Control System Software Design Methodology for Automotive Industry

Giovanni Di Orio, José Barata
CTS – UNINOVA, Dep. de Eng. Electrotécnica, FCT
Universidade Nova de Lisboa
2829-516 Caparica, Portugal
{gido, jab}@uninova.pt

Carlos Sousa, Luis Flores
INTROSYS, SA – Global Control System Designers
2860 Moita, Portugal
{carlos.sousa, luis.flores}@introsys.eu

Abstract—As automated manufacturing systems become more and more complex, the need for new methodologies to improve the design and development of industrial monitoring and control solutions is becoming peremptory. Programmable Logic Controllers (PLCs) dominates the application domain meaning that they are established as the most popular industrial controllers used in factory and shop floor. Although the capabilities of these controllers have strongly improved in the last decades, their historical background as the easy understand by the electricians who had previously worked on electrical systems has meant that the de-facto standard for the implementation of control and monitoring solution for these devices remains the ladder logic. However, to face the globalization challenges, the manufacturing companies need to improve their productivity by reducing the costs, delivering high-quality of products with high variety and improving their responsiveness to changing market condition. Moreover productivity gains can also be achieved by reducing the life-cycle costs of manufacturing production systems implying the usage of more flexible and agile approaches. In this scenario, the proposed methodology aims to provide a homogeneous and optimum process that, starting from a set of mechanical specifications and behavioural models of machines, enable the generation of industrial logic automatically ensuring the same structure and naming standard in every project and the quality of the code.

Index Terms—Behavioural Modelling, Design Methodology, Programmable Logic Controller, IEC-61131-3, Automotive Industry

I. INTRODUCTION

The increasing demand and competition in the automotive manufacturing market are radically changing the production system design and manufacturing in terms of flexibility, life-cycle cost, customization of products and time to market [1].

Due to the competition for key markets share, the Original Equipment Manufacturers (OEMs) are engaged in an innovation race [2] to implement more and more exclusive and efficient production systems able to produce as many different product variations as quickly as possible [3]. Therefore, manufacturing companies are striving to introduce flexible manufacturing techniques in order to better meet market needs whilst maintaining the low cost base of heavily automated mass production techniques [4]. In this context, the key to competitiveness is the ability to have systems able to quickly respond to markets variations and demands for new and innovative products which means frequently changes in products that in turn cause changes in the plant’s hardware structure and functionality. As existing plants cannot be rebuild due to high investment volumes, the changes in hardware components affect particularly the software: it has to be modified continuously and its amount and complexity increases [5]. Furthermore, the manufacturing systems of today are complex and heterogeneous in terms of hardware and software since they are characterized by a lot of different devices, such as sensors, actuators, etc., interconnected across multiple controllers and communicating with each by means communications networks in order to accomplish the given automation tasks. At the same time, the presence of industrial proprietary standards, encompassing electrical systems and diagnostic as well as PLC types and software structure, means that each manufacturer have its own exclusive production and control solution, developed considering a set of mechanical, electrical and electronics proprietary rules and software philosophies. These aspects together highlight that every change in the shop floor environment has heavy consequences on the production since brings the production system from the normal production state to a new one that is the system reengineering state characterized by zero-production, where the system is shut down for reprogramming task to face of the process of change. The time needed for this task depends strictly by the dimension of changes, presence of a well-suited documentation, particular standard familiarity and last software version [6]. Moreover, the programming task represents an error-prone activity since could introduce new possible errors in the system that might affect the way the system will operate. The design and development of methods and tools to eliminate or reduce time, costs and errors related to shop floor reengineering as much as possible would offer a competitive advantage to contemporary manufacturing companies.

The paper proposes a new methodology for modeling and programming manufacturing control systems. The methodology is intended to improve shop floor agility and global competitiveness of companies, providing a homogeneous and optimum process that, starting from a set of mechanical specifications and behavioural models of machines, enables the generation of reliable industrial Ladder Logic Diagram (LLD) automatically ensuring the same structure and naming standard in every project and the quality of the code.

The reminder of this paper is organized as follows: sec-
Section II presents a review on common practices for PLC software development in industry. Section III focuses on the proposed methodology. Section IV provides a brief description of the demonstrator used for applying the proposed methodology. Finally, section V presents the conclusions and future work.

II. A REVIEW ON PRACTICES FOR PLC SOFTWARE DEVELOPMENT IN INDUSTRY

In its root the effectiveness and efficiency of monitoring and control programs relates with the methodology used to implement them. Actually tools and methodologies used to develop such systems, do not possess enough level of representation making the design and development process a very painful activity in terms of time and costs. Moreover, the challenges imposed by globalization ask for more evolvable and high level control systems underlining the need for new methodologies that are able to move high level software engineering techniques to the automation world in order to have solutions not so much bit-oriented and based on scan paradigm. Another important limitation is the lack of a neutral representation for programs, i.e. current program representations depend on the type of used PLC as well as on the end-user implying that the same process can be designed and implemented in completely different ways. Worse than that, the same process in the same PLC is designed and developed in different way according to the particular manufacturing company, in automotive industry this problem is amplified by the presence of hardware and software standard. To really understand the innovative role played by the proposed methodology in design and development of monitoring and control programs, existing approaches are discussed.

A. Current Design Methodologies

The design methodologies are a fundamental issue and unfortunately little work has been done in this direction. The most significant works describing logic design principles for monitoring and control systems was done in 2003 [7], [8], 2007 [9] and 2008 [4], [10], [11]. As stated in [11] there is no definitive report on main challenges faced by the users of industrial control and supervision systems, nor is there any published material on the tools and techniques and methodologies used in order to overcome these problems. The Hajarnavis and Young’s works tries to alert industrial users to different practice and/or approaches used outside their company, in order to open a discussion inside the industrial community and the research community for potential ways for improving usability and flexibility of PLC systems. In Lucas’s work the current methods of creating control logic are observed. Those observations are, then, used to measure the effectiveness of the logic control design methodologies enabling the comparison with existing academic methods. The final result not demonstrates which design method is better but, on the contrary, demonstrates that a room for improvement exists. Ljungkrantz, in his work, provides a study of logic control programming practices and describes a component based approach to develop industrial logic control programs. Component based approach promises to shorten development and modification times, and lessen programming errors improving at the same time readability, code auto-description and code reusability. Finally, Hajarnavis describes the results of an investigation into the techniques used to design factory control system in the body-in-white section of car manufacturing plants located in Europe.

Since PLCs are established as the device of choice for the implementation of control algorithms in many manufacturing companies [11] a good starting point is to present the industrial specification IEC 61131-3 [12] which contains five programming languages for developing PLC programs that are supposed to be supported by any compliant vendor. The main goals of the IEC 61131-3 standard is to unify all the programming concepts for developing industrial control applications [13].

The supported programming language are:

1) Instruction List (IL) - A textual low level programming language similar to assembly and consisting of simple operation codes.
2) Structured Text (ST) - A high-level programming language with syntax similar to Pascal, Basic or Fortran suited for programming more complex algorithms.
3) Function Blocks Diagram (FBD) - A graphical language providing a mechanism for encapsulating functionality into a module with a common external interface, and designed to promote reuse of code.
4) Sequential Function Chart (SFC) - A graphic language derived from the GRAFCET (a simplified version of Petri Nets). It is made up of graphical elements called steps and transitions; a step represents a specific state in process and/or machine sequence while a transition represents the set of conditions that allows the evolution from one step to another.
5) Ladder Diagram (LLD) - A graphical language derived directly from the electrical relay diagram and created to be easy to understand for booth engineers and technician without any software skills.

Most of the industrial applications are designed using one of these languages. However as exposed in [8] and recently confirmed a poll realized by the Control Engineering U.S. and Control Engineering Poland magazine [14], 96% of industrial developers uses LLD diagrams. FBDs are the next most popular at 67%, followed by IL (37%) as depicted in figure 1.

Therefore, the LLDs represent the most common method for creating control solutions, especially as regards the car

![Fig. 1. Industrial programming languages dissemination](image)
manufacturing industry, that represents the background for the ProFlex methodology. The LD diagrams are, probably, the simplest mechanism for constructing industrial control applications, since are rooted directly to the electrical drawings for hardwired control panel. Several reasons exist for ladder diagrams prevalence in the industry. First of all a historical reason, since the ladder diagram was created to enable a soft paradigm switch from electromechanical systems to new microprocessor-based system allowing creation of control applications that look like the electrical diagrams. According to Walter [15], LD diagrams are an excellent graphical representation based on well-understood circuit design concepts allowing to quickly develop basic programming skills. Furthermore, since LLDs are designed to mirror electrical circuits then it naturally is a good way to represent discrete logic. However Walter also points out some drawbacks when using LLDs such as the poor data structure, the limited execution control and arithmetic capabilities. The lack of a good semantic level in LLDs is still a discussion element in the scientific community. As exposed in [16], semantics describe the meaning of diagrams, i.e., the behaviour of the computer when such diagram is “executed”. In a LLD the semantic is defined by spatial relationship between components implying that LLDs could be hard to visualize and become unreadable even for small to medium size programs resulting in a monitoring and control application hard to debug and modify [17].

B. Alternative Design Methodologies

As pointed in II-A, LLDs weaknesses lead the research community to find new techniques, methodologies and/or approaches to improve ladder code even if without abandoning it. A lot of work has been carried out by several authors on the application of structured programming techniques, software engineering methods including modularity and top-down design as well as object oriented paradigm to the design of LLDs for advanced automation engineering [18]–[20]. Java based solutions have also been used to design industrial control applications that satisfy requirements imposed by agility such as reconfiguration allowing at the same time to generate a IEC 61131-3 standard compliant code starting from a behavioral model of the constituent resources of a manufacturing assembly line (see Fig. 2).

 controllers as well as generating LLDs [26], [27]. A set of software tools that have been developed to create, manipulate, verify, and execute logic control systems written using modular finite state machines (MFSMs) is described in [28], [29].

Despite the ladder diagram disadvantages and the proliferation of more powerful and richer semantic representation programming languages, it is believed that ladder diagrams will continue to be wide used for constructing industrial applications. The reason why the industry is still betting on the ladder diagrams is basically due to its simplicity with respect of more advanced programming languages allowing to be handled by operators, as well as the lack of better alternatives that justify costs and risks associated with the shift to another design methodology. Moreover the existence of several rigid software and hardware standards, especially in the automotive industry, quite complicates the introduction of any new tool and methodology and discourages their acceptance by the industrial community.

Nevertheless, the using of LLDs not prevent the developing of new methodologies that based on good programming practices do not need ladder for creating industrial applications. The challenges for agility demand new methods and approaches for creating industrial application even if the final result has to be a ladder diagram. In this scenario, the proposed methodology defines a set of rule based on high level programming language paradigm for creating industrial applications assuming that the original representation will always be transformed into ladder.

III. METHODOLOGY

The methodology proposed in this paper addresses an effective way for high level PLCs monitoring and control solution implementation and automatic generation of IEC 61131-3 compliant code starting from a behavioral model of the constituent resources of a manufacturing assembly line (see Fig. 2).

![Fig. 2. Methodology Concept](image-url)
and intended to transform real manufacturing components into agents which behaviour is modelled using a formal and unique representation. In this scenario, each manufacturing component will be then represented by a self-contained piece of software (agent) inside the PLC that is responsible to control and monitor the status and activities of its own associated manufacturing equipment in order to meet a specific objective. Even if each agent is capable to sense changes in the state of its own resource and act in response of these, only the interactions between them enable the correct execution of the activities of the manufacturing production line and/or cells in terms of functionalities and sequence of operations.

Therefore the main activities of the methodology are five, namely:

- **A0**: Monitoring and Control system requirements analysis.
- **A1**: Identification of the constituent resources of the manufacturing assembly line.
- **A2**: Construct an universal behavioural model of each resource.
- **A3**: Software Generation.
- **A4**: Deployment and Tests.

The activities A0, A1 and A2 are executed during the Model Development Cycle.

The Monitoring and Control system Requirements Analysis activity is intended to identify the monitoring and control system requirements starting from the entire set of mechanical and electrical specifications of the manufacturing assembly line. In this context the following documents are analyzed: the mechanical and electrical drawings, the description of the hardware, the sequences of works of the machines/stations and the description of the proprietary software standard. The output of the activity A0 is used to start the Agentification Cycle that, in turn, includes the activities A1 and A2. The activity A1 performs the identification of the constituent resources of the manufacturing assembly line. During this activity all the machines/stations of the manufacturing line are identified and their sensors and actuators are grouped together. Moreover all the interactions of machines/stations are explained. In this context, two kinds of interactions can occur, namely internal interactions (between machine/station internal components) and external interactions (with other machines/stations). The results of this activity are a set of Manufacturing Resources Entities and/or agents which behaviour is modelled using a formal and unique representation (see Fig. 4) during the activity A2. Once the behavioral model of the manufacturing components is obtained, the Control Development Cycle starts and the activity A3 is performed. The control software is developed starting from the behavioral model and considering necessary information such as the specific software and hardware standards. Furthermore, several templates are used during the Software Generation activity in order to keep the software structure as much as possible similar to the different proprietary standards. This way the generated code satisfies both the functional requirements and the specific formalism and syntax.

The output of the activity A3 will be an eXtensible Markup Language (XML) file that can be easily deployed into several PLCs.

Finally, the Deployment and Tests activity represents a set of tasks for mapping the XML file to IEC-61131-3 standard compliant language. This activity is performed using the PLC vendor development environment.

The proposed methodology offers an effective and simple way for developing monitoring and control solutions using a formal representation of the manufacturing resources while keeping the specific proprietary standards. The methodology focuses on a kind of “lingua franca”, represented by the behavioral model, as a way to enable fast coding, improve industrial acceptance of developed code and, most important, reduce the error during the programming activity. As a result, the following section IV shows some experimental result obtained by applying such methodology.
IV. A MANUFACTURING CELL FOR APPLYING THE METHODOLOGY

In order to validate and test the proposed methodology, two robotized manufacturing cells located at IntRoSys have been used (see Fig. 5).

The cells are identical in terms of functionalities, however they rely on two distinct hardware and software standards from two automotive industry manufacturer. The cells include a loading station used to load car components and a welding robot. The main differences between the manufacturing cells are resumed in table I. These jigs presented in both cells are symmetric and their purpose is to hold and guarantee a firm position to allow operations over the parts with the two robots (KUKA and Fanuc). These two robots, despite being programmed according to distinguished Standards, (composed by 6 axes, controller, and the possibility to plug a tool on the sixth axis) can perform the same detailed actions in parts.

The LLDs for monitoring and control these two cells has been implemented applying two distinct approaches. First of all, a traditional programming approach has been considered and the time needed to create LLDs has been monitored. Secondly, the manufacturing cell has been programmed, i.e. the LLDs have been implemented using the proposed approach.

During the design and development of the offline monitoring and control solution two fundamental differences between the applied approaches arose, namely:
1) The time needed to model the problem;
2) and the time needed to create the LLDs;

As a matter of fact, applying a traditional programming approach the modelling time is extremely short (the coding starts as soon as possible without any modelling activity), while on the contrary the coding time varies between five and seven days. On the other hand, applying the proposed methodology the modelling time is greater than before and varies between one or two days (depending on how complex the manufacturing cell is), however the coding time is almost zero since the LLDs are automatically generated and the programmer needs only to check the generated files. The tests performed, clearly shown that the application of the proposed methodology could improve the coding activity performed especially by system integrators while keeping the code standard compliant. As a matter of fact, the results of the application of the proposed methodology can be resumed in a reduction of almost 70% of the time needed to generate the LLDs and as a consequence a significant reduction of the programming effort whenever modifications should be realized inside the PLC code (see Fig. 6).

V. CONCLUSION

The following paper presents an innovative methodology for automatically generate the LLDs. The key element and innovative aspect of these methodology is that it relies on a unique model that is generated from the entire set of manufacturing process specifications. The choice of the formal representation for manufacturing machines/stations has been realized considering the main documents and principal techniques used by system integrators to design and implement process specific LLDs. All the steps of the methodology have been described and an application scenario has been considered to show the feasibility and repeatability of the proposed approach. Furthermore, relevant results has been presented showing the reduction of the time necessary to create LLDs also called coding time. Finally, it is imperative to state that the reduction of the coding time is not the only achievement since typically the development of LLDs is an error-prone activity especially when 100% performed by human. In this scenario, the proposed methodology allows the reduction of the errors during the coding activity while guaranting a standard compliant code since it is template-based. The LLDs created
using the proposed methodology can be directly downloaded inside PLCs and is ready to be tested.

ACKNOWLEDGMENT

This work was funded by the Regional Operational Programme of Lisbon (POR Lisboa), in the scope of the National Strategic Reference Framework of Portugal (QREN), part of the European Regional Development Fund (FEDER). This work is also supported by FCT Fundação para a Ciência e Tecnologia under project grant Pest-OE/EEI/UI0066/2011.

REFERENCES


<table>
<thead>
<tr>
<th>Type of Controller</th>
<th>Main characteristics</th>
<th>Type of Controller</th>
<th>Main characteristics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rockwell</td>
<td>Safety softwired</td>
<td>Siemens</td>
<td>Safety softwired</td>
</tr>
<tr>
<td></td>
<td>Ethernet IP</td>
<td></td>
<td>ProfiNet</td>
</tr>
<tr>
<td></td>
<td>Implicit Sequencer</td>
<td></td>
<td>Explicit Sequencer</td>
</tr>
<tr>
<td></td>
<td>(DCP)</td>
<td></td>
<td>(S7 Graph)</td>
</tr>
<tr>
<td></td>
<td>Dynamic Address</td>
<td></td>
<td>Static Address Space</td>
</tr>
<tr>
<td></td>
<td>space</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>RISC</td>
<td></td>
<td>CISC</td>
</tr>
<tr>
<td></td>
<td>LLDs</td>
<td></td>
<td>LLDs, STL, FBD</td>
</tr>
<tr>
<td></td>
<td>Integrated Development Environment</td>
<td></td>
<td>Modularized Development Environment</td>
</tr>
</tbody>
</table>

TABLE I

COMPARISON BETWEEN FORD MOTOR COMPANY AND VOLKSWAGEN AG STANDARDS