Automatic welding-robot programming based on product-process-resource models

Sarivan, Ioan-Matei; Madsen, Ole; Wæhrens, Brian Vejrum

Published in:
International Journal of Advanced Manufacturing Technology

DOI (link to publication from Publisher):
10.1007/s00170-024-13409-x

Publication date:
2024

Document Version
Publisher's PDF, also known as Version of record

Citation for published version (APA):
Sarivan, I.-M., Madsen, O., & Wæhrens, B. V. (2024). Automatic welding-robot programming based on product-process-resource models. *International Journal of Advanced Manufacturing Technology*, *132*(3-4), 1931-1950. https://doi.org/10.1007/s00170-024-13409-x

The International Journal of Advanced Manufacturing Technology
https://doi.org/10.1007/s00170-024-13409-x

ORIGINAL ARTICLE

Automatic welding-robot programming based on product-process-resource models

Ioan-Matei Sarivan[1] · Ole Madsen[1] · Brian Vejrum Wæhrens[1]

Received: 11 September 2023 / Accepted: 9 March 2024

Abstract
This paper describes a novel end-to-end approach for automatic welding-robot programming based on a product-process-resource (PPR) model, for one-of-a-kind manufacturing systems. Traditionally, the information needed to program a welding robot is processed and transferred along the manufacturing organisation's value chain by using several stand-alone digital systems which require extensive human input and high skill to operate. A PPR model is proposed through this research as a platform for storing and processing the necessary information along the value chain seamlessly. Unlike existing approaches which make use of complex algorithms to automatically identify the weldment seams, the approach suggested in this research makes use of information already digitalised by design engineers under the form of ISO 2553:2019 compliant weldment annotations. Hence, the PPR model contains the weldment annotations; it enables the automatic programming of welding robots and reduces human input down to a few minutes only. The applicability in manufacturing of the theoretical concept is demonstrated through technical implementations tested in the laboratory and on the value chain of an engineering-to-order (ETO) industrial partner involved in the metal fabrication industry. The experiments were conducted by creating several products using the proposed artefact. Experiments show that automatic programming of welding robots can be achieved using PPR models. The conducted experiments showed a reduction of about 80% in human input measured in terms of time, when using the proposed solution. The reduction of the human input can free up skilled labour resource which ETO SMEs can reallocate to other tasks.

Keywords Welding robots · Automatic robot programming · Product model · End-to-end integration

1 Introduction

In this paper, the theoretical concept of a product-process-resource (PPR) model for automatic welding-robot programming is proposed and then its potential to be adopted by practitioners is tested through practical experiments in the laboratory and at a partner manufacturing company. This concept lays the basis of organising the PPR [1] data required to program a welding robot in a centralised manner which facilitates the fast and automatic programming of welding robots proving their high utilisation rate in high-mix low-volume production environments to be feasible. Traditionally, the computer-aided design (CAD) environment is disconnected from the offline programming (OLP) environment for welding robots. This discontinuity within the digital value chain of the company requires redundant human input to re-digitalise data which is often already present in CAD models but lost upon transfer in the OLP environment. The proposed PPR model enables the seamless integration of the multiple operations necessary to be conducted before obtaining a functional robot-welding program, by making use of the CAD system's compliance with the ISO 2553 standard. Thus, an end-to-end digital pipeline can be put in place to automatically program welding robots without the need of highly skilled labour, resulting in lower production cost and dramatically faster lead time, without affecting the flexibility associated with a one-of-a-kind manufacturing system.

Enterprises located in high-cost environments, e.g. Denmark, face the challenge of an ever-increasing competition from low-cost environments [2]. Furthermore, the Danish metal fabrication industry is facing a shortage of skilled welders [3]. This shortage compels enterprises in

✉ Ioan-Matei Sarivan
ioanms@mp.aau.dk

[1] Department of Materials and Production, Aalborg University, Fibigerstræde 16, 9220 Aalborg Øst, Denmark

Published online: 25 March 2024

high-cost environments to rely on automation, which is highly challenging in one-of-a-kind manufacturing setups. With every product being unique, companies with one-of-a-kind manufacturing systems must invest heavily in engineering resources to create new product designs and to reprogram the welding robots every time a new order is received. Similarly with the situation of skilled welders, robotics engineers are expensive too. Therefore, labour cost is one of main challenges which ETO manufacturing enterprises are facing day by day. By conducting the research presented in this paper, it is desired to investigate a novel and effective tool which ETO manufacturing enterprises can use to reduce the manual inputs needed from robotics engineers to program welding robots. The time which is normally used by the robotics engineers to program the robots can then be reallocated to other tasks, thus reducing the cost of programming a welding robot with manual inputs.

Programming of welding robots is a process involving advanced competencies and time resources. Bejlegaard et al. and Tolio et al. show however in their studies that digital technologies tailored to fit the needs of the ETO small-medium-enterprises (SME) can reduce many of the trade-offs which concern these companies, such as flexibility versus efficiency [4, 5]. The artefact which is described in Section 4 of this paper is meant to drastically reduce the time and the competencies level required to reprogram welding robots, while at the same time facilitating lower production cost, faster lead time and increased utilisation of welding robots when compared with traditional, manual programming methods. It is showed through the practical experiments which were setup to demonstrate the potential of the proposed solution that a reduction of 80% in human input, measured by time, can be achieved when using the proposed solution.

2 Theoretical background and related work

To program welding robots, a thorough understanding of the product, the manufacturing process and the manufacturing system's capabilities are required. In ETO value chains, engineering resources are used to either create new products based on customer's requirements or adapt existing product designs to satisfy the customer's demand. A literature study was conducted on existing literature in the domain of product-process-resource models and how product-process-resource data can be structured in such manner that it allows fast redesign based on the customer's requirements and automatic programming of welding robots. The findings in the fields of PPR models and methods for automatic programming of welding robots are presented in the following.

2.1 PPR models

Integrating the customer's requirements into the product design is a matter of high importance when it comes to ETO supply chains and this can put extraordinary strain on the delivery dependability of products, especially for highly individualised products, where the customer is involved in the product design and engineering process [6]. The scalability and the personalisation of the product's architecture in the most efficient manner possible is a topic investigated extensively in the literature [6–8]. Tan et al. propose an open product architecture design, which makes use of several optimisation criteria to come up with the best solution to satisfy both the customer's requirements and the manufacturing company's targets for production cost. By using genetic algorithm solvers, the optimal attribute selection for the product is provided [8]. Customers can interact with the product architecture through product configuration systems, which facilitates customer-centric product development. By using the open architecture product platform conceptualised by Zheng et al., customers can either choose a configure-to-order approach on their order or an engineer-to-order approach to have access to the full suite of engineering services provided by the company [6]. A similar approach is investigated by Bejlegaard et al. where the case company provides emerging customers access to a focused product portfolio, in a made-to-order regime, while frequent customers are given access to innovative products in an ETO regime [4].

To program welding robots, information is required concerning the product design, the process and the resources needed to perform the weldments [9]. The systematic overview on product-process-resource information is also known as the "PPR perspective" and it is formalised in existing literature through PPR models [10–12]. PPR models include in their scope details about the resource capability, or skill required to conduct the manufacturing processes necessary to product an item. Brecher et al. and Ferrer et al. use PPR models to particularly formalise the process required and the skills needed to sustain assembly operations. The information required to set up the assembly process and the resource dimensions is extracted from the CAD through means of ontology-tailored meta models or semantic web rule language [13, 14]. A similar approach is taken by Ahmed et al. by using a PPR model in their study case for automotive engine assembly lines. The PPR model is queried for information in two stages, first to determine the capabilities of the available resources given the product requirements, and then to align the resource capabilities with the proper assembly process parameters [1]. As Ahmed et al. point out, the PPR model has the potential to increase engineering productivity and bring together information and knowledge under a single knowledge base. The same view on the PPR concept in shared by Winkler et al. who use PPR to trace the product requirements along the value chain and facilitate the propagation of information across the product-process and resource domains which form an engineering asset network [15]. The implementation of PPR view actively contributes at the increase of the adaptability of the production systems and allows for faster new product insertion with reduced human intervention [16, 17].

While the work mentioned above showcases the benefits of using the PPR perspective on assembly lines, no substantial literature and empirical research was found on how PPR models can be used for welding applications, specifically for automatic programming of welding robots in one-of-a-kind manufacturing systems. Therefore, the approach presented in this paper aims towards defining a PPR model, using the PPR perspective to enable automatic programming of welding robots and fast redesign of products, all within an end-to-end integrated information flow. The environment in which product data can be manipulated by the product engineers is the CAD software tool. A deep dive into existing methods on how automatic programming of welding robots can be obtained with the data existing in CAD models is given in the following.

2.2 CAD-based automatic programming of welding robots

Welding is one of the most exploited applications for industrial robotics [18] and the programming of welding robots is a well-researched field that has been in the attention of both researchers and practitioners for several decades. Current research in this field is focused on the reduction of time, skills and overall costs involved in programming and deploying a welding-robot program. This is especially important for one-of-a-kind production setups where welding robots must be reprogrammed every time a new customer order is received. There are three major approaches towards automatic programming of welding robots [19]: CAD-based programming, sensor-based programming and hybrid programming.

As mentioned in the previous subsection, the main environment where product data is manipulated is the CAD model. Therefore, the focus of the manuscript is placed on CAD-based programming methods only. The computer-aided design (CAD) system lays at the core of the engineering-to-order (ETO) supply chains facilitating the creation and the transformation of the product's design by design engineers [20]. Methods for automatic programming of welding robots using CAD models are at the core of intelligent weldment systems (IWS) [21]. The CAD model is needed within offline programming systems (OLP) for collision detection and avoidance algorithms [22–26] which allow automatic robot programming (ARP) [27, 28]. However, this is only possible if information about the locations and the geometry of the weldments exists. Traditionally, this information is manually inserted by the robotics engineers using various human machine interfaces (HMI) either readily available in the OLP system or custom made to make the programming process as intuitive as possible and independent of programming languages [29]. The information that is manually inserted in the OLP is gathered from the technical drafts which are automatically generated by modern CAD systems under the form of 2D technical drawings annotated with the positions of the weldments as stipulated by ISO Standard 2553:2019.

Recent developments in the field of ARP for welding applications have been focused on the automation of weldment-related information transfer from the CAD environment directly to the OLP environment. Feature recognition algorithms are used to process the CAD model stored in STEP (Standard for the Exchange of Product Data ISO 10303) files [30]. Xuan and Ngoc make use of the hierarchical structure of the STEP format to access the set of faces and edges which make up the geometry of the product designed in CAD [31]. The relationships between the faces and the edges are evaluated and the coordinates of the weldments are determined. Kuss et al. exploit a similar approach augmented by rules, which delimit the recognition procedure to fillet weldments only [32]. By using the surface normal of the extracted faces, Kuss et al. can determine the angle between the surfaces which are to be welded. Fang et al. also make use of data extracted from the CAD model to discretise the weldment by using methods available in the SolidWorks CAD system [33].

Modern CAD systems allow product design engineers to model the geometry of the product, perform engineering analysis, review the design and finally create the technical drafts of the product. The technical draft contains 2D drawings annotated with the positions of the weldments [34]. Weldments are considered rigid engineering connections in CAD assemblies and by using modern CAD tools; weldment information can be overlaid on CAD models as directed by ISO Standard 2553:2019 [35–39]. This is also supported to some extent in the second edition of STEP AP242 and an implementation for extracting information about the weldments directly from the CAD model without feature recognition algorithms is given by Mohammed et al. [40]. Mohammed et al. show that process information like current, voltage and torch speed can be additionally overlaid on the STEP file format by intervening directly on the extensible markup language (XML) structure of the STEP file.

To obtain reliable information for ARP purposes, Tran et al. use a composed implementation using information obtained both from STEP files with ISO Standard

2553:2019 annotations, and feature recognition tools [41]. The annotations present in the STEP files are created by Tran et al. using the tools available in the Siemens NX CAD system [39]. The weldment-related information obtained by Tran et al. can be further used to automatically generate a robot program with collision-free movements between and across the weldments [42].

Larkin et al. classify the methods for extracting weldment information from CAD models using the CAD tools available, as either automatic, using the feature recognition algorithms, or semi-automatic, by using manual inputs to create the necessary annotations on the CAD model which define weldments [43]. It is to be noted, however, that weldment annotations fall under the ISO Standard 2553:2019, which stipulates the proper drafting of products with rigid engineering connection held together by weldments. And as mentioned before, there are several CAD systems commercially available which are compliant with this standard. Therefore, the proper annotation of weldments falls under the responsibility of the product design engineer. Moreover, as shown in the related work presented above, information encapsulated in both the product and process domain can be stored in modern CAD file formats. While research similar with the one made by Tran et al. [41, 42] and by Mohammed et al. [40] offer examples on how data can be extracted directly from CAD models to program welding robots, a research gap is identified in the lack of a general approach for extracting relevant information from CAD systems compliant with the ISO 2553:2019 standard. The approach presented in this paper and the proposed theoretical concept addresses CAD software compliant with the ISO 2553:2019 standard independent of the supplier.

It is concluded from the literature study made that the ETO case company may benefit from a technical solution which allows fast adjustment of product designs without the need for an iterative process, ideally where the customer can have more control over the product's design, but without the assistance of the engineers. This is highlighted in Table 1 by directly comparing the methods mentioned above with the method proposed in this research. The main comparison dimension used is the amount of human input necessary to obtain a new robot program after having modified the CAD model to obtain a new product variant. Currently, when a company is using CAD software and OLP software, the data exchange between these two environments is made manually and is disconnected. The digital integration of CAD and OLP is expected that the information flow will become streamlined, and reduce the required human input needed every time a new product variant is created.

3 Product-process-resource model

By linking the findings in the literature survey on the PPR perspective and automatic programming of welding robots based on CAD models, a PPR model for automatic programming of welding robots is hereby proposed. The product design engineers use CAD tools to document information about the product beyond the geometry of the product. The information can be overlaid on CAD models using modern CAD tools, can be used to set up the production processes and ensure that the process can be conducted using the available resources.

3.1 Theoretical foundations

Ahmad et al. offer a decomposition of the PPR model [1] used for automatic assembly applications as follows:

- **The product domain** contains all the available information about the product's characteristics, the product's family and variations. The product domain is often fully encompassed in the CAD model of the product. The product can be an assembly of multiple components which share various engineering connections or liaisons (fixed, cylindrical, prismatic or rigid).
- **The process domain** encapsulates the knowledge and information necessary to set up the operations required to manufacture the product. As described by Ahmad et al. [1], a production operation is realised through several processes which are sustained using resources. Some processes require certain skills and resources to be conducted.
- **The resource domain** concerns the set of physical and non-physical entities involved in supporting the production process. The physical resources may be manufacturing tools and various consumables, while non-physical resources can be the tacit or documented knowledge regarding how the production process can be set up.

The product model concept proposed is shown in the diagram displayed in Fig. 1. The Unified Modelling Language (UML) was selected to describe the PPR model proposed. UML provides a general and flexible modelling language using specific graphical elements which are widely understood in the software development community [44]. UML is fit for the development of a theoretical concept that has its foundation in the functionalities of existing CAD software tools. The general nature of the language allows for applicability across the whole domain of CAD software compliant with the ISO 2553:2019 standard. The

Table 1 A summary of existing method for programming of industrial robots. The table is used to highlight advantages and disadvantages when compared to the solution proposed in this research

Solution	Is executed by	Amount of human input needed	Advantages	Disadvantages
Online manual programming using the robot's HMI	Robotics engineers and welders	Very high	Highly flexible method for programming of welding robots. The program is accurately fitted according to the work piece's geometry	The programming process requires for the production line to be stopped and it is very time consuming
Offline manual programming using the OLP	Robotics engineers	High	Highly flexible method for programming of robots	The programming process required a high amount of human input and high skill in using OLP systems
Automatic programming using feature extraction tools from CAD models	Automatically with input needed from robotics engineers	Medium	The method reduces the necessary human input to program the robots	This method treats every intersection of surfaces within the CAD model as possible locations for a weldment seam. This requires further input from engineers to properly adjust the program
Automatic programming using weldment annotations from CAD models	Automatically	Medium–low	The method drastically reduces human input for programming the robots	When modifying the CAD model to obtain a new product variant, it is necessary to adjust the weldment annotations manually
Automatic programming using PPR models (method proposed in this paper)	Automatically	Low	The method drastically reduces human input and it the weldment annotations are automatically adapted based on the product's geometry and information associated with the manufacturing process	The method requires a comparatively high initial investment in order to digitalise all the necessary information for a process–product-resource model

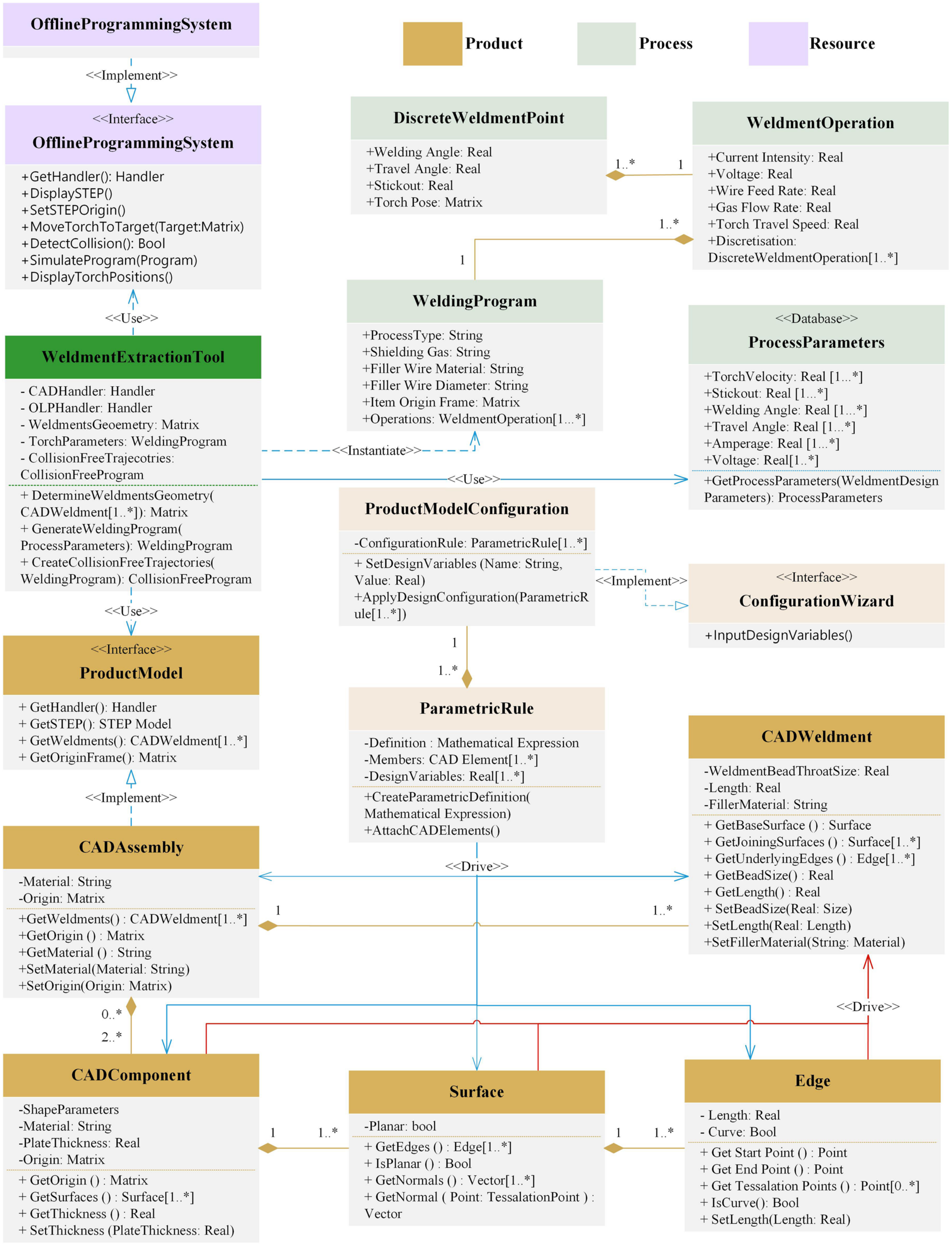

OfflineProgrammingSystem
<<Implement>>
<<Interface>>
OfflineProgrammingSystem
+GetHandler(): Handler
+DisplaySTEP()
+SetSTEPOrigin()
+MoveTorchToTarget(Target:Matrix)
+DetectCollision(): Bool
+SimulateProgram(Program)
+DisplayTorchPositions()
<<Use>>
WeldmentExtractionTool
- CADHandler: Handler
- OLPHandler: Handler
- WeldmentsGeoemetry: Matrix
- TorchParameters: WeldingProgram
- CollisionFreeTrajecotries: CollisionFreeProgram
+ DetermineWeldmentsGeometry(CADWeldment[1..*]): Matrix
+ GenerateWeldingProgram(ProcessParameters): WeldingProgram
+ CreateCollisionFreeTrajectories(WeldingProgram): CollisionFreeProgram
<<Use>>
<<Interface>>
ProductModel
+ GetHandler(): Handler
+ GetSTEP(): STEP Model
+ GetWeldments(): CADWeldment[1..*]
+ GetOriginFrame(): Matrix
<<Implement>>
CADAssembly
-Material: String
-Origin: Matrix
+GetWeldments() : CADWeldment[1..*]
+GetOrigin () : Matrix
+GetMaterial () : String
+SetMaterial(Material: String)
+SetOrigin(Origin: Matrix)
0..*
2..*
CADComponent
-ShapeParameters
-Material: String
-PlateThickness: Real
-Origin: Matrix
+ GetOrigin () : Matrix
+ GetSurfaces () : Surface[1..*]
+ GetThickness () : Real
+ SetThickness (PlateThickness: Real)
Product
Process
Resource
DiscreteWeldmentPoint
+Welding Angle: Real
+Travel Angle: Real
+Stickout: Real
+Torch Pose: Matrix
1..*
1
WeldmentOperation
+Current Intensity: Real
+Voltage: Real
+Wire Feed Rate: Real
+Gas Flow Rate: Real
+Torch Travel Speed: Real
+Discretisation: DiscreteWeldmentOperation[1..*]
1..*
1
WeldingProgram
+ProcessType: String
+Shielding Gas: String
+Filler Wire Material: String
+Filler Wire Diameter: String
+Item Origin Frame: Matrix
+Operations: WeldmentOperation[1...*]
<<Database>>
ProcessParameters
+TorchVelocity: Real [1...*]
+Stickout: Real [1...*]
+Welding Angle: Real [1...*]
+Travel Angle: Real [1...*]
+Amperage: Real [1...*]
+Voltage: Real[1..*]
+GetProcessParameters(WeldmentDesign Parameters): ProcessParameters
<<Instantiate>>
<<Use>>
ProductModelConfiguration
-ConfigurationRule: ParametricRule[1..*]
+ SetDesignVariables (Name: String, Value: Real)
+ApplyDesignConfiguration(ParametricRule[1..*])
<<Implement>>
<<Interface>>
ConfigurationWizard
+InputDesignVariables()
1
1..*
ParametricRule
-Definition : Mathematical Expression
-Members: CAD Element[1..*]
-DesignVariables: Real[1..*]
+CreateParametricDefinition(Mathematical Expression)
+AttachCADElements()
CADWeldment
-WeldmentBeadThroatSize: Real
-Length: Real
-FillerMaterial: String
+ GetBaseSurface () : Surface
+ GetJoiningSurfaces () : Surface[1..*]
+ GetUnderlyingEdges () : Edge[1..*]
+ GetBeadSize() : Real
+ GetLength() : Real
+ SetBeadSize(Real: Size)
+SetLength(Real: Length)
+SetFillerMaterial(String: Material)
<<Drive>>
1
1..*
<<Drive>>
Surface
-Planar: bool
+ GetEdges () : Edge[1..*]
+ IsPlanar () : Bool
+ GetNormals () : Vector[1..*]
+ GetNormal (Point: TessalationPoint) : Vector
1
1..*
1
1..*
Edge
- Length: Real
- Curve: Bool
+ Get Start Point () : Point
+ Get End Point () : Point
+ Get Tessalation Points () : Point[0..*]
+ IsCurve(): Bool
+ SetLength(Length: Real)

Fig. 1 UML diagram describing the PPR model for automatic programming of welding robots. The UML version 2.0 is used as described by Eriksson et al. [44] and Scott [47]

flexible nature of UML allows the presented model to be adapted further, as the CAD tools will evolve in the future, or even modified to be compatible with custom-made or open-source CAD software which are not compliant with ISO 2553:2019 but have available tools for weldment annotations.

The major suppliers of CAD systems are extending the capabilities of the tools they supply, to encapsulate more product data [1]. The implementation of rule-based design methods enables the digitalisation of the product knowledge [45] and parametric CAD modelling which enables the reusability, the adaptability and the configurability of a CAD model [46], making the product engineering stage faster and more cost-effective. These capabilities are made use of to build parametric rules and product configurations in the proposed PPR model. The changes in the product domain directly influence the way the manufacturing process is set up and the way the available resources can sustain the manufacturing process [1].

The classification of input parameters for programming of welding robot given by Lauridsen [9] is used to formalise the information contained and processed by the proposed PPR model:

1. **Design parameters**: the product parameters which are related to the workpiece to be welded. These parameters include the geometry of the workpiece, the thickness of the plates joined through welding and the material specification. These parameters are varied by varying the design of the product, as requested by the customer.
2. **Motion parameters**: the process parameters which are specific to the manipulator of the welding torch (welding robot). The motion parameters include the position and the orientation of the torch relative to the workpiece or welding seam which influence the length of the exposed wire at the tip of the welding torch (stickout), the velocity of the torch and the weaving of the torch. Depending on the design parameters, the motion parameters may include certain offsets such as multi-pass offsets which allow an incremental increase of the distance between the torch and the work piece as the weld bead increases in thickness.
3. **Arc parameters**: the process parameters which are specific to the welding apparatus and are independent of the overall geometrical design of the product but are used to fulfil the quality requirements of the weldment, e.g. weldment throat size. These parameters are the arc current and voltage. Modern welding apparatus allows for automatic control of wire feed rate and shielding gas flow depending on the preset material specification and joining plate thicknesses. Other parameters which are important before the welding process can commence and which directly influences the arc parameters are the shielding gas and the filler wire material.

3.2 Proposed PPR model for automatic programming of welding robots

In the proposed PPR model, represented in Fig. 1, the PPR perspective is made apparent by using different colours to represent the classes which constitute the product, process and resource-related elements. The classes represented with dark and light orange are related to the *product* component of the model. The green-coloured class diagrams are related to the *process* component of the model, and the purple-coloured class diagram is related to the *resource* part of the model. The main contribution brought to the ARP field is the process-related component of the model.

Product: The product element of the proposed PPR model has its roots in the generic representation of a product in a CAD system. As observed in Fig. 1, a **CAD component** class is composed of one or more **Surface** instances. A **Surface** class can be composed of one or more **Edge** instances. An **Edge** class is constituted of several tessellation points which can be accessed through the *GetTessellationPoints()* accessor. Depending on the shape of the **Edge** object, it can have two (straight edge) or more tessellation points (curved edge). The **Surface** class contains the normal vector associated with a surface, or several normal vectors, if the surface is curved. The coordinates of the tessellation points and the components of the normal vectors are expressed in relation to the origin frame of the **CADComponent** instance. In the context of this paper, the **CADComponent** is a metal plate that must be welded; the attributes the **CADComponent** class has are the shape parameters, the material (metallic alloy) of the component, the thickness of the plate and the transformation associated with the origin of the product when it composes a **CADAssembly.**

A CAD model can be driven by several **ParametricRule** objects instantiated by the product design engineers. These rules compose a **ProductModelConfiguration** object. This facilitates the accessible configuration of a product model by inserting the design parameters through the interface realised by the **ConfigurationWizard**. A **ParametricRule** contains a mathematical expression which drives various elements inside the model. For example, depending on the thickness of the plates, the throat size of the weldment can be automatically adjusted based on a mathematical rule set by the design engineer, based on specifications. Another example can be a **ParametricRule** object attached to a *joining* **Surface** object and a *base* **Surface** object that controls the length of the two surfaces in the same direction can directly drive the length of the intersection **Edge** and therefore the length of the **Weldment** object attached to that **Edge**. This simplifies the design process from having to modify two parameters to having to modify only one. The marker "<<Drives>>" suggests that the entity towards which the arrow is pointing it is directly controlled by the entity from which the arrow stems from. The marker “1…*” suggests that a **ProductModelConfiguration** object is composed of at least one **ParametricRule** and there is no upper limit on how many **ParametricRule** objects can exist.

A CAD model can be driven by several **ParametricRule** objects instantiated by the product design engineers. These rules compose a **ProductModelConfiguration** object. This facilitates the accessible configuration of a product model by inserting the design parameters through the interface realised by the **ConfigurationWizard**. A **ParametricRule** contains a mathematical expression which drives various elements inside the model. For example, depending on the thickness of the plates, the throat size of the weldment can be automatically adjusted based on a mathematical rule set by the design engineer, based on specifications. Another example can be a **ParametricRule** object attached to a *joining* **Surface** object and a *base* **Surface** object that controls the length of the two surfaces in the same direction can directly drive the length of the intersection **Edge** and therefore the length of the **Weldment** object attached to that **Edge**. This simplifies the design process from having to modify two parameters to having to modify only one. The marker “<<Drives>>” suggests that the entity towards which the arrow is pointing it is directly controlled by the entity from which the arrow stems from. The marker “1…*” suggests that a **ProductModelConfiguration** object is composed of at least one **ParametricRule** and there is no upper limit on how many **ParametricRule** objects can exist.

The interface towards the **CADAssembly** object is realised by the **ProductModel** class. The **ProductModel** interface can be implemented by using the API (application programming interface) available with the CAD system. The interface clearly specifies the methods which must be made available through the API, for the CAD system to correctly adhere to the proposed PPR model [44]. Through the interface, accessors are made available for the design parameters of the weldment. The **CADAssembly** object is composed of at least two **CADComponent** objects. The “0…*” marker suggests that a **CADComponent** object can also exist independently of a **CADAssembly**, although this case is irrelevant in this research.

Process The process component of the product model is the main contribution of this work as part of the proposed PPR model for automatic programming of welding robots. The process model is constituted of a **WeldmentExtractionTool** class, which acts as a bridge between the CAD environment and the OLP environment, by using the available interfaces implemented for these systems. The **WeldmentExtractionTool** directly connects to the **ProductModel** interface and makes use of the available accessors to extract the design parameters of the weldments. The **WeldmentExtractionTool** uses a **ProcessParameters** database, which contains the digitalised tacit knowledge of the engineers under the form of a lookup table. For example, depending on the designed throat size of the weldment and the plate thickness on which the weldment is attached, a certain set of process parameters (motion parameters and arc parameters) are used for the welding process. Some of the motion parameters contained in the database which are dependent on the design parameters are the welding angle $\boldsymbol{\alpha}$ and the travel angle $\boldsymbol{\beta}$, which are represented in Fig. 2, the torch travel velocity, and the stick-out, which is the distance between the welding torch and the welding seam. The arc parameters can be the voltage and the current for the weldment. Some process parameters, like the wire feed and the gas flow, may either be present in this database or be controlled by the welding apparatus.

The **WeldingProgram** class is instantiated by the **WeldmentExtractionTool** class upon call of the *ComputeTorchPositionParameters()* method. The **WeldingProgram** is composed of one or more **WeldmentOperation** objects which share the same process setup parameters, e.g. the materials of the consumables, the welding process type (MIG, TIG, etc.) and the origin frame in which the geometry of the weldments is expressed. These parameters remain constant across the whole welding process. A **WeldmentOperation** object is discretised in one or more **DiscreteWeldmentOperation**

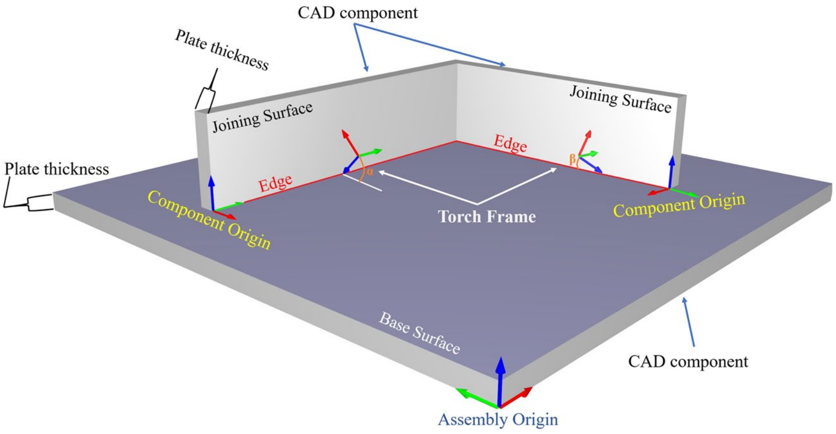


Fig. 2 Graphical representation of a CADAssembly object compose of three CADComponent objects. The CADWeldment objects are attached to the Edge objects between the joining surface of the two components on top and the base surface of the base surface on the bottom. The welding angle $\boldsymbol{\alpha}$ and the travel angle $\boldsymbol{\beta}$ are also represented in this figure

objects, as described by Sarivan et al. [48]. Each discrete weldment operation which composes a **WeldmentOperation** object shares the same process parameters, e.g. the torch travel speed, the wire stickout and so on (the attributes of the **WeldmentOperation** class). Every **DiscreteWeldmentOperation** object is instantiated with its own torch position parameters.

After the **WeldingProgram** is instantiated, collision-free trajectories must be generated for the welding torch in order for the welding process to be successfully supported. This is made possible through the **OfflineProgrammingSystem** interface. The interface is made available by using the OLP's API.

Resource The resource component in the proposed PPR model is represented through the **OfflineProgrammingSystem** interface which offers access to the capabilities of the offline programming system which has available in it a digital twin of the robot-welding cell. These capabilities are the motion solver of the robot, which makes available the possible configurations in which a robotic manipulator can reach a point, and the collision detection functionality. By using these capabilities, it is ensured that the available resources on the manufacturing system can successfully support the welding process. The algorithms used to setup the welding process and the collision-free trajectories for the welding program are given in the next section.

3.3 Automatic programming based on product model

To facilitate the automatic programming of a welding robot based on a PPR model, the following prerequisites are necessary:

1. The creation of a CAD model with weldments annotations as stipulated by ISO 2553:2019 standard by the product design engineers (represented by the **ProductModel** class in Fig. 1).
2. The creation of a configuration through parametric rules which will be used to easily modify the CAD model without skilled intervention (represented by the **ProductModelConfiguration** class, in Fig. 1).
3. The digitalisation of the process parameters (motion and arc parameters) under the form of lookup tables depending on the design parameters of the product which will be organised in the **ProcessParameters** database.

With the prerequisites in place, the following steps must be followed to automatically program a welding robot using the proposed PPR model:

1. The design parameters are input to the **ConfigurationWizard** interface by the sales managers or the design engineers. A new instance of the **CADAssembly** class is created by using the **ParametricRule** objects which compose the configuration created by the product design engineers.
2. The composing elements of the **CADAssembly** instance are driven by the rules, including the weldment information, constituting as such the *Design Parameters* which are used to program the welding robot.
3. The geometry of the weldments is extracted from the **CADAssembly** instance through the **ProductModel** interface.
4. The geometry of the weldments is discretised and for each discrete state, motion and arc parameters are computed through the **WeldmentExtractionTool** class.
5. Collision-free trajectories for the robot are automatically computed using the functions available in the OLP and accessed through the **OfflineProgrammingSystem** interface.
6. The program is saved and ready to be validated through simulations and then deployed on the welding robot.

Algorithms are provided for steps 3–5 of the automatic welding-robot programming process. Each algorithm is associated with the methods available in the **WeldmentExtractionTool** class. A description of how the weldment geometry can be extracted from the product model is provided in Algorithm 1.

Algorithm 1 Implementation of "DetermineWeldmentsGeometry"

```
Input: $CADHandler \leftarrow ProductModel.GetHandler()$
Output: $WeldmentDefinitions$
  $WeldmentDefinitions \leftarrow List < Matrix < Real >>$
  $CADWeldments \leftarrow CADHandler.GetWeldments()$
  for each $CADWeldment \in CADWeldments$ do
      $Edges \leftarrow CADWeldment.GetUnderlayingEdges()$
      $JoiningSurfaces \leftarrow CADWeldment.GetJoiningSurfaces$
      $n \leftarrow 0$
      $BaseSurface \leftarrow CADWeldment.GetBaseSurface()$
      for each $Edge \in Edges$ do
          $n \leftarrow n + Edge.GetTesslationPoints().Count$
      end for
      $M \leftarrow [0..8, 0..n-1]; i \leftarrow 0$
      for each $Edge \in Edges$ & $Surface \in JoiningSurfaces$ do
          if $Edge.IsCurve()$ is true then
              for each $Point \in Edge.GetTessalationPoints()$ do
                  $M[i, 0..2] \leftarrow Point.GetCoordinates()[0..2]$
                  $Vector \leftarrow Surface.GetNormal(Point);$
                  $M[i, 3..5] \leftarrow Vector.GetComponents()[0..2]$
                  $Vector \leftarrow BaseSurface.GetNormal(Point);$
                  $M[i, 6..8] \leftarrow Vector.GetComponents()[0..2]$
                  $M[9] \leftarrow weldBead.GetThickness()$
                  $i \leftarrow i$
              end for
          else
              $Sp \leftarrow Edge.GetStartPoint(); Ep \leftarrow Edge.GetEndPoint()$
              $Vector_j \leftarrow Surface.GetNormals();$
              $Vector_b \leftarrow BaseSurface.GetNormals()$
              $M[i, 0..2] \leftarrow Sp.GetCoordinates()[0..2]$
              $M[i, 3..5] \leftarrow Vector_j[0..2]; M[i, 6..8] \leftarrow Vector_b[0..2]$
              $M[i+1, 0..2] \leftarrow Sp.GetCoordinates()[0..2]$
              $M[i+1, 3..5] \leftarrow Vector_j[0..2]; M[i+1, 6..8] \leftarrow Vector_b[0..2]$
              $M[9] \leftarrow CADWeldment.GetThickness()$
              $i \leftarrow i + 1$
          end if
      end for
      $WeldmentDefinitions.Add(M)$
  end for
```

Once the **ProductModel** instance is created, it is possible to query the list of weldments using the *GetWeldments()* method. The method is accessed by first creating a handler towards the CAD system. For each weldment, the **Edge** to which the weldment is attached is extracted, together with the *joining* and *base* surfaces. The number of **Edge** instances attached to a weldment must be the same as the number of *joining* **Surface** instances. The tessellation points defining the curve underlaying the **Edge** instance are extracted. The tessellation points are important for the discretisation of the weldment into prismatic quasi-stationary elements [48] and are part of the weldment definition. If the underlaying curve is curved and not straight, for each tessellation point, the coordinates are saved and the normal vector components of each *joining* **Surface** and *base* **Surface** is saved. The normal vector components are saved in order to determine the position and the orientation of the torch relative to the weldment seam. In case the underlaying curve of the **Edge** instance is straight, there is only one set of normal vectors of each **Surface** (*joining* and *base*). It is to be mentioned that there can exist only one *base* **Surface** instance, while it is allowed to have multiple *joining* **Surface** instances for any single **Weldment** instance.

To compute the motion parameters for the welding process, namely the torch position and orientation for each discrete weldment point afferent to each extracted tessellation point, the *ComputeTorchPositionParameters(ProcessParameters)* method is called. The method makes use of the **ProcessParameters** database to properly set the orientation of the welding torch relative to the weldment bead. The steps used by the method are presented in Algorithm 2. Horn's Absolute Orientation Method [48, 49] is used in Algorithm 2 when the *ComputeAbsoluteOrientation* method is called to change the reference frame in which the weldments' geometry is expressed.

Algorithm 2 Implementation of "ComputeTorchPositionParameters" interface

```
Input: $WeldmentsGeometry \leftarrow DetermineWeldmentsGeometry()$
Output: $WeldingProgram$
  $DiscreteWeldmentOperation \leftarrow List < Matrix < Real >>$
  $WeldmentOperation \leftarrow List < DiscreteWeldmentPoints >$
  $WeldingProgram \leftarrow List < WeldmentOperation >$
  for each $M \in WeldmentsGeometry$ do
    for each $i \in \{0...M.ColumnsNumber - 1\}$ do
      $c1 \leftarrow M.Column(i); c2 \leftarrow M.Column(i+1)$
      $WeldmentDimension \leftarrow c1[9]$
      $S \leftarrow ProcessParameters.WireStickOut(WeldmentDimension)$
      $\alpha \leftarrow ProcessParameters.WeldingAngle(WeldmentDimension)$
      $\beta \leftarrow ProcessParameters.TravelAngle(WeldmentDimension)$
      $rO \leftarrow \begin{bmatrix} c1[0],\ c2[0],\ c1[0]+\frac{(c1[3]+c1[6])}{2},\ c1[0]+\frac{(c2[3]+c2[6])}{2},\ c1[0]+c1[3],\ c2[0]+c2[6] \\ c1[1],\ c2[1],\ c1[2]+\frac{(c1[4]+c1[7])}{2},\ c1[1]+\frac{(c2[4]+c2[7])}{2},\ c1[1]+c1[4],\ c2[0]+c2[7] \\ c1[2],\ c2[2],\ c1[3]+\frac{(c1[5]+c1[8])}{2},\ c1[2]+\frac{(c2[5]+c2[8])}{2},\ c1[2]+c1[5],\ c2[0]+c2[8] \end{bmatrix}$
      $d \leftarrow \sqrt{(c1[0]-c2[0])^2) + (c1[1]-c2[1])^2) + (c1[2]-c2[2])^2)}$
      $rW \leftarrow \begin{bmatrix} 0,\ 0,\ 0.5,\ 0.5,\ 0,\ 1 \\ 0,\ d,\ 0,\ d,\ 0,\ d \\ 0,\ 0,\ 0.5,\ 0.5,\ 1,\ 0 \end{bmatrix}$
      ${}^{Origin}_{Weldment}R \leftarrow ComputeAbsoluteOrientation(rO, RW)$
      ${}^{Origin}_{Weldment}T \leftarrow \begin{bmatrix} {}^{O}_{W}R[0,0],\ {}^{O}_{W}R[0,1],\ {}^{O}_{W}R[0,2],\ \frac{c1[0]+c2[0]}{2} \\ {}^{O}_{W}R[1,0],\ {}^{O}_{W}R[1,1],\ {}^{O}_{W}R[1,2],\ \frac{c1[1]+c2[1]}{2} \\ {}^{O}_{W}R[2,0],\ {}^{O}_{W}R[2,1],\ {}^{O}_{W}R[2,2],\ \frac{c1[2]+c2[2]}{2} \\ 0 \quad 0 \quad 0 \quad 1 \end{bmatrix}$
      ${}^{Weldment}_{Torch}T \leftarrow \begin{bmatrix} cos(\alpha), & -sin(\alpha)*sin(\beta), & -cos(\beta)*sin(alpha), & S*sin(\alpha)*cos(\beta) \\ 0, & cos(\beta), & -sin(beta), & S*sin(\beta) \\ sin(\alpha), & -cos(\alpha)*sin(\beta), & -cos(\alpha)*cos(\beta), & S*cos(\alpha)*cos(\beta) \\ 0, & 0, & 0, & 1 \end{bmatrix}$
      ${}^{Origin}_{Torch}T \leftarrow {}^{Origin}_{Weldment}T * {}^{Weldment}_{Torch}T$
      $DiscreteWeldmentPoint \leftarrow {}^{Origin}_{Torch}T$
      $WeldmentOperation.Add(DiscreteWelmdentPoint)$
    end for
    $WeldmentProgram.Add(WeldmentOperation)$
  end for
```

The weldments geometry extracted in step 3 is used as input for Algorithm 2. For each weldment definition, the coordinates of each tessellation point and the afferent normal vectors are extracted from each column of the matrix M. This happens as described in step 4 and each discrete state is associated with a tessellation point. The first three entries of the column represent the position of the tessellation point, the next three entries represent the normal vector's components of the base surface, the next three components represent the normal vector's components of the joining surface and the last entry is the dimension of the weldment's throat in millimetres. The dimension of the weldment's throat is used as lookup parameter to extract the associated sickout **S** (distance between welding torch and weldment seam), welding angle of the torch **α** (angle between torch and the base surface), and travel angle **β** (found between the torch and the weld seam) using the lookup table in the **Process Parameters** database. The **α** and **β** angles are also represented in Fig. 2. By using the coordinates of each tessellation point and its afferent normal vectors, discrete weldment data is generated by using Horn's Absolute Orientation method [48, 49]. For each torch position determined using Horn's method, the **S**, **α** and **β** parameters are set and the necessary transformation is computed from the origin frame of the CAD model to the position of the weldment torch afferent to the respective discrete weldment point. The result of Algorithm 2 can be observed in Fig. 3.

With the torch position parameters computed, Algorithm 3 is used to generate collision-free trajectories for the welding robot. This is associated with step 5 in the proposed automatic programming of welding robots' procedure. Algorithm 3 takes as input all the torch position parameters computed in step 4. A handler towards the offline programming software is created in order to access the kinematics of the welding robot and test the reachability of the target point and to access the collision detection functionality of the OLP [22–26]. For each discrete point which composes each weldment, the end effector (the welding torch's tip) of the virtual welding robot is commanded to reach the target point. While collisions are detected, the torch is twisted around the axis of the welding wire both ways until a configuration is reached which is free of collisions. Algorithm 3 is offered as alternative in case the collision detection and automatic collision solver algorithms are not available in the OLP of choice; however, many of the modern OLPs do have such methods available [27, 28].

Algorithm 3 Implementation of CreateCollisionFreeTr ajection"method

```
Input: $WeldingProgram$
Output: $CollisionFreeTrajectories$
  $OLPHandler \leftarrow OLP.GetHandler()$
  $Trajectory \leftarrow List < Matrix < Real >>$
  $Trajectories \leftarrow List < Trajectory >$
  $Default\delta \leftarrow 5$
  for each $WeldmentOperation \in WeldmentProgram$ do
    for each ${}^{Origin}_{Torch}T \in WeldmentOperation$ do
      $TrajectoryTarget \leftarrow {}^{Origin}_{Torch}T.Pose()$
      $OLPHandler.MoveTorchToTarget(TrajectoryTarget)$
      $OffsetNo \leftarrow 0$
      $OffsetDirection \leftarrow 1$
      $TorchOffset \leftarrow I$                              ▷ Identity matrix
      while $OLPHandler.DetectCollision()$ do
        $OffsetNo \leftarrow (OffsetNo + 1)$
        $\delta \leftarrow Default\delta * OffsetNo * OffsetDirection$
        $TorchOffset \leftarrow \begin{bmatrix} \cos(\delta), & -\sin(\delta), & 0, & 0 \\ \sin(\delta), & \cos(\delta), & 0, & 0 \\ 0, & 0, & 1, & 0 \\ 0, & 0, & 0, & 1 \end{bmatrix}$
        $OLPHandler.MoveTorchToTarget({}^{Origin}_{Torch}T * TorchOffset)$
        if $OffsetNo == 7$ then
          $\delta \leftarrow Default\delta$
          $OffsetDirection \leftarrow -1$
          $Offsetno \leftarrow 1$
        end if
        if $OffsetNo == 8\ \&\ OffsetDirection == -1$ then
          Throw Exception:  No collision free solution was found!
        end if
      end while
      $Trajectory.Add({}^{Origin}_{Torch}T * TorchOffset)$
    end for
    $Trajectories.Add(Trajectory)$
  end for
```

3.4 Reflections on the proposed system

This section has provided a thorough description of the proposed PPR model concept for automatic programming of welding robots which facilitates automatic programming of welding robots without the intervention of robotics engineers in defining the weldment paths for the welding robots. The concept was developed based on findings and formalisations from literature, as presented in Subsection 2.1 to achieve a PPR integration addressed to welding applications. The PPR model is composed of a collection of data contained in a CAD system compliant with the ISO 2553:2019 standard and process data that can be digitalised in the **ProcessParameters** database specific to the company where the system is implemented. The proposed PPR model summarised in Fig. 1 extends the scope of extant research, which is applied on a specific CAD system, as shown in Subsection 2.2 [41, 42]. Algorithms were provided on how the data is processed to achieve a program for welding robots in an automatic manner. The proposed system is meant to drastically reduce manual input required to program weldment locations in the OLP.

4 Technical implementation

In order to prove the feasibility of the proposed PPR model, a technical implementation was put together using tools commercially available which are widely used across industry. The elements of the technical implementation are put together as showcased in Fig. 4. For the creation of the product family design to be configured, the SolidWorks CAD software was selected due to availability for the researcher and its compliance with ISO 2553:2019 standard. The CAD assembly created in SolidWorks serves as part of the **Product** component of the PPR model and it can be accessed through the **ProductModel** interface, as presented in Fig. 1. The interface is created as showed in Fig. 4 by using the SolidWorks API. The **ProductModelConfiguration** is created using the DriveWorks add-on for SolidWorks. DriveWorks was selected as it allows the creation of complex **ParametricRule** instances which can drive both the CAD models and the ISO 2553:2019 standard compliant annotations of the weldments. The **ProductModelConfiguration** is created by the design engineers. Upon instantiation of the **Product Model** through the graphical user interface

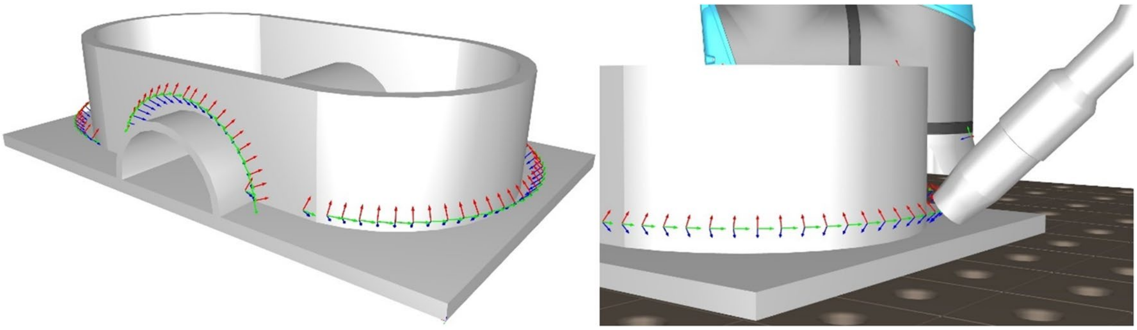


Fig. 3 Visualisation of the discretised weldment points which compose a welding operation and a welding program, resulted from Algorithm 2. On the left-hand side, an overview is provided. On the right-hand side, the torch positioned in one of the discrete weldment points can be observed. Images were generated using RoboDK

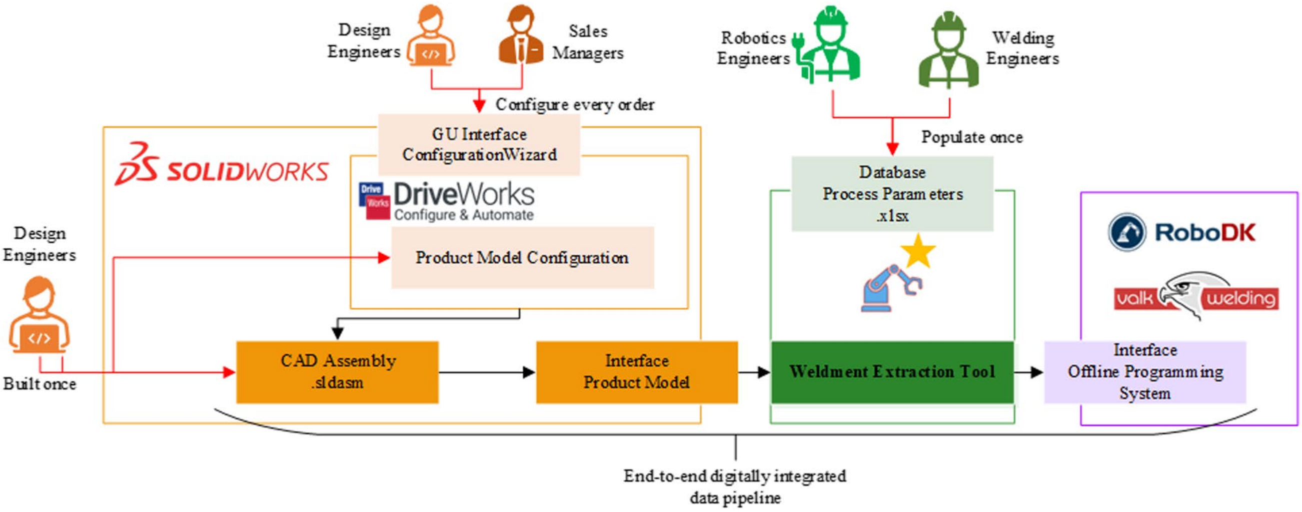


Fig. 4 Diagram showcasing the elements used in the technical implementation of the PPR model. The red arrow indicates manually input information, while the black arrows indicate automatic flow of information along the value chain for programming of welding robots

implemented in DriveWorks, the **CADWeldment** information can be extracted using the **Weldment Extraction Tool** interface annotated with the robot arm in the diagram presented in Fig. 4, and by using the **ProcessParameters** database populated by the welding and robotics engineers in a Microsoft Excel document which serves as a lookup table. The **Weldment Extraction Tool** uses the SolidWorks API in order to extract the necessary information to automatically program the welding robot. The result of the extraction is the.STEP file which can be imported in the OLP and files containing information about the weldments and process parameters. By using Algorithm 3, automatic programming of the welding robot is achieved in the RoboDK OLP software. A special tool called Oqton is used to facilitate the automatic programming of welding robots for the DTPS software supplied by Valk Welding (technology supplier) [28].

There are two approaches towards the automatic programming of the welding robot:

1. Using the RoboDK software, which makes use of the Algorithm 3 described in Section 4, by connecting to the RoboDK API and importing the CAD model in STEP file format and the Welding Program containing weldment information and process parameters.
2. By creating special CSR files which are programs that can be imported in the Desktop Programming Software developed by Panasonic and supplied by Valk Welding. The programming of the welding robot is fully automized using the Oqton add-on supplied by Valk Welding [28].

5 Experiments and results

Experiments were conducted with the technical implementations in the welding laboratory at Aalborg University and on the production line at the case company to test the hypothesis that welding robots can be programmed automatically based on the PPR model with minimum human input necessary. Therefore, two experimental pipelines and two experimental setups are used with the available hardware and software in the two environments. The technical implementation using the RoboDK OLP is used in the laboratory at Aalborg University and the technical implementation using the DTPS OLP is used on the production line at the case company.

5.1 Product design for experiments

An experimental product was designed in collaboration with the case company which will cover in its design the wide range of geometrical elements encountered in the company's real products (observed in Fig. 5 and Fig. 3). The geometrical elements are circle arches of various dimensions with various orientations. The weldments which are tested are fillet weldments. Weldment annotations compliant with the ISO 2553:2019 standard are added using the "Weld Bead" tool available in the SolidWorks CAD software and are visible as black and grey bead-like elements overlaid on the CAD model [37]. The CAD model in Fig. 5 together with the overlaid annotations constitutes a **CADAssembly** instance, accessible through the **ProductModel** interface as explained in the architecture presented in Subsection 3.2.

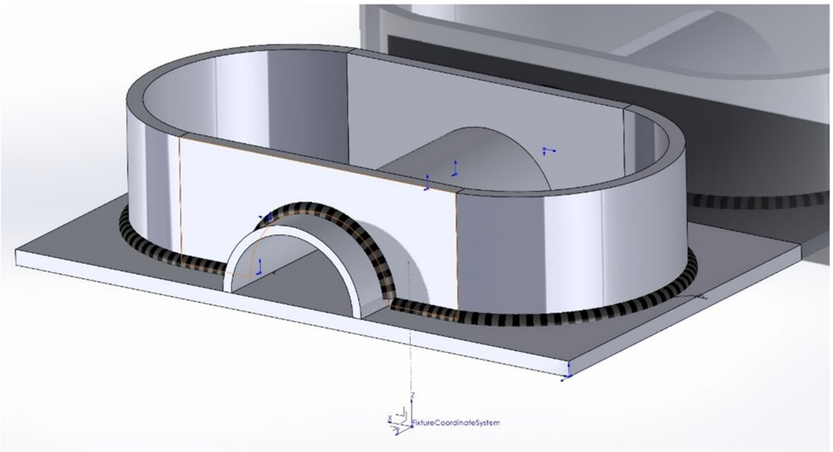
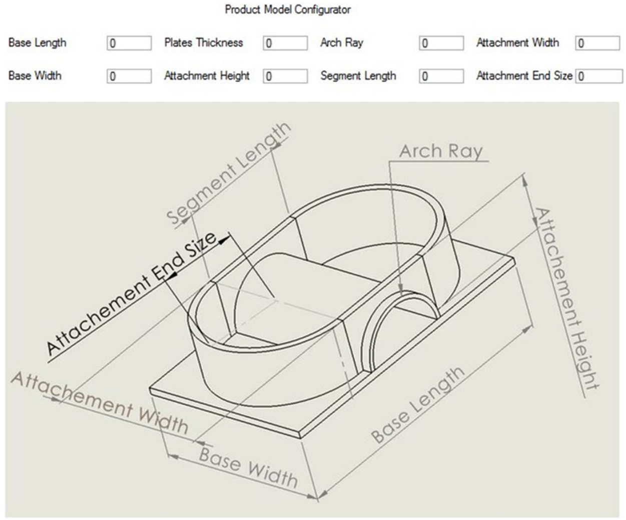


Fig. 5 Test item designed in collaboration with the case company in SolidWorks. The weldments annotations compliant with the ISO 2553:2019 standard, are visible and marked with black and grey

By using the DriveWorks add-on available in SolidWorks, a **ProductModelConfiguration** was put together by the design engineers which will drive the **Product** component of the PPR model. The **ProductModelConfiguration** for the experimental case receives a total of 8 parameters driving the dimensions of various features of the item showed in Table 2.

The design parameters serve as input for a **Configuration Wizard** implemented in DriveWorks and are displayed in Fig. 6. Each design parameter drives one or more **parametric rules** attached to various elements of the **Product**. The process of creating a new product variant takes only a few minutes to complete. For testing the technical implementation presented in the previous section, three products are created using the Product Model Configurator and then fabricated by the case company for running the experiments both on their production line an in the controlled environment at Aalborg University. The design input for each of the test products is shown in Table 2.

The case company was provided with the technical drawings, and the components of the item were fabricated, and tack welded together, being ready to be completely welded by the welding robots. The material used for all the items used in experiments is SSAB 335MC. In order to weld this material, based on the recommendations received from the case company, G3SI1 compliant filler wire with 1.2 mm diameter is used, together with a shielding gas mixture composed of 82% Argon and 18% Carbon Dioxide.


Product Model Configurator

Base Length 0 | Plates Thickness 0 | Arch Ray 0 | Attachment Width 0

Base Width 0 | Attachment Height 0 | Segment Length 0 | Attachment End Size 0

Fig. 6 Product model configurator wizard implemented in DriveWorks, using eight design parameters for controlling the model

The arc and motion parameters which populate the **ProcessParameters** database were also agreed upon together with the case company given the plate thickness used in the items to be welded. The parameters which are used in the experiments are displayed in Table 3. The parameters for fillet weldments between plates of 10 and 12 mm are given; however, the full database contains parameters for plates between 8 and 26 mm. The scope of the experiments

Table 2 Input design parameters of the instantiation of three separate **Product Model** instances. The values are expressed in millimetres

	Base length	Base width	Plates thickness	Attachment height	Arch ray	Segment length	Attachment width	Attachment end size
Item 1	400	250	10	80	50	180	200	90
Item 2	400	250	10	80	65	150	220	110
Item 3	450	300	10	80	50	160	270	120

Table 3 Lookup table which represents the **Process Parameters** database. The database is built using the tacit knowledge of the robotics and welding engineers from the case company. Same kind of lookup table is implemented in a database as showed in Fig. 4

Plate thickness	Torch velocity	Stickout	Welding angle	Travel angle	Amperage	Voltage
10 mm	0.007 m/s	18 mm	45°	0°	290 A	28.2 V
12 mm	0.006 m/s	18 mm	45°	0°	303 A	28.9 V

presented in this paper does not span beyond fillet weldments of plates with 10 mm thickness.

5.2 Experiment in controlled environment at Aalborg University

For the experiments in the controlled environment, at Aalborg University, the technical implementation with the RoboDK OLP is used. A digital model of the robot cell was built using the RoboDK OLP. The digital model of the welding-robot cell can be observed in Fig. 7 together with the physical cell. The experimental setup is composed of a Universal Robots UR10 collaborative manipulator, a Migatronic Sigma Select 400 CW welding apparatus and a Siegmund Professional 750 welding table. The welding table has a measurement mesh engraved which allows for precise placement of the work piece on its surface.

After the geometry and the design parameters of the weldment are extracted using Algorithm 1, the motion parameters for the welding torch can be computed by using the **Weldment Extraction Tool** interface in which Algorithm 2 is implemented. Each position of the weldment torch relative to the weldment seams can be visualised in the OLP as presented in Fig. 3. The motion parameters are imported in the offline programming tool as displayed in Fig. 3 using the application programming interface of RoboDK. Together with the orientation of the torch, the torch velocity is imported too alongside the arc parameters which will be used to obtain the weldment.

With all the necessary data imported in the OLP, the robot program is automatically generated using Algorithm 3 and then ready to be transferred on the robot. A visualisation of the robot programs in RoboDK is provided in Fig. 8 for all three items created using the design parameters in Table 2.

Video of the experiment is available on YouTube at https://www.youtube.com/watch?v=fVslC2x6YVg. The items which were welded using the automatically generated welding program are displayed in Fig. 9.

Results The results of the experiment show that automatic programming of welding robots using PPR models is feasible. After the insertion of the design parameters in the **Configuration Wizard**, no other input was necessary from the human operator and no further intervention was made on the CAD model, on the design parameters of the product and on the process parameters required for the welding process. The employed Algorithm 3 uses the capabilities of the offline programming system to ensure that the available

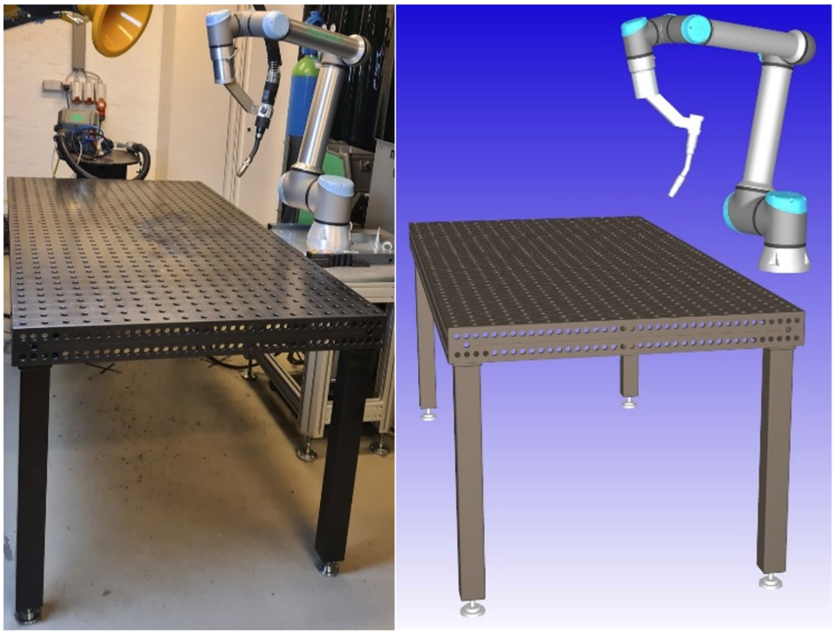


Fig. 7 The robot-welding cell available in the laboratory at Aalborg University on the left-hand side together with the digital twin built using RoboDK on the right-hand side

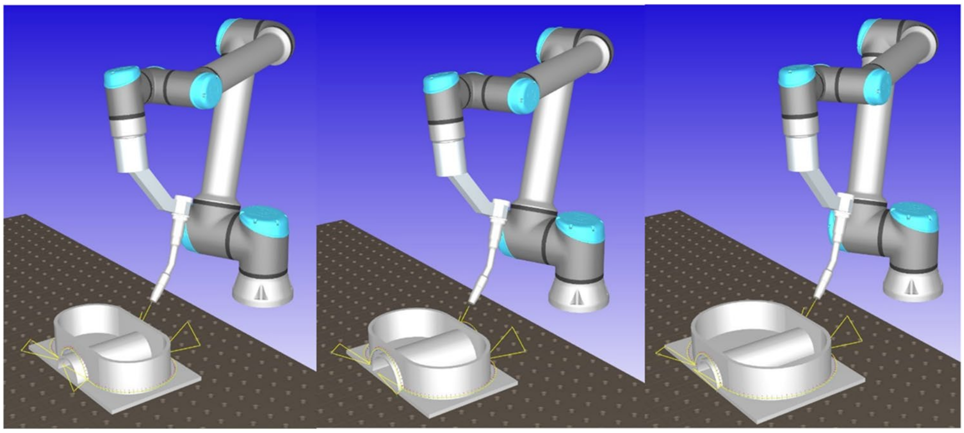


Fig. 8 Visualisation of the welding programs which will be transferred on the robot for each item 1, 2 and 3 (from left to right)

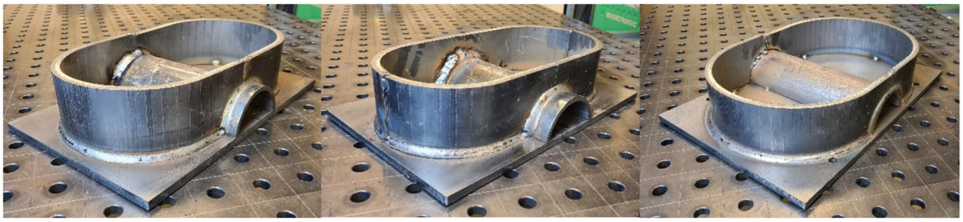


Fig. 9 Resulted weldments following the automatic programming of the welding robot used in laboratory. From left to right: Item 1, Item 2 and Item 3, based on the design parameters displayed in Table 2

resources (the Universal Robots UR10 robot) are able to successfully execute the weldment before the program is sent to the robot, thus proving the end-to-end integration of product and production in a seamless chain of operations.

The resulted weldments were inspected by FORCE Technology, and it was concluded that the weldments can be classified in the class D of the ISO 5817 standard. FORCE Technology is a Danish-approved technological service institute tasked to get technologies into use at Danish companies.

5.3 Experiment in the production line

In collaboration with Sjørring Maskinfabrik A/S, experiments were conducted on a production line in order to check the feasibility of the concept in a real-life scenario. Sjørring Maskinfabrik is an ETO SME located in Denmark, which is involved in the metal fabrication industry. The products produced by the company are in the category of metallic structures used in the construction, automotive and transportation industry.

For the experiments in the production line at Sjørring Maskinfabrik, the technical implementation using the Desktop Programming System (DTPS-Panasonic) provided by Valk Welding was used. The complete robotic setup includes a Panasonic TL 1800 industrial manipulator together with welding apparatus, prismatic mobile base and a two axis-positioner for the workpiece supplied by Valk Welding. The setup together with its digital twin can be observed in Fig. 10.

After the geometry and the design parameters of the weldment are extracted, each position of the weldment torch relative to the weldment seams can be visualised in the OLP as presented in Fig. 11. The parameters which dictate the motion of the torch along the welding seams is imported in DTPS by importing the file generated using the **Weldment Extraction Tool** containing the motion parameters for the welding torch and the design parameters of the product, i.e. the thickness of the plates. The DTPS software uses its own database of arc parameters which was built by the welding engineers at the case company. The motion between weldments and the collision-free trajectories are automatically generated using the Oqton add-on supplied by Valk Welding instead of Algorithm 3 [28]. The Oqton software also facilitates the collision-free programming of the external axis and the mobile robot base, ensuring control over welding puddle of molten metal.

Results The results of the experiment show that automatic programming of welding robots using the PPR model is feasible using the hardware and the software available at the case company along their value chain. After the insertion of the design parameters in the **Configuration Wizard**, the only input required from operators is the import of the

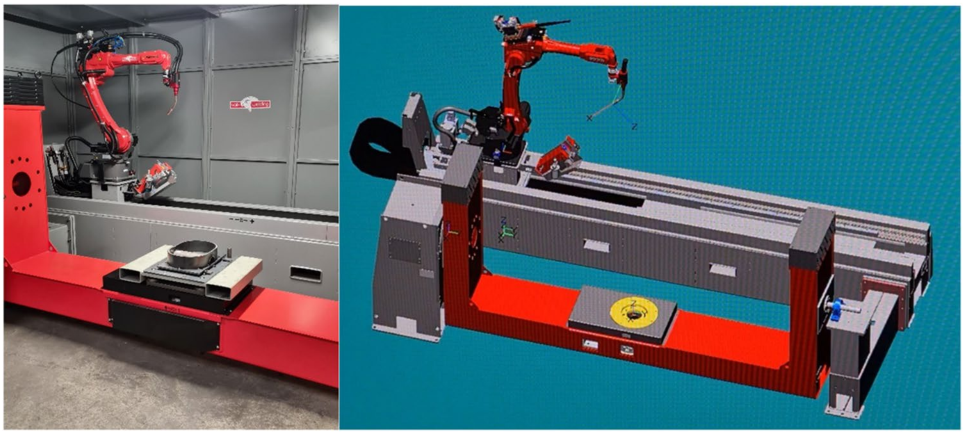


Fig. 10 Welding cell at the case company equipped with a Panasonic TL 1800 welding robot, with prismatic mobile base and 2 axis-positioner for the work piece, supplied by Valk Welding. The physical setup is displayed on the left-hand side together with its digital twin in DTPS on the right-hand side

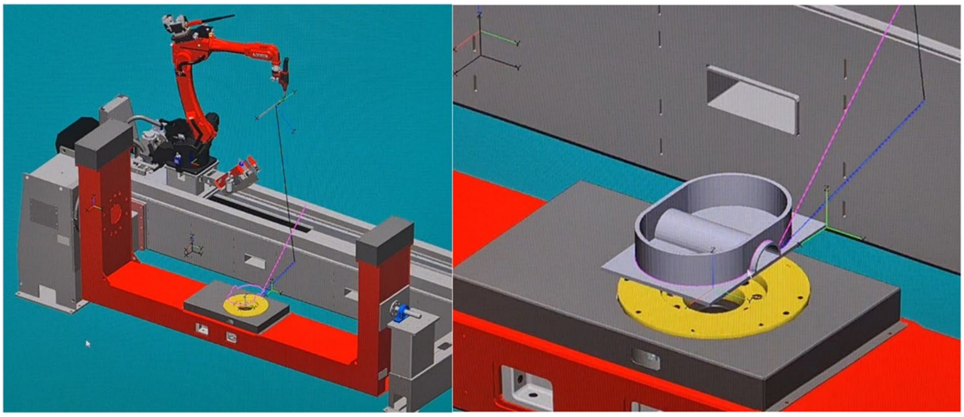


Fig. 11 The motion parameters of the weldment are imported and can be clearly observed in the image in the left-hand side of the figure. On the right-hand side, both the obtained robot program and the imported CAD model of the work piece can be observed

weldment information in the DTPS and Oqton software, which is rudimentary and can be managed with general computer skills. The Oqton software can automatically generate trajectories for both the Panasonic industrial manipulator and the external axis using the information extracted from the product model, thus shortening the programming time of the welding robot drastically.

The quality of the weldment was inspected by the welding engineers at the case company, and they were deemed satisfactory and compliant with the ISO 5817 class D standard. The resulted program is validated through simulations and then sent to the robot. The resulted weldments produced by the welding robot can be observed in Fig. 12. Video of the experiment is available at https://www.youtube.com/watch?v=hX0ZQHrMYWI

5.4 Quantification of results

The experiments above prove the feasibility of the proposed end-to-end digital integration for automatic programming of welding robots using PPR models. To prove the impact of the method, a further scenario was set up by using the traditional method of programming welding robots without having the CAD system and the OLP system integrated. To execute this scenario, Item 3 from Table 2 was selected to be used. First, a robotics engineer will create a robot program using manual input only, both to operate the CAD system to obtain the new product variant and to operate the OLP system to obtain the robot program. The results are compared to those obtained by using the automatic programming of the robot using the PPR model as observed in Table 4.

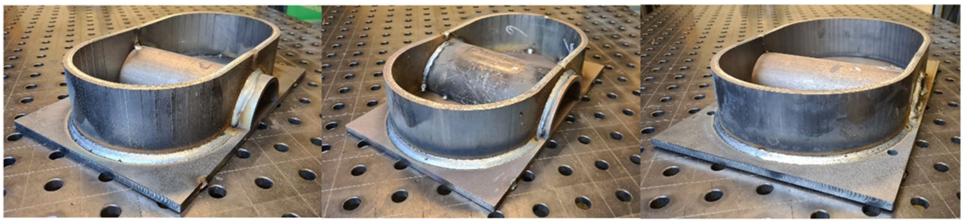


Fig. 12 Resulted weldment following the automatic programming of the welding robot used at the case company. From left to right: Item 1, Item 2 and Item 3, based on the design parameters displayed in Table 2

Table 4 Quantification of results when using the PPR model–based method compared to manual programming of welding robots using CAD and OLP systems without integration in between

Method	Human input (measured in terms of time required for the engineer to operate the systems)	Experiment video
Manual programming	Around 30 min	https://www.youtube.com/watch?v=9BhD3BKojZM
Automatic programming using PPR model with RoboDK	Around 3 min	https://www.youtube.com/watch?v=Xzra1gKHHAw
Automatic programming using PPR model with Oqton DTPS	Around 5 min	https://www.youtube.com/watch?v=9xwGJcSgMqc

The results in Table 4 indicate a drastic reduction of the human input necessary to program the welding robot of about 80% in terms of time spent by the robotics engineer.

6 Discussion and conclusion

In this paper, a PPR model for automatic programming of welding robots is proposed to enable the end-to-end digital integration of product and production. By making use of the PPR perspective over the product design and manufacturing preparation, all the necessary data for programming the welding robot, design parameters and process parameters, are processed automatically using the **Product Configuration** composed of **Parametric Rule** entities, which accepts values for design parameters through a configuration wizard. The necessary data for setting up the process parameters for the welding process are seamlessly extracted from the CAD model which is compliant with ISO 2553:2019 standard, allowing automatic programming of welding robots in one-of-a-kind production environments.

6.1 Academic implications

The knowledge base in the CAD-based programming field is extended by proposing a general framework for programming of welding robots based on data extracted from CAD systems compliant with the ISO 2553:2019 standard, adding to the work conducted by Tran et al. [41, 42] and Mohammed et al. [40]. Unlike feature recognition algorithms, the proposed method makes use of the annotations which the design engineers must add to CAD models and therefore, further input from robotics engineers is not necessary to program the locations of the weldments and motion parameters of the welding robot. The PPR perspective is used to structure the data required to program the welding robot in a rigorous manner defined using the unified modelling language (UML). It is showed through the proposed PPR model, that the PPR perspective is highly relevant to applications beyond assembly operations, like welding. By using the PPR perspective, an engineering framework is obtained which bring together information from the product's design generated through the customer's demands, knowledge regarding the process necessary to produce the item and information about the resources required and resources' capabilities to support the welding process.

6.2 Implications for practitioners

The work presented in this paper is motivated by the context of ETO SMEs, which operate in high-cost environments. Moreover, ETO SMEs are particularly challenged as they need to cope with long lead times for their orders, low utilisation of welding robots and highly skilled engineers required to reprogram the welding robots. The gap between theory and practice in the field of ETO supply chain planning, described by Bhalla et al. [50], is addressed in this work through practical problem-solving by proposing the presented PPR model meant to reduce the lead time and the cost of value adding operations along the ETO value chain. This work is proposed as direct response to Bhalla et al.'s call towards problem-solving research in the field.

While trying to cope with the shortage of welders, the case company of this research tries to rely more on automation. However, as presented and highlighted in the quantification of results, even though welding robots are used, they do require a considerable input from robotics engineers and design engineers to handle the high variation of products associated with an ETO, and to generate new robot programs. In other words, the problem of welders shortage is translated into a production cost–related problem. As suggested in Table 4, by using the proposed PPR model to program robots, there are several advantages to be considered:

- By reducing the necessary human input to generate a CAD model for a new product variant and to generate a new associated robot program, the cost associated with that human worker's time is reduced and the human worker can be reallocated to other tasks.
- Robot utilisation rate: The execution of the robot program while welding the work piece has a duration of about 5 min. In the scenario where the robot is currently

welding a smaller work piece, the downtime until the new robot program is ready can be up to 25 min for Item 3, as suggested in Section 5.4.
- Lead time: With the programming time longer than the actual execution of the welding program, the duration of the programming process directly affects the lead time of the product manufacturing. By drastically decreasing the time necessary to program the welding robots, the lead time of the product is reduced too.

By implementing the PPR model, an end-to-end seamless integration of the sales operations, product engineering operations, and production preparation operations has been achieved. The obtained information flow is streamlined, and the time needed to design a new product variant and program a welding robot is drastically reduced.

6.3 Research limitations

The research that was conducted and presented in this paper was subject to several limitations which must be considered. The focus of the research was limited to robotic welding applications only, specifically MIG welding. Although the implications of this research may be valid for other applications too (e.g. glue dispensing, TIG welding), they were not included in the scope of this research, and they were limited by the equipment available to perform the research. The workpiece which was used to conduct the experiments has only fillet weldments. Other types of weldments (e.g. groove weldments) were not tested in the experiments.

Further limitations were imposed by the systems available for the researcher to design the experiments for the technical realisation of the PPR model concept. Access was limited to the SolidWorks CAD software together with its DriveWorks add-on, and to the RoboDK OLP software and the OLP software provided by ValkWelding. Access for experiments on the production line of case company was limited based on the work-shop activity and had to be synchronised based on robot downtime (e.g. when the utilisation rate is 0).

6.4 Further research and development

In order to reliably ensure the end-to-end integration of operations across the value chain in an ETO enterprise and to address the limitation this research was constrained by, the following further research and development paths are identified by the authors:

- This paper presents a novel approach towards CAD-based programming by extracting all the relevant information to program a welding robot directly from the CAD system if it is compliant with the ISO 2553:2019 standard. It was observed during experiments that it is highly important to ensure that after the welding-robot program is generated, the physical tack-welded product is true to the CAD model. Therefore, the PPR model in combination with sensor-based programming is found to be highly interesting to investigate further in order to compensate for imperfections which might occur in the physical product.
- The relevant work documented in the literature about the PPR perspective covers assembly operations mainly. This work shows that the PPR perspective can be adapted and used to create models for automatic programming of welding robots. Further research is encouraged to investigate the possibility of creating the same kind of models for other production operations, e.g. cutting and bending.
- The generalisability of the proposed PPR model is to be investigated further in other environments than those presented in this paper and by using a wider palette of products.
- The research in this paper was limited to MIG welding process only and fillet weldments; however. there are several other welding processes which involve robotic manipulators: gas tungsten arc welding, laser welding. Moreover, glue dispensing applications involve similar process parameters: end-effector travel speed, glue dispensing rate. It is found highly interesting to further investigate how the PPR model can be adapted to cover these applications too.

6.5 Conclusion

This paper has brought in the reader's attention the current context of high-cost environments where ETO manufacturing enterprises face several challenges: shortage of welders, high production cost and low utilisation of welding robots due to high variety in their product offering. Furthermore, the gap found in the literature regarding CAD-based programming methods and PPR perspective applications in the manufacturing industry is found apparent. The proposed PPR model streamlines the information flow across multiple operations conducted in ETO enterprises allowing for faster lead time. The generalisability of the model is enabled by addressing CAD systems compliant with the ISO 2553:2019 standard. By using the PPR perspective, a comprehensive overview on all the data required to program a welding robot is obtained and the architecture of the proposed model is adapted accordingly. It is concluded that the proposed PPR model is technically feasible, and it holds significant potential in reducing production costs, drastically reduces the time needed to create new product variants, drastically reduces the time needed to program welding robots and increases utilisation rate of welding robots.

Acknowledgements Special thanks to Sjørring Maskinfabrik A/S for allowing unrestricted access to their value chain, their knowledge base and their robot-welding cell for running experiments. We wish to acknowledge the contributions of Klaus Kalstrup, CEO; Morten Hvass, Project Manager; John Yde Hove, Project Manager; Andreas Kjær, Head of Supply Chain; Per Holst Menander, Development Engineer; Mads Fredsøe, Development Engineer, Jens Holm, Head of Production and finally, the work-shop workers who have conducted the fabrication of the experiment items.

The authors express gratitude towards Valk Welding who supplied DTPS licenses and offered attentive assistance in the development of the technical artefact for integration with DTPS. We wish to acknowledge the contributions of James de Villiers, Development Engineer and Adriaan Broere, CTO.

We express our gratitude towards FORCE Technology, who helped with the objective classification of the weldments obtained in the lab environment. We acknowledge the contributions of Michel Honoré, Senior Team Lead and Per Madsen, Robotics and 3D Print Operator.

We express our gratitude towards DriveWorks and RoboDK for supplying educational licenses which made possible the technical implementation which supported the tests conducted for validating the proposed theory.

Author contribution All authors have contributed to the research hereby presented and its documentation in this paper. Ioan-Matei Sarivan has conducted the conceptualisation of the PPR model and the technical implementation of the model in collaboration with external parties (Sjørring Maskinfabrik, FORCE Technology and Valk Welding), and wrote the submitted version of the manuscript. Professor Brian Vejrum Wæhrens has contributed in ensuring the research process's balance between relevance and rigour by making and made sure the novelty of this work is properly documented. Professor Ole Madsen has contributed to framing the literature study in the field of CAD-based programming of welding robots and PPR models, while ensuring that the proposed PPR concept is properly grounded in the existing knowledge base. All authors have read and approved the manuscript that is hereby submitted for The International Journal of Advanced Manufacturing Technology.

Funding Open access funding provided by Aalborg University The authors declare that this work was funded by Manufacturing Academy of Denmark (MADE). MADE is a Danish national association of universities, research and technology organisations, and private companies which work together towards bringing the Danish manufacturing industry at the highest global competitive level possible.

Declarations

Competing interests The authors declare no competing interests.

References

1. Ahmad M, Ferrer BR, Ahmad B, Vera D, Martinez Lastra JL, Harrison R (2018) Knowledge-based PPR modelling for assembly automation. CIRP J Manuf Sci Technol. https://doi.org/10.1016/j.cirpj.2018.01.001
2. McKinsey and Company (2016) Danish manufacturing - winning in the next decade. https://www.mckinsey.com/featured-insights/europe/danish-manufacturing-winning-in-the-next-decade. Accessed Mar 2024
3. Sebastian Wittrock (2023) Researchers at SDU receives 13 million DKK grant to develop a robot that will be crucial to the green transition. https://www.sdu.dk/en/om_sdu/fakulteterne/teknik/nyt_fra_det_tekniske_fakultet/sdu-forskere-faar-13-millioner-til-ny-robot
4. Bejlegaard M, Sarivan I, Waehrens BV (2021) The influence of digital technologies on supply chain coordination strategies. J Glob Oper Strateg Sourc. https://doi.org/10.1108/JGOSS-11-2019-0063
5. Tolio T, Sacco M, Terkaj W, Urgo M (2013) Virtual factory: an integrated framework for manufacturing systems design and analysis. Procedia CIRP. https://doi.org/10.1016/j.procir.2013.05.005
6. Zheng P, Xu X, Yu S, Liu C (2017) Personalized product configuration framework in an adaptable open architecture product platform. J Manuf Syst. https://doi.org/10.1016/j.jmsy.2017.03.010
7. Berry C, Wang H, Hu SJ (2013) Product architecting for personalization. J Manuf Syst. https://doi.org/10.1016/j.jmsy.2013.04.012
8. Tan C, Chung H, Barton K, Jack HuS, Freiheit T (2020) Incorporating customer personalization preferences in open product architecture design. J Manuf Syst. https://doi.org/10.1016/j.jmsy.2020.05.006
9. Lauridsen JK (1991) Computer aided off-line programming of multipass TIG-Welding. Dissertation or Thesis, Institute of Production, Aalborg University. https://vbn.aau.dk/en/publications/computer-aided-off-line-programming-of-multipass-tig-welding
10. Meixner K, Decker J, Marcher H, Luder A, Biffl S (2020) Towards a domain-specific language for product-process-resource constraints. ETFA. https://doi.org/10.1109/ETFA46521.2020.9212063
11. Meixner K, Rinker F, Marcher H, Decker J, Biffl S (2021) A domain-specific language for product-process-resource modeling. ETFA. https://doi.org/10.1109/ETFA45728.2021.9613674
12. Agyapong-Kodua K, Haraszkó C, Németh I (2014) Recipe-based integrated semantic product, process, resource (PPR) digital modelling methodology. Procedia CIRP. https://doi.org/10.1016/j.procir.2014.03.118
13. Brecher C, Kusmenko E, Lindt A, Rumpe B, Storms S, Wein S, von Wenckstern M, Wortmann A (2018) Multi-level modeling framework for machine as a service applications based on product process resource models. Proceedings of the 2nd International Symposium on computer science and intelligent control. https://doi.org/10.1145/3284557.3284714
14. Ferrer BR, Ahmad B, Lobov A, Vera DA, Martinez Lastra JL, Harrison R (2015) An approach for knowledge-driven product, process and resource mappings for assembly automation. CoASE. https://doi.org/10.1109/CoASE.2015.7294245
15. Winkler D, Novak P, Meixner K, Vyskocil J, Rinker F, Biffl S (2021) Product-process-resource asset networks as foundation for improving CPPS engineering. ETFA. https://doi.org/10.1109/ETFA45728.2021.9613253
16. Schleipen M, Drath R (2009) Three-view-concept for modeling process or manufacturing plants with AutomationML. ETFA. https://doi.org/10.1109/ETFA.2009.5347260

17. Pfrommer J, Schleipen M, Beyerer J (2013) PPRS: production skills and their relation to product, process, and resource. ETFA. https://doi.org/10.1109/ETFA.2013.6648114
18. Bergur Thormundsson (2023) Total number of industrial robots installed globally from 2019 to 2021, by application. https://www.statista.com/statistics/1383931/industrial-robot-installation-by-field-worlwide/. Accessed Mar 2024
19. Zych A (2021) Programming of welding robots in shipbuilding. Procedia CIRP. https://doi.org/10.1016/j.procir.2021.03.107
20. Kristianto Y, Helo P, Jiao RJ (2015) A system level product configurator for engineer-to-order supply chains. Comput Ind. https://doi.org/10.1016/j.compind.2015.04.004
21. Wang B, Li Y, Freiheit T (2022) Towards intelligent welding systems from a HCPS perspective: a technology framework and implementation roadmap. J Manuf Syst. https://doi.org/10.1016/j.jmsy.2022.09.012
22. FANUC ROBOGUIDE - Intelligent 3D simulation. https://www.fanuc.eu/dk/en/robots/accessories/roboguide. Accessed Mar 2024
23. KUKA From offline programming to virtual commissioning. https://www.kuka.com/en-de/products/robot-systems/software/planning-project-engineering-service-safety/kuka_sim. Accessed Mar 2024
24. ABB Detecting Collision. https://developercenter.robotstudio.com/api/robotstudio/articles/How-To/Miscellaneous/CollisionDetectionExample.html. Accessed Mar 2024
25. ValkWelding (2011) Panasonic, the best tool for the arc welding robot industry. ValkMailing. https://valkwelding.com/media/site/3556cf32b1-1682495626/valk-mailing-2011-1-en.pdf. Accessed Mar 2024
26. RoboDK Collision Detection. https://robodk.com/doc/en/Collision-Avoidance.html. Accessed Mar 2024
27. Pan Z, Polden J, Larkin N, Van Duin S, Norrish J (2012) Recent progress on programming methods for industrial robots. Robot Comput Integr Manuf. https://doi.org/10.1016/j.rcim.2011.08.004
28. Valk Welding (2023) Oqton teams with Valk welding for automatic robotic arc welding programming. MetalForming. https://www.metalformingmagazine.com/article/?/pressroom-automation/robotics/oqton-teams-with-valk-welding-for-automatic-robotic-arc-welding-programming. Accessed Mar 2024
29. Ferreira LA, Figueira YL, Iglesias IF, Souto MÁ (2017) Offline CAD-based robot programming and welding parametrization of a flexible and adaptive robotic cell using enriched CAD/CAM system for shipbuilding. Procedia Manuf. https://doi.org/10.1016/j.promfg.2017.07.228
30. Kiani MA, Saeed HA (2019) Automatic spot welding feature recognition from STEP data. RAEE. https://doi.org/10.1109/RAEE.2019.8886989
31. Xuan LanPhung, Ngoc Linh Tao (2021) Automatic extraction and welding feature recognition from STEP data. Advances in engineering research and application. Springer International Publishing AG, Switzerland, pp 210–215
32. Kuss A, Dietz T, Ksensow K, Verl A (2017) Manufacturing task description for robotic welding and automatic feature recognition on product CAD models. Procedia CIRP. https://doi.org/10.1016/j.procir.2017.01.045
33. Fang H, Ong S, Nee A (2017) Robot path planning optimization for welding complex joints. Int J Adv Manuf Technol. https://doi.org/10.1007/s00170-016-9684-z
34. Groover MP (2016) Automation, production systems and computer-integrated manufacturing, Fourth Global Edition, Pearson Education Limited
35. AutoDesk Weldments in Fusion. https://help.autodesk.com/view/fusion360/ENU/?guid=DWG-REF-WELDING. Accessed Mar 2024
36. PTC Welding Design in Creo. In: https://support.ptc.com/help/creo/creo_pma/r10.0/usascii/?_gl=1*3ljo5r*_ga*MTc5NzAzNzY2My4xNjg4MTM4NDIy*_ga_1QBT6P6HR1*MTY4ODEzODQyMS4xLjEuMTY4ODEzODUyMS4wLjAuMA..*_ga_CBN5QVB9VJ*MTY4ODEzODQyMS4xLjEuMTY4ODEzODUyMS4wLjAuMA..#page/welding/welding.html. Accessed Mar 2024
37. Dassault systemes cosmetic weldments solidworks. https://help.solidworks.com/2024/english/SolidWorks/sldworks/t_Creating_Weld_Beads.htm. Accessed Mar 2024
38. Siemens Weldments in SolidEdge. https://community.sw.siemens.com/s/article/frames-and-weldments-in-solid-edge. Accessed Mar 2024
39. Siemens Weldment Assistant—Siemens NX. https://docs.plm.automation.siemens.com/tdoc/nx/1899/nx_help/#uid:best_practices_bp_weldments_assistant. Accessed Mar 2024
40. Mohammed SK, Arbo MH, Tingelstad L (2021) Constraint identification from STEP AP242 files for automated robotic welding. ICMIMT. https://doi.org/10.1109/ICMIMT52186.2021.9476209
41. Tran TA, Lobov A, Kaasa TH, Bjelland M, Midling OT (2021) CAD integrated automatic recognition of weld paths. Int J Adv Manuf Technol. https://doi.org/10.1007/s00170-021-07186-0
42. Tran TA, Njåstad EB, Midling OT, Bjelland M, Lobov A (2023) Generation of rule-adhering robot programs for aluminium welding automatically from CAD. Int J Adv Manuf Technol. https://doi.org/10.1007/s00170-023-10996-z
43. Chen S, Zhang Y, Feng Z (2017) Automated programming for robotic welding. Transactions on intelligent welding manufacturing. Springer Singapore Pte. Limited, Singapore, pp 48–59
44. Eriksson H, Fado D, Lyons B, Penker M (2003) UML 2 Toolkit. Wiley, Indianapolis, Ind
45. Chavali SR, Sen C, Mocko GM, Summers JD (2008) Using rule based design in engineer to order industry: an SME case study. Comput Aided Des Appl. https://doi.org/10.3722/cadaps.2008.178-193
46. Camba JD, Contero M, Company P (2016) Parametric CAD modeling: an analysis of strategies for design reusability. Comput Aided Des. https://doi.org/10.1016/j.cad.2016.01.003
47. Scott K (2004) Fast Track UML 2.0. Apress, Berkeley, CA. https://doi.org/10.1007/978-1-4302-0720-7
48. I Sarivan JS Larsen O Madsen BV Wæhrens 2023 Elementary welding operations for automatic robot programming https://doi.org/10.1007/978-3-031-27933-1_9
49. BKP Horn HM Hilden ShahriarNegahdaripour 1988 Closed-form solution of absolute orientation using orthonormal matrices J Opt Soc Am A Opt Image Sci https://doi.org/10.1364/JOSAA.5.001127
50. Bhalla S, Alfnes E, Hvolby H (2023) Tools and practices for tactical delivery date setting in engineer-to-order environments: a systematic literature review. Int J Prod Res. https://doi.org/10.1080/00207543.2022.2057256