Business Process Models for Analysis of Industrial IoT Applications

Ajay Krishna, Gwen Salaün

Business Process Models for Analysis of Industrial IoT Applications

Ajay Krishna
Univ. Grenoble Alpes, Inria, CNRS, Grenoble INP, LIG
Grenoble, France

Gwen Salaün
Univ. Grenoble Alpes, CNRS, Grenoble INP, Inria, LIG
Grenoble, France

ABSTRACT

Industrial automation is a complex process involving various stakeholders. The two important aspects to consider during the automation system development are its business production goals and its technical implementation. The international standard IEC 61499 helps to specify distributed automation using a generic architectural model, targeting the technical development of the automation. However, it is not easy to analyse whether these IEC 61499 models satisfy production goals due to their informal semantics and inherent complexities of distributed logic. In this paper, we propose to use Business Process Model Notation (BPMN) to express the underlying IEC 61499 specification. We propose a transformation from an IEC 61499 specification to the BPMN model. This model presents the automation from a business point-of-view, and it also enables quantitative analysis of process models. Specifically, it allows business analysts to perform analysis related to cost, resource allocation, and time in automation. This analysis is achieved by transforming the business processes to formal model in Maude rewriting logic. The viability of our proposals is illustrated using an industrial case study of a packaging system.

CCS CONCEPTS

• **Software and its engineering → Context specific languages**; **Formal software verification**.

KEYWORDS

rewriting logic, IEC 61499, IIoT, optimization

ACM Reference Format:
Ajay Krishna and Gwen Salaün. 2021. Business Process Models for Analysis of Industrial IoT Applications. In *IoT '21: 11th International Conference on the Internet of Things, November 08–12, 2021, St. Gallen, Switzerland*. ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3494322.3494336

1 INTRODUCTION

Industrial automation consists of complex control systems that are designed to handle industrial processes and machines requiring limited human intervention. The goal of automation is to improve production efficiency in terms of outputs and costs. The automation systems have evolved from simple feedback controllers to large connected systems with hundreds of distributed components leading the fourth industrial revolution (Industry 4.0) [23]. The global Industry 4.0 market size is expected to reach $210 billion by 2026and one of the factors driving this trend is the promise of increased productivity (up to 30%) [22].

In recent years, efforts have been made to standardize the development of production systems to enable portability and wider adoption of automation systems. Standards and frameworks such as the International Electrotechnical Commission (IEC) 61499 [17], Arrowhead [5], OPC-UA [16] target different aspects of automation design such as modelling distributed processes, network communication, and interoperability of connected devices [28]. Specifically, IEC 61499 standard aims at distributed automation by encoding automation logic in modular and portable units known as Function Block (FB). IEC 61499 allows system developer to define the system in an abstract manner without worrying about the deployment hardware architecture.

The IEC 61499 model is useful for technical development of the system, business users however may find it difficult to understand. One needs to know the syntax and execution semantics to completely understand the underlying automation. Moreover, the control logic of the automation is encoded within the FB using algorithms and state-transition systems. Thus, if business users are to view the FBs as a black box, they would miss out on the information related to dependencies and order of execution of the control logic.

Industrial automation, especially in the context of Industrial IoT (IIoT) and Industry 4.0, is a complex process aimed at increasing productivity without exceeding the business constraints related to resources and costs. These constraints are sensitive to market factors such as supply and demand, material availability, and time to market. Therefore, product planners and business analysts need to efficiently manage the available resources to maintain optimal production output. Another challenge in industrial automation is the cost of deployment, which contributes to a large part of capital expenditure in an organization. The decisions related to scaling of automation need to be carefully thought out before deployment. IEC 61499 modelling allows developers to verify the functional correctness of the system. They can simulate the design using IEC 61499 runtime environments before deployment. However, the model specification is not suited for quantitative analysis which could help in making business decisions [2, 27]. For instance, it is difficult to answer questions such as: does an additional machine in the automation increase the production throughput? Is the return on investment acceptable? What is the bottleneck in the automation? What is the impact of speeding up a certain part of the automation process? Answering these questions at design time would go a long way in optimizing costs and increasing productivity.

IoT '21, November 08–12, 2021, St. Gallen, Switzerland

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/10.1145/3494322.3494336

Business Process Model Notation (BPMN) is a graphical notation for modelling business workflows. The goal of BPMN is to describe the different activities of a business through a common notation for both technical and business stakeholders, thereby bridging the gap between business processes and its implementation. BPMN has gained wide adoption across organizations due to its ability to represent complex process semantics as a network of graphical objects understandable by all stakeholders.

The broad goal of this work is to increase productivity by involving both technical and business users. This work proposes to transform the IEC 61499 models to BPMN with an aim of presenting the automation to business users. By modelling the system as a business process, we allow business users to easily understand the automation as BPMN is widely adopted in the industry. Also, through BPMN, we can express both communication and internal behaviour of the FBs in a unified notation. We specifically use collaboration diagrams, which are suited for modelling distributed automation and message communication. This model can be enriched with cost and resource information to perform quantitative analysis. The analysis is achieved by transforming the BPMN model to a formal specification in Maude rewriting logic [4, 11]. The results of the analysis can help business users make better decisions regarding the deployment of the automation. In this paper, we illustrate our proposals through an industrial case study and the experimental results indicate that optimal deployment of automation can be achieved through quantitative analysis.

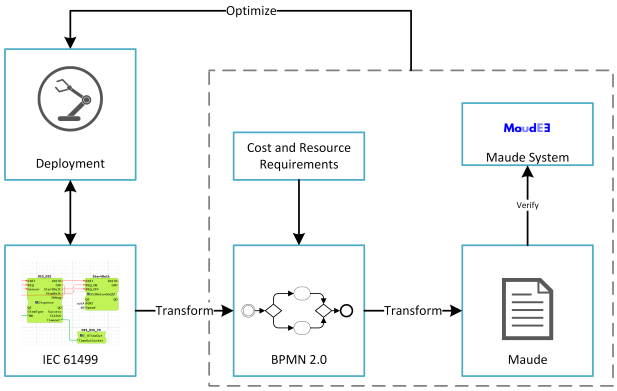

Figure 1: Productive Deployment of Industrial Automation

An overview of our approach is shown in Figure 1. The IEC 61499 model of the industrial system is the basis for productive deployment of the system. The model is transformed to a BPMN 2.0 specification. Business analysts can introduce the cost and resource information to this BPMN model in the form of annotations. Further, a Maude specification is generated from the BPMN model which is executed by the Maude compilers to perform the analysis. Decisions for optimal deployment of resources can be made based on this analysis. To summarize, the key contributions of this work are: i) A business process view of industrial automation based on IEC 61499 standard; ii) A model-to-model transformation of IEC 61499 to BPMN; iii) The verification of properties related to execution times, resources and costs by analysing the Maude specification generated from the BPMN model; iv) An industrial case study illustrating the viability of the proposals.

The remainder of the paper is organized as follows: Section 2 introduces the IEC 61449 concepts followed by a description of BPMN standard. Section 3 describes the transformation of IEC 61499 models to BPMN for analysis using Maude. This is followed by Section 4 where an industrial case study is presented. Section 5 covers the related works and finally, Section 6 concludes the paper with an insight on future work.

2 BACKGROUND: IEC 61499 AND BPMN

This section provides a brief introduction to modelling in IEC 61499 and BPMN.

2.1 IEC 61499

IEC 61499 is a standard for distributed industrial automation to enable portability, interoperability, and reconfiguration of distributed applications. IEC 61499 is based on the IEC 61131 standard for programmable controllers. It defines event-driven execution architecture along with a block-based programming language [26]. The applications are defined by networks of interconnected Function Blocks (FBs) that can be subsequently deployed across available resources, i.e., the FBs can run on the Programmable Logic Controllers (PLCs) used to control the machines.

FB is the key element of the IEC 61499 standard. The left part of Figure 2 shows a generic FB. The FB provides interfaces for event and data flows. The event flows are marked in red, and the data flows are shown in green. The diagram convention places events on the top half of the FB (data on the bottom half), the inputs are shown on the left side and the outputs are shown on the right. Events can be associated with data inputs and outputs on their side. The blocks encapsulate the desired behaviour defined using an Execution Control Chart (ECC), a state-transition system and algorithms. Events trigger the execution of FB and input data may be used during the execution as part of data expressions.

There are three kinds of FBs: Basic FBs, Composite FBs and Service Interface FBs. The FBs contain event and data interfaces that are classified as inputs or outputs. Data interfaces can be associated to event interfaces. The ECC is a state-transition system that receives input events and depending on the current state, it can execute a transition to move to the next state. Each state can have actions associated with it. An action consists of an algorithm and an output event. The algorithms can be defined using different IEC 61131-3 programming languages or other languages such as Java. They can use internal data variables and the data associated with the input events to generate output events. The output events are generated once the algorithm terminates, and these events can be used to trigger the execution of another FB. In IEC 61499, four types of transitions are possible from a given state: i) Event transition, where the move to next state solely depends on an event ($s_1 \xrightarrow{ei} s_2$) ; ii) Event-data expression, where an event and associated data expression needs to be evaluated to true for the transition to occur ($s_1 \xrightarrow{ei[exp]} s_2$) ; iii) Data (guard) transition, where the move to next state depends on the data ($s_1 \xrightarrow{di} s_2$) ; iv) Default transition,

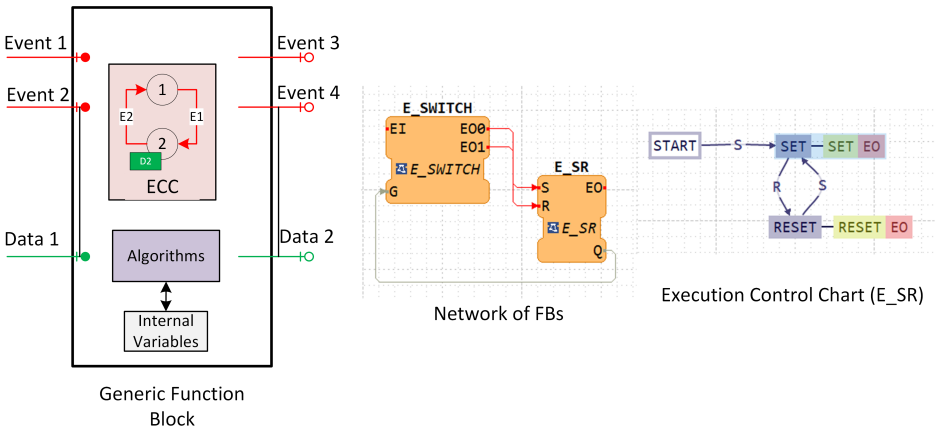

Figure 2: IEC 61499 Model: FB, Network of FBs, and ECC

where no event or data input is required to move to the next state ($s_1 \xrightarrow{1} s_2$). This transition is denoted by a special label 1.

It is to be noted that an event is valid only once when it is evaluated in a transition. Also, no more than one input event is delivered at any instant in time to the FB.

A Network of FBs consists of Basic FBs connected together via event and data bindings (connections). The outputs of a FB are fed as inputs to another FB to form a network of FBs. According to the specification, event and data connections cannot be mixed.

The IEC 61499 standard provides loosely defined semantics [26, 29]. The IEC 61499 tools make different choices in the implementation of the semantics for some specific constructs. In this work, we follow the semantics defined by the 4DIAC tool as it is open source, and its runtime is used by the commercial nxtStudio tool as well [19].

Figure 2 (centre) shows a network of two FBs, modelled using 4DIAC. The names of the block instances are displayed on the top (E_SWITCH and E_SR) and the blue squares within the FB represents the ECC encoded in the block. The E_SWITCH has an event input EI and a data input G. It is connected to a switch set-set block E_SR via output events EO0 and EO1. The data output Q is fed back to input G of the E_SWITCH to form a feedback loop. The ECC of the switch set-set block E_SR is shown on the right of Figure 2. It has an initial state START and the event set (S) triggers the move to state SET. This state is associated with an action consisting of the algorithm SET and output event EO. The reset event R triggers the transition to RESET state. In this ECC, the transitions are simple events, but they may contain expressions in the form $E[exp]$, where E is the event and exp is the event condition that needs to be satisfied for the transition to occur.

2.2 Business Process Model Notation

BPMN provides a graphical notation for modelling business processes. Currently, BPMN 2.0 [1] is the ISO/OMG standard that is in practice. BPMN supports three types of diagrams: process diagrams, collaboration diagrams, and choreographies. Process diagrams model a business process as a sequence of activities. Collaboration diagrams are an extension of process diagrams that can be used to model interaction between two or more business processes. The interactions occur through the exchange of messages. Choreography is used to model message interactions between participants. In this work, we focus on BPMN collaboration diagrams, which provide the right level of expressiveness for modelling IEC 61499 specifications. More precisely, modelling of industrial automation can be achieved using the following set of BPMN elements: Pools, Tasks, Gateways, Events, and Flows.

Pool: Pool represents an organization in a business process. It indicates the participants or resources involved in a process. Communication with processes outside the boundaries of a pool is done through messages.

Start Event: It indicates the beginning of a process. Triggering of a Start Event initiates the flow of a token across the process.

End Event: It indicates where the process execution will terminate. A process can have multiple end events.

Intermediate Message Event: These are the intermediate events between Start and End events that are used to send or receive messages. Throw Message Event sends a message to the communicating BPMN element and continues the process execution onwards. Catch Message Event waits for the arrival of message and proceeds with the process execution upon receiving the message.

Tasks: Task is an atomic activity, and it generally represents a piece of work in the process. From a modelling perspective, we treat all kinds of tasks as an abstract task, and it is assumed that a task has only one outgoing flow.

Exclusive Gateway: A split Exclusive Gateway is used to create alternative paths, where only one of the proposed paths can be taken. Merge pattern of the gateway is used to converge alternative paths and also to construct looping behaviour. Each incoming flow token is sent across outgoing flow without any synchronization.

Parallel Gateway: A parallel gateway split creates parallel paths, without checking any conditions. A parallel merge waits for all incoming flows before triggering the flow through its outgoing sequence flows. It synchronises tokens from multiple parallel paths.

Inclusive Gateway: An Inclusive split pattern behaves like a logical OR ($\vee$) clause. Unlike Exclusive split, it evaluates all the

outgoing flows and triggers all the flows that evaluate to *true* and in case of inclusive merge, the tokens may be synchronized.

Sequence Flow: Each sequence flow has one source and one target element. It can also encode a condition, wherein the token will flow through it, only if the condition evaluates to *true*.

Message Flow: Message flow is a type of flow which is characterised by exchange of messages. In a sequence flow, the execution token moves from the source BPMN element (e.g., a task or a gateway) to the target element without a message, whereas in a message flow, the movement of the token is associated with a message communication.

Annotations: Annotations are textual information that can be added to the BPMN elements to provide additional context or detail for the reader of the BPMN model.

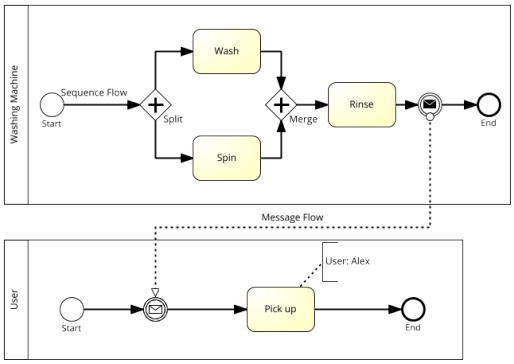

Figure 3: BPMN Collaboration Diagram with Message Flows

Figure 3 shows a simple collaboration diagram. It consists of two pools representing a washing machine and a user. The parallel gateway with split and merge pattern indicates that the machine washes and spins the clothes before rinsing them. Once the wash cycle is complete, a message is sent to the user to pick up the clothes as indicated by the message flow. A text annotation indicating the resource (a user named Alex) associated with the pick-up task can also be seen in the figure.

3 TRANSFORMATION FROM IEC 61499 TO BPMN FOR ANALYSIS USING MAUDE

In this section we describe the patterns for transforming an IEC 61499 model to a BPMN specification. Then we briefly describe the encoding of BPMN in Maude for analysis.

3.1 Transformation from IEC 61499 to BPMN

Let us define the transformation starting from the network of FBs, followed by the ECC and finally the patterns for different types of transitions in the ECC.

Network of Function Blocks. Each FB in an application is modelled as a BPMN Pool. This mapping is chosen because each FB can be mapped to resources and a Pool offers such behaviour in BPMN. The data and event bindings are specified as message flows as they indicate communication in a certain direction. The source and target of message flows are intermediate message events. Events and data are mapped to intermediate message events as they occur during the execution of a FB. More precisely, event and data inputs are transformed to message catch events, i.e., they wait for the arrival of the message. Similarly, the event and data outputs are modelled as message throw events. Since a data output can be sent to multiple data inputs and similarly, event inputs can be combined or split, they require special treatment during transformation. Fan-out of messages is modelled using a parallel split gateway to indicate the flow of messages to different inputs and Fan-in of messages is modelled as an exclusive merge, indicating more than one message may be merged across the flow. The relationship between data and event in the FB is modelled using a pair of parallel split and parallel merge gateways with corresponding Intermediate Message Events in between them. Parallel merge gateway synchronizes on all incoming flows, which implies both data and event need to be available before continuing the execution.

Execution Control Chart. The transformations above relate to the interfaces of the FBs and the connections. Now, we need to transform the ECC inside an FB (described in Figure 4). The ECC behaviour is modelled as a BPMN process inside the FB Pool. The initialization of a FB (start) is modelled as a start event in BPMN. A state in an ECC is mapped to a task. IEC 61499 does not provide information related to state attributes, therefore the task can be viewed as an abstract task. The algorithm associated with the state is specified as a task, followed by the task representing its state (it can be specified within the task representing the state, if the algorithm operation does not need to be explicitly specified). The output associated with the state is represented by an intermediate message throw event. This ordering respects the execution semantics of IEC 61499, i.e., once a state is reached, algorithm(s) associated with it are executed and upon completion of the execution, data output is sent. The terminal states are connected to the end event using a sequence flow. If there are more than one such states, then an inclusive gateway is used to merge such states to the end event. When the

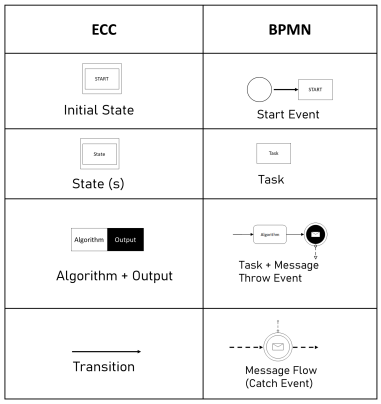

Figure 4: Transformation of ECC

ECC of the FB is not explicitly described in the specification, we model the FB as a black box, i.e., empty box pool with incoming and outgoing message flows or add a generic task with associated message flows in a pool.

Transitions . The transitions are transformed into message flows and sequence flows depending on the type of the transition. IEC 61499 supports different types of transitions which can be associated with event and data inputs. The associated inputs are captured through intermediate message catch events. The transitions with data and event inputs are associated with intermediate message catch events. Since the transitions occur upon the arrival of an event or data input, the message catch event pattern is chosen. The transitions in ECC where an event is evaluated with a data expression ($s_1 \xrightarrow{ei[exp]} s_2$), requires both event and data inputs to be available. This is modelled in a parallel split and merge pattern. The data and event input (message catch events) are placed in between the pair of gateways, this ensures that the execution cannot proceed further unless both data input and event input have been received for evaluation of the expression. The default transition in ECC ($s_1 \xrightarrow{1} s_2$), where no input is required to move to the next state is modelled as an unconditional sequence flow in BPMN. When there is more than one transition possible from a given state, they are modelled as an exclusive split gateway to account for all the possible choices from the state.

IEC 61499 allows assigning of priorities for transitions when more than one transition is possible from a given state. However, BPMN does not provide mechanisms to assign priorities to its flows. This is handled by assigning probabilities to the transitions during quantitative analysis. The state space of the BPMN model consists of all possible executions. So, when we perform quantitative analysis, we assign probabilities to the possible outgoing flows in the BPMN model that correspond to the possible data inputs. For instance, if there are three transitions possible from a state and for a given input data, only two transitions evaluate to true, then we assign a probability of zero to the transition that evaluates to false. Thus, we can achieve a realistic execution model from the overapproximate model.

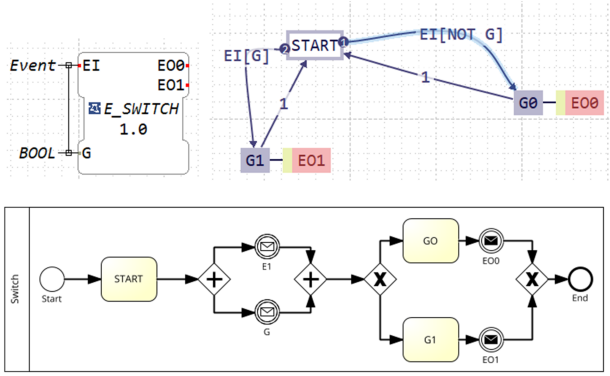

Figure 5: ECC to BPMN Transformation

Figure 5 illustrates the transformation of an ECC associated with a switch. The FB has two inputs *EI* and *G* as shown in its FB interface (top left). The ECC (top right) has two states, in addition to the initial start state. These states are transformed to tasks in BPMN. The data and event inputs are related (as seen in the FB interface on the top left) and the transitions are in the form *event[exp]*, so the parallel split-merge is modelled for intermediate message catch events associated with the event and data inputs. The possibility of two transitions after the state is modelled as a choice using an exclusive gateway. Readers may note that we have not modelled the looping default transition in BPMN because a new instance of the process would be initiated from the beginning once the current event completes its execution. The lifetime of an event in IEC 61499 is only until it is evaluated in a transition. So, any subsequent transition should be a result of a new event.

3.2 Transformation from BPMN to Maude

The BPMN model generated from the IEC 61499 specification allows us to reuse existing tools for analysis of BPMN models. In this work, we rely on recent results [10, 11], which support the verification of several quantitative properties over BPMN models. This approach proposes to automatically translate BPMN models into Maude's rewriting logic, thus enabling the use of Maude's tools for automated verification.

Rewriting logic [18] is a logic of change that is suited to model states and nondeterministic concurrent computations. Maude System [4] provides a formal toolbox in which Maude, a high-level language can be used to specify an executable formal specification.

The approach presented in [10, 11] supports a subset of BPMN process elements and this subset is expressive enough to model IEC 61499 applications. The Maude specification relies on simulation techniques and compute average execution times of the BPMN process as well as several properties related to resource usage over time with varying workload and number of resources. This information is useful in order to identify execution time, resource occupancy, dependencies and bottlenecks in the designed system. The resources correspond to the machines (e.g., workstations or dispensers) used in the application.

In the Maude specification, tasks and flows are associated with a stochastic distribution modelling the duration of execution and delay in task and flow, respectively. The tasks can also be associated with a set of resources required for its execution. The outgoing flows in an exclusive split gateway can be assigned probabilities for each outgoing branch. The message flows are treated as special flows with a message identifier (message event) associated with it. Users can specify the resources, time and probabilities associated with the tasks and flows in the process as annotations (supported by BPMN specification) in the model. These parameters are used to verify properties in Maude. Specifically, three properties are encoded in Maude: average execution time, synchronization time, and resource usage. The average execution time is an indicator of the throughput. If the execution is faster, it implies that more items can be processed by the automation. The synchronization time measures the time taken for synchronising merge flows in parallel gateway. It is a relevant measure as it helps to identify bottlenecks in a system (time spent waiting). The resource usage indicates the occupancy of the resource. Readers interested in knowing more about this transformation and properties can refer to [10, 11].

4 CASE STUDY

This section illustrates an industrial automation case study and presents experiments related to it.

4.1 Packaging Automation

We consider the case study presented in [27], which implements an industrial automation system based on IEC 61499 standard. The production line is a packaging system that does the task of commissioning shipments stored in an automated storage facility consisting of the following control units - *Job scheduler:* It is the top level controller which manages the jobs. It owns the list of jobs to be commissioned and monitors the status of the workstations to dispatch the jobs to the available workstations. *Workstation:* It is a robotic arm that can pick up different items from the platform and package them. *Conveyor belt:* This is the transport system that moves the items from the storage facility to the robotic arm. *Dispensers:* These are the control units in the automated storage facility. There are multiple dispensers and each of the dispenser hands out a specific type of item on the conveyor belt.

The aforementioned units form a distributed application communicating through event and data flows. The job scheduler initiates the packaging process by dispatching a job to the workstation. Once the workstation receives the job, the dispenser drops the item on the conveyor belt. The belt transports the item to the assigned workstation. Then the workstation picks up the item and packs them for shipment.

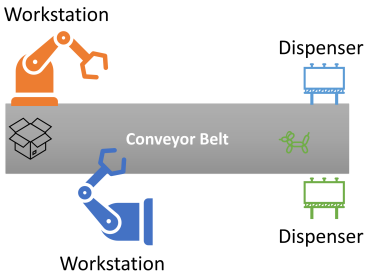

Figure 6: Control Units in a Packaging System [27]

Figure 6 shows the different control units of the automation. We chose this case study as the implementation was readily available and more importantly, it consists of dispenser, conveyor belt, and workstation resources that can be tweaked to obtain optimal production output. The number of workstations or the dispensers can be changed to measure the variation in production throughput. The original specification was modelled using the 4DIAC tool and it used the runtime environment provided by the tool for execution. The application has more than 30 instances of FBs.

4.2 BPMN Model

An excerpt of the BPMN model corresponding to the IEC 61499 specification of the case study is shown in Figure 7. For the sake of readability, only the pool corresponding to the DispenseRequestor FB is shown in Figure 7. In cases where ECC was not explicitly defined, we present a generic task with associated event and data flows instead of using an empty collaboration.

The internal execution logic of the dispenser specified in ECC is transformed into the activities within the DispenseRequestor pool using the mapping described in Section 3. In the IEC function block, there are two input events INIT and RSP and two output events INITO and IND. The events are translated as catch and throw message events in BPMN. The ECC also handles three data events, MB which relates to the input from the message bus, QO and Timeout are the output data events. The work status response input event RSP is associated with data input QI, this association is modelled using the parallel gateway between the message events in Figure 7. Once the response and data is received, there are two outgoing transitions in the ECC, which are modelled as a choice (exclusive gateway). The two possible paths lead to either state ActDeInit or AckJob. An output event INITO is associated with the state ActDeInit. The BPMN model represents the algorithms associated with the states as tasks. In order to distinguish states and algorithms, we have added an asterisk (*) on the tasks corresponding to the algorithms.

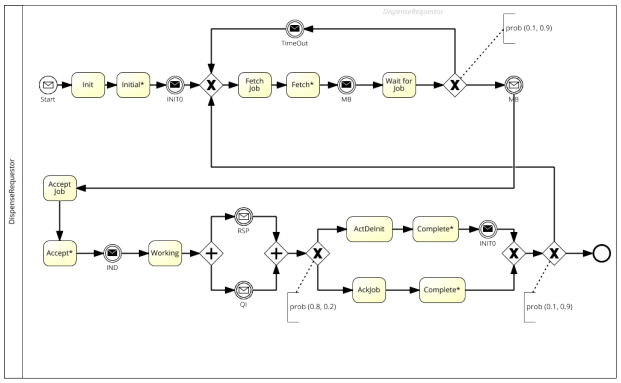

Figure 7: BPMN model of the Dispense Requestor ECC

4.3 Quantitative Analysis

Once the Maude specification was generated, we conducted experiments to compute the properties of average execution time, synchronization time, and resource usage. The experiments in Table 1 were carried out by specifying the workload in terms of number of instances of work. The arrival of instances (work) is specified through an exponential distribution denoting the interarrival time ($\lambda = 4$). In the first set of experiments, we use the following set of resources: 1 workstation, 2 dispensers, and 1 conveyor belt. Regarding the parameters, we specified task execution times in line with the real world execution times. For example, sending a job request takes a few milliseconds, but moving the item from one end of the conveyor belt to another end takes a few seconds. Specifically, we used the normal and uniform distributions for specifying the logical time units as they model natural phenomenon and variability of occurrence, respectively. Also, while assigning the probabilities associated with exclusive gateway (choice) transitions, we assigned higher probability to higher priority transitions and assigned lower

Instances	AET	Sync	Resource Usage (RU)		
			Dispenser	Conveyor	Workstation
100	53	32	42	46	86
200	96	52	44	61	93
500	234	138	45	70	97
1000	499	276	48	73	98

Table 1: RU: 2 Dispensers, 1 Workstation, and 1 Conveyor

probability to failure (exception) paths when priorities were not explicitly defined.

The first column in Table 1 denotes the number of **instances**, Average Execution Time is denoted by **AET**. The synchronization time of the gateways is shown in the **Sync** column. The resource usage percentage of dispenser, conveyor belt, and workstation are shown in the subsequent columns. The analysis was carried out on a host machine running WSL/Ubuntu 20.04 on a hardware of Core i7-7600U processor, 256GB M.2 PCIe SSD, and 32GB of RAM.

Naturally, the AET increases with increasing instances. However, the resource usage percentage varies. Workstation is under load as two dispensers are feeding the items to dispense for a single workstation. The results also indicate that the conveyor belt is not as busy as the workstation because it takes less time to transport items (transport multiple items across the belt, in practice). The increase in **Sync** time is the result of the dispenser waiting for the workstation to complete the task.

In order to optimize the resource usage, we varied the number of available resources, i.e, changing the number of available instances of dispenser, workstation, and conveyor belt. Figure 8 shows the usage percentage (UP) of resources with varying number of resource instances, with 200 token instances. The resource occupancy of dispenser is quite low when the number of dispensers is higher than the number of workstations. Increasing the number of workstations increases the overall resource usage across the other components as more workstations are available to process the packaging requests. The ideal configuration for this model is configuration 6: 2 dispensers, 1 conveyor belt, and 3 workstations, where the resource usages are 58%, 72%, and 63%, respectively indicating a balanced resource usage.

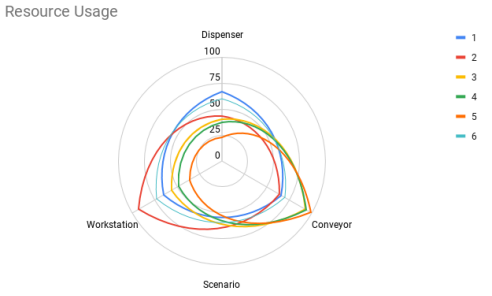

Figure 8: Usage % of Dispenser, Conveyor, and Workstation

The resource allocation and execution data can be used to optimize costs. The resources can be assigned costs. As an example, the cost of operating the workstation is $100 per hour, dispenser and conveyor belts are cheaper to operate costing $30 per hour. The goal is a multicriteria optimization, where one needs to minimize the costs and at the same time reduce the average execution time. Depending on the production requirements, business users can prioritize on either cost or throughput (average execution time). Consider a production requirement of packaging 500 items and if a configuration of one instance of each component is used, then the execution time goes up to 350 time units. Since we are using one instance of each component, the total operating cost is $933. However, if the business needs to ramp up the production to meet the rising demand for the product (e.g., vaccine supply), then the number of resource instances can be increased. For instance, a configuration of 4 workstations, 3 conveyor belts, and 3 dispensers would package 500 items in 142 time units, but at the same time the cost of operation increases to $1383, and this cost does not include the additional capital expenditure (CapEx) required for the resources. We explored different solutions by varying the number of resource instances and for the given production requirement, we found the configuration of 2 workstations, 1 conveyor belt, and 2 dispensers to be more optimal both in terms of cost and throughput as it would bring down the operating cost by 31% to $953 compared to the previous configuration and provides 43% increased throughput compared to the configuration with one instance of each resource.

5 RELATED WORK

This section covers three different areas of related work: modelling IEC 61499, BPMN for industrial automation in general, and quantitative analysis for industrial IoT.

Various modelling techniques have been used for analysis of IEC 61499 applications [12]. Earliest approaches tried to establish a relationship between IEC 61499 and existing modelling languages. The authors in [25] propose an extension of Unified Modeling Language (UML) tuned for real-time systems named UML-RT to model characteristics of timeliness and performance in system deployment. Similarly, in [8], the authors present UML-FB as a language for modelling IEC 61499 systems. UML-FB extends UML by stereotyping and it uses class diagrams, sequence diagrams, and state charts. The authors envisage this UML model to serve for testing and functional validation. Further in [3], the authors propose to implement IEC 61499 automation in low-cost embedded systems using UML specification. Compared to these works, our approach provides a business-oriented view of the system whereas UML-based models present an object oriented view of the system. Indeed, BPMN has its origins as an UML profile, so it is possible to use them interchangeably.

There are numerous works related to formal modelling of IEC 61499 applications [7, 13, 24, 30]. In [7], the authors propose an Abstract State Machine based formalism to model IEC 61499 systems. This model is used as the basis for symbolic model checking using SMV and SPIN model checkers. In [13], the authors model

the applications using Petri Nets. BlokIDE [30] is a development environment for electronic devices and it has an IEC 61499 compiler that generates C code to run on PLCs. Timed Automata is used to model IEC 61499 applications in [24]. Each FB is modelled as a Timed Automata and synchronization of FBs guided by the events and data invocation order in the ECCs. The formal models proposed in the aforementioned works are mostly used for checking functional or behavioural correctness (qualitative) of IEC 61499 systems. The Maude specification in this work is a formal model used for resource and cost analysis (quantitative). Since there are already works related to formal modelling of BPMN specification [6, 15], by transforming the IEC 61499 specification to BPMN, we can take advantage of the existing tools for qualitative analysis. Arrowhead is an IIoT automation framework and the authors in [14] integrate BPMN and colored Petri Nets (CPN) in Arrowhead to manage the industrial workflows. In [9], authors model rule-based IoT applications in Maude rewriting logic to verify application reconfiguration. Industry 4.0 Process Modelling Language (I4PML) [20] is an UML profile with BPMN elements to specify Industry 4.0 applications. It extends BPMN with IoT-aware process elements such as Sensing Task, Actuation Task, Mobility Aspect (Pool) etc., to model applications. In comparison to these works, our model provides a BPMN representation of the system with standard BPMN elements for cost-resource analysis, it does not integrate multiple platforms and standards which would require additional model-specific knowledge to understand the process.

Regarding cost and resource optimization for industrial production, in [21], the authors propose dispersed automation where the workloads are dynamically shared across underutilized devices. They use a DSL to specify application logic and the available resources. This approach focuses on computational resources and our work considers resource requirements at the business level. Moreover, optimization by dispersed automation will be limited by the security and physical constraints in an industrial setup.

6 CONCLUSION

We have proposed a business process-based modelling approach for industrial automation. The BPMN model is derived from the IEC 61499 specification of the automation. The business process model provides a complementary view of the IEC 61499 specification, which allows business analysts to integrate cost and resource requirements during the system design. This BPMN model is transformed into a formal specification in Maude for average execution time, resource usage, and cost analysis, thereby enabling business stakeholders to make decisions related to the productive deployment of the automation as illustrated by the case study. As far as future work is concerned, we plan to extend our work to verify functional correctness of IEC 61499 application designs.

REFERENCES

[1] Thomas Allweyer. 2016. *BPMN 2.0: Introduction to the standard for business process modeling.* BoD–Books on Demand.

[2] Zeeshan E Bhatti, Partha S Roop, and Roopak Sinha. 2016. Unified functional safety assessment of industrial automation systems. *IEEE Transactions on Industrial Informatics* 13, 1 (2016), 17–26.

[3] Esteban X Castellanos, Carlos A Garcia, Cesar Rosero, Carlos Sanchez, and Marcelo V Garcia. 2017. Enabling an automation architecture of CPPs based on UML combined with IEC-61499. In *2017 17th International Conference on Control, Automation and Systems (ICCAS).* IEEE, 471–476.

[4] Manuel Clavel, Francisco Durán, Steven Eker, Patrick Lincoln, Narciso Martí-Oliet, José Meseguer, and Carolyn Talcott. 2007. *All About Maude-A High-Performance Logical Framework: How to Specify, Program, and Verify Systems in Rewriting Logic.* Vol. 4350. Springer.

[5] Jerker Delsing. 2017. *IoT automation: Arrowhead framework.* CRC Press.

[6] Remco M Dijkman, Marlon Dumas, and Chun Ouyang. 2008. Semantics and analysis of business process models in BPMN. *Information and Software technology* 50, 12 (2008), 1281–1294.

[7] Dmitrii Drozdov, Victor Dubinin, Sandeep Patil, and Valeriy Vyatkin. 2021. A Formal Model of IEC 61499-Based Industrial Automation Architecture Supporting Time-Aware Computations. *IEEE Open Journal of the Industrial Electronics Society* 2 (2021), 169–183.

[8] Victor Dubinin, Valeriy Vyatkin, and Thomas Pfeiffer. 2005. Engineering of validatable automation systems based on an extension of UML combined with function blocks of IEC 61499. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation.* IEEE, 3996–4001.

[9] Francisco Durán, Ajay Krishna, Michel Le Pallec, Radu Mateescu, and Gwen Salaün. 2021. Seamless Reconfiguration of Rule-Based IoT Applications. In *SEAMS 2021-16th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, Vol. 1. 142–148.

[10] Francisco Durán, Camilo Rocha, and Gwen Salaün. 2018. Stochastic analysis of BPMN with time in rewriting logic. *Science of Computer Programming* 168 (2018), 1–17.

[11] Francisco Durán, Camilo Rocha, and Gwen Salaün. 2019. A rewriting logic approach to resource allocation analysis in business process models. *Science of Computer Programming* 183 (2019), 102303.

[12] Georg Frey and Tanvir Hussain. 2006. Modeling techniques for distributed control systems based on the IEC 61499 standard-current approaches and open problems. In *2006 8th International Workshop on Discrete Event Systems.* IEEE, 176–181.

[13] Nils Hagge and Bernardo Wagner. 2005. A new function block modeling language based on petri nets for automatic code generation. *IEEE Transactions on Industrial Informatics* 1, 4 (2005), 226–237.

[14] Dániel Kozma, Pál Varga, and Felix Larrinaga. 2019. Data-driven Workflow Management by utilising BPMN and CPN in IIoT Systems with the Arrowhead Framework. In *2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA).* IEEE, 385–392.

[15] Ajay Krishna, Pascal Poizat, and Gwen Salaün. 2019. Checking business process evolution. *Science of Computer Programming* 170 (2019), 1–26.

[16] Stefan-Helmut Leitner and Wolfgang Mahnke. 2006. OPC UA–service-oriented architecture for industrial applications. *ABB Corporate Research Center* 48 (2006), 61–66.

[17] Robert Lewis. 2001. *Modelling control systems using IEC 61499: Applying function blocks to distributed systems.* Number 59. Iet.

[18] Narciso Martı-Oliet and José Meseguer. 2002. Rewriting logic: roadmap and bibliography. *Theoretical Computer Science* 285, 2 (2002), 121–154.

[19] Cheng Pang, Sandeep Patil, Chen-Wei Yang, Valeriy Vyatkin, and Anatoly Shalyto. 2014. A portability study of IEC 61499: Semantics and tools. In *2014 12th IEEE International Conference on Industrial Informatics (INDIN).* IEEE, 440–445.

[20] Roland Petrasch and Roman Hentschke. 2016. Process modeling for Industry 4.0 applications: Towards an Industry 4.0 process modeling language and method. In *2016 13th International Joint Conference on Computer Science and Software Engineering (JCSSE).* IEEE, 1–5.

[21] Gustavo Quirós, Diansong Cao, and Arquimedes Canedo. 2020. Dispersed Automation for Industrial Internet of Things. *IEEE Transactions on Automation Science and Engineering* 17, 3 (2020), 1176–1181.

[22] Michael Rüßmann, Markus Lorenz, Philipp Gerbert, Manuela Waldner, Pascal Engel, Michael Harnisch, and Jan Justus. 2015. *Industry 4.0: The Future of Productivity and Growth in Manufacturing Industries.* Boston Consulting Group.

[23] Gideon Rose. 2016. *The Fourth Industrial Revolution: A Davos Reader.* Foreign Affairs.

[24] Marius Stanica and Hervé Guéguen. 2004. Using timed automata for the verification of IEC 61499 applications. *IFAC Proceedings Volumes* 37, 18 (2004), 375–380.

[25] Kleanthis C Thramboulidis. 2004. Using UML in control and automation: a model driven approach. In *2nd IEEE International Conference on Industrial Informatics, 2004. INDIN'04. 2004.* IEEE, 587–593.

[26] Valeriy Vyatkin. 2009. The IEC 61499 standard and its semantics. *IEEE Industrial Electronics Magazine* 3, 4 (2009), 40–48.

[27] Jörg Walter, Kim Grüttner, and Wolfgang Nebel. 2018. Using IEC 61499 and OPC-UA to implement a self-organising plug and produce system. In *The 5th International Workshop on Model-driven Robot Software Engineering (MORSE 2018).*

[28] Andreas Wortmann, Olivier Barais, Benoit Combemale, and Manuel Wimmer. 2020. Modeling languages in Industry 4.0: an extended systematic mapping study. *Software and Systems Modeling* 19, 1 (2020), 67–94.

[29] Li Yoong. 2010. *Modelling and Synthesis of Safety-critical Software with IEC 61499.* Ph. D. Dissertation. ResearchSpace@ Auckland.

[30] Li Hsien Yoong, Partha S Roop, Zeeshan E Bhatti, and Matthew MY Kuo. 2015. *Model-Driven Design Using IEC 61499.* Springer.