



Article

A Model-Based Approach to Automated Validation and Generation of PLC Code for Manufacturing Equipment in Regulated Environments

Damian McCarthy^{1,2,3,*} , Dermot McMorro⁴, Noel P. O'Dowd^{1,2,3}, Conor T. McCarthy^{1,2,3}
and Eoin P. Hinchy^{1,2,3} 

- ¹ Confirm Smart Manufacturing Research Center, University of Limerick Digital District, V94 C928 Limerick, Ireland; noel.odowd@ul.ie (N.P.O.); conor.mccarthy@ul.ie (C.T.M.); eoin.hinchy@ul.ie (E.P.H.)
- ² Bernal Institute, University of Limerick, V94 T9PX Limerick, Ireland
- ³ School of Engineering, University of Limerick, V94 T9PX Limerick, Ireland
- ⁴ SL Controls, IDA Industrial Park, F91 WF53 Sligo, Ireland; dermot.mcmorrow@slcontrols.com
- * Correspondence: damian.mccarthy@ul.ie

Abstract: Validation is a critical stage of the equipment design process as it provides documentary evidence that the equipment is performing as per specification and ensures consistent product quality is maintained at all times. The advent of Industry 4.0 has led to a requirement for reconfigurable manufacturing systems as manufacturers adapt to an increased customer demand for personalised products. As equipment control software becomes increasingly complex to accommodate these requirements, a new approach to equipment validation is required. This paper presents a methodology for the design and validation of equipment in regulated manufacturing environments, using a model-based design platform (MathWorks[®] Simulink[®]) to model and digitally validate the Programmable Logic Controller (PLC) code required to control manufacturing equipment. A workflow is presented detailing the steps required to implement this approach and a demonstration model was developed as a proof of concept. Validation documentation and PLC code are automatically generated based on the system model and the functionality of the generated PLC code was successfully verified on a physical demonstrator, proving the feasibility of the proposed approach. Adoption of the approach outlined in this work would enable manufacturers in regulated industries, such as medical devices and pharmaceutical products, to rapidly design, build, reconfigure and revalidate manufacturing equipment as required to accommodate an increased demand for customised products.

Keywords: reconfigurable manufacturing systems; model-based design; equipment validation; Simulink[®]; Automated Code Generation



Citation: McCarthy, D.; McMorro, D.; O'Dowd, N.P.; McCarthy, C.T.; Hinchy, E.P. A Model-Based Approach to Automated Validation and Generation of PLC Code for Manufacturing Equipment in Regulated Environments. *Appl. Sci.* **2022**, *12*, 7506. <https://doi.org/10.3390/app12157506>

Academic Editor: Paolino Di Felice

Received: 11 June 2022

Accepted: 24 July 2022

Published: 26 July 2022

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Copyright: © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Validation activities are a critical stage of the equipment design process for medical devices and pharmaceutical manufacturers, enabling them to provide documentary evidence that the equipment is performing as per specification and ensuring consistent product quality is maintained at all times [1]. As industry transitions from the third to the fourth industrial revolution, new approaches to equipment and process validation will be required to enable the adoption of Industry 4.0 technologies [2]. The Industry 4.0 concept has been described as the utilisation of digital technologies to transform manufacturing facilities into Smart Factories [3]. This transformation is being driven by an increased customer demand for personalised products, leading many manufacturers to shift from mass production of their products and to begin to implement mass customisation strategies to satisfy this demand [4].

One strategy to enable mass customisation is the development of flexible manufacturing systems which can be rapidly modified to account for changes to product configuration,

parts availability and variations in order levels [5]. Manufacturers are attempting to achieve this flexibility using reconfigurable manufacturing equipment and system control software [6]; however, the implementation of Industry 4.0 practices has led to an increase in equipment complexity and an increasing percentage of the equipment functionality being achieved via software configuration [7]. Consequently, the control software for manufacturing equipment must be continuously updated to accommodate modified or new functionalities [8]. This leads to a requirement to re-test existing equipment to ensure that the modified functionalities do not negatively impact product quality. However, with equipment availability being a key element of maintaining a profitable manufacturing enterprise [9], the requirement to regularly stop production for days, or even weeks, to test new equipment functionality is simply not feasible. If product manufacturers in regulated manufacturing sectors, such as medical device and pharmaceutical production, are to implement flexible manufacturing practices, an equipment validation methodology is required which enables this approach but without the requirement to stop production for testing for long periods of time.

The objective of this work was to present a proof of concept, using industrially relevant tools and equipment, which demonstrates how Model-Based Design (MBD) practices can be used to enable automated validation and generation of equipment control codes for medical devices and pharmaceutical manufacturers. Currently, the validation and programming of software for Programmable Logic Controllers (PLC) in regulated manufacturing environments is a manual process. The research carried out in this paper proved that an MBD approach to equipment validation is technically feasible and enables the automatic generation of validation documentation and the associated PLC code required for the equipment to function. Adoption of this approach is expected to provide multiple benefits to manufacturers of regulated products, including shorter validation timelines, reduced validation costs, improved software quality, automated generation of validation documentation and automatic generation of the equipment's PLC code. This would enable engineers to adopt an agile approach to equipment design to create the reconfigurable manufacturing systems that are required to satisfy the mass customisation demands of Industry 4.0. Realisation of flexible manufacturing systems is a significant benefit for the medical device and pharmaceutical manufacturers as this industry is shifting towards personalised medical products based on the individual patient's unique medical requirements. This paper is structured as follows: Background information on relevant topics is presented in Section 2. Section 3 introduces the proposed methodology and describes the development of a proof-of-concept model to demonstrate the proposed methodology. The results of this demonstration, including validation documentation and successful operation of an industrially relevant piece of equipment, are presented in Section 4. Section 5 discusses the results and their relevance to industry. Finally, the paper closes with Section 6 providing a conclusion on the relevance of the proposed method to industry and a brief discussion of future work stemming from the research carried out in this research.

2. Background

2.1. Equipment Validation Practices in Regulated Industries

While all product manufacturers are subject to regulation, few are as strictly regulated as the manufacturers of medical devices and pharmaceutical products. Manufacturers in these sectors must comply with regulations from numerous regulatory organisations, depending on where the product is to be sold, including the European Medicines Agency (EMA) in the EU, Food and Drug Administration (FDA) in the USA and International Medical Device Regulators Forum at a global level [10]. To ensure compliance, manufacturers of medical device and pharmaceutical products validate their equipment and processes in line with the Good Manufacturing Practice (GMP) regulations, outlined in Chapter 5 of the Federal Food, Drug, and Cosmetic Act (FD&C Act) [11]. The FDA has defined validation as “the collection and evaluation of data, from the process design stage through commercial production,

which establishes scientific evidence that a process is capable of consistently delivering quality product" [1].

From a manufacturing equipment and control perspective, validation activities in regulated industries are defined by the Good Automated Manufacturing Practice 5 (GAMP 5) guidelines [12]. GAMP 5 is a set of guidelines which "provides a framework for the risk-based approach to computer system validation where a system is evaluated and assigned to a predefined category based on its intended use and complexity" [13]. It encompasses both hardware and software elements of the equipment and accounts for all stages of the equipment life-cycle, from the initial concept, through design and operation, and finally retirement of the equipment [14]. The design of equipment phases follows the V-Model process (see Figure 1) to provide traceability between the system requirements and the associated verification activities.

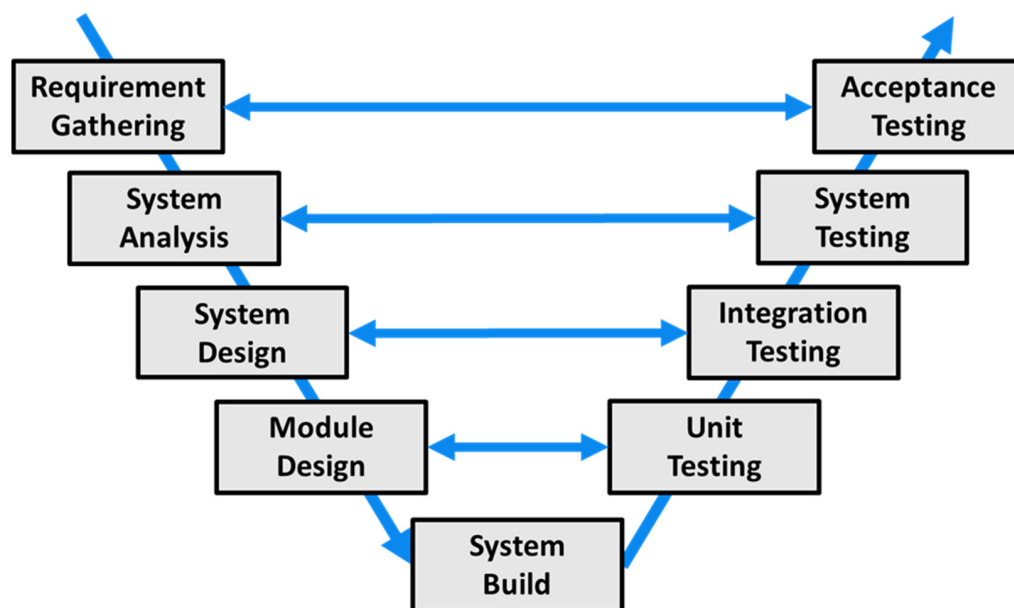


Figure 1. GAMP 5 V-Model.

Although it is a critical step to ensuring that products are correctly manufactured, the validation process is generally viewed in industry as a difficult, costly and resource intensive activity [15]. Validation typically accounts for 30–55% of the total project budget in terms of time and cost [16], and requires highly skilled engineers to analyse the system, author and execute the test scripts and finally document the results in accordance with regulatory requirements. The extensive documentation required to prove a system is validated has also been cited as an impediment to innovation in regulated settings [17], due to regulatory requirements to revalidate an entire system if changes are made to the initial design [12].

2.2. Industry 4.0

The Industry 4.0 initiative was devised in 2011 as part of a German government strategy to maintain and develop its position as a global leader in the manufacturing sector and it was defined as "the technical integration of CPS into manufacturing and logistics and the use of the Internet of Things and Services in industrial processes" [18]. However, as research into this topic has evolved, more comprehensive definitions have been presented, based on what individual authors or institutions perceive to be the essential components of Industry 4.0. One such example of a comprehensive definition is "the Fourth Industrial Revolution can be best described as a shift in the manufacturing logic towards an increasingly decentralised, self-regulating approach of value creation, enabled by concepts and technologies such as CPS, IoT, IoS, cloud computing or additive manufacturing and smart factories, so as to

help companies meet future production requirements" proposed by [19]. Although there is no single definition for Industry 4.0, it can be generally described as the application of enabling technologies, such as the Internet of Things (IoT) devices, Artificial Intelligence (AI) and simulation and modelling techniques to transform manufacturing systems into Cyber Physical Systems (CPS) and enhance processes throughout the value chain within an organisation [20]. Equipment design for Industry 4.0 is defined by four key design principles [21]:

- Interconnection
- Information transparency
- Technical assistance
- Decentralised decisions

Implementation of these technologies can enable organisations to develop new digital business models, reduce operating costs, improve equipment efficiency, enhance product quality, and enable flexible manufacturing operations [22,23].

2.3. Model-Based Design

Simulation and modelling are key technologies for the development and understanding of system behaviour to optimise and support decision making in the design of complex systems [24]. One approach to system simulation is model-based design, which uses system models to support design, analysis and verification activities during the development and operation of a system [25]. The MBD approach places the system model at the centre of all stages of the design process, enabling the design engineers to simulate and assess system behaviour from the initial design through to implementation on the production equipment [26]. This approach is widely used in the automotive and aerospace sectors to assist with the design and validation of complex control and safety systems [27–29]. The use of MBD in these sectors has resulted in reduced development time [30], increased testing coverage [31], shorter commissioning [32] and significant cost savings [33]. More recently, the use of MBD to generate PLC code in the form of structured text is becoming more prevalent in industrial manufacturing environments, with applications including process control [34,35], safety systems [36] and equipment optimisation [37,38].

2.4. Equipment Control and Programmable Logic Controllers

Equipment control in industry is commonly achieved using PLCs, due to their high reliability and ubiquity [39]. Originally developed to replace hard-wired relay logic circuits, PLCs have been developed for many applications, including motion control, batch process control, distributed control systems and sequencing applications [40]. PLCs operate in a cyclical fashion, also known as a scan cycle, which consists of four main steps: input scan, programme scan, output scan and housekeeping (see Figure 2) [41]. They can be programmed using a variety of programming languages outlined in the IEC 61131-3 standard, which has been adopted by most major PLC manufacturers [42]. This standard consists of three programme organisation units (programmes, function blocks and functions) which can be constructed from five programming languages: three graphical (ladder diagram, function block diagram and sequential function chart) and two textual (instruction lists and structured text) [43].

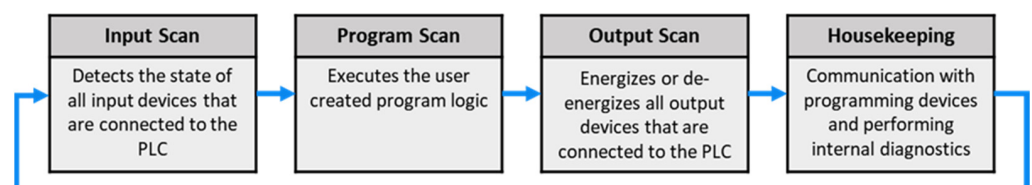


Figure 2. PLC Scan Cycle.

2.5. Model Checking of PLC Code

The application of modelling to test PLC programmings is a widely studied topic of research, with publications in this area generally focusing on the application of model checking tools to various PLC programmes. Typically, this research focuses on the challenges associated with converting a specific PLC language into a suitable model for common model checking tools such as NuSMV or UPPAAL [44]. Examples of this include [45], which details the development of model checking tool to verify PLC programmes written in structured text format [46], which details the formal verification of a PLC programme by translating it into a Unified Modelling Language (UML) model and verifying this model using the NuSMV model checking tool.

From a MathWorks perspective, the use of Simulink for the design and testing of control software is a common approach in the automotive and aerospace sectors. In [27], the Simulink design verifier was used to automatically generate test cases for boundary value testing in accordance with the ISO 26,262 standard. The research in [28] presented a methodology to automatically generate requirements-based tests that comply with the DO-178C standard and demonstrated the use of this methodology on a prototype model of an actuator system. A combination of model checking approaches with MathWorks occurred in [47] where Simulink models were used to design the on-board equipment for an automatic train protection system and were then converted to a NuSMV model for code validation.

3. Methodology

3.1. Proposed Methodology

This paper proposes a methodology using MBD for the design, validation, and automated generation of IEC 61131-3 compliant PLC code for a position control application of a servo motor. It builds on previous work carried out in [48] and applies it to a common industrial application to demonstrate the feasibility of adopting this methodology in regulated environments. The paper demonstrates how the model-based approach to control system design, which is currently used in the automotive and aerospace industries for development and testing of complex control systems, can be applied to the design and validation of control software for manufacturing equipment. This is accomplished using the MathWorks® Simulink® graphical programming environment, where a digital model of the equipment is created. The required control logic is created and validated by simulating its operation on the digital model. Finally, this validated control logic is converted to an IEC 61131-3 compliant function block, which can be operated on a PLC system, and, in the example illustrated here, used to control a servo motor.

3.2. Digital Validation Workflow

The digital validation workflow proposed in this paper is presented in Figure 3 and describes a cyclical approach which uses an equipment digital model to produce validated equipment phases for manufacturing applications. An equipment phase has been defined in the ISA-88 standard “as the lowest element of procedural control” [49] and specifies the commands that are to be sent to the equipment. Based on the initial high-level system requirements provided by the end user, a mechanical concept is created in traditional computer aided design (CAD) packages and then exported into a model-based design environment. Once exported, the CAD model can be used as a platform for developing and testing controller algorithms. These models can then be virtually tested against requirements within the modelling environment and documentation of each test result can be automatically generated to prove it is performing as specified. The validated algorithm can then be exported into the desired PLC programming environment as structured text and downloaded into the equipment. This workflow can be repeated for multiple design cycles to add additional functionality as required. The following sections present a detailed description of a proof-of-concept project on the feasibility of the proposed approach.

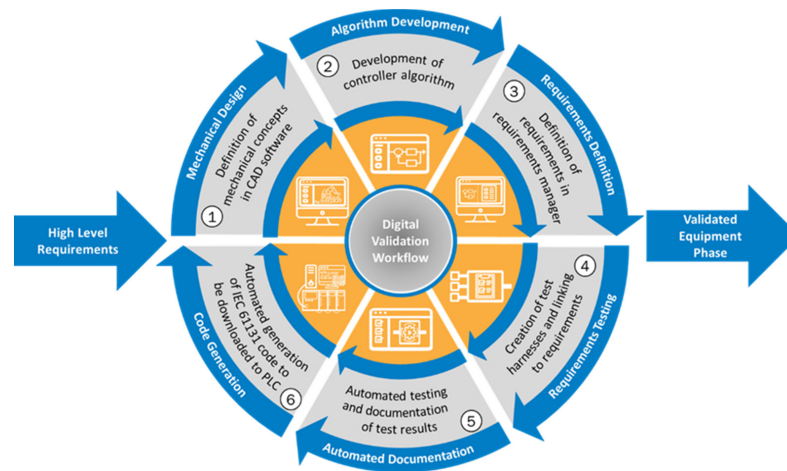


Figure 3. Proposed digital validation workflow for the creation of validated equipment phases.

3.3. Methodology Development and Modelling Approach

The objective of this approach is to enable automated validation of the equipment control logic used to determine machine behaviour. To do this, the modelling approach should reflect the current best practice for software creation and testing in industry. Currently, equipment control logic is manually written in an integrated development environment (IDE) provided by the equipment vendors for their specific equipment [50]. This logic is a combination of configurable function blocks, provided as pre-validated functions in the vendor’s IDE, and custom code is written by the end user specific to their desired function or use case. As per the GAMP 5 software classification, the custom code component of this logic is categorised as a Category 5–Bespoke software, which is a high-risk software component and requires comprehensive specification, risk assessment and testing in order to be validated [12].

Figure 4 presents the MBD architecture used to generate a model which supports digital validation. The model is comprised of two core elements, the physical system model, and the control logic model. The physical system model describes all the physical components of the equipment, including CAD data, physical constraints and kinematic relationships between components. The control logic model encompasses two sub-models, the function block model and the custom code model, which interact with each other to control the physical system. The function block model is comprised of models of pre-validated, configurable function blocks for common equipment functions, which may be provided by the technology provider. The custom code model contains the application specific code that controls the equipment process required by the end user. Dividing the control logic model in this way enables the equipment vendors to supply pre-validated models of their function blocks to the end user. These pre-validated models can then be used in conjunction with other code models, to simulate the functionality of the control logic model on the physical system model and to digitally validate their equipment.

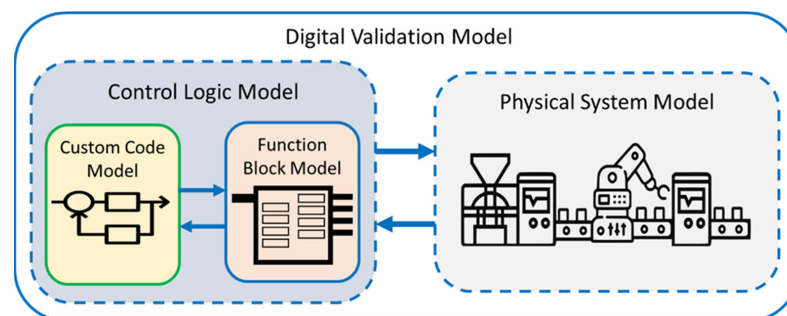


Figure 4. Digital validation modelling architecture.

3.4. Experimental Setup and Application Description

MathWorks Simulink® 2021B was selected as the model-based software platform due to its comprehensive catalogue of software tools, which provide all of the functionality required to enable digital validation. One key feature of Simulink that was critical to its selection as the model-based platform was the Simulink PLC Coder tool. No other model-based platform has integrated PLC code generation capabilities, to the best of the authors' knowledge, making Simulink uniquely suited to industries where equipment control is predominantly programmed with PLCs. Its wide range of compatible PLC software targets PLC software vendors, such as Rockwell, Siemens, B&R, PLC Open and others, enabling the equipment designer to take a vendor agnostic approach to equipment design.

Table 1 presents the Simulink modules and their associated function used throughout this work. Solidworks® 2021 from Dassault Systems was used to create the initial model of the mechanical system and Studio 5000® Logix Designer from Rockwell Automation was used to configure the control environment of the physical system.

Table 1. Simulink modules and respective functions required for digital validation.

Module	Function
Simscape Multibody	Simulation Environment for 3D Mechanical Systems
StateFlow	Control Logic Modelling Environment
Simulink Requirements	Requirements Management
Simulink Test	Test Management
Simulink Design Verifier	Design Error Detection
Simulink PLC Coder	Automated PLC Code Generation

The chosen application for the proof of concept of the approach was the creation of a state machine to interface with the Rockwell Automation Motion Axis Move (MAM) function block and control the position of a servomotor. Servomotors are a commonly used actuating component in manufacturing equipment, thus the demonstrator is applicable to multiple manufacturing domains. A state machine is a computational model which may only be active in one of a finite number of states at a given time and is defined by its initial state, a list of its possible states, and the inputs required to trigger a state transition. State machines are commonly used to perform a predetermined set of functions based on a specific combination of inputs [41]. The physical demonstrator used in this work was a Rockwell Automation Training Workstation (Figure 5), controlled by an Allen-Bradley CompactLogix® 5380 PLC.

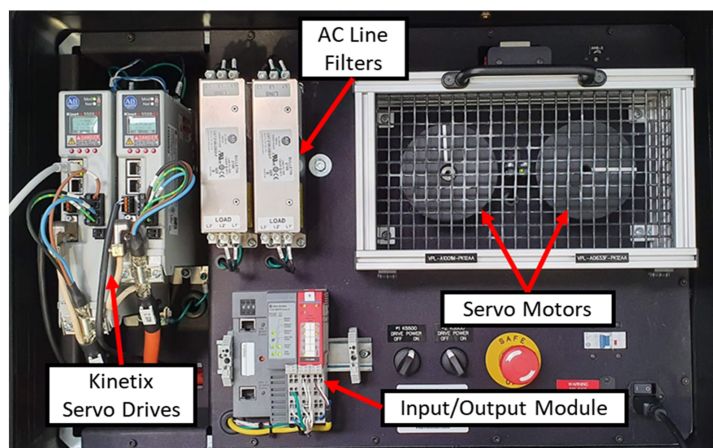


Figure 5. Rockwell Automation Training Workstation, incorporating two servo motors (only one is used in the application in this paper).

3.5. Physical System Model Development

The creation of a digital model of the mechanical equipment is the first step towards digital validation. The CAD model used in this paper is shown in Figure 6a, modelled using Solidworks. The completed model was exported from Solidworks using the Simscape™ multibody link plugin and converted by Simscape Multibody within Simulink into an equivalent Simscape Multibody block diagram Figure 6b.

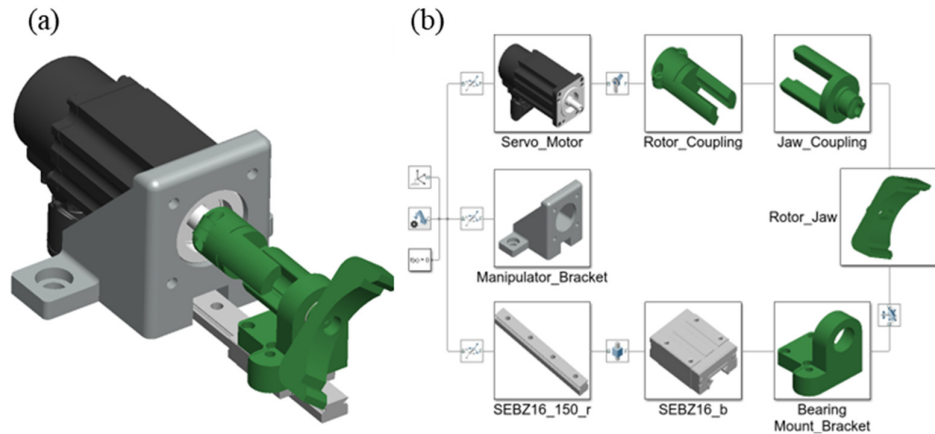


Figure 6. (a) CAD model of System in Solidworks®: (b) Simscape™ Multibody Block diagram in Simulink®.

3.6. Control Logic Model Development

As previously mentioned, the control logic model contains two sub models: the pre-validated models of the technology providers function blocks and the model of the end user’s custom code. The standard industry practice when programming motion control is to use preconfigured function blocks supplied by a technology provider [51]. With this work being a proof of concept, there are currently no technology providers who supply Simulink models which perform the functionality of their equipment function blocks.

Therefore, to prove the feasibility of this work, a model, representing the behaviour of the Rockwell Automation MAM status bits, was created in StateFlow to model this equipment function block. The state logic of the function block model (MAM model) is displayed in Figure 7, which represents the behaviour presented in Table 2 [50].

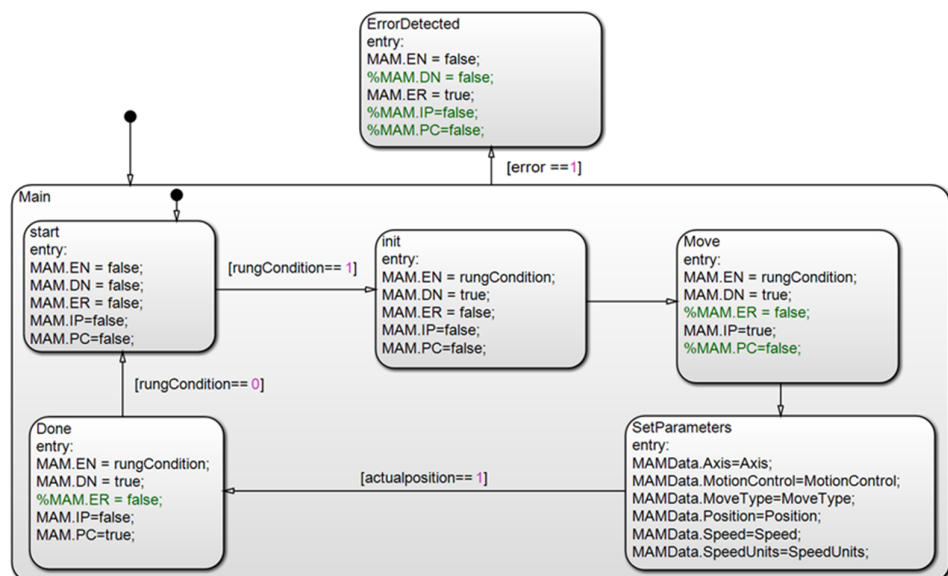


Figure 7. State Logic of the MAM model (function block model).

Table 2. Motion Axis Move Status Bit Behaviour from [50].

To See If	Check If This Bit Is Set To	Data Type	Notes
A false-to-true transition caused the instruction to execute	EN	BOOL	The EN bit stays set until the process is complete and the rung goes false
The move was successfully initiated	DN	BOOL	
An error happened	ER	BOOL	
The axis is moving to the end position	IP	BOOL	Any of these actions stop this move and clear the IP bit: <ul style="list-style-type: none"> • The axis gets to the end Position • Another MAM instruction supersedes this MAM instruction • Motion Axis Stop instruction • Merge from another instruction • Shutdown command • Fault Action
The axis is at the end position	PC	BOOL	<ul style="list-style-type: none"> • The PC bit stays set until the rung makes a false-to-true transition. • The PC bit stays cleared if some other action stops the move before the axis gets to the end Position.

The model of the custom code is presented in Figure 8. This model is based on the ISA-88 Procedural Model for batch control, which divides the operating cycle of equipment into a series of states [52]. Each state describes the equipment actions in response to various commands. In this model, the *running state* contains an equipment phase “pos” which contains the logic required to interact with the MAM model and control the position of the servo motor.

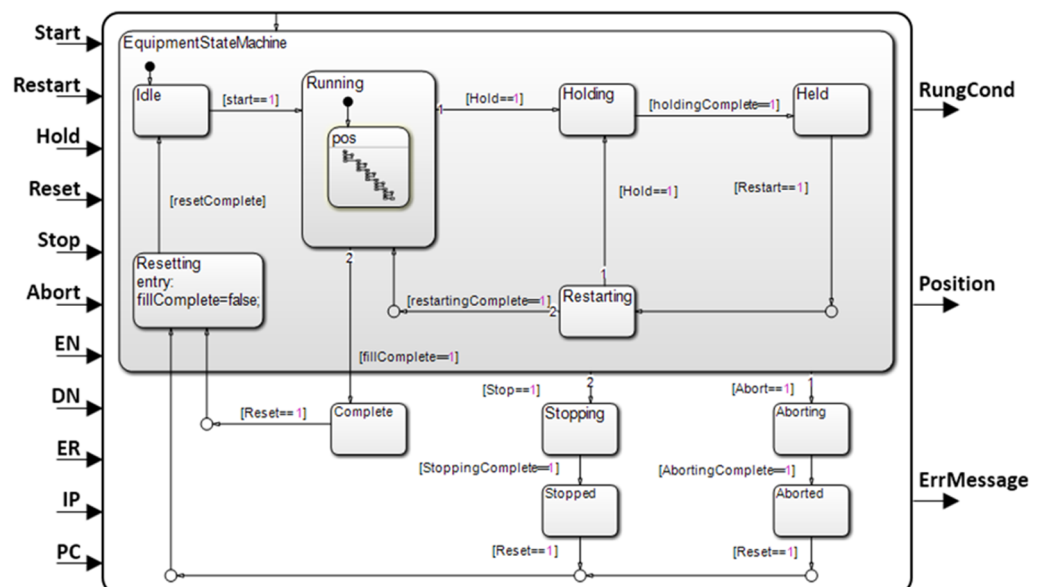


Figure 8. State machine model for custom code.

This logic would move the servo motor through five position commands with a two second pause between each position. The system was initialised with the servo motor positioned at 0° and the arbitrary position commands for the logic were 90°, 180°, 270°, 360° and back to 0°. An example of the StateFlow logic for setting the position value to 90° is provided in Figure 9.

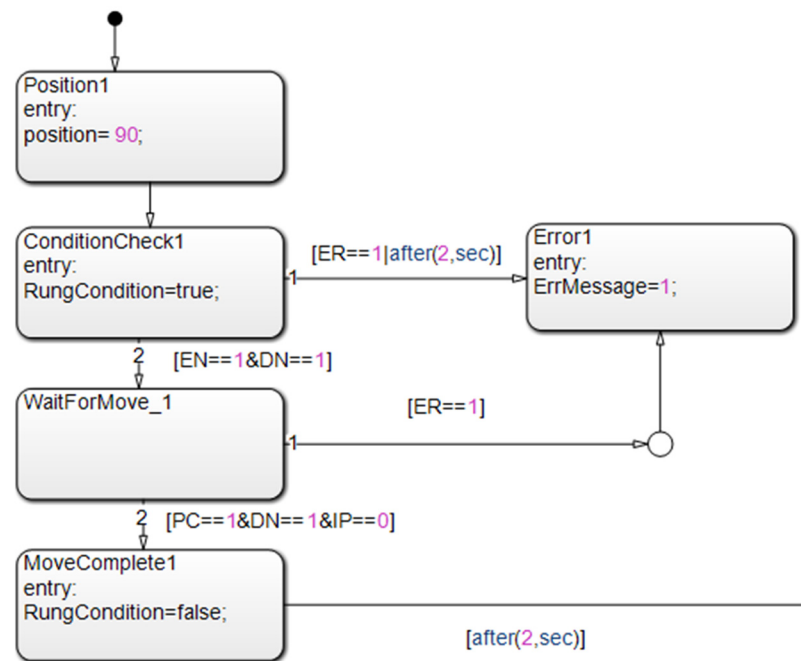


Figure 9. Sample StateFlow logic used to set position value to 90°.

The objective of this work was to show that the control logic developed in the MBD environment could interact with standard PLC programming functions to perform an operation. It should be noted here that, although 0° and 360° physically occupy the same position in terms of rotation, in the PLC control environment they were two unique position commands, and the behavior of the motor reacted differently to each one. For example, if the motor was to move from 180° to 0° it would rotate counterclockwise until it reached 0°, but if it was to move from 180° to 360° it would rotate clockwise until it reached 360°.

3.7. Definition of Requirements and Automated Testing

Traceability between requirements and test scripts is a key output of the validation process. Simulink Requirements Editor was used in this work to author and manage the system requirements as it offers the capability to link requirements to various model elements and test scripts, thus achieving the traceability required to prove that the model was validated.

Table 3 displays the sixteen requirements which characterise the behaviour of the model in response to the input signals. Simulink Design Verifier was used to automatically generate a test harness for the model. A test harness is an instance of a model where input values can be simulated, and the corresponding outputs are monitored [53]. In this instance, the test harness was generated in Simulink Design Verifier by defining the model element for custom code (Figure 8), as the component to be tested. The test inputs were automatically generated by the design verifier tool with the model coverage objective being Modified Condition Decision Coverage. These analyse whether the conditions within decisions independently affect the decision outcome during execution [54]. The resulting test harness contained forty-one test cases simulating all possible operating scenarios of the state machine and its equipment phases. Appendix A of this paper contains an example of a typical test case, its associated data and an explanation of each element of the test case.

The generated test harness was then used to form a test suite in Simulink Test Manager, where logical assessments could be applied to assess event occurrence, and all test scenarios could be automatically simulated. The results of the test simulation were then examined and linked to the relevant requirements to validate the model. Documentation of the test results were automatically generated using the built-in report generator in the Test Manager tool.

Table 3. Summary of sixteen functional requirements describing the model behaviour.

ID	Summary	Description
1	Initialise State	The system must Initialise into the Idle state
2	Idle to Running Transition	When the system is idle and the start input becomes true, then it must transition into the Running state and execute the running logic.
3	Running to Holding Transition	When the Running state is active and the hold command becomes true, then it must transition to the holding state and execute the holding logic.
4	Holding to Held Transition	Once the holding logic has been executed, the system must transition to the Held state.
5	Held to Restarting Transition	When the Held state is active and the Restart command becomes true, then the system must transition to the Restarting state and execute the restarting logic.
6	Restarting to Running Transition	Once the restarting logic is complete, the system must transition to the Running state.
7	Restarting to Holding Transition	When the Restarting state is active and the hold command becomes true, then the system must transition to the Holding state and execute the holding logic.
8	Running to Complete Transition	When the running logic is complete, the system must transition from the Running state to the Complete state.
9	Complete to Resetting	When the Complete state is active and the Reset command becomes true, then the system must transition to the Resetting state and execute the resetting logic.
10	Resetting to Idle Transition	Once the Resetting logic is complete, the system must transition to the Idle state.
11	Transition to Stopping	If the stop command becomes true, then the system must immediately transition to the Stopping state and execute the stopping logic.
12	Stopping to Stopped Transition	Once the stopping logic is complete the system must transition to the Stopped state.
13	Stopped to Resetting Transition	When the Stopped state is active and the Reset command becomes true, then the system must transition to the Resetting state and execute the resetting logic.
14	Transition to Aborting	If the abort command becomes true, then the system must immediately transition to the Aborting state and execute the aborting logic.
15	Aborting to Aborted Transition	Once the aborting logic is complete the system must transition to the Aborted state.
16	Aborted to Resetting Transition	When the Aborted state is active and the Reset command becomes true, then the system must transition to the Resetting state and execute the resetting logic.

3.8. Automated PLC Code Generation and Operation on the Physical Equipment

Figure 10 illustrates the process for automatic generation of PLC code from the Simulink model. The validated model was exported from the MBD platform using the Simulink PLC coder tool to the relevant equipment IDE, providing a hardware-independent IEC 61131-3 compliant code from the modelling environment. The model was exported as an Add On Instruction (AOI) for the Rockwell Automation Studio 5000 programming environment. Once imported into Studio 5000, the AOI was inserted into a ladder diagram routine containing the MAM instruction. The required inputs for both the imported AOI and the MAM instruction were then mapped to the relevant data tags required for operation within the routine.

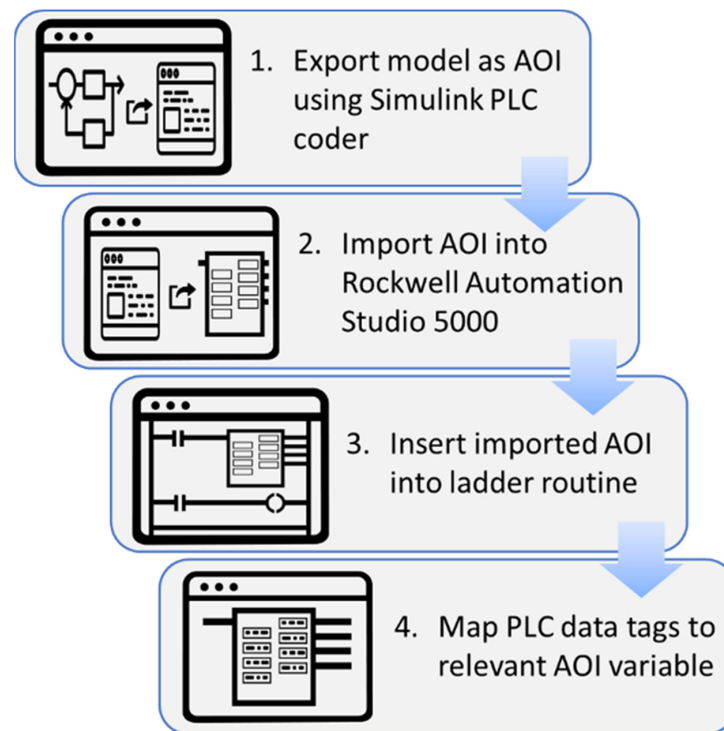


Figure 10. Automatic PLC code generation process.

4. Results

4.1. Results of Automated Testing and Report Generation

Simulink Test manager was used to automatically simulate the test harness generated in Section 3.7. All forty-one test cases were successfully simulated, and all logical assessments were satisfied. A test report was automatically generated from these tests to provide documentation on test metrics, such as pass/fail results of each test and cyclomatic complexity to indicate the complexity of the logic and coverage metrics, such as decision and condition coverage to assess test completeness. The results of these metrics for the testing carried out in this work is presented in Figure 11. From this, it can be seen that 100% test coverage of decision and condition metrics across all model elements was achievable using the Simulink design verifier tool.

Model Hierarchy	Complexity	Decision	Condition
1. EquipmentFillPhase	91	100%	100%
2. . SF: EquipmentFillPhase	90	100%	100%
3. . . SF: EquipmentFill	79	100%	100%
4. . . . SF: Running	65	100%	100%
5. SF: pos	65	100%	100%
6. SF: EquipmentFillPhase/EquipmentFill.Running.pos	65	100%	100%

Figure 11. Test Metrics Summary displaying the Model Hierarchy and the associated Complexity, Decision and coverage metrics for each model element.

4.2. Traceability between Tests and Requirements

As each test is linked to its associated requirement, the Simulink Requirements module was used to automatically generate a traceability matrix and report detailing the implementation and verification status of each requirement and its corresponding model element. The requirements traceability matrix for the model used in this work is displayed in Figure 12 and showcases the links between each requirement and its corresponding test case.

The requirements report that was automatically generated by Simulink requirements contained all details associated with each requirement including requirement type, description, links to other model elements, its implementation status and verification status.

	abort_SingleAxis_tests	New_TestSuite_1	SingleAxis_Harmon1	TestCase_02	TestCase_03	TestCase_04	TestCase_05	TestCase_06	TestCase_07	TestCase_08	TestCase_09	TestCase_10	TestCase_11	TestCase_12	TestCase_13	TestCase_14	TestCase_15	TestCase_16	TestCase_17
SingleAxisRequirements																			
Req1 Initialise State				↙															
Req2 Idle to Running Transition					↙														
Req3 Running to Holding Transition						↙													
Req4 Holding to Held Transition							↙												
Req5 Held to Restarting Transition								↙											
Req6 Restarting to Running Transition									↙										
Req7 Restarting to Holding Transition										↙									
Req8 Running to Complete Transition											↙								
Req9 Complete to Resetting												↙							
Req10 Resetting to Idle Transition													↙						
Req11 Transition to Stopping														↙					
Req12 Stopping to Stopped Transition															↙				
Req13 Stopped to Resetting Transition																↙			
Req14 Transition to Aborting																	↙		
Req15 Aborting to Aborted Transition																		↙	
Req16 Aborted to Resetting Transition																			↙

Figure 12. Requirements Traceability Matrix displaying the Links between each requirement (left) and its corresponding test case (top), with the soft return figure identifying the link.

4.3. Operation of the Code on the Physical Equipment

Figure 13 displays the monitored values of the following: (a) the position command sent to the MAM instruction and (b) the resulting position of the physical servo motor, sampled at 0.02 s intervals, while the automatically generated code was running on the PLC. It can be seen from Figure 13 that the position of the servo motor followed the prescribed motion profile set by the AOI generated from the Simulink model. There were slight discrepancies between the two signals (not visible in Figure 13), such as the slight overshoot of the actual position value as the servo moved to a new position command and the slope of the servo motor position as it moved from one position to the next. These discrepancies could be attributed to the fact that the position command signal was a digital signal which could change instantaneously during a PLC scan cycle, whereas the actual position was the measured value of the servo motor as it moved through the series of position commands.

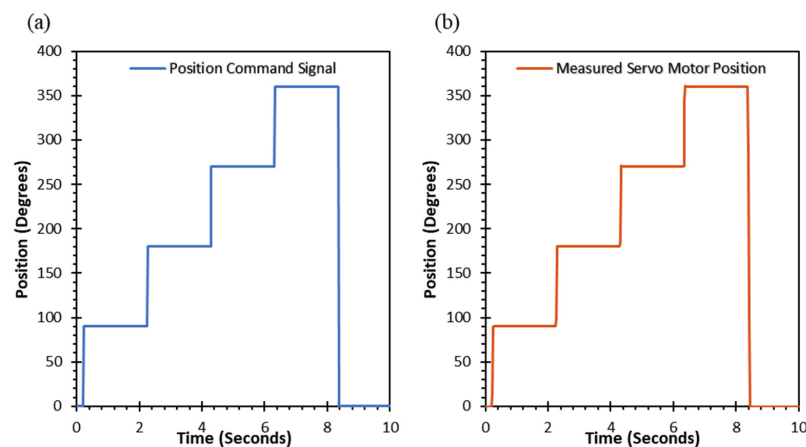


Figure 13. Graphs plotting (a) the required servo motor angular position against time and (b) the measured servo motor position against time.

Based on these results it could be determined that the automatically generated code functioned correctly and successfully controlled the MAM instruction so that the servomotor followed the prescribed motion profile set by the equipment phase. This demonstrated

that it is feasible to use a model-based approach to generate equipment phases which can interact and control technology provider specific function blocks.

5. Discussion

The shift to Industry 4.0 and flexible manufacturing systems will significantly change how equipment is designed, validated and operated in manufacturing environments. One of the major trends emerging in the medical device and pharmaceutical industries because of Industry 4.0 is personalised healthcare [55]. This promises the delivery of custom medical products based on the individual patient's condition and physiology. Although a significant research effort is being invested in the development of patient specific products, ranging from custom drug delivery solutions [56] to bespoke orthopaedic implants [57], this approach generally appears to be clinically driven and is usually only taken in exceptional circumstances where semi-urgent care is required. If personalised healthcare is to be accessible to every individual, the effects of this shift on the manufacturers of regulated products must be considered. It is expected that personalised healthcare will result in many manufactures having to embrace the "batch of one" concept, where each batch contains its own unique product [55,58]. This will require manufacturers to be able to rapidly reconfigure and validate machine functions to produce patient specific products.

While manufacturers in regulated industries are generally slow to adopt new manufacturing concepts and technologies, they recognise that there is a requirement for a new approach to validation to account for recent developments in Industry 4.0 [2]. If manufacturing equipment configurations are to change repeatedly to accommodate changes to the product design for customised products, there must be a method to ensure the validated state of the equipment is maintained. The current approach to validation, the GAMP 5 V model, is a rigid process which would result in equipment repeatedly being taken offline to carry out manual validation activities and is, therefore, not suitable to the proposed flexible manufacturing approaches. Additionally, the cost of repeatedly validating equipment would be prohibitive to the majority of manufacturers. The digital validation workflow presented in Figure 3 was developed with the objective of addressing these issues. With the model as the focus of all stages of the design process, engineers can continually test and validate new functions virtually before the code is generated and sent to the physical equipment. The digital validation modelling architecture described in Figure 4 was structured to minimise the amount of validation required when changes are made to the model. By separating the custom code models from the standard function blocks, a design engineer is only required to revalidate the custom code which interacts with these function blocks when a functional change is requested.

In contrast to the traditional checking approaches described in [44–46], where the PLC code is written first and is then converted into a model to be analysed by a model checking tool, this paper proposes that Simulink be used to create a model of the system control logic. This model can then be used as the basis to generate test cases for validation and to generate PLC code using the built in PLC code generator. Designing the control logic in this manner eliminates any issues associated with translation between PLC code and the model. Furthermore, adoption of this approach enables testing to occur much earlier in the development cycle, resulting in the identification of errors or bugs in the control logic before the code is generated.

Leveraging the automated test generation capabilities of Simulink Design Verifier in this work enabled complete coverage testing of the control logic while also eliminating the requirement to manually create test scripts to validate the system model. Combining this with the capabilities of Simulink Requirements to link model elements and requirements with their associated tests, this paper has shown that it is feasible to automatically generate documented evidence that the model performs as specified to satisfy regulatory requirements. It is expected that this approach would result in improved quality and a significant reduction in the cost and time required to validate a system. This is corroborated by [26], who displayed how automated testing with MBD resulted in increased test coverage com-

pared to manual methods in an industrial project to develop a spatial frequency sensor system, and [59], who reported that the use of MBD for testing and documentation during the development of rail propulsion systems resulted a cost reduction of 45%. Another benefit to this approach is the capability to automatically rerun the initial test scripts following an update to ensure that the original functionality has not been affected by the addition of new features and is still performing as specified. This has the potential to enable equipment designers to adopt an agile approach to equipment design by creating standard function blocks which can have additional features added in future as per user desires.

Although MBD has previously been used to generate PLC code for industrial equipment in [36], there is little evidence in the literature to suggest that the use of MBD to develop and validate equipment software has been considered by manufacturers of equipment for regulated environments. This paper aimed to address this gap in the literature by presenting a novel approach to the design and validation of custom software for equipment control. This was accomplished by leveraging the capabilities of MBD to automate the testing, documentation and code generation steps in equipment design and applying this approach to a common industrial application. It is expected that this work will provide a basis for demonstrating the benefits of adopting model-based practices to equipment manufacturers in regulated industries.

The successful operation of the PLC code for the test case examined in this paper verified that the simulation model and results accurately represented the physical system, proving that the proposed approach is a viable method to produce validated equipment control logic. Although the Simulink PLC coder presents the opportunity to generate code automatically for multiple technology provider platforms and, in principle, enables a technology provider agnostic approach to equipment design, some considerations for the target hardware are required. The design engineer requires a detailed understanding of the PLC platform or equipment on which the code will run, as many PLC environments include libraries of function blocks that perform similar functions but receive and process data in slightly different manners. This is a potential barrier to adoption of the approach in an industrial environment, due to the requirement for model-based definitions of the technology provider specific function blocks, such as the MAM block developed in this work. A significant modelling effort would be required to generate accurate model-based definitions of all commonly used function blocks for different PLC platforms, and this is unlikely to be feasible for a single manufacturer. However, the potential benefits of adopting a MBD approach to validation, including automated requirements testing, automated software verification, automated documentation and code generation, indicates that the creation of these models would improve validation activities across the regulated industry and warrants further research to enable this approach, perhaps adopting an all-industry approach where technology providers provide their function block models in accordance with standards, such as PLC Open.

6. Conclusions

The methodology proposed in this paper has proven that a model-based approach to equipment control design is technically feasible, can be used to rapidly test equipment control logic and to automatically generate the required PLC code. From the proof of concept, it has been determined that implementation of the proposed workflow on a common industrial application resulted in the successful validation of equipment modules for use in regulated manufacturing environments. Using the Simulink modules as described in the proposed workflow would enable engineers to achieve complete test coverage of the equipment control logic. It has been shown that test reports could be automatically generated to serve as validation documentation and prove that the equipment control logic is performing as specified.

The work carried out in this study has proven that it is technically feasible to use an MBD approach to model, test and generate validated equipment control logic for manufacturing equipment in regulated manufacturing sectors, such as medical device and

pharmaceutical manufacturing facilities. Further work on this approach aims to determine potential methods to enable mass adoption of this approach for equipment validation activities in regulated manufacturing environments. Other topics in this area that may warrant further investigation include the use of additional applications, such as Simulink Check, to further test and improve model quality before code generation.

Author Contributions: Conceptualization, E.P.H.; Funding acquisition, N.P.O. and C.T.M.; Methodology, D.M. (Damian McCarthy); Project administration, D.M. (Dermot McMorrow), N.P.O. and C.T.M.; Software, D.M. (Damian McCarthy); Supervision, D.M. (Dermot McMorrow), N.P.O. and C.T.M.; Visualization, D.M. (Damian McCarthy) and E.P.H.; Writing—original draft, D.M. (Damian McCarthy); Writing—review & editing, D.M. (Dermot McMorrow), N.P.O., C.T.M. and E.P.H. All authors have read and agreed to the published version of the manuscript.

Funding: This publication has emanated from research conducted in the Confirm Smart Manufacturing Research Centre, with the financial support of Science Foundation Ireland (SFI) under Grant Number SFI/16/RC/3918, co-funded by the European Regional Development Fund.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: Damian McCarthy acknowledges the assistance of Shane Loughlin of SL Controls in the development of the model used in this paper.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A

Test Case 3

Summary.

Length: 1 second (6 sample periods)
Objectives Satisfied: 9

Objectives.

Step	Time	Model Item	Objectives
2	0.2	Transition "[Stop==1]" from "EquipmentFill" to "Stopping" State "EquipmentFill" Transition "[start==1]" from "Idle" to "Running"	14. expression "Stop==1" false [0] 17. Substate executed "Idle" [0] 31. expression "start==1" true [0]
3	0.4	State "EquipmentFill" Transition "[Hold==1]" from "Running" to "Holding" State "pos"	20. Substate executed "Running" [0] 37. expression "Hold==1" true [0] 84. Substate exited when parent exits "Position1" [0]
4	0.6	State "EquipmentFill" Transition "[holdingComplete==1]" from "Holding" to "Held"	16. Substate executed "Holding" [0] 30. expression "holdingComplete==1" false [0]
6	1	State "EquipmentFill"	22. Substate exited when parent exits "Holding" [0]

Generated Input Data.

Time	0–0.1	1
Step	1–5	6
start	1	1
Restart	0	1
Hold	1	1
Reset	1	0
Stop	0	0
Abort	0	1
EN	1	1
DH	0	1
ER	0	1
IP	1	0
PC	0	1

Expected Output. These output values are expected assuming that inputs that do not affect the test objectives (- in the table above) are given a default value - 0 for numeric types, and default value for enumerated types.

Time	0	0.2–1
Step	1	2–6
start	0	0
Restart	0	90
Hold	0	0

Figure A1. Example of typical test case and its associated data.

Explanation:

This is an example of a test case.

This test case satisfies Requirement 2–Idle to Running Transition.

The objectives table is a list of the model elements which were tested in this case and the associated test objective that this test case satisfied.

The Generated Input Data table records the value of each model input at every simulation step of the test case.

The Expected Output Table presents the expected simulation output values for this test case.

References

1. Food and Drug Administration. *Process Validation: General Principles and Practices*; U.S. Department of Health and Human Services: Washington, DC, USA, 2011.
2. Margetts, D.; Vuolo, M. Breaking with Tradition: Laying the Foundation for Validation 4.0. Available online: <https://ispe.org/pharmaceutical-engineering/march-april-2021/breaking-tradition-laying-foundation-validation-40> (accessed on 9 December 2021).
3. Weyer, S.; Schmitt, M.; Ohmer, M.; Gorecky, D. Towards Industry 4.0—Standardization as the crucial challenge for highly modular, multi-vendor production systems. *IFAC-PapersOnLine* **2015**, *48*, 579–584. [CrossRef]
4. Aheleroff, S.; Philip, R.; Zhong, R.Y.; Xu, X. The Degree of Mass Personalisation under Industry 4.0. *Procedia CIRP* **2019**, *81*, 1394–1399. [CrossRef]
5. Cohen, Y.; Faccio, M.; Galizia, F.G.; Mora, C.; Pilati, F. Assembly system configuration through Industry 4.0 principles: The expected change in the actual paradigms. *IFAC-PapersOnLine* **2017**, *50*, 14958–14963. [CrossRef]
6. Yu, J.; Yin, Y.; Sheng, X.; Chen, Z. Modelling strategies for reconfigurable assembly systems. *Assem. Autom.* **2003**, *23*, 266–272. [CrossRef]
7. Thramboulidis, K. The 3+1 SysML View-Model in Model Integrated Mechatronics. *J. Softw. Eng. Appl.* **2010**, *3*, 109–118. [CrossRef]
8. Vogel-Heuser, B.; Feldman, S.; Folmer, J.; Ladiges, J.; Fay, A.; Lity, S.; Tichy, M.; Kowal, M.; Schaefer, I.; Haubeck, C.; et al. Selected challenges of software evolution for automated production systems. In Proceedings of the 015 IEEE 13th International Conference on Industrial Informatics (INDIN), Cambridge, UK, 22–24 July 2015; pp. 314–321.
9. Ståhl, J.; Gabrielson, P.; Andersson, C.; Jönsson, M. Dynamic manufacturing costs—Describing the dynamic behavior of downtimes from a cost perspective. *CIRP J. Manuf. Sci. Technol.* **2012**, *5*, 284–295. [CrossRef]
10. U.S. Food and Drug Administration. International Medical Device Regulators Forum. Available online: <https://www.fda.gov/medical-devices/cdrh-international-programs/international-medical-device-regulators-forum-imdrf> (accessed on 15 November 2021).
11. ISPE. What Is GMP? Available online: <https://ispe.org/initiatives/regulatory-resources/gmp/what-is-gmp> (accessed on 15 November 2021).
12. ISPE. *GAMP® 5: A Risk-Based Approach to Compliant GxP Computerized Systems*; International Society for Pharmaceutical Engineering: Tampa, FL, USA, 2008.
13. International Society for Pharmaceutical Engineering. What Is GAMP? Available online: <https://ispe.org/initiatives/regulatory/what-gamp> (accessed on 30 March 2022).
14. Fanmuy, G.; Szczepaniak, R. Requirements Engineering in the bio medical industry: GAMP 5 and tooling. *INCOSE Int. Symp.* **2010**, *20*, 2430–2445. [CrossRef]
15. Margetts, A.J.; Lundsberg-Nielsen, L. The History & Future of Validation. Available online: <https://ispe.org/pharmaceutical-engineering/march-april-2021/history-future-validation> (accessed on 19 November 2021).
16. Engel, A. Systems Quality Costs in the Literature. In *Verification, Validation, and Testing of Engineered Systems*; John Wiley & Sons: Hoboken, NJ, USA, 2010.
17. Bennet, C.; Heesakkers, H.; Horneborg, S.; Langer, G.; Lundsberg-Nielsen, L.; Margetts, A.; Roder, F. Industry Perspective: Validation 4.0—Shifting Paradigms. 2020. Available online: <https://ispe.org/pharmaceutical-engineering/november-december-2020/industry-perspective-validation-40-shifting> (accessed on 21 November 2021).
18. Kagerman, H.; Wahlster, W.; Helbig, J. *Recommendations for Implementing the Strategic Initiative INDUSTRIE 4.0*; Final Report of the Industrie 4.0 Working Group: Frankfurt, Germany, 2013.
19. Hofmann, E.; Rusch, M. Industry 4.0 and the current status as well as future prospects on logistics. *Comput. Ind.* **2017**, *89*, 23–34. [CrossRef]
20. Culot, G.; Nassimbeni, G.; Orzes, G.; Sartor, M. Behind the definition of Industry 4.0: Analysis and open questions. *Int. J. Prod. Econ.* **2020**, *226*, 107617. [CrossRef]
21. Hermann, M.; Pentek, T.; Otto, B. Design Principles for Industrie 4.0 Scenarios. In Proceedings of the 49th Hawaii International Conference on System Sciences, Koloa, HI, USA, 5–8 January 2016.
22. Sony, M.; Antony, J.; McDermott, O.; Garza-Reyes, J.A. An empirical examination of benefits, challenges, and critical success factors of industry 4.0 in manufacturing and service sector. *Technol. Soc.* **2021**, *67*, 101754. [CrossRef]
23. Masood, T.; Sonntag, P. Industry 4.0: Adoption challenges and benefits for SMEs. *Comput. Ind.* **2020**, *121*, 103261. [CrossRef]

24. de Paula Ferreira, W.; Armellini, F.; De Santa-Eulalia, L.A. Simulation in industry 4.0: A state-of-the-art review. *Comput. Ind. Eng.* **2020**, *149*, 106868. [[CrossRef](#)]
25. INCOSE. *Systems Engineering Vision 2020*; International Council on Systems Engineering: Seattle, WA, USA, 2007.
26. Bergmann, A. Benefits and Drawbacks of Model-based Design. *KMUTNB Int. J. Appl. Sci. Technol.* **2014**, *7*, 15–19. [[CrossRef](#)]
27. Khastgir, S.; Dhadyalla, G.; Jennings, P. Incorporating ISO 26262 Concepts in an Automated Testing Toolchain Using Simulink Design Verifier™. *SAE Int. J. Passeng. Cars Electron. Electr. Syst.* **2016**, *9*, 59–65. [[CrossRef](#)]
28. Miranda, B.; Masini, H.; Reis, R. Using Simulink Design Verifier for Automatic Generation of Requirements-Based Tests. *FM 2015: Form. Methods* **2015**, *9109*, 601–604.
29. Lee, D.; Lee, D.; Na, J. Automatic Failure Modes and Effects Analysis of an Electronic Fuel Injection Model. *Appl. Sci.* **2022**, *12*, 6144. [[CrossRef](#)]
30. Mathworks(C). Alstom Generates Production Code for Safety-Critical Power Converter Control Systems. Available online: https://uk.mathworks.com/company/user_stories/alstom-generates-production-code-for-safety-critical-power-converter-control-systems.html (accessed on 13 December 2021).
31. Mathworks(C). ENGEL Speeds Development of Injection Molding Machine Controllers. Available online: https://uk.mathworks.com/company/user_stories/engel-speeds-development-of-injection-molding-machine-controllers.html (accessed on 13 December 2021).
32. Mathworks(C). Siemens Applies Model-Based Development and Commissioning for Industrial Assets. Available online: https://uk.mathworks.com/company/user_stories/case-studies/siemens-applies-model-based-development-and-commissioning-for-industrial-assets.html (accessed on 13 December 2021).
33. Maplesoft. Using Virtual Commissioning for a New, Competitive Injection Molding Machine. Available online: <https://www.maplesoft.com/company/casestudies/stories/virtual-commissioning-for-new-injection-molding-machine.aspx> (accessed on 13 December 2021).
34. Tapak, P.; Huba, M. Experimenting with Modified Smith Predictors Using B&R Automation Studio Target for Simulink. *IFAC Proc. Vol.* **2012**, *45*, 366–371.
35. Mystkowski, A.; Kierdelewicz, A. Fractional-Order Water Level Control Based on PLC: Hardware-In-The-Loop Simulation and Experimental Validation. *Energies* **2018**, *11*, 2928. [[CrossRef](#)]
36. Niang, M.; Riera, B.; Philippot, A.; Zaytoon, J.; Gellot, F.; Coupat, R. A methodology for automatic generation, formal verification and implementation of safe PLC programs for power supply equipment of the electric lines of railway control systems. *Comput. Ind.* **2020**, *123*, 103328. [[CrossRef](#)]
37. Mathworks(C). B&R Industrial Automation Improves Servo Drive Performance with Virtual Sensor Algorithms Developed Using Model-Based Design. Available online: https://uk.mathworks.com/company/user_stories/br-industrial-automation-improves-servo-drive-performance-with-virtual-sensor-algorithms-developed-using-model-based-design.html (accessed on 14 December 2021).
38. Bakht, M.P.; Salam, Z.; Bhatti, A.R.; Anjum, W.; Khalid, S.A.; Khan, N. Stateflow-Based Energy Management Strategy for Hybrid Energy System to Mitigate Load Shedding. *Appl. Sci.* **2021**, *11*, 4601. [[CrossRef](#)]
39. Bissell, C. A History of Automatic Control. In *Springer Handbook of Automation*; Springer: Berlin, Germany, 2009; pp. 64–65.
40. Love, J. Programmable Logic Controllers. In *Process Automation Handbook: A Guide to Theory and Practice*; Springer: London, UK, 2007; pp. 337–343.
41. Advanced Micro Controls Inc. What Is a PLC? Available online: <https://www.amci.com/industrial-automation-resources/plc-automation-tutorials/what-plc/> (accessed on 15 December 2021).
42. John, K.-H.; Tiegelkamp, M. The IEC 61131 standard. In *IEC 61131-3: Programming Industrial Automation Systems*; Springer: Berlin, Germany, 2011; pp. 12–20.
43. *IEC 61131-3; Programmable Logic Controllers—Part 3: Programming Languages*. International Electrotechnical Commission: Geneva, Switzerland, 2009.
44. Ovatman, T.; Aral, A.; Polat, D.; Unver, A.O. An overview of model checking practices on verification of PLC software. *Softw. Syst. Model.* **2016**, *15*, 937–960. [[CrossRef](#)]
45. Darvas, D.; Adiego, B.F.; Vinuela, E.B. PLCverif: A tool to verify PLC programs based on model checking techniques. In Proceedings of the 15th International Conference on Accelerator and Large Experimental Physics Control Systems (ICALPECS2015), Melbourne, Australia, 17–23 October 2015.
46. Klotz, T.; Fordran, E.; Straube, B.; Haufe, J. Formal Verification of UML-modeled Machine Controls. In Proceedings of the Proceedings of 12th IEEE International Conference on Emerging Technologies and Factory Automation, Palma de Mallorca, Spain, 22–25 September 2009.
47. Ferrari, A.; Grasso, D.; Magnani, G.; Fantechi, A.; Tempestini, M. The Metrô Rio ATP Case Study. In *International Workshop on Formal Methods for Industrial Critical Systems*; Springer: Berlin/Heidelberg, Germany, 2010.
48. McCarthy, D.J.; Hinchey, E.P.; O'Dowd, N.P.; McCarthy, C.T.; McMorro, D. Using Model Based Design as an Enabler for Digital Validation of Discrete State Machines in Regulated Manufacturing Environments. *Procedia Manuf.* **2021**, *55*, 365–370. [[CrossRef](#)]
49. Hawkins, W.M.; Fisher, T.G. 88 Batch Control Concepts, Part 1. In *Batch Control Systems*; ISA—The Instrumentation, Systems, and Automation Society: Durham, NC, USA, 2006; p. 113.

50. Logix 5000 Controllers Motion Instructions. Available online: https://literature.rockwellautomation.com/idc/groups/literature/documents/rm/motion-rm002_-en-p.pdf (accessed on 11 December 2020).
51. PLCopen.org. Creating Reusable, Hardware Independent Motion Control Applications via IEC 61131 3 and PLCopen Function Blocks. Available online: <https://plcopen.org/technical-activities/motion-control> (accessed on 15 May 2022).
52. *Integrated Architecture™: Foundations of Modular Programming*; Rockwell Automation: Milwaukee, WI, USA, 2009.
53. Mathworks(C). Test Harnesses. Available online: <https://uk.mathworks.com/help/sltest/test-harnesses.html> (accessed on 16 February 2022).
54. Mathworks(C). Types of Code Coverage. Available online: <https://www.mathworks.com/help/slcoverage/ug/types-of-code-coverage.html> (accessed on 24 May 2022).
55. DBEI. *Ireland's Industry 4.0 Strategy 2020–2025: Supporting the Digital Transformation of the Manufacturing Sector and Its Supply Chain*; Government of Ireland: Dublin, Ireland, 2019.
56. Tan, D.K.; Maniruzzaman, M.; Nokhodchi, A. Advanced Pharmaceutical Applications of Hot-Melt Extrusion Coupled with Fused Deposition Modelling (FDM) 3D Printing for Personalised Drug Delivery. *Pharmaceutics* **2018**, *10*, 203. [[CrossRef](#)] [[PubMed](#)]
57. Willemsen, K.; Nizak, R.; Noordmans, H.J.; Castelein, R.M.; Weinans, H.; Kruijt, M.C. Challenges in the design and regulatory approval of 3D-printed surgical implants: A two-case series. *Lancet Digital Health* **2019**, *1*, e163–e171. [[CrossRef](#)]
58. Manufacturing Personalised Meds: What Needs to Change? 2018. Available online: https://pharma.nridigital.com/pharma_special_oct18/manufacturing_personalised_meds_what_needs_to_change (accessed on 17 January 2022).
59. Mathworks(C). Bombardier Transportation Implements Model-Based Design to Accelerate Rail Propulsion System Development. Available online: https://www.mathworks.com/company/user_stories/bombardier-transportation-implements-model-based-design-to-accelerate-rail-propulsion-system-development.html (accessed on 3 December 2021).