Design pattern for industrial control applications based on one-line IEC 61499 adapter connections

Pranay Jhunjhunwala*, Polina Ovsiannikova†, Valeriy Vyatkin∗†

*Department of Electrical Engineering and Automation, Aalto University, Espoo, Finland
†Department of Computer Science, Computer and Space Engineering, Lulea Tekniska Universitet, Sweden

Email: {pranay.jhunjhunwala,polina.ovsiannikova}@aalto.fi, vyatkin@ieee.org

Abstract—IEC 61499 is an emerging standard for distributed automation which requires well-defined design practises to improve development efficiency. In this paper, we extend the one-line engineering design pattern and provide step-by-step guidelines on its implementation keeping in mind key features of Industry 4.0. The one-line design pattern simplifies the modification of the application structure and the addition of new components to the system without disrupting the existing control application. It also enables easy fault identification and debugging of the system. The design pattern was developed and validated in three different use cases that demonstrated the applicability of the pattern.

Index Terms—Design pattern, one-line engineering, modularity, IEC 61499

I. INTRODUCTION

Industry 4.0 is gaining traction and being adopted by all industries around the world. Industries are looking at flexible automation, energy-efficient solutions, data connectivity and analysis, etc. Flexible automation systems are the focus of Industry 4.0 achieved by exploring the object-based software design principles, implemented by means of major automation standards IEC 61131-3 [1] and IEC 61499 [2].

The IEC 61499 standard has begun to emerge as a major enabler of flexible automation due to its capability to support distributed and decentralised automation. Using the standard, various directions of research have been carried out. For example, one-line engineering was invented to keep the block diagrams of object-based systems tidy [3]. There have been works related to the development of applications based on service-oriented architecture, skill-based engineering, etc., which will be discussed in detail in Section II.

Researchers have also carried out research to extend and improve the standard. In [4], [5], the authors have proposed the extensions of IEC 61499 to address dynamicity and complexity.

Although the literature and research have helped the standard and its applicability, the authors believe that there is a lack of guidelines with well-defined steps and limitations for the development of automation software, especially having applicability across application domains such as discrete or process automation. In this paper, we propose a well-defined step-by-step guideline for the development of control applications based on the one-line engineering invested by the authors in [3]. The one-line pattern helps improve readability, reusability, and reconfigurability by connecting significant parts of the software in a single line.

The remainder of the paper is structured as follows. In Section II we briefly introduce the IEC 61499 programming language and place a one-line design pattern among the known ones. Section III gives the step-by-step guideline on the application of the design pattern, starting with the equipment analysis. Sections IV and V present two applications from discrete and process domains, where the one-line architecture was successfully applied. In Section VI we discuss the major benefits and drawbacks of the proposed design pattern. Section VII concludes the paper.

II. PRELIMINARIES

A. IEC 61499†

IEC 61499 defines an architecture for distributed automation and control systems. According to the standard, the control program is implemented using a graphical programming language of Function Block Diagrams (FBDs). Unlike the FBD language of IEC 61131-3, in IEC 61499 Function Blocks (FBs) communicate with each other through message passing, implemented by means of event and data flow. In addition to the data interface (that is, input and output data variables), IEC 61499 FBs have the event interface (that is, input and output event variables). This changes the execution type of programmes from predefined cycle-based to event-based. Any event input or output of an IEC 61499 FB (hereinafter, an FB) can be associated with a subset of data input or output of this FB, respectively, which means that the corresponding data will be received, processed or sent only if the particular event fires.

There exist three types of FB: basic, composite, and service interface, which differ by the way their internal logic is implemented. The logic of basic FBs is defined by their Execution Control Chart (ECC), which is, essentially, a Moore state machine. The actions that can be performed in the states of an ECC are programmed using PLC languages, e.g., the Structured Text language. ECC states are also the initial event emitters. Service interface FBs are wrappers for code libraries, for example, functionality implemented using programming

†The subsection is intended for an audience without prior IEC 61499 knowledge and can be skipped otherwise.
languages such as C#. Composite FBs are defined by networks of FBs of any kind. An application is a network of function block instances.

B. IEC 61499 FBD-level design patterns

The scope of this work includes the FBD-level design patterns for IEC 61499 programs. We strive to find a pattern that will allow writing better (more readable, reconfigurable, and reusable) code, and which is applicable to both discrete and process domains. We address general architectural design patterns that are not focused on particular logical subsystems of the application, such as, for example, fault handling.

The on-line design pattern presented in this paper can utilise the lower-level patterns focused on solving specific program code/FBs arrangement/communication problems and can be combined with higher-level architectural design principles.

An example of a higher-level design principle is Service-Oriented Architecture (SOA), which is on the frontier of design patterns, allowing reconfigurability and applicable to IEC 61499 programs [6]. It defines the main layers of the application, such as production, services, and execution. However, the engineer still decides how the FBs are organised within the layers, while instead a one-line engineering approach can be applied.

The same holds for other high-level architectural patterns, such as skill-based design discussed in [7], [8], [9], modularization technique from [10], and hierarchical automation design [11]. One-line engineering can be used to further enhance the readability of product-centric [12] and component-based [13] designs.

More case-focused design patterns, for example, dynamic adapter connections from [5] can be incorporated into the one-line architecture to solve the subsystem interaction problem. Another example is a monitoring adapter connections design pattern from [14] and a general code-simplification design pattern from [15].

III. GUIDELINE OR METHODOLOGY

For the design pattern, we take as the backbone the one-line adapter connection architecture presented by Gernot et al. in [3]. The authors propose the development of a distributed belt line application in which various FBs are connected to each other in a single line using adapter links. These adapter connections would be responsible for data exchange with neighbouring FBs. The application showcased performed a basic operation of numerous conveyor lines connected to one another.

Using the one-line engineering design pattern as a reference point, we have developed a guideline for the development of control application and the outline is shown in Fig. 1.

The methodology will be explained, using the example of the EnAS demonstrator shown in the bottom right corner of Fig. 2.

A. Stage 1: Analysis of the structure of the equipment

In the first state of the methodology, we analyse the physical structure of the equipment under consideration. Various mechanical components are analysed, and an interconnection diagram is developed. The diagram not only showcases the interconnection between various mechanical components but also indicates the capability of each component. Put together, we will define the final product or process being undertaken by the equipment.

For example, the case of the EnAS demonstrator is shown in Fig. 2, the equipment consists of six conveyors connected in a cyclic chain. Furthermore, the equipment consists of pneumatic operators such as a pair of jacks.

Together, the demonstrator is capable of producing workpieces of different colours by circulating them across 6 conveyors and using pneumatic operators to pick or place components as required.

B. Stage 2: Architecture planning

Focussing the design pattern around modularity and intensive reusability of developments, a layered architecture is proposed. Layered architecture helps to reduce the effort to incorporate new production sequences with the same hardware. Furthermore, the modular design enhances the possibility of adding new components to the system without disrupting the flow of existing applications.

The bottom layer, that is, the execution services, would correspond to the physical hardware in terms of connections. The highest layer would be the production layer which would contain the production recipe. In between the production and execution layers are the planning and services layers which would be responsible for managing the execution services based on the requirements from the production recipe. The number of service layers can vary according to the application and use case as showcased in the outline in Fig. 1.

The architecture is designed such that the bottommost layer replicates the interconnection diagram developed in stage 1,
which means the components of the execution layer would be connected in a pattern similar to the physical hardware. This would ensure ease of assembly of the system and would be helpful during software debugging in case of failure situations.

In the case of the EnAS demonstrator, we observed the presence of six conveyors connected in a cyclic pattern and the presence of additional independent pneumatic components. Using this information, the execution layer would have six FBs connected in a cyclic pattern, each representing one conveyor, as shown in Fig. 2. Subsequently, there would be additional FBs to represent the pneumatic jacks marked with the green line in Fig. 2.

In this case, two types of services were identified, that is, 1) Delivery Services responsible for the transportation of the workpiece using conveyors and 2) Pneumatic Services responsible for the pick-and-place operations of the jacks. Each of these services would have their respective FB and be connected to the execution layer. A key point to be noted is the connection between the services and execution layer, in case of cyclic connections, the services layer would only be connected to the first element in the chain, for example, conveyor 1 in EnAS whereas there would be individual connections for standalone operators. Finally, the highest layer would have the production recipe. An additional 'organisation service' was used to differentiate between the need for delivery or pneumatic services.

C. Stage 3: Development

Based on the architecture and structure obtained from the previous stages, we will design the FBs and communication mechanisms for each layer. To achieve the operation based on the one-line design pattern, a specific message-passing format has been proposed in the methodology. The format of the messages depends on the services and capabilities available. In this pattern, instead of directly giving commands to each component, the service layer generates a general command and passes it downstream to the first component. It is the responsibility of the components to check if the command belongs to them and then act accordingly.

For example, in the case of the EnAS demonstrator, the conveyors are connected in a cycle chain and can carry the workpiece from one conveyor to another. A workpiece can be transported from conveyor 2 to 4, and that information can be used to define the commands for the production recipe, for example, C2_to_C4. To achieve this, the workpiece has to start motion from conveyor 2, pass conveyor 3 and finally be delivered at conveyor 4. This showcases that conveyors in the EnAS demonstrator can go through three different modes of operation, which are 1) Mode C - assuming the workpiece is
on the conveyor and operation starts from there 2) Mode P - the workpiece is coming from the previous conveyor and needs to be passed on to the next conveyor 3) Mode D - workpiece needs to be delivered to the conveyor.

Based on this definition, the conveyor function blocks can expect messages in three different formats, and this would be accomplished by the services layer, which would take as input the command from the recipe and produce the output usable by the execution blocks. This message passing has been shown with the help of a sequence diagram in Fig. 2. Similarly, a message-passing system was developed for the pneumatic components.

Fig. 2 also shows the final developed application broken down into various layers as highlighted in the proposed outline.

The design pattern was tested and verified by developing the automation applications for two different use cases. The use cases were 1) modular production island and 2) vertical farming. The use cases and developed applications are explained below.

IV. MODULAR PRODUCTION ISLAND USE-CASE

This use case is based on the Festo CP Lab system which is a compact modular assembly process model of a laboratory scale. CP Lab covers complex software-related Industry 4.0 topics in mechatronics and automation technology. The system shown in the lower right corner of Fig. 3 is the CP lab station which is used to demonstrate modularity and flexibility through independent processing stations, allowing changes in routing, process, product, and operation.

A. Stage 1

The structure of modular production stations was analysed and various components were identified. The setup consisted of four conveyor stations and four application modules, which were two pneumatic magazines, a pneumatic press, and a camera inspection station. The structure was such that each conveyor station would have one of the application modules placed on top.

Using the structure, we draw the interconnection diagram shown in Fig. 3. Together, the stations are able to produce a dummy mobile phone by following a sequence of operations that helped to decide the order of production.

B. Stage 2

Following the proposed pattern, the bottom layer would contain four FBs connected in a cyclic chain to represent the four conveyors. Furthermore, there would be individual FBs representing the application modules.

In the case of CP Lab, two types of services are identified, that is, delivery services and application services. Delivery services would be responsible for providing the execution blocks with the correct commands based on the message received from the production recipe. The application services will ensure the operation of the correct application module based on the recipe, but would also be responsible for mapping the location of each module upon initialisation.

C. Stage 3

In this stage, the message passing and communication between various layers have been developed.

The conveyors of CP Lab can perform the exact same operations as that of the EnAS demonstrator but are equipped with additional sensors at the start and the end of the conveyors, enabling the workpieces to be delivered at the start or the end of the conveyor as well. To achieve this, two additional modes were added to the existing list of modes C, P, and D. The new modes S and E would be responsible for indicating the delivery at the start or end of the conveyor, respectively. These modes of operation would be provided to the conveyor line using delivery services, and an example has been shown in Fig. 3.

We assigned codes to each of the application modules such as 1) MB: Magazine Back cover, 2) MF: Magazine Front Cover, 3) KA: Camera inspection, and 4) MP: Muscle press. These codes were not only used during initialisation but also used by the production recipe and the application services layer to ensure correct operations. The resulting structure and application have been shown in Fig. 3.

V. VERTICAL FARM USE-CASE

Vertical farms have gained popularity in the last decade and shown in Fig. 4 is a vertical farming module present at the Aalto Factory of the Future. The vertical farm composed of such modules supports the addition of numerous growing modules and is the research ground for modular process control technologies. The structure of such a farm allows the reconfigurability of the modules and enables research on comprehensive software solutions to achieve automated reconfigurability.

Unlike the other use cases investigated above, the vertical farming module falls under the process control industry due to the presence of continuous processes. Even though the use case is drastically different, the same methodology can be applied to develop the control application.

A. Stage 1

The vertical farming module consisted of various components that act as individual entities. The core identified elements are a) proximity service (plant detection), b) fan execution (air quality control), c) lights and d) pump (water supply control). Some of these elements were just sensors, whereas others were composed of both sensors and actuators. For example, air quality control consists of a sensor to check CO2, humidity, and temperature, and has two fans as actuators.

The only common link between these components is the plant detection sensors based on which the services are allocated. Using this information, the interconnection diagram shown in Fig. 4 was developed.

---

2https://www.aalto.fi/en/futurefactory
B. Stage 2

The architecture for this control application retained a similar structure as observed in the applications above, that is, the bottommost layer consisted of FBs forming the execution services. Above which was the services layer responsible for organising the execution.

The critical difference identified between a discrete control application and a process control application was communication between different layers. In the EnAS and CP Lab application, we observed that each time a process was completed, the information was sent upwards to the organisation and production layer, whereas in the case of the vertical farming module, the conditions such as temperature or CO2 level would remain the same for a growing plant, hence the recipe would be a single-line recipe indicating growing parameters for the plants, and the execution blocks would continuously operate to achieve and maintain the desired parameters. For example, basil needs 20°C temperature, 54% humidity, 10% oxygen and 15 hours of light, the recipe would be \[ T20,H54,C1200,L15,P15,O10 \] as shown in Fig. 4. The operating conditions for the various blocks would be regulated by the respective service.

C. Stage 3

Vertical farming assumes a continuous process to grow a particular plant, therefore, the recipe was a single-line message containing all the growing parameters which is shown in the sequence diagram Fig. 4. The services layer would interpret the recipe and instruct all execution services to continuously monitor and achieve those parameters.

The message passing was developed such that when a recipe is processed, the system would wait for the proximity service to confirm the presence of a plant or growing medium, following which the desired settings would be communicated to the respective services. These services would break down the commands and accordingly process instructions for the blocks in the execution layer.

Shown in Fig. 4 is the resulting control application developed based on the pattern, in which at the highest layer we have the Recipe FB, followed by five different services at the ‘Services Layer’. Finally, as identified in stage 1, there were three components corresponding to the execution layer, which
communicate with the services using the one-line adapter connection.

VI. DISCUSSION

The proposed design pattern was used to develop control applications for three significantly different use cases. During the development and testing of these applications, various benefits and drawbacks of the pattern were identified, as discussed in the following sections.

A. Benefits

The design pattern was shown to be applicable both in the process and in the discrete domains of industrial automation.

The design pattern shows great potential for software reusability, modularity, and ease of system assembly. Furthermore, by using the design pattern, it is very easy to modify the structure of the application and add new components to the system without disrupting the existing control application, and it enables easy fault identification and debugging of the system.

1) Ease of system assemble and reusability of software components: For example, in the case of the modular production island CP-Lab, adding a fifth conveyor station would only require adding an instance of the conveyor FB and connecting the desired adapter connections, following which the setup would be ready to process a recipe that includes operations for conveyor 5. If no application module was added to the new conveyor, the system would still function as desired due to modularity in software development.

2) Expansion to Multi-Agent systems: Due to the modular nature of the architecture, composing a multi-agent system is easier compared to hardcoded applications. Each hardware element could be treated as an individual agent and could be enhanced with open-ended communication mechanisms such as OPC-UA or MQTT enabling it to communicate with different agents in the architecture without preprogrammed communication mechanisms.

Furthermore, an application representing one agent could be encapsulated to a function block, and a number of such modules can be deployed together to form a multi-agent system. For example, several growing modules in the vertical farm could be deployed together with the same code. The code will be enhanced with an agent manager, which would coordinate the presence of different modules in the system. Developing the application using the proposed pattern would make it possible to add such additional components and test during operation without disturbing the existing workflow of
the application, potentially reducing the time to market for the products.

One basic requirement of multi-agent systems is peer-to-peer communication of agents which may emerge dynamically during the run-time, that is, the recipient of the message may be unknown during the compile time. The proposed pattern copes well with this requirement. However, it introduces communication overhead at the application layer.

3) Debugging: In case of system failure or failure of a certain component, it is extremely easy to trace back the fault to the respective software component because of the software architecture replicating the physical hardware setup and connections.

B. Drawbacks

A significant drawback of the design pattern was the communication overhead observed in the cyclic connections of the conveyor FB on the EnAS demonstrator and the CP Lab production island. For the messages had to be passed through each conveyor FB to be delivered to the desired location, resulting in a significant increase in communication time.

Furthermore, if devices at any of the conveyor stations fail, messages would not be sent to the blocks, causing the entire production scenario to stop until the device is restored or a backup device takes control.

VII. Conclusions and Future Work

The proposed design pattern was adopted from [3], in which the authors developed software for a belt line application. We used the development by the authors as a backbone and proposed a step-based design pattern to incorporate various aspects of a machine into the control application following the one-line adapter pattern. Our solution enhanced the original work by providing a layered structure to have a complete software solution for the production unit.

The pattern was developed and validated in three different use cases demonstrating the applicability of the pattern. A step-based method has been used to develop control applications keeping in mind key features of Industry 4.0. The pattern brings up opportunities for significant future work, which have been highlighted below.

A. Introduction of bus architecture in IEC61499

In order to resolve the communication overhead drawback, the adapter technology of the IEC 61499 standard could be upgraded with a bus-based architecture of adapter connections, which would allow for one-to-many communication. This, in turn, would more efficiently map to wireless communication networks.

B. Expansion to Industry 4.0 Standards

Various open standards such as Asset Administration Shell (AAS), Skill-based engineering, and Module Type Package (MTP) have been intensively researched over the past couple of years and are now gaining popularity. Future work could be the incorporation of these open standards into the design pattern. Moreover, enhancing the pattern in a way such that these standards can be incorporated into existing control applications in a non-disruptive form.

ACKNOWLEDGMENT

This work was sponsored, in part, by the Horizon Europe project Zero-SWARM (grant agreement: 101057083) funded by the European Commission.

REFERENCES