Journal of Manufacturing Systems 71 (2023) 172–187

Contents lists available at ScienceDirect

Journal of Manufacturing Systems

journal homepage: www.elsevier.com/locate/jmansys

PLC orchestration automation to enhance human–machine integration in adaptive manufacturing systems

Fan Mo [a,1,*], Miriam Ugarte Querejeta [b,1], Joseph Hellewell [a], Hamood Ur Rehman [a,c], Miren Illarramendi Rezabal [b], Jack C. Chaplin [a], David Sanderson [a], Svetan Ratchev [a]

[a] Institute for Advanced Manufacturing, University of Nottingham, Nottingham, Nottinghamshire, NG8 1BB, United Kingdom
[b] Mondragon Unibertsitatea, Arrasate-Mondragon, 20500, Spain
[c] TQC Automation Ltd., Nottingham, Nottinghamshire, NG3 2NJ, United Kingdom

ARTICLE INFO

Keywords:
PLC
Automatic program generation
Structural testing
Adaptive manufacturing systems
Industry 4.0
Knowledge graph
Graph neural network

ABSTRACT

Current approaches to manufacturing must evolve to respond to increasing demands for short product life cycles and customised products. Adaptive manufacturing systems integrate advanced technologies, automation, and data-driven methodologies to develop adaptable, efficient, and responsive production processes. Central to this concept is the emphasis on human involvement and fostering synergy between human operators and the manufacturing system. Significant changes to the system's controller are required to achieve adaptivity, with programmable logic controllers (PLCs) being a common controller type. After the necessary changes to the configuration of the manufacturing system, the PLC should be reconfigured to orchestrate the new required behaviour. Automated reconfiguration is vital to rapidly responding to change, but some changes cannot be entirely achieved without human input in collaboration with automated methods. Conventional practices in PLC programming include manual, repetitive coding practices subject to errors. As a result, to ensure operational safety, the changes must be tested before being deployed to operations, ensuring it is error-free. This paper presents a methodology to automatically reconfigure the simulation environment and controller in response to a new product request. We automate the PLC code generation and testing practices to support and free up the operators when performing repetitive manufacturing reconfiguration tasks. The methodology is based on human learning, software automation, customised program development, knowledge graphs, and Graph Neural Networks (GNNs). The presented solution is a generic, vendor-agnostic, and interoperable solution that facilitates information exchange among multiple heterogeneous environments. Lastly, we have validated the methodology as a proof of concept at an adaptive assembly cell at the University of Nottingham in the United Kingdom.

1. Introduction

Modern manufacturing companies are facing increasing variability in market demands and uncertainty in supply chains. A competitive global market is forcing manufacturing companies to respond quickly to customer requirements and to produce high-quality products with shorter product life cycles and higher levels of customisation [1–3]. These key factors require a high degree of flexibility in manufacturing processes.

Humans are flexible by nature, and they represent an opportunity to enhance the flexibility of manufacturing systems by being creative and being able to operate with different tools and equipment. A worker can handle a non-standard situation where an automated resource would fail. As a result, the integration of human decision-making and expertise increases the flexibility and adaptability of manufacturing systems. Humans can coexist with machines, but they should use their cognitive capabilities and flexibility to perform complex activities, whereas routine manual tasks and number crunching should be automated by machines [4].

Adaptive Manufacturing Systems (AMS) are manufacturing systems equipped with flexible and reconfigurable manufacturing capabilities to cope with dynamically changing demands. AMS refers to production systems that are defined as "adaptive", and emphasises the importance of human processes, human decisions, and specific user requirements [5–8]. These systems are complex systems capable of continuously monitoring the values and trends of the external environment or internal variables, maximising their objectives based on particular response

* Corresponding author.
E-mail address: fan.mo@nottingham.ac.uk (F. Mo).
[1] These authors contributed equally to this work.

https://doi.org/10.1016/j.jmsy.2023.07.015
Received 25 April 2023; Received in revised form 11 July 2023; Accepted 25 July 2023
Available online 19 September 2023

rules, and altering their behaviour to achieve their goals, while RMSs are more concerned with the physical layout and production planning changes [5].

Controller devices are utilised to achieve the adaptivity and reconfigurability of AMSs [9,10]. Due to their reliability and durability, Programmable Logic Controllers (PLCs) are an industry standard for low-level control and automation. Engineers can program Human–Machine Interfaces (HMIs) to enable human workers to interact with, control, and monitor the activities of a PLC [11]. HMIs and PLCs enable humans and machines to work together to monitor and control manufacturing equipment. Building an agile PLC orchestration environment that suits both machine and human input is crucial to dynamically adapt existing equipment behaviour to new environmental changes or customer needs. This requires seamlessly reconfiguring, testing, and validating the PLC code whenever a new change is inserted in the layout, as the increasing frequency of product or process change increases the likelihood of an error being made.

This paper presents a methodology to automate manufacturing system reconfiguration in the context of AMS, focusing on the controller as the orchestrator of the manufacturing processes. The PLC must be reprogrammed to address new process requests, which currently require manual intervention. This leads to a significant investment of time and effort, but also an increased likelihood of errors, all of which increase in adaptive environments. We have therefore developed a dynamic PLC code generation and testing approach to automate the orchestration of the manufacturing system when handling new requests. The proposed methodology aims at automating error-prone manual programming tasks to reduce errors and shorten commissioning time while letting humans focus on the most complex and flexible tasks. Humans will therefore coexist with automated machines in a continuously adaptive environment. This way, humans will have a distinctive role in which they might be able to focus on novel PLC challenges rather than repetitive programming and monotonous practices.

The remainder of this paper is organised as follows. Section 2 reviews PLC code formats and standards, PLC modelling, PLC code testing, dynamic software reconfiguration, Graph Neural Network (GNN), human–machine role and coexistence in manufacturing and the existing gaps. Section 3 describes the methodology of automating PLC code reconfiguration, generation, and testing in AMSs. Section 4 provides a case study in the aerospace industry which utilises our proposed methodology. Section 5 describes our conclusions and outlines future work.

2. Literature review

The work presented in this paper focuses on enabling AMSs to adapt to changing customer needs seamlessly. This requires both hardware and software adaptation, and typical applications require large volumes of repetitive code and configuration changes that slow down the speed of adaptation, and introduce opportunities for error [12,13]. To enable the automatic generation of PLC code to accelerate and simplify the adoption of adaptive manufacturing systems, as well as to allow human workers to focus on novel and interesting challenges, several new or emerging technologies are used, which will be detailed in the next sections.

2.1. Control architecture

In traditional manufacturing systems, the control architecture is centralised, which means it lacks the capability to respond to failures or disruptions in the system effectively. Consequently, decentralised control approaches have been implemented to address failures and adapt to the dynamic environment, such as multi-agent systems [14] or holonic manufacturing systems [15]. The concept of decentralisation pertains to the degree of autonomy that individual entities or modules possess to control themselves or make independent decisions [16].

The extent of decentralisation is determined based on whether there are any specific requirements or tasks that need centralised control. According to Trentesaux et al. [16], the control architecture can be categorised as follows:

- Class 0: centralised control systems.
- Class I: fully hierarchical control systems.
- Class II: semi-heterarchical control systems, integrating both hierarchical and heterarchical relationships.
- Class III: fully heterarchical control systems.

Regarding the control architecture of PLCs, the IEC 61499 standard [17] was introduced to facilitate the design of distributed architectures in industrial automation systems. However, the IEC 61131-3 standard [18] remains predominant in the field, as there are essential processes that still require hierarchical centralisation.

2.2. PLC code formats and standards

PLCs are a standard industrial control method, especially in Industry 3.0-era systems. However, given the multi-vendor nature of the sector, there exists a wide range of proprietary programming languages and data formats. The internal details and operation of PLCs are often proprietary, and interoperability among different vendor solutions is typically a major challenge.

A common framework for programming PLCs is defined by the IEC 61131-3 standard, which specifies the number of programming formats, including Function Block Diagrams (FBDs). Structured Text (ST) is an advanced PLC programming language established in the IEC 61131-3 standard, also known as Structured Control Language (SCL) in Siemens software [19]. The use of these standard in the industrial control field provides a common programming interface, which allows people with different backgrounds and expertise in different vendor solutions to create different parts of an automation project during different phases of the development life cycle [20].

PLC code logic is typically programmed manually. Several researchers have therefore proposed frameworks to generate FBD code from state chart diagrams. For example, Vogel-Heuser et al. [21] described the automatic generation of PLC code from UML diagrams. The generated code was programmed in ST and SFC language, making reconfiguration more challenging as adding or removing functions is more difficult. Similarly, Hametner et al. [22] described a model-to-model transformation process based on UML state charts. However, this adds complexity as models must be first mapped, resulting in a loss of information.

The majority of PLC suppliers have varying degrees of compliance with IEC 61131 standard. However – even with IEC 61131-3 – users cannot always share their program, libraries, and projects between PLC brands or development environments due to variability in file formats, development environments, and additional features not represented in the standards [23].

The international organisation PLCopen has proposed an open XML interface based on the IEC 61131-3 standard to facilitate tool interoperability and code reuse [24]. PLCopen XML is a markup language that specifies all the textual and visual notations used in the PLC languages as specified by IEC 61131-3 [25], including FBDs. PLCopen XML defines a common representation of the information exchanged by tools during the different phases as well as supporting the transformations between such representation and each tool [20].

2.3. Semantics for PLC modelling

The increasing dynamism of modern manufacturing challenges requires PLCs to be reprogrammed and reconfigured by human workers at an increasingly regular rate. This is complex, time-consuming, and repetitive. To remove this repetition, code reuse can be implemented,

which takes previously developed code and adapts it where possible for new situations. Ontology models can be used to describe the characteristics and concepts of PLCs [26]. An ontology model is a structured representation of knowledge within a specific domain designed to facilitate information sharing and enable reasoning about that domain. It is often used in artificial intelligence, the semantic web, and knowledge management systems to provide a proper and shared understanding of a particular subject area [27–29]. Skill and capability models are two important areas for understanding the characteristics and concepts of PLCs. In manufacturing equipment, the skill model focuses on machines or systems' specific abilities and functions. It represents the tasks or operations that equipment can perform, such as cutting, welding, or assembly. The skill model helps in resource allocation, process optimisation, and system integration, ensuring efficient and productive manufacturing. This capability model concerns the overall capacities, performance, and functionalities of machines, equipment, or systems in a manufacturing setting. It provides a holistic view of each machine or equipment's attributes, such as processing types, speed, accuracy, and other factors contributing to the manufacturing process. The capability model aids in planning, resource allocation, and optimising production processes based on available capabilities.

The PLC domain knowledge is built by considering PLC domain characteristics and by designing a layered ontology [30]. Domain knowledge is the technical understanding and expertise in a particular subject area, encompassing its concepts, terminology, principles, theories, and practices. Domain characteristics are a subject area's defining features and attributes, shaping its scope and nature and influencing the methods used within that field. Layered ontology organises knowledge in a hierarchical structure with multiple levels or layers. Each layer represents a different level of abstraction, where higher layers contain more general concepts, and lower layers include more specific details. This organisation helps in managing complexity and enables efficient knowledge representation and reasoning.

The development of such models, however, requires a high degree of expertise in semantic technologies. In conventional automation engineering workflows, such models would have been created manually, which is an error-prone and time-consuming task.

Given the need to automate the process of model creation in continuously adaptive systems, Industry 4.0 promotes skill and capabilities-based modelling to share information in dynamic engineering [31]. In this way, already-defined skills can be reused when needed. New production tasks can be dynamically assigned by invoking a given product's required sequence of skills. For example, Köcher et al. [32] described an approach to automatically create skill models from IEC 61131-3 code; however, it is limited to IEC 61131-3.

2.4. PLC code testing

The manual nature of most PLC programming leaves PLC code open to error, especially if the code is being changed rapidly, or if the changes are repetitive. In terms of testing FBD programs, the industry mainly relies on manually conducting functional tests from requirements [33], known as functional testing. This is a tedious task, as it takes considerable time and effort. Moreover, functional testing is not effective enough at detecting implementation errors, as it does not take into account the internal design or the structure of the FBDs. Automated structural testing of PLC programs is therefore a question of great interest in industrial automation. In this context, Reinhard et al. [34] described a dynamic test case generation approach based on UML state charts on IEC 61499 standard. The automation model was generated automatically through the model-to-model transformation process. Similarly, Wu et al. [25] and Enoiu et al. [35] presented an approach to automatically generate test data for FBDs from intermediate data models such as UPPAAL (an integrated model checking tool for timed automata) following traditional software engineering techniques. This requires converting FBDs to intermediate data models first, which does not accurately represent some key aspects of PLC code, such as cyclic execution.

Existing research recognises the critical role played by automated test generation for FBDs. Jee et al. introduced a new test generation approach directly applied to FBD diagrams without the need for any further conversion; they proposed Basic Coverage (BC), Input Condition Coverage (ICC), and Complex Condition Coverage (CCC) metrics for structural coverage testing [36] based on FBD data path conditions. Recently, KAIST university did further research on automated test data generation for cyclic FBDs [37]. However, the industry has not yet applied and validated these approaches.

Mutation testing is also a widely used technique for fault detection in software engineering [38,39]. To this end, Enoiu et al. [40] and Liu et al. [41] successively proposed mutation-based test generation approaches for FBD programs. According to Liu et al. Mutation-based test generation showed promising results in terms of effectiveness. However, there are still gaps in the current research that need to be addressed in future studies, such as reducing time and cost associated with these techniques, as well as knowledge transfer to real industrial applications.

Fernandez et al. [42] have been developing the automation and testing of PLC programs for their unified industrial control system framework. They analysed formal algorithmic verification and automatic testing for PLC validation and testing, where data is exchanged via SCADA. However, the interaction with SCADA interfaces depends on the communication protocol. Therefore, a more recent study proposed an alternative approach that relies on OPC UA [43]. The presented method focused on continuous integration, considered one of the key elements in supporting agile software development and testing environment [44].

On this basis, Talkhestani et al. [45,46] described the use of a digital twin as an agile technology to enable automatic control code generation for the newly added machines based on the Anchor Point Method. This method mainly focuses on the digital twin's automated change detection and model adjustment to enable an automated PLC code generation via Siemens TIA Portal. Likewise, Koziorek et al. [47] suggested TIA Portal as a tool for automating the generation of the PLC code and testing activities. However, these solutions are Siemens-specific, and there is no vendor-agnostic solution yet in the market that offers an open and seamless PLC code generation and testing approach.

2.5. Dynamic software reconfiguration

Dynamic software reconfiguration is an area of research focusing on providing faster software solutions to changing problems by adding more flexibility to any given software solution, enabling the same solution to be rapidly repurposed for new situations [48]. Distinct from writing new code from scratch, software reconfiguration is a strategy that allows code to be modifiable, extensible (capable of responding to new characteristics and specifications), reusable (applicable to multiple programs), and, most importantly, reconfigurable (capable of retaining driver configurations and supporting internal and external interactions between modules) with no additional modifications. This concept is of great interest to the manufacturing control research community. Several methods exist [49], including Petri Nets (PN) model [50], PLC and PC based control [50,51], Domain Specific Modelling (DMS) [52], and Self-Adaptive Smart Assembly System (SASAS) [53].

Though there is research on achieving software control reconfiguration, only a few consider developing an automated model to generate the program and reconfigure it automatically [32,54,55]. With the development of artificial intelligence technology, OpenAI company has released ChatGPT. ChatGPT is an advanced natural language processing model that uses deep learning algorithms to generate human-like responses to user input. It can also automatically generate code snippets based on a developer's problem statement or code requirement. This technology has the potential to revolutionise software development by

reducing the time and effort required to write code, and improving speed and accuracy [56]. With this kind of technology, the performance of dynamic software reconfiguration has the potential to be enhanced.

Furthermore, the data model in software reconfiguration should be standardised to make the approach more extensible and easily handle compatibility issues.

2.6. Graph Neural Network (GNN) and GraphSAGE model

In an adaptive system, asset connections and relationships change to cope with the ever-changing environment and requirements. Therefore, semantic relations among the assets must be represented to allow changes in the production system to be dynamically orchestrated. Semantic relations refer to the relationships between words and concepts based on their meaning. In natural language processing, understanding semantic relations is important because it allows systems to interpret the meaning of words and phrases in context, rather than simply identifying them as discrete units. In addressing changes in production systems, semantic relations are crucial because they enable the system to recognise and respond to changes in the language used to describe production processes. By analysing the semantic relationships between different concepts and phrases, the system can identify new or modified processes and adapt accordingly. This can help ensure that production systems remain up-to-date and effective, even in the face of changing technologies, processes, and business needs. Graphs are a method of representing data in a way that captures the structural relationships between data. Storing data as graphs may encode structural information to describe the relationships between entities, yielding more insightful insights into the data. Graphs are ubiquitous in many application disciplines, ranging from social analysis [57], fraud detection [58,59], traffic prediction [60], and computer vision [61].

However, it is frequently difficult to utilise graphs as a data storage method as many different forms of data, such as images and text data, are not initially arranged as graphs. Even for graph-structured data, the underlying connectivity patterns are often complex and diverse, making interpretation difficult. A potential solution is to learn the representation of graphs in a low-dimensional Euclidean space, such that the graph properties can be preserved [62].

Graph Neural Networks (GNN) are neural models that use message exchange between graph nodes to reflect the reliance on graphs [63]. In recent years, GNN variations such as graph convolutional network (GCN), graph attention network (GAT), and graph recurrent network (GRN) have shown ground-breaking performance on numerous deep learning tasks. In numerous applications, deep learning models have proven to be effective. For large-scale graphs, however, the training procedure of GCN might be memory-intensive. In addition, the transduction of GCN interferes with generalisation, making it more challenging to learn representations of unseen nodes in the same network and nodes in an entirely different graph [64]. To effectively address changes in production systems using PLCs, it is necessary to utilise inductive learning methods. This is because production system changes can often be unpredictable, making it difficult to anticipate and respond to them using traditional rule-based approaches. The use of inductive learning in PLCs is becoming increasingly important as production systems become more complex and dynamic. By leveraging the power of machine learning, PLCs can adapt to changes in the production environment and optimise performance without the need for extensive manual programming or maintenance. This can help companies remain competitive by improving efficiency, reducing downtime, and ensuring consistent product and service quality.

GraphSAGE model addressed this issue. It is a framework for inductively learning huge graph representations. Graphs are utilised to construct low-dimensional vector representations for nodes, which is particularly effective for graphs with abundant node attribute data. Unlike embedding approaches based on matrix factorisation, the node features (e.g., text attributes, node profile information, node degrees) based on GraphSAGE are leveraged to learn an embedding function that generalises to unseen nodes [65]. The GraphSAGE model has the advantage of inductively learning huge graph representations, which can help the system deal with new unknown requests.

2.7. Human–machine coexistence in manufacturing

Some autonomous machines are designed to adapt to new and unforeseen scenarios. Their abilities, however, do not yet approach the adaptive skills of a human being. Humans are and will remain an essential part of manufacturing due to their cognition and problem-solving skills [66]. Human–machine coexistence is, therefore, a key aspect of dynamically adapting the system to new situations, as indicated by [67]. Despite being highly adaptable and flexible, human labour can be comparatively expensive and requires significant time and effort. Productivity can be increased with automation, although there must be a balance between fully automated production systems and highly adaptable human work. In this matter, the A4BLUE project is developing assembly systems that will integrate digital and automated mechanisms in conjunction with human workers to respond to the rapidly evolving requirements of manufacturing processes [8].

The majority of industrial research and development under the Industry 4.0 banner is on enhancing the performance of manufacturing systems without a clear emphasis on human aspects and their relationship to production systems [5]. ‘Industry 5.0’ responds to this with a value-driven approach towards practical implementations of available enabling technologies in the industry. The key ideas of Industry 5.0 are underpinned by three interwoven pillars: human-centricity, sustainability, and resilience [68].

Humans and machines must work collaboratively to achieve the same goals rather than solely as individuals [69,70]. Humans should therefore play an active and distinctive role rather than performing repetitive tasks that could be automated and replaced by machines. According to Autor [71], automation brings new job opportunities, as it increases outputs in a way that leads to higher demand for labour. Automation should therefore complement human work. As such, Cimini et al. [4] focused their study on the role of humans in smart factories and highlighted the importance of properly integrating humans in production systems. This however requires significant changes to the human role in today’s industry.

2.8. Existing gaps

Most manufacturing approaches (e.g. dedicated manufacturing systems such as production lines and mass manufacture) are not designed for regular change. However, other manufacturing system paradigms such as reconfigurable or flexible manufacturing systems are not able to adapt autonomously to changing manufacturing environments [72]. Control architectures should be adaptive enough to respond to manufacturing requirements as a company reacts to market demands. New approaches such as autonomous manufacturing systems achieve these needs but often miss the human role and focus on the adaptability of machines alone [72], which require a lot of engineering time as reconfiguration and testing are required every time there is a change in the controller. To achieve this, the adoption of software engineering in automation is key [73].

PLC code, however, is mostly dedicated to specific machines, and requires significant changes in frequently changing environments, not being able to reuse FBD blocks to build up new ones. Moreover, With the increasing complexity of automation systems, engineers face more challenging scenarios in integrating products from multiple vendors into the same system [74]. This is a barrier when exchanging PLC programs and libraries between multiple heterogeneous environments. In this regard, according to Vogel-Heuser et al. [73], research on software automation should focus on building reusable and interoperable solutions to make it extensible.

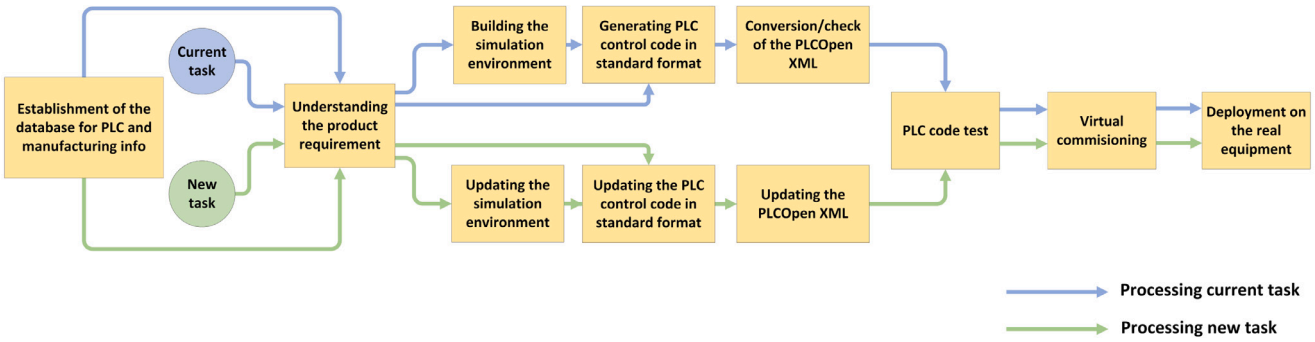

Fig. 1. Overview of the PLC reconfiguration framework in an adaptive manufacturing system.

Given the existing challenges described above, a new approach is required to succeed in automating PLC orchestration for Adaptive Manufacturing Systems. The approach should focus on AMS, in which humans and machines coexist, and should meet the following key objectives:

- O1: Orchestrating new manufacturing processes in frequently changing manufacturing environments autonomously.
- O2: Avoiding repetitive manual work, reducing significant programming errors and time.
- O3: Involving humans to address highly flexible tasks in coexistence with automated machines.
- O4: Reusing modules to quickly build new FBDs or reconfigure existing ones, making them extensible and scalable.
- O5: Building a generic, vendor-agnostic, and interoperable solution to exchange information among multiple heterogeneous environments.

3. Proposed methodology

3.1. Overall structure

With the increased demand for adaptive production, there is a need to update the PLC code whenever a new request is received. Motivated by the literature review and the current limitations of the techniques, our paper proposes a novel framework of human–machine collaborative PLC reconfiguration in an adaptive manufacturing system that automates the PLC code generation and testing. This was achieved with the use of human learning, software automation, customised program development, ontology reasoning, knowledge graphs, and GNN.

We have implemented the simulation environment to generate and test the code in our methodology. A simulation environment is crucial for updating PLC code when a new product request occurs. Because it allows engineers to test new code changes in a virtual environment, reducing the risk of errors and downtime in the actual production system. This saves time and ensures that the production system meets the required specifications for the new product. This section describes in detail the process of updating the simulation environment either manually or automatically and the respective PLC code when a new product request comes. Fig. 1 depicts the main steps involved when processing the existing product and a new one. Some of the steps are fully automated, whereas others present the coexistence of humans and machines in the process.

New product requirements are addressed by creating a new Bill of Processes (BoP). This triggers the adaptation of the manufacturing system. The BoP includes all the necessary processes or tasks involved with the new customer request. These tasks could be fully automated but could also require human labour to perform complex tasks that require more flexibility. In this way, the framework should first understand the requirements, and match the required capabilities from the existing Bill of Resources (BoR) to accommodate the new BoP. Then our framework will generate PLC code and build (or update) a simulation environment for testing the code, as detailed in Fig. 2.

3.1.1. Establishment of the database for PLC code

Key to this approach is the reuse of existing PLC code to minimise effort on behalf of the human workers. A database for storing the PLC information and the manufacturing information is therefore generated as the first step. In our approach, we utilise a graph database, as they are effective for discovering and giving meaning to complex interdependencies and relationships between entities [75]. Using the graph database, the following manufacturing information is represented:

(1) **Task information**
The task information describes the information about how customer orders and requests that are initiated internally within the factory are processed. It represents the tasks or activities needed to complete a customer order or fulfil a request. It provides a high-level overview of the steps involved in the production process. It is critical in our methodology as it describes the new customer request information properly.

(2) **Product information**
In reconfigurable manufacturing systems, product dimensions and geometric features are key to determining appropriate manufacturing processes. Process selection is typically based on factors such as size, shape, and material properties. For instance, sheet metal cutting processes such as laser, plasma, and water jet cutting may be suitable for products with large dimensions. Conversely, products with intricate geometries may require CNC milling, EDM wire cutting, or CNC turning. Material properties, such as hardness, may also dictate the manufacturing process, with CNC grinding or honing processes used for hard materials. Similarly, processes like polishing or sandblasting may be employed for products requiring a high surface finish.

(3) **Bill of process (BoP)**
A BoP in manufacturing outlines the processes needed to make a product. It is important to understand customer requests and optimise efficiency. When a customer request for a change is received, the BOP is used to assess the feasibility and make necessary modifications to the production system.

(4) **Bill of resource (BoR)**
A BoR in manufacturing lists the resources needed to produce a product. It is important to understand product requirements and adapt to changes in customer requests by analysing potential resource constraints or bottlenecks and optimising resource allocation. When a customer request for a change is received, the BoR is used to assess the availability of the required resources and identify any necessary modifications to the production system. In summary, the BoR is a critical document in manufacturing that outlines the necessary resources to produce a product. It helps manufacturers understand product requirements and adapt to changes in customer requests, allowing them to optimise resource allocation and minimise disruptions to the production schedule.

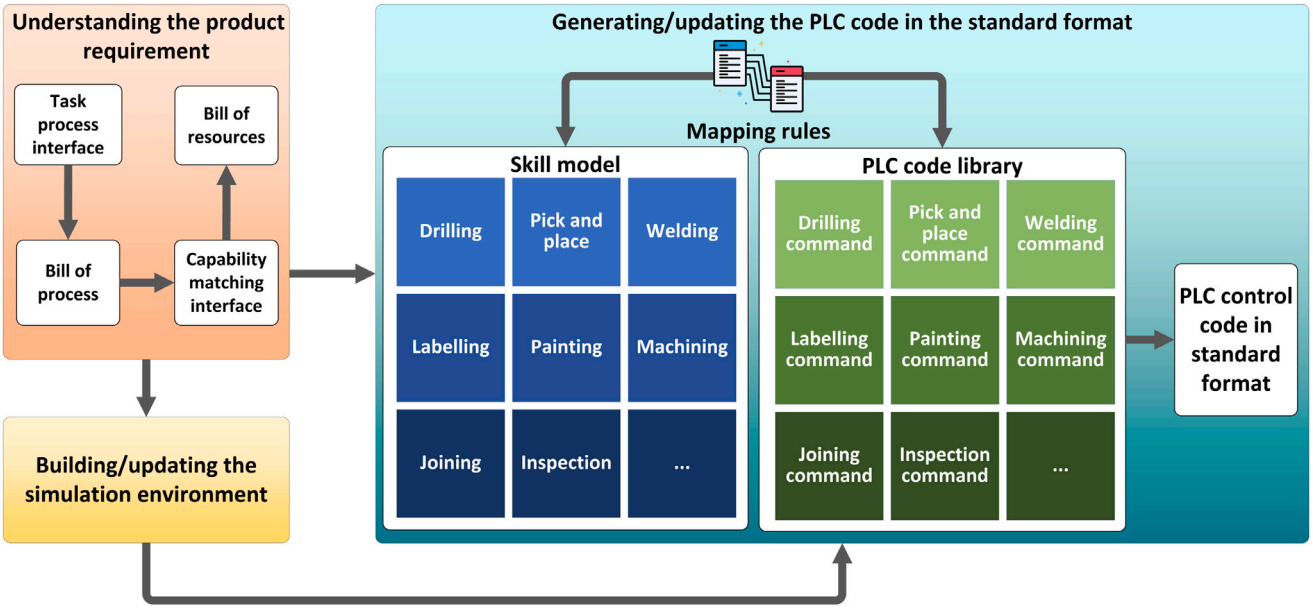

Fig. 2. Process of generating the PLC code.

(5) **Capability model of the resources**
A Capability Model represents a factory's abilities and limitations regarding processes, technologies, resources, skills, and regulations in manufacturing. It provides a comprehensive view of the factory's capabilities, aiding informed product and process feasibility decisions. The model also identifies improvement opportunities, such as acquiring new resources or technologies or skill development to enhance the factory's competitiveness.

(6) **Capacity model of the resources**
A capacity model in manufacturing is a mathematical representation of a factory or production line's production capacity. It considers resources, processes, and constraints to determine maximum output under specific conditions. The model adapts to changes in production conditions, such as new products, demand changes, or technology updates, to achieve reconfiguration. This structured representation of manufacturing KPIs monitors performance on the shop floor, focusing on reconfiguration. It helps identify areas for improvement in capacity and informs decision-making for achieving optimal performance.

(7) **Plc control code library**
A code library for PLC control code contains pre-written software modules that control various aspects of a manufacturing process. It saves time, reduces errors, and improves efficiency by allowing programmers to incorporate pre-tested code blocks. The library includes primary and complex functions and can be customised for specific industries and applications.

An ontology and industry standards are utilised to represent the information in a formal and structured way. In our proposed framework, existing ontologies describing PLC code are used to build the layered ontology. In particular, Common Core Ontology (CCO) [26] as there is a need to represent common concepts in PLC projects. In addition, the Application Specific Ontology (ASO) [26] is also needed to represent the heterogeneous PLC development environment (O5), code comments, test logs, and bug reports. With these two ontology models, the PLC domain knowledge base can be built with high accuracy and completeness by considering PLC domain characteristics, designing a layered ontology, and implementing the matching process on the schema level instead of the instance level. The schema level represents the formal definition of the PLC domain's classes, properties, and relationships. It consists of the general, abstract concepts and the structure of the ontology, providing the basis for the specific instances that will populate the ontology. The instance level represents the actual instances of the classes and relationships defined in the schema level. This level includes specific entities, their properties, and the relationships between them, all based on the schema level's structure.

Any previous modular PLC code is also stored in the database so that it can be reused to build new FBD modules, making it extensible (O4), and reducing significantly manual work (O2).

3.1.2. *Understanding the product requirement*

This step is critical to getting all the necessary information to process the new task. The product requirement, BoP, and BoR will be generated based on the experience of the engineer or utilising automated methods such as the knowledge graph reasoning [76], and semantic search [77]. This step specifies the required manufacturing capabilities and feasible manufacturing resources (including humans). The approach developed by Mo et al. [54] is used in our methodology to detail the procedure of processing the new product requirement.

Algorithm 1 details the procedure of processing the new product requirement. There are three ways of finding the BoP. First of all, the product requirement and the BoP can be provided by the system. A rule-based approach will be applied if the BoP is not pre-provided. It is a traditional approach that extracts knowledge in the form of rules. For example, the sub-product requirement is to have a hole with a diameter of 50 mm in the product. Then drilling operation is needed. The end-effector can drill a hole of 50 mm, and a related robot is needed. The main problem with rule-based systems is that the programmer must specify all "rules" or <pattern> <template> pairs.

If the rule-based search does not yield the desired BoPs, a semantic search approach can be employed. Semantic search focuses on understanding the intent and context behind a product requirement, rather than just keyword matching. For instance, suppose a product requirement mentions "a durable material resistant to outdoor conditions." A semantic search could infer the need for materials like stainless steel or treated wood, even if these specific terms aren't present in the requirement, based on knowledge graphs or ontologies that understand the relationships and properties of materials.

Should the rule-based and semantic searches fail to secure the BoPs, the semantic embedding search becomes the primary recourse. This technique identifies potential processes related to a product node by leveraging semantic insights grounded in graph embeddings. To

facilitate this, a graph neural network is trained, converting graph-centric knowledge into a cohesive vector space, which in turn fosters efficient graph embeddings. Given the unpredictable nature of some product modifications, it is paramount for this neural network to feature inductive learning capabilities. In line with this, our methodology integrates the inductive learning approach, designed especially for handling expansive graphs. This approach not only streamlines data storage but also significantly reduces computational overhead in subsequent reasoning stages. Hence, when faced with a product inquiry, the adeptly pre-trained model recommends appropriate manufacturing processes and pinpoints the necessary resources.

Algorithm 1 Processing the product requirement

Input: Initialised Database (DB), Task (T)
Output: Bill of process, Bill of process

```
Load initial DB
ProductRequirement(PR) ← Find_requirement(T)
SubProductRequirement(SPR) ← Divide_ProductRequirement(PR)
for each SPR in PR do
    BillofProcess(BoP) ← PreProvided(SPR)
    if BoP ≠ Null then
        BoPs.add(BoP)
        continue
    else
        BoP ← Process_search_rule(SPR)
        if BoP ≠ Null then
            BoPs.add(BoP)
            continue
        else
            BoP ← Process_search_semantic(SPR)
            if BoP ≠ Null then
                BoPs.add(BoP)
                continue
            else
                BoP ← Process_search_semantic_embedding(SPR)
                if BoP ≠ Null then
                    BoPs.add(BoP)
                    continue
                else
                    "No bill of process can be found"
                end if
            end if
        end if
    end if
end for
Initialise bill of resource (BoRs)
BoRs ← Find_resources(BoPs)
return BoPs, BoRs
```

The outputs of this step are the BoP and BoR, which are necessary for building the simulation environment and generating the PLC code accordingly in an autonomous way (O1). In the BoPs, if the process and the resources are human-involved operations, the operations will be marked as manual operations (O3). The generated BoPs and BoRs will be utilised in the next steps to build the simulation environment and generate the PLC control code.

3.1.3. Building/updating the simulation process

Our approach aims at utilising and developing simulation software capable of simulating a broad set of vendor-specific robots and assets. We have therefore proposed a vendor-agnostic solution to update simulation environments, as depicted in Fig. 3. The simulation environment is generated based on the BoP, BoR, and the current system configuration, such as the location of each device. All the information will be saved in a JavaScript Object Notation (JSON) format to facilitate information exchange in heterogeneous environments (O5). The simulation

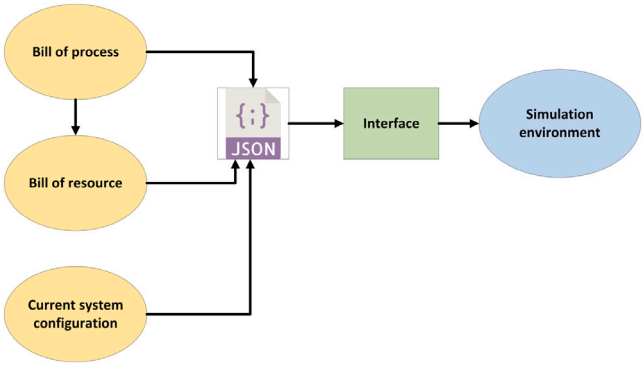

Fig. 3. Process of building the simulation environment.

environment is generated or updated manually and with the possibility of automatically from the JSON files (O2). When a new request comes, the simulation environment will be updated based on the updated BoPs and BoRs for the new product request. This interface also facilitates simulation re-use. For example, if the new product (e.g.: Product II) has some common parts to the old product (e.g.: Product I), then the common parts in the simulation environment can be reused for Product II and do not need to be newly modelled.

A human digital model has also been developed to simulate a worker's talents, behaviour, and well-being index. A digital human well-being impact profile would evaluate an activity's physical, cognitive, and psychological burden if it were viable for industrial workers to perform it. This is a requirement of the human-centric manufacturing system [69].

The creation of a simulation environment depends very heavily on the specific environment used, and will be a combination of automation and manual work. An example of how to create and update the simulation environment will be explained in detail in the implementation section.

3.1.4. Generation of the PLC control code

A PLC is used to orchestrate manufacturing processes and resources to control the overall production process. As such, a new PLC control code needs to be generated every time there is a new upcoming product request.

Automatic code generation for this PLC is generated by parsing the BoP and BoR to determine the process sequence, determining the order in which code needs to be executed and which resources must be triggered for a given process (O2). The code generated by this process can then be stored in any standardised format, such as SCL source code or PLCopen-compliant XML, being interoperable with the software used in the other stages of the proposed methodology (O5). To generate the specific code required, a set of process mapping rules are applied to the manufacturing sequence defined within the BoPs, see Fig. 2.

A mapper between the skill interfaces, which is generated based on the BoPs and BoRs, and the PLC generation modules is generated. Then the mapping between the skill interfaces and the PLC code is executed. This mapping is required to transform the physical process created within the simulation environment into the sequence of PLC instructions that are required to orchestrate the process. These two representations of the process can be quite different, so these mapping rules allow for a consistent approach to orchestrate a given process type (O1). Examples of these process types and mapping rules are illustrated in Table 1.

The code generation process uses the database of process-specific PLC code, which it uses as part of the PLC instructions generated by the mapping process. These pre-existing pieces of code can come from various sources but are most likely to have been developed by a suitably experienced engineer with knowledge of the specific process. Using this

Table 1
Examples of mapping rules to generate PLC code from simulation information and the product requirement (The mapping rules need not be fixed and can be tailored to a manufacturer's needs)

Operation Type	PLC action
Robot (Automation)	(1) Send a request to the robot controller to run offline programming (OLP) code, specified by filename on the controller. The OLP code is generated by relevant robot simulation environments such as Siemens Process Simulate (2) Wait for a signal from the robot controller that the robot has completed the robot operation specified in the robot code (or handle error condition)
Human (Manual)	(1) Send a message to HMI with work instructions to the human operator (2) Wait for a user input on HMI to acknowledge that the task is complete
Metrology (Automation)	(1) Send a message to the metrology controller (e.g: PC) to trigger the measurement procedure (2) Wait for a signal from the metrology controller that the metrology process is complete

approach allows for the coordination of tasks that would be otherwise too difficult to automate or even for repetitive tasks that do not require the level of simulation fidelity that would be otherwise required to perform automatic code generation from the simulation environment. This way, when a new request comes, some of the existing PLC code can be reused and updated if there are overlapping skill models (O4).

3.1.5. Conversion/check and update of the PLCopen XML

A method of generating a standard format of the PLC code should be developed to enable the user to exchange the PLC software programs from different vendors, libraries, and projects between development environments. In this regard, we have followed the PLCopen group committee guidelines, which have established an open interface that supports a broad set of software tools (O5). We have focused on TC6 for XML workgroup, as it enables the transmission of screen-based data to other platforms [78]. This way, the PLC code is converted to PLCopen XML to perform a vendor-independent PLC testing approach in the next step.

3.1.6. PLC code testing

PLC testing is vital in adaptive manufacturing systems, as the environment and respective control code are continuously updated to proceed with new tasks. Such changes to the PLC code must be fully tested to ensure there are no implementation errors in the software. This is a tedious task, as the PLC code will be frequently modified to respond to new requirements.

As seen in the literature, structural testing is a common practice when testing the control software. FBD structural coverage is calculated by a sequence of data paths. Each data path describes all the conditions that the input edges must go through to reach the output edges of the FBD program. The output of each Function or Functional Block is determined by the input value(s), which results in input condition-based data paths. FBD coverage is then calculated by a combination of multiple function conditions. In this matter, Jee et al. [36] proposed Basic Coverage, Input Condition Coverage, and Complex Condition Coverage, as specified below:

- Basic coverage (BC): every data path in the FBD program must be tested at least once.
- Input Condition coverage (ICC): every data path in the FBD tester must be tested with all combinations of Boolean input edges.
- Complex Condition coverage (CCC): every data path in the FBD tester must be tested with all combinations of Boolean input and output edges.

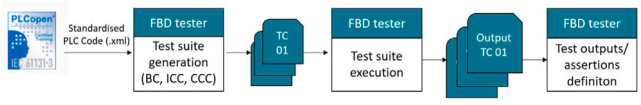

Fig. 4. PLC code testing process.

FBD structural testing will be therefore carried out based on the aforementioned metrics. In specific, we build our solution on the FBD tester developed by KAIST university [37], as it does not require any additional model-to-model transformations to generate structural data path-based test cases. While coverage-based testing aims to maximise the structural coverage of a program, it does not necessarily correlate with high effectiveness in detecting faults. Therefore, we also generated mutation-based test cases to leverage the benefits of both techniques.

As outlined within the objectives, the approach should be vendor agnostic and interoperable with a wide range of PLC controllers; therefore, we have designed a testing methodology for PLCopen standardised functional blocks (O5). The presented process involves the following steps, detailed in Fig. 4:

The FBD tester first gets the PLC code in the PLCopen XML format to define the functions, functional blocks, constant values, and input and output signals. Such information is utilised to define the data paths and data path conditions for each input and output edge of the FBD. It is worth emphasising that the library should encompass all the essential functions. If a specific function is not available, it is essential to update the library accordingly. The FBD tester then generates test suites based on BC, ICC, and CCC structural coverage criteria aiming at maximising the coverage with the minimum number of test cases. These tests can be valuable in assessing the quality of the model design for subsequent SiL (Software-in-the-Loop) or HiL (Hardware-in-the-Loop) testing.

A test case includes input values for a minimum set of scan cycles required in the FBD. Each test case addresses basically a certain number of data path combinations; hence the coverage level increases with the number of test cases. Some of the test cases may also have overlapping data path conditions. Thus the test generation algorithm only generates test cases for those data paths that have not been addressed until maximum coverage is achieved.

The generated test suites are then executed in the FBD standalone tester to define assertions based on the output values. Test outputs will be used as oracles when executing the PLC code in the simulation environment (SiL) and the physical PLC (HiL) before deploying the PLC code into operations. Tests will be passed if the asserted output value is the same as the FBD tester execution results in Java. If not, a test failure will be reported as seen in Figs. 5–6:

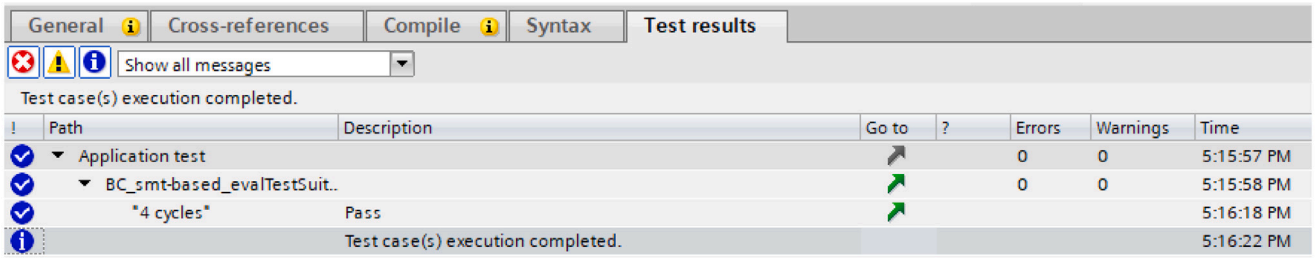

Fig. 5. Successful test execution.

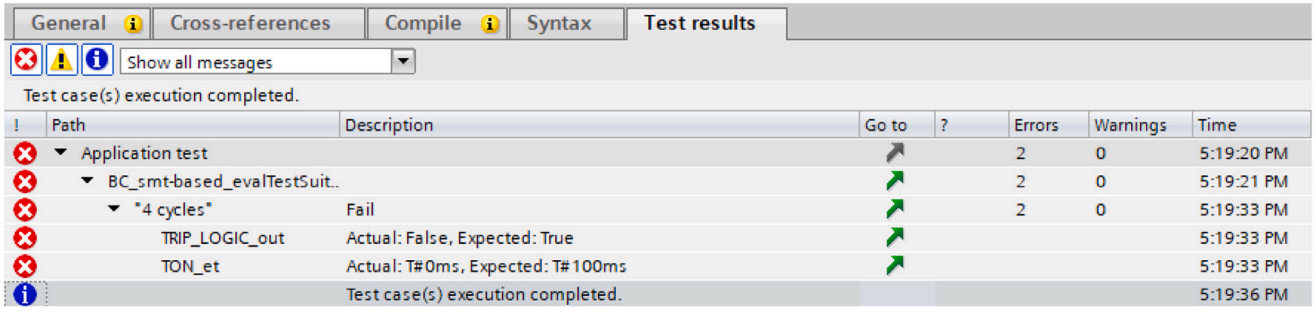

Fig. 6. Failed test execution.

To date, this approach enhances PLC testing practices, a process that has been manually performed in the industry, leading to human errors (O2). This way, the proposed methodology is expected to reduce implementation errors while shortening the commissioning time significantly.

3.1.7. Virtual commissioning and deployment

Once the simulation environment is built and the PLC code is thoroughly tested, Virtual commissioning (VC) can be executed to test the effectiveness of the PLC code. VC is the process of validating the software code for PLCs, HMIs and SCADA equipment in the virtual world before deploying it on the factory floor. The code of the PLC controls a virtual model, also called the behaviour model, which behaves like the real machine [79]. The PLC code generated from the modular database will be optimised and updated based on the VC results from the simulation software. To enable virtual commissioning, the PLC code, which is generated in Section 3.1.4, is imported to the PLC simulation software (e.g., TIA Portal [80], CODESYS [81], TwinCAT [82]).

Dynamic reconfiguration of a PLC system introduces complexities that require careful management to ensure safety. Any change in the control logic or machine configuration can potentially impact safety, making it critical to synchronise safety measures with the implemented changes. Safety verification is performed within the virtual environment during VC. This includes testing for potential hazardous states, verifying fail-safe mechanisms, and validating safety functions within the updated logic. VC aids in ensuring that changes do not compromise system safety before they are implemented on real equipment.

After successful commissioning, the updated configuration and control code will be deployed to the real equipment. However, applying these updates to real equipment presents some challenges;

- **I/O Modules:** The updated configuration must be supported by the correct I/O module (in terms of bandwidth, channels etc.). It is necessary to thoroughly examine field equipment for compatibility before implementing updates.
- **In or Out:** Wiring errors and incompatibility can occur as the program is updated, leading to undesired behaviour and potential damage from a short circuit. An "integrity check" for in and out wiring must be conducted to ensure correct memory addressing and power connection.
- **Interference:** External influences such as electromagnetic or radio frequency interference can cause issues as program logic is updated and executed. Proper shielding and grounding practices must be in place to mitigate these effects.
- **Memory:** Frequently changing the programs in the PLC can lead to memory corruption, necessitating a system refresh. It is crucial to verify that the configuration operates as expected after changes.
- **Troubleshooting:** Changing configurations can cause conflicts or errors in PLCs. Different manufacturers have unique troubleshooting methods that need to be understood and effectively applied.

By managing these challenges effectively, the updated PLC code, optimised via virtual commissioning, can be safely and successfully deployed to the real-world system.

4. Implementation and case study

A use case was designed to verify the feasibility of our proposed methodology. The production of aircraft parts still relies heavily on human labour, with parts often loaded into jigs manually, representing a significant health and safety risk for the workers and an increased likelihood of damaging components during the manufacturing process [83]. In the near future, the demand for aircraft is expected to increase considerably and thus, there is a need to produce a large number of aircraft units [84]. This will consist of existing aircraft designs, re-engineering of existing designs, and new designs yet to be launched. Tier 1 suppliers – which supply both large structural components and small box assemblies to the original equipment manufacturers (OEMs) – will particularly benefit from optimising the production line with modular, automated, intelligent, and reconfigurable solutions to meet the varying demands of the OEMs. Reconfigurable automated solutions are considered key to the assembly strategy, as they bring benefits such as increased automation across assembly operations, fewer variations, standardised processes, increased process capability, reduced tool design lead time, increased in-house knowledge, reduced integration out-sourcing, increased re-use of expensive capital equipment, increased in-house assembly, and a more sustainable approach [85,86].

Fig. 7. Architectural layout of a test plant within the OMNIFACTORY.

Table 2
Current process - producing product I.

Step 1	WS1	Pick and Place Upper/Lower Beam (Load)
Step 2	WS2	Manual Assembly - Pick parts
Step 3	WS2	Manual Assembly - Place parts
Step 4	WS2	Manual Assembly - Mount parts
Step 5	WS3	Inspection - Step check
Step 6		Pick and Place Upper Skin (Load)
Step 7		Drilling Upper Skin
Step 8		Pick and Place Profile Board (Remove)
Step 9		Pick and Place Profile Boards (Load)
Step 10		Pick and Place Lower Skin (Load)
Step 11		Drilling Lower Skin
Step 12		Pick and Place Profile Board (Remove)
Step 13	WS1	Inspection - Profile check

At the University of Nottingham, an adaptive assembly facility in the OMNIFACTORY [87] test plant is designed to be applied to a range of small box assemblies for aircraft. Opportunities to reduce adjustment and shimming on assembly operations will be explored through the development of virtual assembly models, allowing virtual assemblies to be built with data collected significantly further upstream. This cell will also look at all data collection and processing opportunities to support a multi-product production line concept. Our method is validated in this assembly facility, enabling it to react more quickly to new aircraft model launches, reduce product life cycles, and deal with fluctuations in product demand. Fig. 7 shows a station of this assembly cell.

The current production line is intended to assemble *Product I* and other variants. It consists of three workstations, with a mix of both automated and manual assembly solutions. The initial reconfiguration and inspection will happen at Station 1. Station 2 will then utilise human workers for the manual assembly of small components. At Station 3, automatic assembly of the large components will happen. The required operations to produce a *Product I* are detailed in Table 2. These operations will be carried out at different workstations (WP1, WP2, WP3) as detailed in Fig. 8.

In this use case, the assembly line will suddenly receive a new customer request (*Product II*). We have therefore applied our methodology to enable the human–machine collaborative PLC adaptation and proceed with the new request. The detailed implementation steps are described in the following sections:

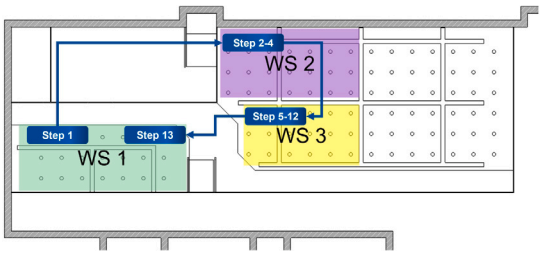

Fig. 8. Current process.

4.1. Establishment of the database for PLC and manufacturing information

Regarding implementing ontologies, the major technologies used in creating the database include knowledge modelling, knowledge extraction, ontology matching, and knowledge inferences. In specific, Protégé was selected to define the ontology, and Neo4j to build the graph database platform, as detailed below.

The database is built based on the ontology model we defined in Protégé. The presented use case includes task information, product requirements, part information, required process information, capability information of the assets, and device information. The ontology model is illustrated in Fig. 9. Then this ontology information is converted to a JSON model, which can be used in the data importer of Neo4j to build the graph database.

The skill model is generated based on the capability model described in Neo4j. For the purpose of this demonstration, we clarify the relationship between skills and capabilities as follows: a skill is an executable action based on an underlying capability. In simpler terms, a capability is the potential to perform a specific action, while a skill is the actual execution of that action. For example, if the robot has the capability to move, then in our demonstration, the corresponding skill allows the robot to perform the actual movement. Regarding the PLC code, we used Siemens TIA Portal for PLC programming in the demo. The PLC code was generated through a custom XML generation program we designed.

Based on the mapping rules, the customised program can generate the PLC code of specific skills from the modular database. The generated PLC code is stored in Siemens-compliant XML format following the structure depicted in Fig. 10.

4.2. Understanding the product requirements

The steps for understanding the product requirements are executed based on the Algorithm 1 described in Section 3.1.2. The BoPs and BoRs of the current product (*Product I*) listed in Table 2 are generated from the experience of the engineer (pre-provided and rule-based search). If the system encounters product requirements that do not directly align with the engineer's predefined methods but can be interpreted or converted using domain-specific ontologies to uncover implicit relationships, then the semantic search is employed. However, if the semantic search proves insufficient, especially for product with unknown requirements (e.g. *Product II*), the semantic search with embedded vector representation (semantic embedding search) is executed in the algorithm. As a result, the semantic embedding search find the most suitable BoPs and BoRs. This is achieved by a GraphSAGE graph neural network model. The network is trained to transfer the graph data into the embedding vector to execute the semantic search. The training process was done with StellarGraph [88], which is a Python library that offers state-of-the-art algorithms for graph machine learning.

Once BoPs and BoRs are generated, the information is stored in a JSON file [89], which is essential for constructing the simulation environment. Fig. 11 outlines the structure of the JSON file used in our

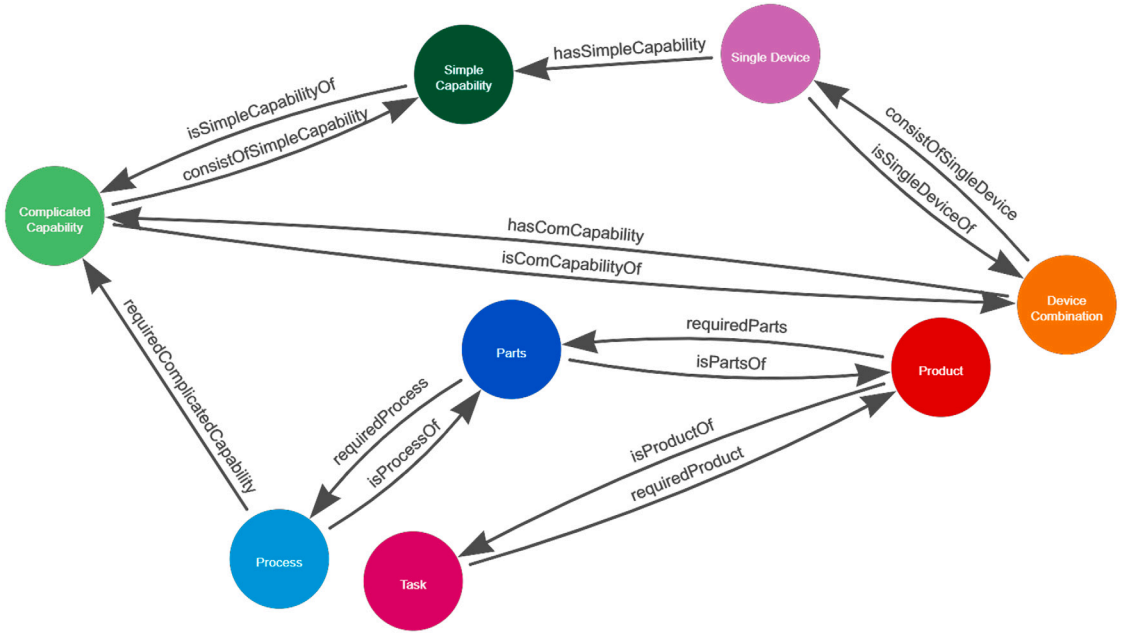

Fig. 9. Ontology model of our proposed framework.

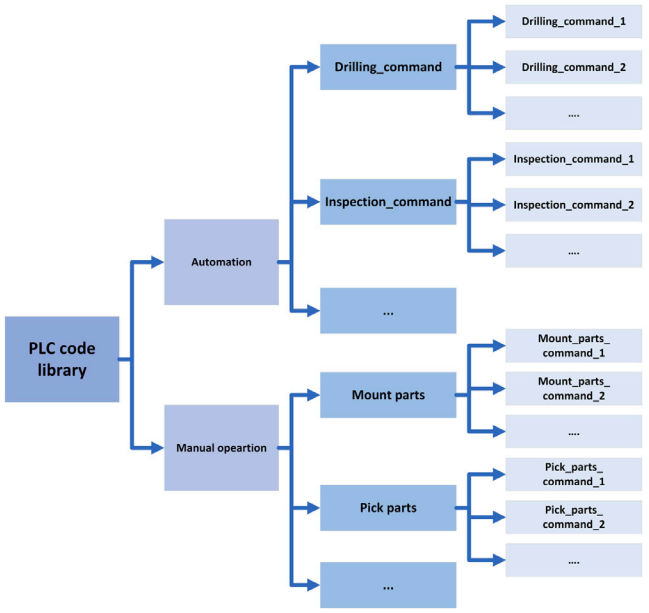

Fig. 10. PLC code library structure.

Fig. 11. Json information about the BoPs and BoRs.

approach. The JSON file gathers the necessary information for building the simulation environment, including details on parts, resources, manufacturing features, and compound operation information.

In the current implementation, the simulation environment is updated through a semi-automatic process. The JSON file is manually loaded, and then the simulation environment is updated based on the provided information. While this approach reduces some manual labour, there is still potential for further optimisation to streamline the process and improve overall efficiency.

4.3. Building the simulation environment

In this case study, Siemens Tecnomatix Process Simulate is utilised for the simulation, as it can simulate a wide set of robot controllers and supports secondary development to customise behaviour. The Tecnomatix .NET API [90] was employed to enable the secondary development of Tecnomatix software. The developed modules with this development tool exhibit fast running speeds and high reliability. With the assistance of the Tecnomatix .NET API, the simulation environment is generated based on the information from the BoPs, BoRs, and the current system configuration, following the approach presented in Fig. 3.

The Tecnomatix .NET API plays a crucial role in bridging the communication between the simulation environment and various elements, such as the BoPs (Bills of Processes), BoRs (Bills of Resources), and the current system configuration. By leveraging the capabilities of this API, we were able to effectively extract relevant features and import necessary data, enabling seamless integration of these elements into the simulation environment.

Fig. 12 is the user interface for importing the JSON file to generate or update the simulation environment in Tecnomatix .NET API. This interface allows for the efficient importation of data, streamlining the process and reducing the possibility of errors. Fig. 13 showcases the generated simulation environment in the Tecnomatix Process Simulate, providing a comprehensive and accurate representation of the system being modelled.

Overall, the use of the Tecnomatix .NET API in this case study has demonstrated its effectiveness in streamlining the development process, ensuring high reliability and efficiency, and providing a foundation for further development and customisation as needed.

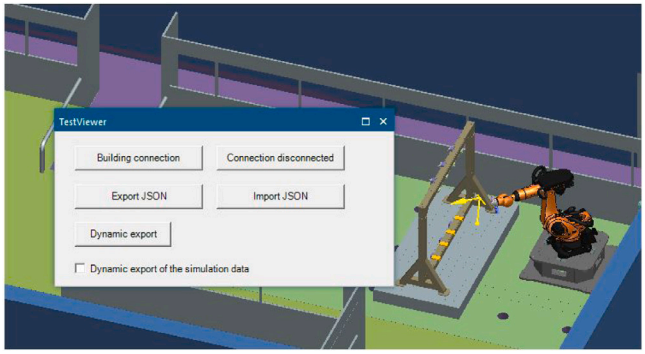

Fig. 12. GUI interface for building the simulation environment.

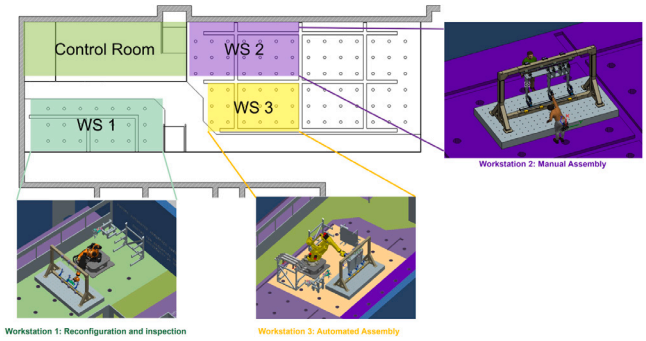

Fig. 13. Generated simulation environment.

4.4. Generation of the PLC code

As proposed in Section 3, a mapper between the skill model interface and PLC code library is designed. In the validation use case, we created the mapper with Protégé and Owlready2, which is a package for manipulating OWL 2.0 ontologies in Python [91]. At this step, the modular process-specific PLC codes are stored in the database, which are later needed for the mapping rules. The required information from the BoPs and BoRs is applied to the skill model with the ontology model we designed. The corresponding PLC code is then generated based on the skill model. As mentioned before, the PLC database already stores some PLC code.

A benefit of using this modular approach is how flexible the system is in incorporating new processes or technologies; once a new process-specific code is developed, it can be easily used within the automatic code generation process by entering it into the database and adding a new rule into the mapping process.

Based on this approach and the mapping tools, the related PLC code from the BoPs and BoRs was generated as indicated in Fig. 14.

4.5. Conversion/check of the PLCopen XML

The generated PLC code is in Siemens-compliant format, as described in Section 4.4. In this case, a conversion to the PLCopen XML is needed to convert the Siemens-compliant format to the PLCopen XML format. PLC automated conversion work remains in progress, and for this use case was manually done to demonstrate the methodology. PLC code was replicated in the Codesys development environment so that it could be exported automatically on an IEC 61131-3 standardised XML format.

4.6. PLC code testing

The developed PLC testing solution is implemented in the Siemens TIA Portal environment by following the approach described in Fig. 15. TIA Portal automatically creates a PLCSIM Advanced instance – a virtual replica of the hardware in place – and downloads the PLC configuration to the PLC simulation instance for executing the generated test cases. Thus, a software-in-the-loop approach is proposed to validate the changes before commissioning and deploying the new PLC code into operations.

The presented approach was implemented for the “Drilling” operation of the OMNIFACTORY. Specifically, the following FBD networks of the drilling process were selected: FBD, which generates position flags, and FBD, which sets the execute bit when a position is required.

The FBD tester was developed by KAIST [37] for FBD automated coverage-based test data generation and MuFBD tester [41] for mutation-based test data generation. We extended these tools for their use in the OMNIFACTORY by 1) updating the respective FBD libraries, 2) developing cost-effective metrics calculation – Test Execution Time (TET), Fault Detection Capability (FDC), and coverage (BC, ICC, CCC) – to cost-effectively select the test cases, and 3) developing automated test execution to obtain test oracles, which are required to benchmark TIA Portal execution results. In addition, we implemented a new SMT solver – Yices 2 SMT2 – to generate test cases for a wide range of IEC 61131-3 function groups, including non-linear arithmetic functions, as opposed to the legacy KAIST tool.

Given a large number of test cases per each Functional Block diagram, a cost-effective test selection approach was applied based on the evaluation of cost-effective metrics, i.e. TET, FBD, and coverage. The developed test selection testing was based on a multi-objective search-based algorithm that maximises FDC and coverage (BC, ICC, CCC) while reducing execution time.

A total of 134 test cases were generated for the selected FBD networks of the drilling operation: 2 test cases for the BC test suite, 5 test cases for ICC and CCC test suites, and 122 mutation-based test cases. Fig. 16 shows an example test case of the basic coverage test suite with three cyclic iterations, as indicated on the left side. These test cases were then executed in Java to record the expected output values of every intermediate cycle of each test case, as outlined on the right side.

A converter was developed to map the generated test cases into the TIA Portal Test Suite Application format. TIA Test Suite application allows us to define the input values for a given number of scan cycles or execution time. Hence, every output of each execution run can be assessed according to previous test execution results.

Finally, the generated TIA Test Suite application test cases were imported to TIA Portal via TIA Openness. Siemens TIA Openness is an API that allows the engineer to interact with TIA Portal using a customised application. It offers many advantages that lead to more efficient code development. For example, TIA Openness allows us to create modular code that can be used across many devices, with application-specific changes being made automatically. By defining templates, users can generate entire projects without any specific knowledge of PLC and HMI programming.

TIA Portal then establishes a connection with the S7-PLCSIM Advanced simulation environment to execute the generated test cases in the FBD program (see Fig. 17).

TIA Portal Test Suite first compiles the program and downloads the hardware configuration to PLCSIM Advanced. All variables are also loaded into the device. Once successfully completed, application tests were executed and asserted with test oracles one by one. All tests were successfully executed with a success rate of 100.00%, which means that TIA execution results matched the test oracles generated in Java. On average, each test case execution lasted around 40 s, resulting in a total execution time of 132 min. This significantly reduces time to market, as traditional manual testing requires a lot of time and is error-prone, depending on the experience of the engineer.

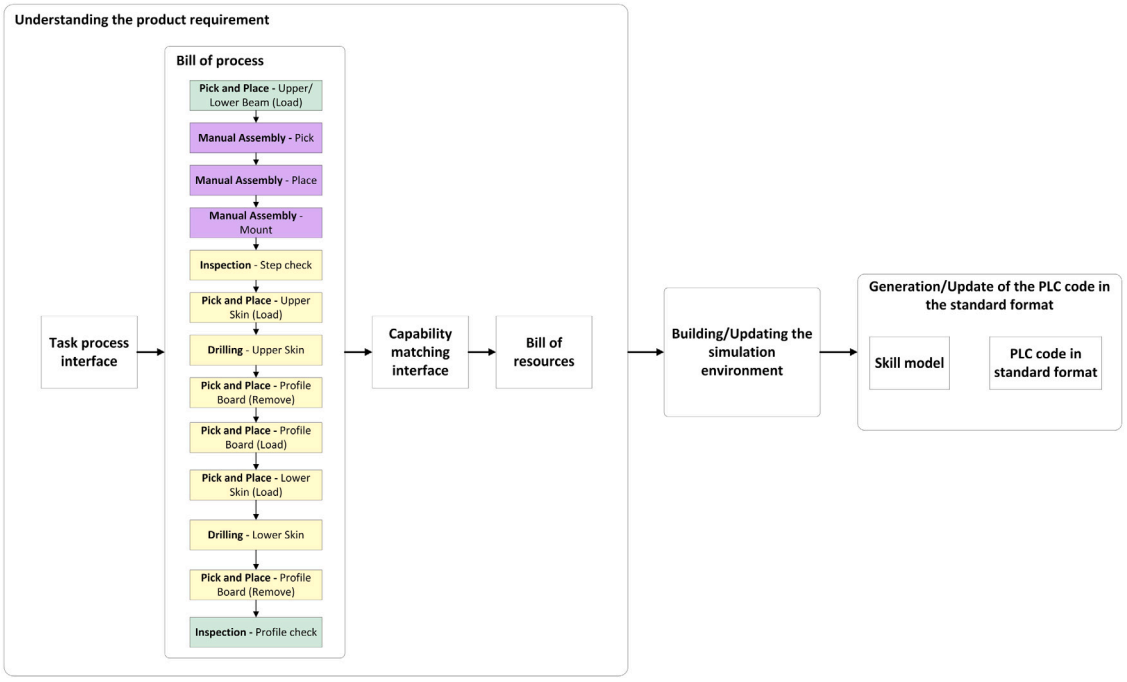

Fig. 14. PLC code generation process.

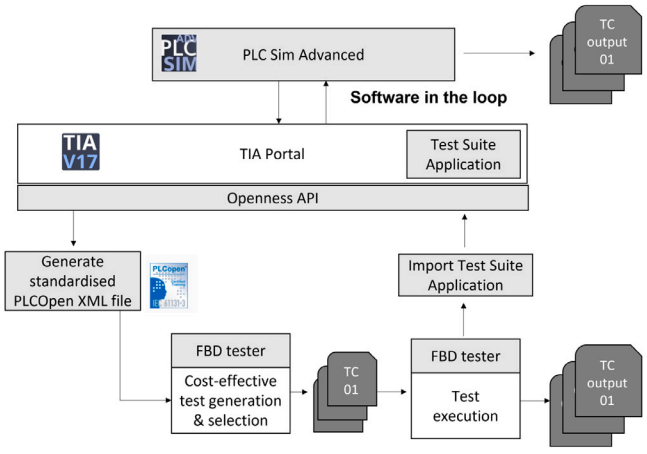

Fig. 15. PLC Code testing - the overall structure.

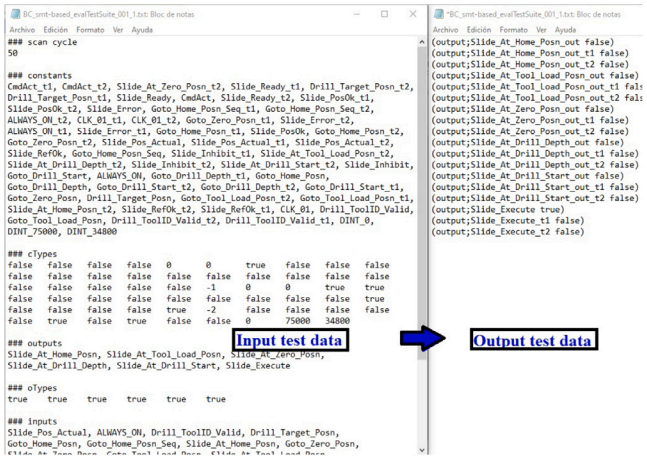

Fig. 16. Example of a test case.

4.7. Virtual commissioning

As mentioned in the previous steps, Siemens Tecnomatix Process Simulate is utilised in our approach. Virtual commissioning was performed via TIA PLCSIM Advanced to validate the PLC code against the newly updated simulation environment.

In Tecnomatix Process Simulate, there are two modes: standard mode and line simulation mode. The standard mode is used for detailed robot planning, which is a time-based simulation. Compared with the standard mode, the line simulation mode is an event-based simulation in which operations are triggered continuously by signals. In our proposal, the line simulation mode was chosen for virtual commissioning because it is more suitable to describe the use case and implementation. This choice is made as the regular factory control is typically performed through event-triggered operations rather than time-sequenced actions. By using the line simulation mode, we can accurately represent the real-world behaviour of the system and ensure a more effective evaluation of its performance during the commissioning process. To enable virtual commissioning, the PLC code generated in our last step was converted to the format used in TIA Portal. As mentioned in Section 4.4, the generated PLC code is in the Siemens XML format. Therefore, the generated PLC code was imported to the Siemens automation software TIA portal in this step. This was done with the help of the TIA Openness program.

This API interface supports exporting consistent blocks and user data types to an XML file. The XML file receives the name of the block. The following block types are supported: Function blocks (FB), Functions (FC), Organisation blocks (OB), and Global data blocks (DB). According to the International Electrotechnical Commission 61131-3, the following programming languages are supported: TL, function block diagrams (FBD), ladder logic (LAD), GRAPH, and SCL.

We have developed a customised program leveraging Siemens TIA Openness, which can automatically extract PLC code from the TIA portal. Our tailored PLC generation program offers several advantages over the standard TIA Openness demo program provided by Siemens:

- Our program can import different blocks at the same time, whereas the TIA openness demo can only import one block at a time.

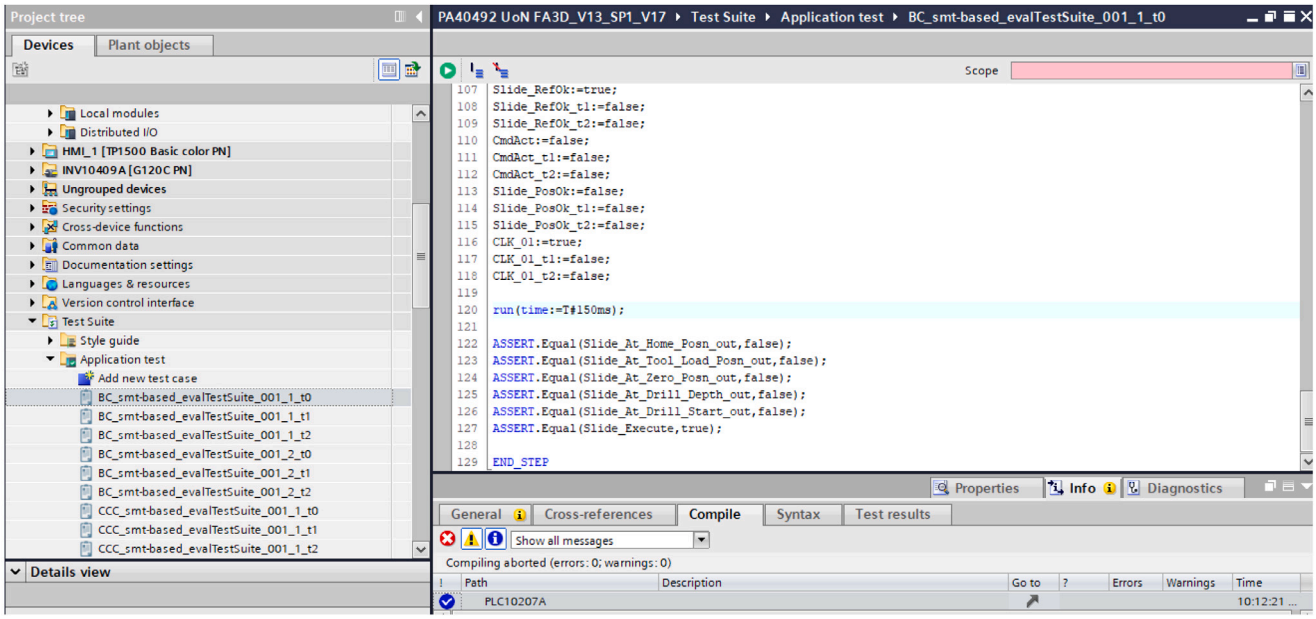

Fig. 17. Example of TIA Portal Test Suite application test.

- Our program can automatically integrate and extract the PLC code via automation methods.

The process of importing the PLC code from the original Siemens-compliant XML format is defined in the next steps:

- **Selecting the version**: Select the version of the software, as the project file is related to the TIA Portal software version. A compatibility problem will exist if the software version is different from the project file version.
- **Selecting the project**: The TIA Openness program will select the target TIA Portal project stored locally or in the cloud. The corresponding TIA project will then be automatically launched in TIA Portal.
- **Updating the project information**: The information on the project in the TIA portal will be updated in the TIA Openness Program user interface. All the information on the selected TIA Portal project will be updated with the help of the developed import automation command.
- **Importing the PLC code**: With the help of TIA Openness API, the generated PLC code in Siemens-compliant XML format can be imported into the TIA Portal.

5. Conclusions

This paper presents a methodology aimed at enhancing human–machine integration in adaptive manufacturing systems through automatic PLC code generation and testing. By automating the PLC code generation and testing process, the proposed approach addresses the evolving manufacturing demands and reduces the need for human workers to perform repetitive tasks. This enables humans to play a more active role in adaptive manufacturing systems, working in harmony with machines.

The applicability of this approach has been demonstrated through an industrial use case. In this implementation, a modular PLC database is employed, facilitating the generation and updating of PLC code based on new requirements and previously available PLC function blocks. This allows existing modules to be efficiently reused and extended as needed, depending on the processes to be operated. PLCopen, a vendor-independent solution, is utilised as the PLC code format, broadening the potential scope of applications. An automatic interface is implemented for generating/updating the simulation, while a customised TIA Openness program automates the import and export of PLC data from the TIA Portal.

The following manual commissioning procedures have been successfully automated, significantly reducing error-prone routine tasks, human effort, and time:

- Automated PLC code generation
- Automated coverage and mutation-based test case generation
- Automated multi-objective test case selection
- Automated test case execution and assertion with the use of test oracles

For future work, we plan to extend the implementation to encompass physical hardware, going beyond simulations. The performance of the proposed methodology will be benchmarked against conventional manufacturing assembly systems to justify the return on investment in terms of costs and time. Furthermore, we aim to validate the methodology on other PLC vendors, such as Beckhoff and Mitsubishi, to ensure maximum extensibility. To achieve this, the ontology model for the PLC database will be enriched with existing Industry 4.0 Standards [92]. As the data stored in the PLC database grows, deep learning algorithms will be employed to select relevant PLC code more accurately [93]. Finally, vision sensors, virtual reality, and augmented reality will be integrated to implement the proposed method in human–machine collaborative manufacturing systems.

By automating key aspects of the manufacturing process, this paper highlights the potential for improved human–machine integration, allowing for a more efficient and collaborative work environment.

CRediT authorship contribution statement

Fan Mo: Conceptualization, Methodology, Writing – original draft, Software. **Miriam Ugarte Querejeta:** Conceptualization, Methodology,

Writing – original draft, Software. **Joseph Hellewell:** Conceptualization, Writing – original draft, Software. **Hamood Ur Rehman:** Conceptualization, Writing – original draft. **Miren Illarramendi Rezabal:** Supervision. **Jack C. Chaplin:** Writing – review & editing, Supervision. **David Sanderson:** Writing – review & editing, Supervision. **Svetan Ratchev:** Supervision, Funding acquisition, Resources.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This research is supported byDiManD Innovative Training Network (ITN) project funded by the European Union through the Marie Skłodowska-Curie Innovative Training Networks (H2020-MSCA-ITN-2018) under grant agreement number no. 814078 and the Elastic Manufacturing Systems project (Project Reference EP/T024429/1) funded by the UK Engineering and Physical Science Research Council. The authors would like to acknowledge the support from Innovate UK project ELCAT (ref 113235) and GKN Aerospace.

References

[1] Storrie Donald. The future of manufacturing in Europe. Eurofound; 2019.
[2] Mo Fan, Chaplin Jack C, Sanderson David, Rehman Hamood Ur, Monetti Fabio Marco, Maffei Antonio, et al. A framework for manufacturing system reconfiguration based on artificial intelligence and digital twin. In: International conference on flexible automation and intelligent manufacturing. Springer; 2022, p. 361–73.
[3] Mo Fan, Monetti Fabio Marco, Torayev Agajan, Rehman Hamood Ur, Mulet Alberola Jose A, Rea Minango Nathaly, et al. A maturity model for the autonomy of manufacturing systems. Int J Adv Manuf Technol 2023;126(1–2):405–28.
[4] Cimini Chiara, Pirola Fabiana, Pinto Roberto, Cavalieri Sergio. A human-in-the-loop manufacturing control architecture for the next generation of production systems. J Manuf Syst 2020;54:258–71.
[5] Peruzzini Margherita, Pellicciari Marcello. A framework to design a human-centred adaptive manufacturing system for aging workers. Adv Eng Inform 2017;33:330–49.
[6] Lu Yuqian, Xu Xun, Wang Lihui. Smart manufacturing process and system automation–a critical review of the standards and envisioned scenarios. J Manuf Syst 2020;56:312–25.
[7] Bortolini Marco, Faccio Maurizio, Galizia Francesco Gabriele, Gamberi Mauro, Pilati Francesco. Adaptive automation assembly systems in the industry 4.0 era: A reference framework and full–scale prototype. Appl Sci 2021;11(3):1256.
[8] Fletcher Sarah R, Johnson Teegan, Adlon Tobias, Larreina Jon, Casla Patricia, Parigot Laure, et al. Adaptive automation assembly: Identifying system requirements for technical efficiency and worker satisfaction. Comput Ind Eng 2020;139:105772.
[9] Rehman Hamood Ur, Chaplin Jack C, Zarzycki Leszek, Mo Fan, Jones Mark, Ratchev Svetan. Service based approach to asset administration shell for controlling testing processes in manufacturing. 2022.
[10] Bortolini Marco, Faccio Maurizio, Galizia Francesco Gabriele, Gamberi Mauro, Pilati Francesco. Design, engineering and testing of an innovative adaptive automation assembly system. Assem Autom 2020;40(3):531–40.
[11] Sărăcin Cristina Gabriela, Deaconu Ioan Dragoş, Chirilă Aurel Ionuţ. Educational Platform Dedicated to the Study of Programmable Logic Controllers and the Human-Machine Interface. In: 2019 11th international symposium on advanced topics in electrical engineering (ATEE). IEEE; 2019, p. 1–4.
[12] Manesis S, Akantziotis K. Automated synthesis of ladder automation circuits based on state-diagrams. Adv Eng Softw 2005;36(4):225–33.
[13] Ulm G, Bellorini F, Brodrick D, Fernandes R, Levchenko N, Fernandez D Piso. PLC factory: Automating routine tasks in large-scale PLC software development. In: 16th int. conf. on accelerator and large experimental control systems (ICALEPCS'17), Barcelona, Spain, 8-13 October 2017. JACOW Geneva; 2018, p. 495–500.
[14] Rehman Hamood Ur, Pulikottil Terrin, Estrada-Jimenez Luis Alberto, Mo Fan, Chaplin Jack C, Barata Jose, et al. Cloud based decision making for multi-agent production systems. In: Progress in artificial intelligence: 20th EPIA conference on artificial intelligence, EPIA 2021, Virtual Event, September 7–9, 2021, proceedings 20. Springer; 2021, p. 673–86.
[15] Babiceanu Radu F, Chen F Frank. Development and applications of holonic manufacturing systems: a survey. J Intell Manuf 2006;17:111–31.
[16] Trentesaux Damien. Distributed control of production systems. Eng Appl Artif Intell 2009;22(7):971–8.
[17] Lyu Guolin, Brennan Robert William. Towards IEC 61499-based distributed intelligent automation: A literature review. IEEE Trans Ind Inf 2020;17(4):2295–306.
[18] Tiegelkamp Michael, John Karl-Heinz. IEC 61131-3: Programming industrial automation systems. Vol. 166, Springer; 2010.
[19] Páez-Logreira Heyder David, Zamora-Musa Ronald, Bohorquez-Perez Jose. Programming logic controllers (PLC) using ladder and structured control language (SCL) in MATLAB. Revista Facultad de Ingenieria 2015;24(39):109–19.
[20] Estévez E, Marcos Marga, Lüder Arndt, Hundt Lorenz. PLCopen for achieving interoperability between development phases. In: 2010 IEEE 15th conference on emerging technologies & factory automation (ETFA 2010). IEEE; 2010, p. 1–8.
[21] Vogel-Heuser Birgit, Witsch Daniel, Katzke Uwe. Automatic code generation from a UML model to IEC 61131-3 and system configuration tools. In: 2005 international conference on control and automation. Vol. 2, IEEE; 2005, p. 1034–9.
[22] Hametner Reinhard, Kormann Benjamin, Vogel-Heuser Birgit, Winkler Dietmar, Zoitl Alois. Test case generation approach for industrial automation systems. In: The 5th international conference on automation, robotics and applications. IEEE; 2011, p. 57–62.
[23] van der Wal Eelco. PLCopen. IEEE Ind Electron Mag 2009;3(4):25.
[24] Wang Han, Tang Xiaoqi, Song Bao, Wang Xiaoyu. A novel architecture of the embedded computer numerical control system based on PLCopen standard. Proc Inst Mech Eng B 2014;228(4):595–605.
[25] Wu Yi-Chen, Fan Chin-Feng. Automatic test case generation for structural testing of function block diagrams. Inf Softw Technol 2014;56(10):1360–76.
[26] An Yameng, Qin Feiwei, Sun Danfeng, Wu Huifeng. A multi-facets Ontology matching Approach for generating PLC Domain Knowledge Graphs. IFAC-PapersOnLine 2020;53(2):10929–34.
[27] Aminu Enesi Femi, Oyefolahan Ishaq Oyebisi, Abdullahi Mohammad Bashir, Salaudeen Muhammadu Tajudeen. A review on ontology development methodologies for developing ontological knowledge representation systems for various domains. 2020.
[28] Elshafei Basem, Mo Fan, Chaplin Jack C, Arellano Giovanna Martinez, Ratchev Svetan. Capacity modelling and measurement for smart elastic manufacturing systems. Tech. rep., SAE Technical Paper; 2023.
[29] Martínez-Arellano Giovanna, Niewiadomski Karol, Mo Fan, Elshafei Basem, Chaplin Jack C, Mcfarlane Duncan, Ratchev Svetan. Enabling Coordinated Elastic Responses of Manufacturing Systems through Semantic Modelling. 2023.
[30] An Yameng, Qin Feiwei, Chen Baiping, Simon Rene, Wu Huifeng. OntoPLC: semantic model of PLC programs for code exchange and software reuse. IEEE Trans Ind Inf 2020;17(3):1702–11.
[31] Bayha Andreas, Bock Jürgen, Boss Birgit, Diedrich Christian, Malakuti Somayeh. Describing capabilities of industrie 4.0 components: Joint white paper between plattform industrie 4.0, VDI GMA 7.20, BaSys 4.2. 2020.
[32] Köcher Aljosha, Jeleniewski Tom, Fay Alexander. A method to automatically generate semantic skill models from PLC code. In: IECON 2021–47th annual conference of the IEEE industrial electronics society. IEEE; 2021, p. 1–6.
[33] Jee Eunkyoung, Shin Donghwan, Cha Sungdeok, Lee Jang-Soo, Bae Doo-Hwan. Automated test case generation for FBD programs implementing reactor protection system software. Softw Test Verif Reliab 2014;24(8):608–28.
[34] Hametner Reinhard, Kormann Benjamin, Vogel-Heuser Birgit, Winkler Dietmar, Zoitl Alois. Automated test case generation for industrial control applications. In: Recent advances in robotics and automation. Springer; 2013, p. 263–73.
[35] Enoiu Eduard P, Čaušević Adnan, Ostrand Thomas J, Weyuker Elaine J, Sundmark Daniel, Pettersson Paul. Automated test generation using model checking: an industrial evaluation. Int J Softw Tools Technol Transf 2016;18(3):335–53.
[36] Jee Eunkyoung, Yoo Junbeom, Cha Sungdeok, Bae Doohwan. A data flow-based structural testing technique for FBD programs. Inf Softw Technol 2009;51(7):1131–9.
[37] Song Jiyoung, Jee Eunkyoung, Bae Doo-Hwan. FBDTester 2.0: Automated test sequence generation for FBD programs with internal memory states. Sci Comput Program 2018;163:115–37.
[38] Jia Yue, Harman Mark. An analysis and survey of the development of mutation testing. IEEE Trans Softw Eng 2010;37(5):649–78.
[39] Papadakis Mike, Kintis Marinos, Zhang Jie, Jia Yue, Le Traon Yves, Harman Mark. Mutation testing advances: An analysis and survey. In: Memon Atif M, editor. Advances in computers. Vol. 112, Elsevier; 2019, p. 275–378.
[40] Enoiu Eduard P, Sundmark Daniel, Čaušević Adnan, Feldt Robert, Pettersson Paul. Mutation-based test generation for PLC embedded software using model checking. In: Wotawa Franz, Nica Mihai, Kushik Natalia, editors. Testing software and systems. Lecture notes in computer science, Vol. 9976, Cham: Springer; 2016, p. 155–71. http://dx.doi.org/10.1007/978-3-319-47443-4_10.
[41] Liu Lingjun, Jee Eunkyoung, Bae Doo-Hwan. Mugenfbd: Automated mutant generator for function block diagram programs. KIPS Trans Softw Data Eng 2021;10(4):115–24.
[42] Fernández B, Blanco E, Merezhin A. Testing & verification of PLC code for process control. In: Proceedings of ICALEPCS. 2013.

[43] Schofield B, Viñuela E Blanco, et al. Continuous integration for PLC-based control systems. In: 17th int. conf. on acc. and large exp. physics control systems, ICALEPCS2019. 2019.

[44] Stolberg Sean. Enabling agile testing through continuous integration. In: 2009 agile conference. IEEE; 2009, p. 369–74.

[45] Talkhestani Behrang Ashtari, Jung Tobias, Lindemann Benjamin, Sahlab Nada, Jazdi Nasser, Schloegl Wolfgang, et al. An architecture of an intelligent digital twin in a cyber-physical production system. at-Automatisierungstechnik 2019;67(9):762–82.

[46] Talkhestani Behrang Ashtari, Braun Dominik, Schloegl Wolfgang, Weyrich Michael. Qualitative and quantitative evaluation of reconfiguring an automation system using Digital Twin. Proc CIRP 2020;93:268–73.

[47] Koziorek J, Gavlas A, Konecny J, Mikolajek M, Kraut R, Walder P. Automated control system design with model-based commissioning. Int J Circuits Syst Signal Process 2019;13(2019):6–12.

[48] Robert Szepesi, Horia Ciocârlie. An overview on software reconfiguration. Theory Appl Math Comput Sci 2011;1(1):74–9.

[49] Aksit Mehmet, Choukair Zièd. Dynamic, adaptive and reconfigurable systems overview and prospective vision. In: 23rd international conference on distributed computing systems workshops, 2003. proceedings. IEEE; 2003, p. 84–9.

[50] Li Jun, Dai Xianzhong, Meng Zhengda. Automatic reconfiguration of petri net controllers for reconfigurable manufacturing systems with an improved net rewriting system-based approach. IEEE Trans Autom Sci Eng 2008;6(1):156–67.

[51] Mohamad NR, Rahman AA Abdul, Mohamad BM Bali, Rahman MAA, Jafar FA, Muhamad MR, et al. Architecture of reconfigurable conveyor system in manufacturing system. J Adv Manuf Technol (JAMT) 2018;12(1 (2)):117–28.

[52] Niang Mohamed, Riera Bernard, Philippot Alexandre, Zaytoon Janan, Gellot François, Coupat Raphaël. A methodology for automatic generation, formal verification and implementation of safe PLC programs for power supply equipment of the electric lines of railway control systems. Comput Ind 2020;123:103328.

[53] Bortolini Marco, Accorsi Riccardo, Faccio Maurizio, Galizia Francesco Gabriele, Pilati Francesco. Toward a real-time reconfiguration of self-adaptive smart assembly systems. Procedia Manuf 2019;39:90–7.

[54] Mo Fan, Rehman Hamood Ur, Monetti Fabio Marco, Chaplin Jack C, Sanderson David, Popov Atanas, et al. A framework for manufacturing system reconfiguration and optimisation utilising digital twins and modular artificial intelligence. Robot Comput-Integr Manuf 2023;82:102524.

[55] Jbair Mohammad, Ahmad Bilal, Mus'ab H Ahmad, Vera Daniel, Harrison Robert, Ridler Tony. Automatic PLC code generation based on virtual engineering model. In: 2019 IEEE international conference on industrial cyber physical systems (ICPS). IEEE; 2019, p. 675–80.

[56] Floridi Luciano. AI as agency without intelligence: On chatgpt, large language models, and other generative models. Philosophy Technol 2023;36(1):15.

[57] Backstrom Lars, Leskovec Jure. Supervised random walks: predicting and recommending links in social networks. In: Proceedings of the fourth ACM international conference on web search and data mining. 2011, p. 635–44.

[58] Akoglu Leman, Tong Hanghang, Koutra Danai. Graph based anomaly detection and description: a survey. Data Min Knowl Discov 2015;29(3):626–88.

[59] Zhang Si, Zhou Dawei, Yildirim Mehmet Yigit, Alcorn Scott, He Jingrui, Davulcu Hasan, et al. Hidden: hierarchical dense subgraph detection with application to financial fraud detection. In: Proceedings of the 2017 SIAM international conference on data mining. SIAM; 2017, p. 570–8.

[60] Li Yaguang, Yu Rose, Shahabi Cyrus, Liu Yan. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. 2017, arXiv preprint arXiv:1707.01926.

[61] Monti Federico, Boscaini Davide, Masci Jonathan, Rodola Emanuele, Svoboda Jan, Bronstein Michael M. Geometric deep learning on graphs and manifolds using mixture model cnns. In: Proceedings of the IEEE conference on computer vision and pattern recognition. 2017, p. 5115–24.

[62] Cai Hongyun, Zheng Vincent W, Chang Kevin Chen-Chuan. A comprehensive survey of graph embedding: Problems, techniques, and applications. IEEE Trans Knowl Data Eng 2018;30(9):1616–37.

[63] Zhou Jie, Cui Ganqu, Hu Shengding, Zhang Zhengyan, Yang Cheng, Liu Zhiyuan, et al. Graph neural networks: A review of methods and applications. AI Open 2020;1:57–81.

[64] Kipf Thomas N, Welling Max. Semi-supervised classification with graph convolutional networks. 2016, arXiv preprint arXiv:1609.02907.

[65] Hamilton Will, Ying Zhitao, Leskovec Jure. Inductive representation learning on large graphs. In: Advances in neural information processing systems. Vol. 30, 2017.

[66] Hermann Mario, Pentek Tobias, Otto Boris. Design principles for industrie 4.0 scenarios. In: 2016 49th hawaii international conference on system sciences (HICSS). IEEE; 2016, p. 3928–37.

[67] Hoc Jean-Michel. From human–machine interaction to human–machine cooperation. Ergonomics 2000;43(7):833–43.

[68] Xu Xun, Lu Yuqian, Vogel-Heuser Birgit, Wang Lihui. Industry 4.0 and Industry 5.0—Inception, conception and perception. J Manuf Syst 2021;61:530–5.

[69] Lu Yuqian, Zheng Hao, Chand Saahil, Xia Wanqing, Liu Zengkun, Xu Xun, et al. Outlook on human-centric manufacturing towards Industry 5.0. J Manuf Syst 2022;62:612–27.

[70] Michalos George, Makris Sotiris, Papakostas Nikolaos, Mourtzis Dimitris, Chryssolouris George. Automotive assembly technologies review: challenges and outlook for a flexible and adaptive approach. CIRP J Manufact Sci Technol 2010;2(2):81–91.

[71] David HJJOEP. Why are there still so many jobs? The history and future of workplace automation. J Econ Perspect 2015;29(3):3–30.

[72] Park Hong-Seok, Tran Ngoc-Hien. An autonomous manufacturing system for adapting to disturbances. Int J Adv Manuf Technol 2011;56(9):1159–65.

[73] Vogel-Heuser Birgit, Fay Alexander, Schaefer Ina, Tichy Matthias. Evolution of software in automated production systems: Challenges and research directions. J Syst Softw 2015;110:54–84.

[74] Schneider Georg Ferdinand, Wicaksono Hendro, Ovtcharova Jivka. Virtual engineering of cyber-physical automation systems: The case of control logic. Adv Eng Inform 2019;39:127–43.

[75] Fosić I, Šolić K. Graph database approach for data storing, presentation and manipulation. In: 2019 42nd international convention on information and communication technology, electronics and microelectronics (MIPRO). IEEE; 2019, p. 1548–52.

[76] Chen Xiaojun, Jia Shengbin, Xiang Yang. A review: Knowledge reasoning over knowledge graph. Expert Syst Appl 2020;141:112948.

[77] Xiong Chenyan, Power Russell, Callan Jamie. Explicit semantic ranking for academic search via knowledge graph embedding. In: Proceedings of the 26th international conference on world wide web. 2017, p. 1271–9.

[78] Simros Markus, Wollschlaeger Martin, Theurich Stefan. Programming embedded devices in IEC 61131-languages with industrial PLC tools using PLCopen XML. In: CONTROLO'2012. 2012.

[79] Lechler Tobias, Fischer Eva, Metzner Maximilian, Mayr Andreas, Franke Jörg. Virtual Commissioning–Scientific review and exploratory use cases in advanced production systems. Proc CIRP 2019;81:1125–30.

[80] Salih Husam, Abdelwahab Hammam, Abdallah Areej. Automation design for a syrup production line using Siemens PLC S7-1200 and TIA Portal software. In: 2017 international conference on communication, control, computing and electronics engineering (ICCCCEE). IEEE; 2017, p. 1–5.

[81] Salari Mikael Ebrahimi, Paul Enoiu Eduard, Afzal Wasif, Seceleanu Cristina. Choosing a test automation framework for programmable logic controllers in CODESYS development environment. In: 2022 IEEE international conference on software testing, verification and validation workshops (ICSTW). 2022, p. 277–84.

[82] Langlois Kevin, van der Hoeven Tom, Cianca David Rodriguez, Verstraten Tom, Bacek Tomislav, Convens Bryan, et al. Ethercat tutorial: An introduction for real-time hardware communication on windows [tutorial]. IEEE Robot Autom Mag 2018;25(1):22–122.

[83] Jayaweera Nirosh, Webb Phil. Metrology-assisted robotic processing of aerospace applications. Int J Comput Integr Manuf 2010;23(3):283–96.

[84] Nicksch C, Kluge-Wilkes A, Huber M, Schmitt RH. Global Reference System for factory-wide integration of metrology enabling flexible automation in aeroplane assembly–requirements, concept and suitable technologies. Procedia Manuf 2020;52:89–94.

[85] Morgan Jeff, Halton Mark, Qiao Yuansong, Breslin John G. Industry 4.0 smart reconfigurable manufacturing machines. J Manuf Syst 2021;59:481–506.

[86] Bortolini Marco, Galizia Francesco Gabriele, Mora Cristina. Reconfigurable manufacturing systems: Literature review and research trend. J Manuf Syst 2018;49:93–106.

[87] Introduction of the Omnifactory, https://www.omnifactory.co.uk/.

[88] Waikhom Lilapati, Patgiri Ripon. An empirical investigation on BigGraph using deep learning. In: Advances in computers. Vol. 128, Elsevier; 2023, p. 107–33.

[89] Pezoa Felipe, Reutter Juan L, Suarez Fernando, Ugarte Martín, Vrgoč Domagoj. Foundations of JSON schema. In: Proceedings of the 25th international conference on world wide web. 2016, p. 263–73.

[90] Givehchi Mohammad, Ng Amos, Wang Lihui. Evolutionary optimization of robotic assembly operation sequencing with collision-free paths. J Manuf Syst 2011;30(4):196–203.

[91] Tomaszuk Dominik, Szeremeta Łukasz. The molecular entities in linked data dataset. Data in Brief 2020;31:105757.

[92] Jaskó Szilárd, Skrop Adrienn, Holczinger Tibor, Chován Tibor, Abonyi János. Development of manufacturing execution systems in accordance with Industry 4.0 requirements: A review of standard-and ontology-based methodologies and tools. Comput Ind 2020;123:103300.

[93] Kussul Nataliia, Lavreniuk Mykola, Skakun Sergii, Shelestov Andrii. Deep learning classification of land cover and crop types using remote sensing data. IEEE Geosci Remote Sens Lett 2017;14(5):778–82.