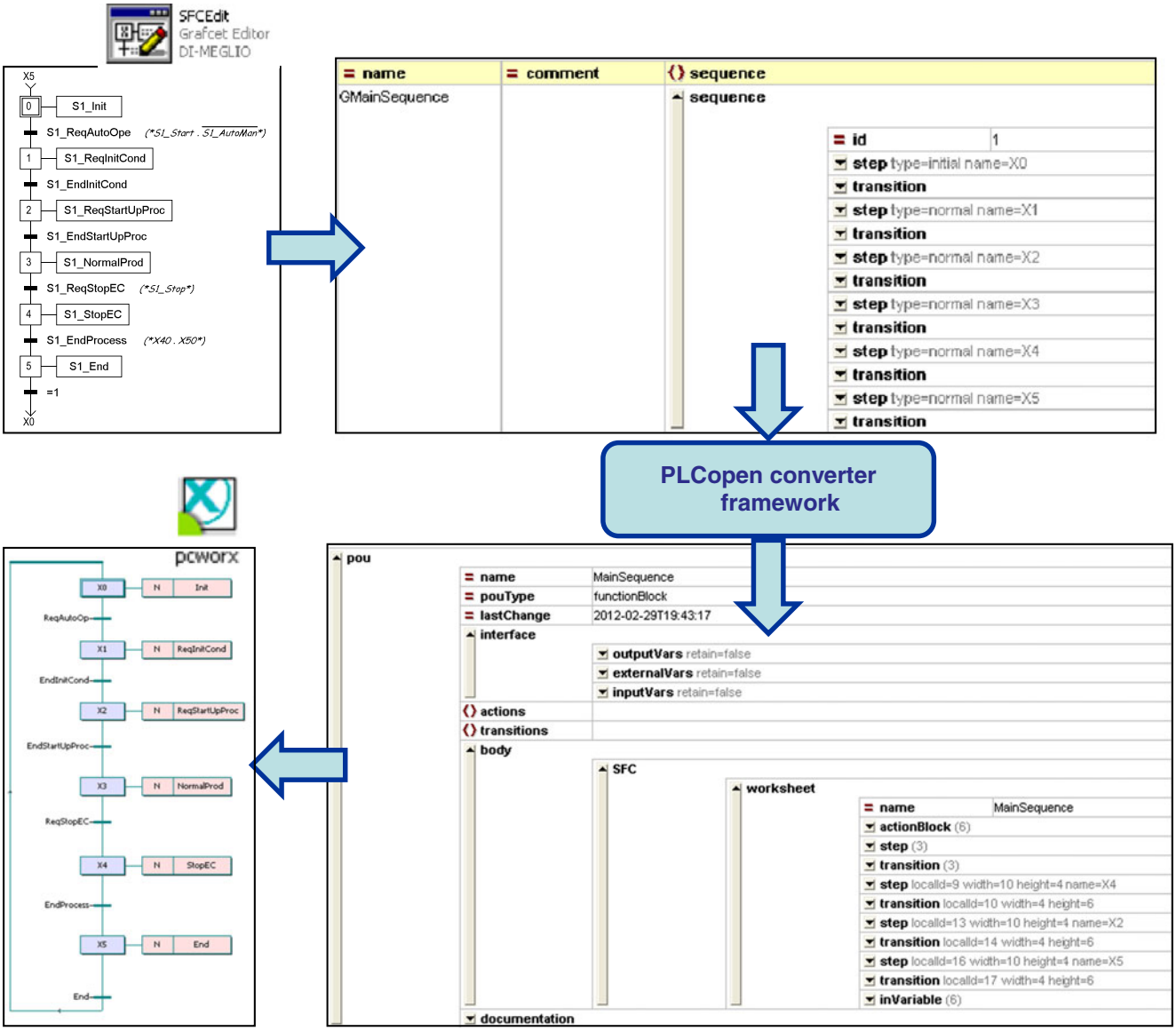


Fig. 8 PLCopen XML conversion example

the resulting file of this transformation follows the PLCopen XML. Hence, this framework has been extended to support input files expressed in SFCEdit.

Figure 7 illustrates the three type of plug-ins used by the framework. The loader plug-in is the file that is expressed in a format used by the tool (in this case, SFCEdit). It is the responsible for transforming the tool format to IML data format. The conditioner plug-in transforms IML data to data following the PLCopen syntax without taking into account graphical information. The writer plug-in implements the printing algorithm defined in PLCopen, such as size of steps and positions.

Accordingly, the extension of this framework for being able to transform SFCEdit files to PLCopen consists on developing its corresponding loader. It implements the mapping rules identified in order to transform SFCEdit files (Fig. 8) to IML data format [31]. Table 3 summarizes identified mapping rules.

Table 3 SFCEdit — tool to IML mapping rules

SFCEdit tool	IML data format
Step	State
Transition	State transition
Link	
div and	Simultaneous divergence
conv and	Simultaneous convergence
div or	Selection divergence
conv or	Selection convergence
Jump	Jump
Action	Activity

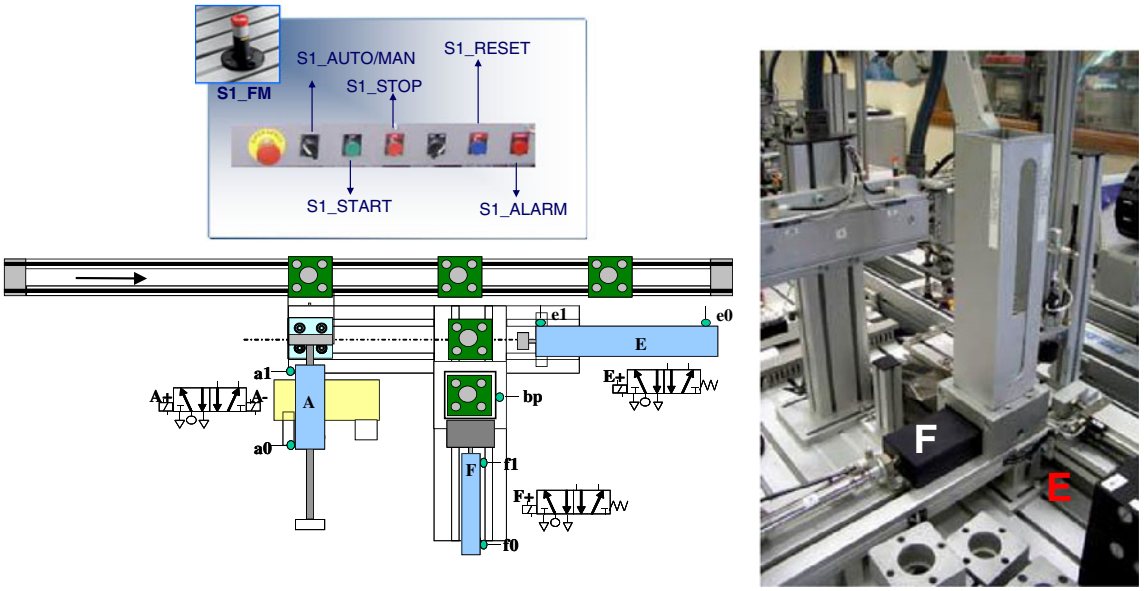


Fig. 9 General scenario of the warehouse

4 Case study: a warehouse's control system development

This section illustrates the applicability of proposed methodology by means of a simple mechatronic unit illustrated in Fig. 9. A warehouse, that supplies parts by gravity, consists of three cylinders and eight sensors. A single-acting cylinder (*F*) extracts a part from the warehouse, a second single-acting cylinder (*E*) moves the piece to the transfer point and a double-acting cylinder (*A*) loads the part on the transport system. Each cylinder has two limit-switch sensors that detect if the cylinder is retreated or expanded. The warehouse has two more sensors: one indicates that the warehouse is full and the other one that there is a part at the bottom of the warehouse.

The operation panel consists of three pushbuttons (Start, Stop and Reset), an automatic/manual selector (Auto/Man), an alarm indicator, an emergency stop pushbutton and a

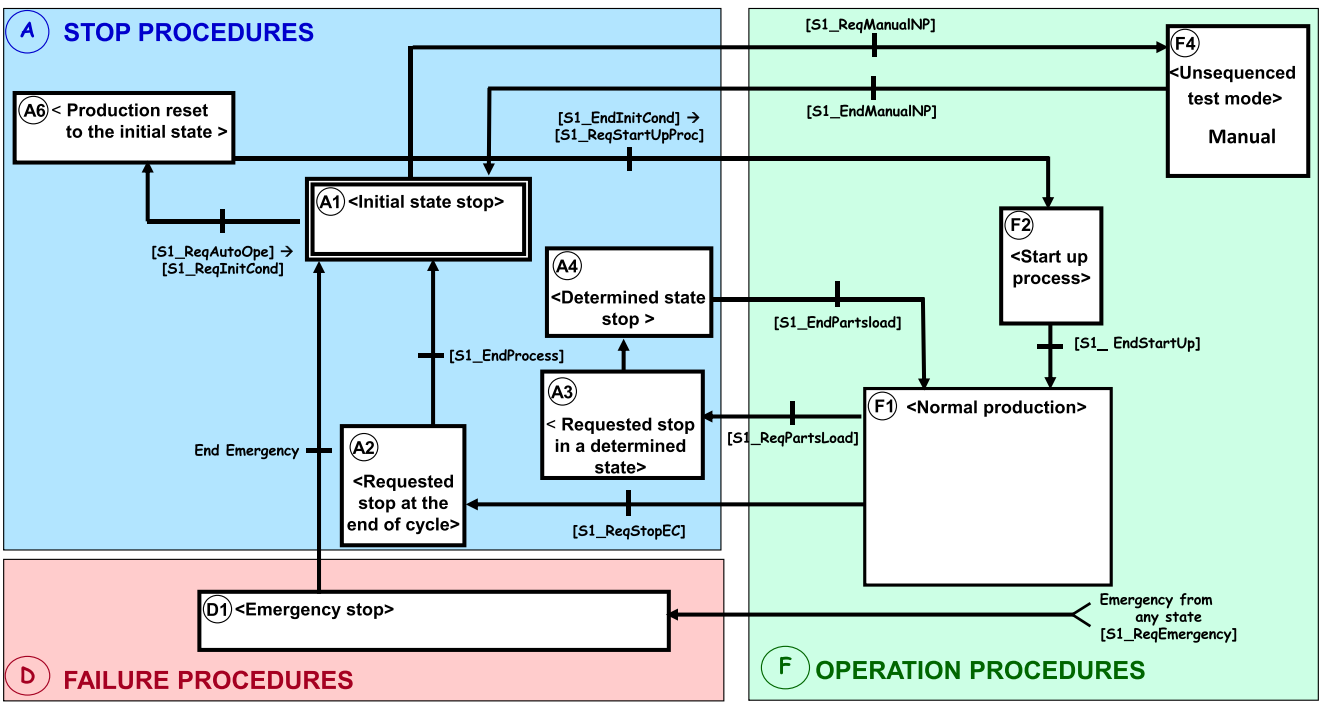


Fig. 10 Final GEMMA for the warehouse

pushbutton for each manual action related to the two single-pilot valves and the double-pilot valve.

The cycle starts when the automatic/manual selector indicates automatic operation mode and the shop-floor personnel presses the start pushbutton. Then, the F cylinder extracts the part detected by the sensor at the bottom of the warehouse and places it in front of the E cylinder. Once the E cylinder is retreated, the A cylinder moves the part to the conveyor that is running from the system startup. The cycle will finish when the stop pushbutton is pressed and the last part has been loaded on the conveyor. In case of an emergency, the activation of the emergency stop pushbutton implies an immediate process stop and the deactivation of this operation mode will be performed by means of the reset pushbutton and the unlocking of the emergency stop pushbutton.

According to the proposed development process, nine GEMMA states have been identified with their evolution conditions (Fig. 10).

In the first phase that consists of establishing the main sequence, five GEMMA states have been identified: "Initial state stop [A1]", "Production reset to the initial state [A6]", "Start up process [F2]", "Normal Production [F1]" and "Requested stop at the end of cycle [A2]". As only one actor interacts with the system, the evolution conditions between GEMMA states depend on shop-floor personnel requests.

Next step consists of adapting the GRAFCET template called "Main Sequence" and, if necessary, the related ones. In this sense, three Decision GRAFCETs have been obtained by adapting the "Main Sequence", "Initial Conditions" and "Start up process" templates, respectively.

During the second phase, "Un-sequenced test mode [F4]" state is identified due to the need of manual operation mode, jointly with the evolution conditions. A new Decision GRAFCET is obtained by adapting the "Manual with priority" GRAFCET template.

The fourth phase leads to the definition of the production cycle. Two new GEMMA states and their evolution conditions are identifies: the "Requested stop in a determined state [A3]" and "Determined state stop [A4]". In addition, three more Production Grafcets are developed, one for each of the following coordinated tasks: the part extraction from the warehouse and its load on the conveyor, and other two tasks in charge of the warehouse management.

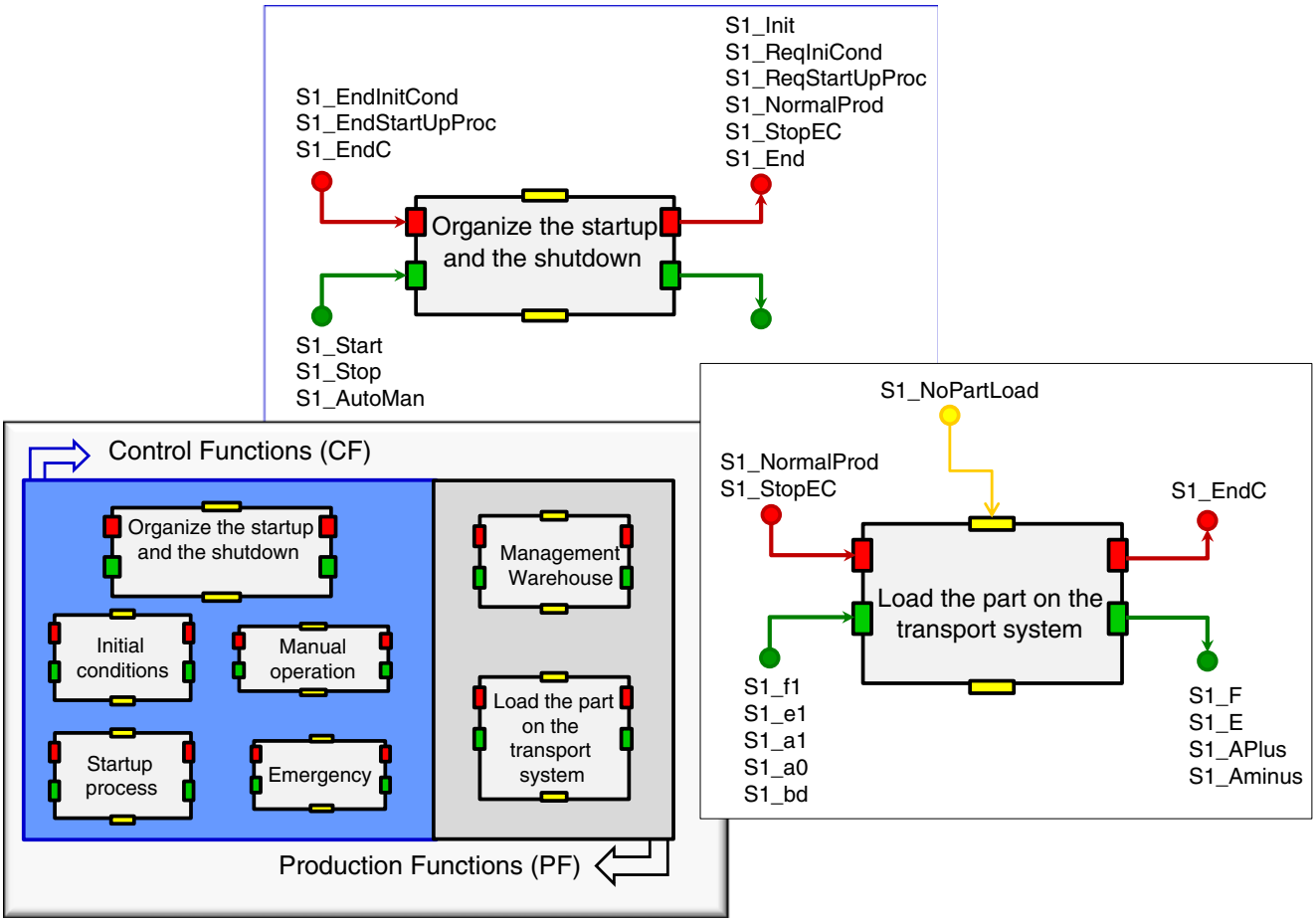


Fig. 11 Functional model of the warehouse

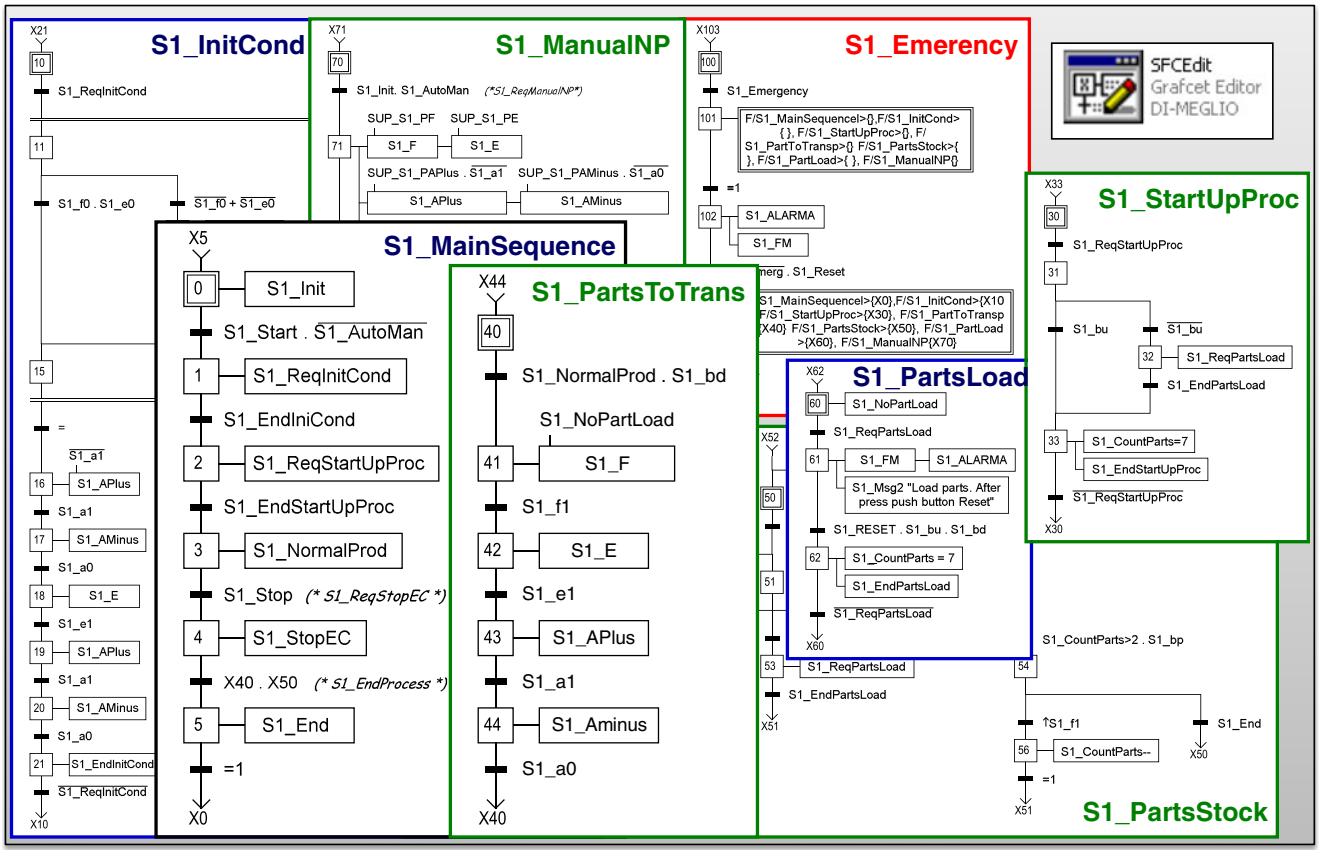


Fig. 12 GRAFCETs for the warehouse

The sixth phase comprises the definition of the emergency protocol. Thus, the "Emergency stop [D1]" GEMMA state is required, its evolution conditions are identified and the "Emergency" GRAFCET is obtained.

The resulting functional model is illustrated in Fig. 11. Seven functional components have been obtained: five control components and two production components. Furthermore, field, internal and control connectors have been identified and inserted in the corresponding ports.

The complete GRAFCET design includes five Decision GRAFCETs and three Production GRAFCETs. Figure 12 illustrates them which have been generated with the SFCEdit tool.

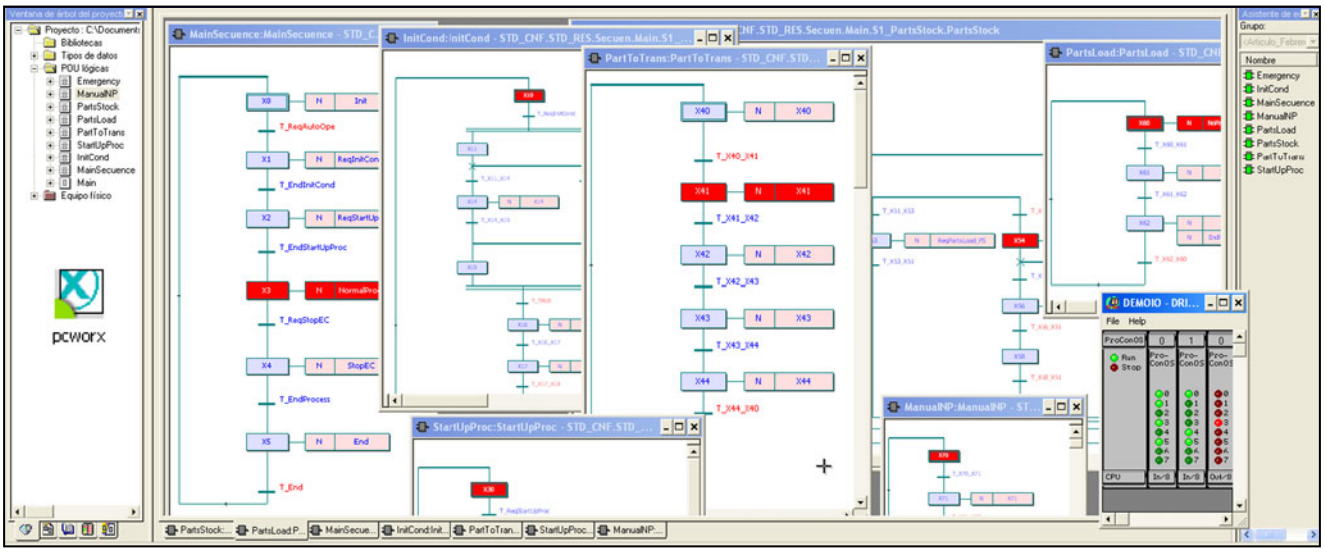


Fig. 13 POUs for warehouse control system in PC Worx of Phoenix Contact

Finally, the corresponding POUs in PLCopen XML format are generated (Fig. 8). The complete automation project could be imported in any PLCopen XML compliant tool. Figure 13 illustrates the warehouse's control system in a PCWorx PLC programming tool of Phoenix Contact vendors.

If the PLC programming tool does not support PLCopen XML format, SFCEdit [30] details the steps to follow in order to adapt to an understandable format.

5 Conclusions

This study has presented a novel approach that uses model based engineering principles for developing automation control systems, but using domain specific modeling methods that combines GEMMA, UML use case diagrams and GRAFCET for assisting the designer during the analysis and design phases.

The MeiA**.** methodology guides the final user offering methods widely used in the automation field. To do this, it offers design guidelines and templates to define the industrial control systems methodologically using the syntax and lexicon that the final users commonly are used to. As a result, the platform independent model is generated automatically making particular emphasis on operation modes. So, MeiA**.** methodology provides the following benefits: time savings by means of model reuse, templates and code generation; quality improvement by employing design guidelines, validated models and tested transformations; documentation improvement by using models with defined semantics; and communication improvement by employing the domain terminology

Furthermore, another two important results are derived: the documentation of the design using domain terminology, and the minimal units of source code needed to generate the software model.

An SFC IEC 61131–3 code generator from GRAFCET has been developed. It is a new plug-in of a pipeline-based framework defined by Lüder et al. [31]. This latter, allows the engineers to automatically generate the POUs from the GRAFCETs designed following the methodology. Hence, SFCEdit tool (GRAFCET editor) has been added as a new tool from which PLCopen files could be generated.

The use of PLCopen XML interface assures portability of applications, providing they are IEC 61131–3 and PLCopen compliant. It also allows assuring no information lost and no extra effort. Additionally, it provides a powerful interface to other software tools, such as project design tools, simulation tools and documentation tools among others.

Finally, the analysis and design documentation, the functional model and the POUs are stored in distributed repositories as XML files for later management. Thus, reuse can be achieved by means of model reuse.

Acknowledgments This work was financed in part by the MCYT&-FEDER under DPI 2012-37806-C02-01 and by UPV/EHU under GIU 10/20 and UFI 11/28.

References

1. IEC International Standard IEC 61131–3 (2003) Programmable Controllers, Part 3: Programming Languages". International Electrotechnical Commission
2. John KH, Tiegelkamp M (2001) Programming industrial automation systems. Berlin:Springer
3. PLCopen (1992). Available at: http://www.plcopen.org. Accessed 25 July 2012
4. Kandare G, Strmcnik S, Godena G (2010) Domain specific model-based development of software for programmable logic controllers. Comput Ind 61:419–431
5. Schmidt D (2006) Model driven engineering. IEEE Computer Society 39:25–31
6. Selic B (2010) The pragmatics of model-driven development. IEEE Softw 20(5):19–25
7. Streitferdt D, Wendt G, Nenninger P, Nyßen A, Lichter H (2008) Model driven development challenges in the automation domain. Annual IEEE International Computer Software and Applications Conference. Turku, Finland
8. OMG (2003) MDA Guide Version 1.0.1. Available at: http://www.omg.org/news/whitepapers/index.htm#MDA. Accessed 25 July 2012
9. Panjaitan S, Frey G (2007) Operation modes handling in distributed automation systems. Proc. of the 1st IFAC Workshop on Dependable Control of Discrete Systems (DCDS'07), Cachan (France), June 13–15
10. Booch G, Rumbaugh J, Jacobson I (2005) The Unified Modeling Language user guide. Addison-Wesley Educational Publishers
11. Panjaitan S, Frey G (2007) Development process for distributed automation systems combining UML and IEC 61499. Int J Manuf Res 2(1):1–20
12. AFCET (1977) Normalisation de la Représentation du Cahier des Charges d'un Automatisme Logique. Technical report, AFCET Commission
13. ADEPA (1981) GEMMA (Guide d'Étude des Modes de Marches et d'Arrêts). Agence nationale pour le DEveloppment de la Production Automatisée
14. Ponsa P, Vilanova R (2005) Automatización de procesos mediante la guía GEMMA. Edicions UPC (in Spanish)
15. Ponsa P, Vilanova R, Díaz M, Gomí A (2007) Application of a guideline for design improvement in supervisory control. Int J Hum Comput Interact 1:21–36
16. Ponsa P, Vilanova R, Amante B (2011) Human intervention and interface design in automation systems. Int J Comput Comm Contr 6(1):166–174
17. González VM, Mateos F, Ng A (2004) MLAV: the object-oriented methodology of the Virtual Automation Lab. Proc. of the IEEE International Conference on Robotics and Automation (ICRA 2004), New Orleans, LA, USA, pp 5153–5158
18. Machado J, Seabra E (2010) A systematized approach to obtain dependable controllers specifications. ABCM Symp Series Mechatronics 4:408–417
19. IEC 60848 (2002) GRAFCET specification language for sequential function charts. International Electrotechnical Commission

20. Estévez E, Marcos M, Gangoiti U (2005) A tool integration framework for industrial distributed control systems. Proc. of the 44th IEEE Conference on Decision and Control, pp 8373–8378
21. Beremiz (2007). Available at: http://www.beremiz.org. Accessed 25 July 2012
22. Hästbacka D, Vepsäläinena T, Kuikka S (2011) Model-driven development of industrial process control applications. J Syst Softw 84(7):1100–1113
23. Tranoris C, Thramboulidis K (2006) A tool supported engineering process for developing control applications. Comput Ind 57 (5):462–472
24. Thramboulidis K, Perdikis D, Kantas S (2007) Model driven development of distributed control applications. Int J Adv Manuf Tech 33(3–4):233–242
25. Vyatkin V, Hanisch HM, Pang C, Yang CH (2009) Closed-loop modeling in future automation system engineering and validation. IEEE Trans Syst Man Cybern C Appl Rev 39(1):17–28
26. Chiron F, Kouiss K (2007) Design of IEC 61131–3 function blocks using SysML. Mediterranean Conference on Control & Automation, Athens
27. Thramboulidis K, Frey G (2011) Towards a model-driven IEC 61131-based development process in industrial automation. J Softw Eng Appl 4:217–226
28. Reinhartz-Berger I, Sturm A (2009) Utilizing domain models for application design and validation. Inform Software Tech 51 (8):1275–1289
29. Estévez E, MarcosM (2008) Model-driven design of industrial control systems. Proc. of IEEE Multi-Conference on Systems and Control, 9th IEEE International Symposium on Computer-Aided Control System Design (CACSD), pp1253–1258
30. Estévez E, Marcos M, Orive D (2007) Automatic generation of PLC automation projects from component-based models. Int J Adv Manuf Tech 35(6):527–540
31. Lüder A, Estévez E, Hundt L, Marcos M (2010) Automatic transformation of logic models within engineering of embedded mechatronical units. Int J Adv Manuf Tech 54(9–12):1077–1089
32. IEC (1988) Preparation of Function Charts for Control Systems. No 848. Technical report. International Electrotechnical Commission
33. SFCEdit (2012) Available at: http://stephane.dimeglio.free.fr/. Accessed 25 July 2012