



**University of  
Nottingham**  
UK | CHINA | MALAYSIA

# Development of Data Models and Adaptation Strategy for Self-Configuring Production Systems


Thesis submitted to the University of Nottingham for the degree of  
**Doctor of Philosophy, July 2024.**

**Hamood Ur Rehman**

**20240963**

Supervised by

**Prof. Svetan Ratchev  
Dr. Jack C. Chaplin  
Mr. Leszek Zarzycki**

Signature  \_\_\_\_\_

Date 10 / 07 / 2024

# Abstract

Manufacturing intelligence is the ability to gather and analyse data for decision-making in production systems towards reaching an objective. This project involves introducing intelligence in production systems for achieving self-configuration. This is done by conceptualising and developing the intelligent components that act as building blocks of the self-configuring production systems and their application in the system. The approach taken for the project is iterative and built from the bottom-up. A classification tool is developed to study the capability of self-configuration in the current industrial production system, followed by insight gathered through surveys. The theoretical aspects involving self-configuration are discussed, and a general adaptation strategy is developed that integrates self-configuration capabilities in the intelligent components. These components are then manipulated and controlled through the use of technologies. Tools and techniques involving asset administration shell, state charts/state machines, multi-agent system, information model and machine learning approaches were studied. These technologies were implemented to achieve self-configuration, leveraging data gathered during operation. This research is applied to use cases of an industrial leak test equipment MALT and on a force testing station of the PRIME assembly system. The dissemination of work is highlighted, and future possibilities are expanded.



## Acknowledgements

I extend my deepest gratitude to the University of Nottingham and TQC Ltd. for their support throughout this research project. I am immensely grateful to my supervisors Prof. Svetan Ratchev, Dr. Jack C. Chaplin and Mr. Leszek Zarzycki, whose guidance, expertise, and encouragement were invaluable in shaping the direction of this research. I want to thank Dr. Samanta Piano for providing feedback for improving the thesis.

I also wish to express my appreciation to the research participants and work colleagues, especially Mr. Mark Jones, Mr. Phil Tonge and Mr. Gavin Murray, whose contributions provided critical insights and enriched the findings of this work.

I am indebted to the hosting partners of the project for their continual encouragement and fruitful discussions, which significantly contributed to the development of this research. I would like to express my appreciation towards collaborators who worked together with me to produce research outputs.

Additionally, I extend my heartfelt thanks to my family, especially my father Nawaid Iqbal Sheikh (Late) for the unwavering support, patience, and understanding during the long hours devoted to this research.

Finally, I am thankful to all members of the DiManD Innovative Training Network (ITN) project for funding this research, arranging training schools, and providing networking opportunities.

This research is funded by the European Union Marie Sklodowska-Curie Innovative Training Networks (H2020-MSCA-ITN-2018) program under grant agreement number no. 814078.

# Contents

<b>Abstract</b>	<b>i</b>
<b>Acknowledgements</b>	<b>ii</b>
<b>List of Tables</b>	<b>vii</b>
<b>List of Figures</b>	<b>viii</b>
<b>Acronym</b>	<b>1</b>
<b>Glossary</b>	<b>3</b>
<b>Chapter 1 Introduction</b>	<b>7</b>
1.1 Introduction . . . . .	8
1.2 Problem Statement . . . . .	9
1.3 Research Domain . . . . .	12
1.4 Aims and Objectives . . . . .	14
1.5 Thesis Structure . . . . .	15
1.6 Publications . . . . .	18
<b>Chapter 2 Literature Review</b>	<b>19</b>
2.1 Introduction . . . . .	21
2.2 Concept of Configuration in Production Systems . . . . .	23
2.3 Self-Configuration in Production Systems . . . . .	29
2.4 Machine Learning Applications . . . . .	35
2.5 Optimisation Algorithms for Parameter Determination . . . . .	38
2.6 Interoperability and Data Integration . . . . .	40

2.7	Applications of Self-Configuration . . . . .	43
2.8	Research Gap . . . . .	48
<b>Chapter 3</b>	<b>Methodology</b>	<b>63</b>
3.1	Introduction . . . . .	64
3.2	Requirements of the Approach . . . . .	65
3.3	Detailed Methodology . . . . .	67
3.4	Validation Methods . . . . .	75
3.5	Overview of the Approach . . . . .	77
3.6	Thesis Structure . . . . .	79
3.7	Conclusion . . . . .	80
<b>Chapter 4</b>	<b>Industrial Practice Survey For Adoption Of Self-Configuring Production Systems</b>	<b>81</b>
4.1	Introduction . . . . .	83
4.2	Survey Description . . . . .	85
4.3	Analysis And Result . . . . .	91
4.4	Threats to Validity . . . . .	103
4.5	Reflection on Impact . . . . .	104
4.6	Conclusion . . . . .	110
<b>Chapter 5</b>	<b>A Level-Based Classification of Technical En- ablers in Production Systems</b>	<b>112</b>
5.1	Introduction . . . . .	113
5.2	A Level-Based Classification for Self-Configuration . . . . .	114
5.3	Evaluating Enablers of Self-Configuration with the Level- Based Classification . . . . .	125
5.4	Conclusion . . . . .	137
<b>Chapter 6</b>	<b>Module Driven Configuration</b>	<b>139</b>
6.1	Introduction . . . . .	140

6.2	Overview of Framework . . . . .	141
6.3	Module Driven Configuration System	
	Model . . . . .	151
6.4	Conclusion . . . . .	182
<b>Chapter 7</b>	<b>Standard Configuration Model for Produc-</b>	
	<b>tion Systems in Manufacturing</b>	<b>184</b>
7.1	Introduction . . . . .	185
7.2	Concept of Configuration in Production Systems . . . . .	186
7.3	Configuration Model . . . . .	186
7.4	Impact on Configuration . . . . .	197
7.5	Towards Capturing Configuration for Production Systems . .	201
<b>Chapter 8</b>	<b>Self-configuring Production System: An Adap-</b>	
	<b>tation Strategy</b>	<b>203</b>
8.1	Introduction . . . . .	205
8.2	Overview of the Adaptation Strategy . . . . .	206
8.3	Capturing Configuration Module Information . . . . .	215
8.4	Deployment of Engine for Capturing Information . . . . .	216
8.5	State Chart to State Machine for Functionality Representation	220
8.6	Machine Learning for Self-Configuration . . . . .	223
8.7	Multi-Agent System Integration . . . . .	227
8.8	Real Time Control . . . . .	235
8.9	Simulation . . . . .	238
8.10	Conclusion . . . . .	240
<b>Chapter 9</b>	<b>Industrial Use-Case: Validating the Adap-</b>	
	<b>tation Strategy for Self-Configuring Produc-</b>	
	<b>tion Systems</b>	<b>242</b>
9.1	Introduction . . . . .	243
9.2	Methodology . . . . .	244

9.3	Implementation . . . . .	248
9.4	Experiments . . . . .	271
9.5	Conclusion . . . . .	317
<b>Chapter 10</b>	<b>Conclusion</b>	<b>321</b>
10.1	Research Questions and Contributions . . . . .	322
10.2	Summary of the Findings . . . . .	328
10.3	Outputs, Implications, and Novel Findings . . . . .	334
10.4	Limitations . . . . .	336
	<b>Bibliography</b>	<b>347</b>
	<b>Appendices</b>	<b>379</b>
<b>Appendix A</b>	<b>Survey Questionnaires</b>	<b>380</b>
A.1	Survey Questions . . . . .	380
A.2	Detailed Survey Questionnaire . . . . .	384
A.3	Survey Response . . . . .	398
<b>Appendix B</b>	<b>Code Walkthrough</b>	<b>411</b>
B.1	Code Walkthrough . . . . .	412
<b>Appendix C</b>	<b>State Chart</b>	<b>419</b>
C.1	Understanding States and Actions in State Charts . . . . .	419
<b>Appendix D</b>	<b>CAEX Standard</b>	<b>428</b>
D.1	Utilising the CAEX Standard . . . . .	428

# List of Tables

1.1	The accepted papers from this PhD research. . . . .	18
2.1	Definitions of Self-Configuration extracted from literature in Chronological Order . . . . .	32
3.1	Components of generalised data model for self-configuration.	71
3.2	Mapping the research questions to the industrial use-cases. .	76
4.1	Maturity level of Production Systems for Enabling Self-Configuration	107
4.2	Value Proposition Matrix for enabling Self-Configuration, linking stakeholders to lifecycle. . . . .	108
5.1	Enablers of Self-Configuration . . . . .	127
8.1	Data Pipeline Component Description for Production System	209
8.2	Agent Description for Production System . . . . .	229
10.1	Contributions to the Research Related to Chapters in the Thesis . . . . .	328

# List of Figures

1.1	Thesis structure, relating chapters to the specific research questions. The chapter-wise mapping of each research question is shown. . . . .	16
2.1	Adopted methodology for literature review . . . . .	22
2.2	Automated Configuration by Policy Rules, Database and Network Association aided by Daemon Processes (Bachula and Zajac (2013)). The configuration database is governed by the associated policy rules. The production systems have a digital interface on which these configurations can be adjusted. Background processes that ensure communication, and connections are also running on the system. . . . .	26
2.3	Configuration Cloning by Template in Production Environment (Jones and Romig (1991)). The cloning template can be altered through the interface and is displayed on HMI. The machine data is used to configure the production system through the template. . . . .	28
2.4	Relating Self-Configuration to value extraction levels (Kao et al. (2015)) in Intelligent Systems. As the level increases, more “value ” is obtained from the system. Configuration Level is the highest data and machine interaction level, where the machine can make independent decisions. . . . .	30

2.5	Hierarchy of Self-properties. Relating self-configuration to self-organisation/self-adaptation. The hierarchy begins at the base, progresses to the major, and finally reaches the general level. The self-* properties necessary for achieving self-adaptation in production systems are highlighted at each level with their respective self-* property. . . . .	34
2.6	Asset Administration Shell (AAS). Representation of an asset with its main components for I4.0 compliant communication with IoT infrastructure within a production facility. .	41
2.7	A production system consists of multiple granular objects (modules). These modules can have different granular levels. In this illustration, the granularity level is considered at the gripper. Each module adds value, imparting functionality to the system. . . . .	51
2.8	Decomposition Style in Production Systems. Modularity is defined at different granularity levels depending on decomposition styles. (a) Physical Decomposition, where granularity is defined at each component in the system. For example, in the picture, granular boundaries are considered at each manufacturing asset. (b) Functional Decomposition, where functionality is defined where the granularity boundary lies. In the picture, the two assets and the conveyor system are considered as a single granular boundary since their functionality adds value. . . . .	53



2.9	Modularity is dependent on a careful assessment of the physical and cyber components of the production system. Granularity is linked with modularity. The granularity level is linked with the collection of mechatronic components responsible for functionalities within the production system. In the picture, there are four modules. . . . .	54
3.1	The methodology approach taken for this research. . . . .	67
3.2	Simple data flow diagram of the research. Data from the production system is stored in its asset administration shell. A change in production conditions or performance requirements results in configuration setting change through the interaction of agent systems and machine learning model. . .	70
3.3	Data modelling generalisation for capturing capability, relationship, constraint, and operation information for self-configuration. Information interchange happens between hardware abstraction and AAS for self-configuration by querying from endpoints during the execution of functionality. . . . .	71
3.4	Technological development for the self-configuration research. Each stage behind the development of the adaptation strategy is shown. . . . .	74
3.5	The leak testing setup: (a) the cylinder volumes under test and MALT test system being a part of the test bench for general leak testing (b) The PRIME force testing setup involving a force test station, two robots and a shuttle on the rail. . . . .	76

3.6	Proposed Self-Configuration Approach for the research. Integrating Asset Representation, Agent System, and Parameter Optimisation. The diagram shows component mapping with research problem objectives. These components work together to assist in the realisation of the self-configuration adaptation strategy . . . . .	78
3.7	Ph.D. Thesis structure, relating the contributions to needs and dividing them into chapters. . . . .	79
4.1	Collaboration related to production system development. The adaptation strategy targets the last configuration stage of the production system development. . . . .	84
4.2	Structure of the questions in the surveys. . . . .	90
4.3	Focus of the surveys carried out. Planning is done to gain insight from professionals on self-configuring aspects of production systems. . . . .	91
4.4	An infographic of the Surveys. The main findings of the survey are presented. Relation between industrial challenges, data in manufacturing, and self-configuring systems explored. A need exists for addressing challenges in data security, demand, and adaptation. Self-configuring systems will provide a solution to the lack of skills in machine settings and the shortage of field experts. . . . .	92
4.5	Criteria of intelligence in production system. Each criterion of intelligence in the production system in the industry is shown and participants are asked to rank their current system on it. . . . .	94

4.6	Conceptual business model indicating in detail the strategic components, customer and market components, & value and creation components for self-configuring production systems.	109
4.7	The status for innovation; A reflection on the initial and desired business model through the introduction of self-configuration in production systems. . . . .	110
4.8	An illustration for potential value utilisation/realisation by self-configuration strategy in production systems. New value generated by the solution can serve as direction for business model. . . . .	110
5.1	Level Based Classification for Self-Configuration. The degree of the capability in production system is outlined with dividing the levels into two aspects of System Readiness and System Execution. The System Readiness is from Level-0 to Level-3 and System Execution is from Level-4 to Level-5. The Stage-Wise Transition (S1-S8) presents the transformation of production system for incorporating self-configuration across levels. . . . .	116
5.2	Level Based Classification for Self-Configuration applied to Enablers along with Stage-Wise transition Identification. Note that there is only one occurrence of Stage 7 in the diagram and none of Level-5. . . . .	131
5.3	Industrial production systems for demonstrating the applicability of level-based classification: (a) Basic differential pressure Leak Tester; (b) Multi-Application Leak Tester (MALT); (c) Multi-Agent integrated with MALT System . . . . .	132
5.4	Level Based Classification for Self-Configuration applied to basic differential pressure Leak Tester . . . . .	134

6.1	Conceptual illustration of the framework, showing aspects of the self-configuring production system. Production System Model presents a runtime of the Physical Production System acting as its digital twin. Adaptation Strategy monitors and controls the Production System Model in response to changes. Production System Model updates the physical system under guidance from Adaptation Strategy. . . . .	146
6.2	Adaptation Loop for the process in the framework to achieve self-configuration in production systems. <i>Evaluator</i> , evaluates the rules for the configuration change, <i>change manager</i> , guides the configuration change for functionality and <i>executor</i> changes the configuration on the system. . . . .	150
6.3	The production system model is represented at three levels of abstraction namely: <i>Functionality</i> , <i>Configuration</i> and <i>Constraint</i> . . . . .	152
6.4	Module Evolution in the Configuration Change Process. The illustration shows changes in module w.r.t variable change ( <i>middle</i> )( $v_5$ to $v_6$ ) and ( <i>right</i> ) new module introduction. . .	155
6.5	Representation of a Structural Module. Functionality, configuration, and constraints are encapsulated in this Structural Module defined through operational semantics. A structural module consists of variables, rules (functionality rules and variable rules), allocation and link. . . . .	157
6.6	Module Structure connected with the elements of AAS. Variable rules govern variables. Functionality is executed subjected to constraints defined by rules (functionality and variable). Relationships are encapsulated in allocation and link. . . . .	158

6.7	Configuration space $X \times Y$ of variables $X$ and $Y$ . $CC$ are the candidate configurations satisfying relationship and constraints. $A, B$ and $C$ are those candidate configurations that satisfy the production system requirements. . . . .	160
6.8	An illustration of a Module-based system having two modules hosting functionalities $M_1$ and $M_2$ . Functionality rules can be shared between modules, meaning they can govern the functionality of both modules and also define a dependency between them. . . . .	171
6.9	An example of a Module-based system having two modules $M_1$ and $M_2$ hosting leak test functionality. Both modules can perform leak test functionality subjected to rules $(R)$ . Variable rules $(R_v)$ are governing individual module variables and functionality rules $(R_f)$ govern the functionality behaviour of both modules. . . . .	172
6.10	An illustrative example of static configuration through the framework in a Configurable Object Based System. The $G_{pr}$ and $F_{pr}$ act on variables to update them through interface $I_{pr}$ . . . . .	178
6.11	Module-based system application on a production system used to leak test products developed at an SME (TQC Ltd.). The configuration in terms of variables, relationships, and constraints is identified. The variable rules and functionality rules are presented. . . . .	180

7.1	Envisioning the standard configuration model for production control in Production System. Three layers model is established (1) Configuration modelling layer that capture the information, (2) Mapping layer that maps the model with other submodels ; product, functionality, capability and services, (3) Production control layer that executes functionality in the production system with the configuration. . . . .	187
7.2	Establishing the scope of the configuration modelling in production systems. The production system consists of configurable objects (physical or cyber objects), and there exists a mapping of variables to values in those configurable objects subjected to constraints. . . . .	187
7.3	The UML representation of Configuration in Production system. Functionality presents configuration requirements, depending on capability, product and service requirements. Variable and Functionality rules impact the configuration. (An explanation of the arrows in the UML diagram can be found in figure 7.4) . . . . .	190
7.4	The arrows in UML represent relationships between classes. An explanation of UML arrows in the research is illustrated.	191
7.5	The hierarchy of configuration operation. The dotted lines represent the instances which carry out the specific type of operation. Abstract instances carry out those abstract operations governed by a production plan, while internal physical and external physical operation are carried out by the production system instance. For a successful configuration operation abstract operation, internal physical operation and external physical operation should have matching capabilities and requirements. . . . .	194

7.6	The product contains an ID, specification, and configuration. The attributes can possess values for a property. For classifying the product, categorisation, and specification description may be used. . . . .	199
7.7	A simple capability and skill representation . The capability is based on skill and drives its properties from configuration. Skill is a more descriptive concept containing unique identification, description and assigned controller. Functionality influences the capability directly as functionality generates certain dependencies that must be fulfilled for it to execute.	200
7.8	A simple service representation. The service requirement is dependent on configuration. Service matching is necessary to determine the compatibility of the service with requirements. Each offered service has an Id, Skill (i.e. encapsulating capability), and a service description. . . . .	201
7.9	The UML representation of Configuration in Production system. UML representation of individual components is detailed, which is established in this chapter. . . . .	202
8.1	Mapping Level Based Classification for Self-Configuration to Production system components. The classification features are mapped to the system to provide the focus for manufacturers for improvement. . . . .	208
8.2	Design Principles of Adaptation Strategy. . . . .	210

8.3	The proposed Adaptation Strategy integrated with CAEX Engine. CAEX stands for Computer Aided Engineering eXchange, a standard used to describe the structure of manufacturing data. The CAEX Engine aids the management and exchange of engineering data in a standardized format (more explained in a later section). It consists of three layers: Production System Coordination (High-Level), Production System Runtime (Mid-Level), and Production System Drivers & Control (Low Level). . . . .	212
8.4	A configurable object module is mapped to CAEX engine components. The functionalities are mapped to the system unit class object instantiated in terms of variables as internal elements within an instance hierarchy with the corresponding rules. The allocation and links are mapped to role class library and interface class library respectively. . . . .	217
8.5	An illustration linking elements of an AAS to the CAEX Engine. Variables are housed as IEs, initiated as an instance of system unit class (SUC). The relationships are captured in IEs, interface instances (I) and role classes (R). Constraint information is captured in IEs and Is of CAEX engine. . . .	218
8.6	Asset Administration Shell representation of an asset with its main components for I4.0 compliant communication with IoT infrastructure within a production facility. . . . .	219
8.7	State Machine encapsulating functionality state charts as behaviours. It provides valid transition control mechanism for the production system to follow. . . . .	221
8.8	Proposed workflow for machine learning in the production system. The workflow defines the approach to use dependent on data and requirements. . . . .	224



8.9	Illustration of the mechanism for the prediction of configuration values using the AutoML approach. . . . .	225
8.10	The generalised elaborative approach for predicting configuration values for configurable objects in production systems.	227
8.11	Sequence diagram for functionality execution using agent application in self-configuring production systems. . . . .	230
8.12	RTC and the interaction of its components with hardware abstraction for module-driven self-configuration. The application layer interacts with the configuration module in a production system through its interaction layer. Adaptation strategy acts on the configuration module. . . . .	237
8.13	Illustration of simulation for configuration module. . . . .	239
9.1	Routing design with the interface for self-configuration demonstration. . . . .	249
9.2	Activating the state chart transition for self-configuration . . . . .	253
9.3	Generating part Requirement for the Introduced Part . . . . .	253
9.4	Setting Requirements for the Functionality . . . . .	254
9.5	Matching functionality with requirements . . . . .	255
9.6	Variable Value Determination for Functionalities . . . . .	256
9.7	Deploying variables on production system and executing functionalities . . . . .	257
9.8	A simple glimpse of the Tool Utility developed to aid the Production System Coordination Layer . . . . .	258
9.9	Routing design for MALT with an interface for self-configuration demonstration. . . . .	261
9.10	Routing design for PRIME with an interface for self-configuration demonstration. . . . .	262
9.11	An illustration of the logistic regression algorithm. . . . .	264

9.12	An illustration of Genetic Algorithm . . . . .	266
9.13	NAS algorithm for determination of image detection and classification ML Model . . . . .	267
9.14	Level Based Classification for Self-Configuration applied to Multi-Application Leak Tester (MALT). The identified fea- tures need to be improved to enable MALT to achieve self- configuration. This happens through Stage-Wise Transition explained before. . . . .	273
9.15	Capturing MALT system information in CAEX Engine. . . .	274
9.16	AAS for MALT . . . . .	275
9.17	Generic asset administration shell representation ( <i>left</i> ). Skill submodel is developed, submodel elements contains the skill API calls along with pertaining data that can be used by the client services. Component manager is deployed when the submodels are listed ( <i>right</i> ). . . . .	276
9.18	Detail of each submodel to execute “leak test” functionality.	276
9.19	Summary of experimentation carried out for developing base- line and data-set for self-configuration on MALT. The pa- rameters for each volume are varied over the range men- tioned. The range is volume and MALT-specific (constraints).	280
9.20	Sequence diagram for a distributed manufacturing environ- ment with self-configuration and testing. . . . .	282
9.21	Self-Configuration library formulation with Endpoint hosted on Google Cloud Platform (GCP). Developing parametric optimisation code that is modular and can execute an ex- periment in 4-5 lines. . . . .	285

9.22	The leak testing setup: (a) the cylinder volumes under test (b) MALT test system being a part of test bench for general leak testing (c) Interface for leak testing; agent system drives the execution. . . . .	288
9.23	Execution of the self-configuration framework on MALT sys- tem. (a) PA (representing a product) is entered into the system. (b) TA transports the product to the manufactur- ing system (MALT) location. (c) Manufacturing system is self-configured and test is executed by agent coordination through components of data pipeline.(d) TA transports the product from the system once test is carried out. . . . .	288
9.24	Sequence Diagram for self-configuration and Test Process (JADE Interface). . . . .	289
9.25	Change in configuration settings of the manufacturing sys- tem (MALT). (a) No configuration setting (b) Configuration change after execution before test. . . . .	289
9.26	Validation of Adaptation Strategy for leak test functionality on MALT. The time taken to determine a valid configura- tion is significantly less through the research approach in comparison to an Expert. Also, the approach allows updat- ing this configuration automatically, while the expert has to enter the values manually. . . . .	290
9.27	Mapping project research questions to MALT Use-Case Im- plementation. . . . .	291
9.28	Level Based Classification for Self-Configuration applied to Multi-Agent & ML integrated MALT System. The features have been significantly enhanced making MALT capable of self-configuration. . . . .	293

9.29	Level Based Classification for Self-Configuration applied to PRIME test station . . . . .	297
9.30	Capturing PRIME test station information from the CAEX Engine . . . . .	298
9.31	AAS for PRIME test station . . . . .	299
9.32	Architecture of the Cloud Based Machine Learning Pipeline for Configuration Setting Detection. . . . .	303
9.33	ML Pipeline Deployment. (a) Precision vs Recall against identified part labels, for fixture identification. Precision is a measurement of positive label assignment (ratio between the True Positives and all the Positives) and recall is the measure of the model correctly identifying True Positives. (b) Maximum confidence at Threshold = 0.5. (c) Dataset size (total and test images). (d) Confusion Matrix representing True and Predicted Labels. . . . .	304
9.34	Sequence diagram for the force-test use-case incorporating image oriented configuration setting. . . . .	307
9.35	Sequential diagram for functionality execution on the PRIME test station. . . . .	310
9.36	PRIME experimental setup involving teststation, two robots and a shuttle on rail. . . . .	310
9.37	The force test functionality execution on PRIME. The robots and the test station are allocated respective locations. The agent system routes the part to the test station where depending on the fixture the part is identified and the force test configured. . . . .	311
9.38	Configuration change in the PRIME test station (a) before (b) after detecting the part. . . . .	311

9.39	Validation of Adaptation Strategy for force test functionality on PRIME. The time taken to determine a valid configuration is significantly less through the research approach in comparison to an Expert. Also, the approach allows updating this configuration automatically, while the expert has to enter the values manually. . . . .	312
9.40	Mapping project research questions to PRIME test station Use-Case Implementation . . . . .	312
9.41	Level Based Classification for Self-Configuration applied to Multi-Agent & ML integrated PRIME test station. The features have been significantly enhanced making MALT capable of self-configuration. . . . .	314
9.42	A practical example from TQC (Industrial Partner) on configuration costs incurred in respective projects. This presents a potential for value utilisation by self-configuration strategy, reducing these costs during deployments. . . . .	319
9.43	A plot demonstrating the value utilisation potential on industrial partner data by adaptation strategy. . . . .	320
10.1	Contributions of the research towards the research questions	324
10.2	Level Based Classification for Self-Configuration and Mapping to Production System. . . . .	330
10.3	An illustration of a Module-based system having two modules hosting functionalities $M_1$ and $M_2$ . . . . .	331
10.4	The developed Adaptation Strategy with supporting tools at a Glance. . . . .	331
10.5	Limitations of the research. More detail on this is expanded on in this section. . . . .	337
C.1	A basic state representation using statecharts. . . . .	420

C.2	An collaborative illustration on state charts for a production system. (a) Guards and actions during the transition,(b) time-triggered transition, (c) segmented transition, (d) regions ,(e) concurrent transitions, (f) complex transitions, (g) state transition, (h) history states. . . . .	422
D.1	CAEX architecture for capturing information on the module and operational semantics (Drath (2021)). . . . .	429
D.2	Representation of functionality operation as a Submodel in Asset Administration Shell deployed through CAEX engine. <i>Context</i> is a combination of identifier, kind, semantic ID, qualifiable, and data specification for a production operation. <i>Event</i> defines the change happening as a result of the operation, <i>Submodel Collection</i> contains information and functions about the operations relevant for that production system, <i>Capability</i> is the ability or extent of the operation, and <i>Operation</i> contains information about the input, output and in-out condition for the operation. <i>Value</i> and <i>Value Type</i> are used to assign value as per a unit to the property.	432

---

# Acronym

Acronym	Meaning
AAS	Asset Administration Shell
AI	Artificial Intelligence
API	Application Programming Interface
C	Configuration
CAEX	Computer Aided Engineering Exchange
CO	Configurable Object
CNP	Contract Net Protocol
DiManD	Digital Manufacturing and Design
DF	Digital Factory
DM	Digital Manufacturing
EU	European Union
FMS	Flexible Manufacturing Systems
GA	Genetic Algorithm
GCP	Google Cloud Platform
H2020	Horizon 2020
HMS	Holonic Manufacturing Systems
ITN	Innovative Training Network
I/O	Input and Output
IoT	Internet of Things

---

JADE	Java Agent Development Framework
KPI	Key Performance Indicators
MALT	Multi-Application Leak Tester
ML	Machine Learning
MSCA	Marie Skodowska-Curie Innovative Training Networks
OEE	Overall Equipment Effectiveness
Op	Operation
PA	Product Agent
PLC	Programmable Logic Controller
PSO	Particle Swarm Optimisation
RA	Resource Agent
Rel	Relationship
RL	Reinforcement Learning
RMS	Reconfigurable Manufacturing Systems
RQ	Research Question
RTC	Real Time Control
Q	Question
SME	Small and Medium-sized Enterprise
TA	Transport Agent
UML	Unified Modelling language
V	Variables



---

# Glossary

Some common terms used in this PhD thesis are defined :

<b>Term</b>	<b>Definition</b>
Adaptation	The process of making system configuration suitable for a purpose.
Alarm leak rate	Threshold that triggers a 'fail'.
Alarm differential pressure	Pressure difference threshold also signaling a 'fail'.
Allocation	Relating variables with data values.
Agent	Program that performs actions autonomously based on behaviours to meet goals. It can also interact with other agents and perform negotiations. Collectively agent system, consisting of more than one agent, can show emergent behaviour.
AMPQ	A robust, feature-rich messaging protocol ensuring reliable and secure message delivery for enterprise applications.
Capability	The ability of a system to execute an operation.
Centralised Database	A database that is located, stored, and maintained in a single location

Cloning	Template driven change under the assumption that all machines have identical basic file systems and specific configurations
Configuration	A set of objects, the characteristics of these objects: their extrinsic and intrinsic properties, and the connections between them.
Configurable Object	Those components in the production system that can be adapted (adjusted).
Constraint	The limitations imposed on the production system.
Decomposition	The process of simplifying a mechatronic aggregate.
Fill time	Duration of pressurisation.
Functionality	Is the behaviour in the form of capabilities of configurable object
Functionality Rule	These are used to govern the behaviour of the production system as the functionality is executed.
Granularity	The size of an individual mechatronic aggregate.
Initial Delay	A pause allowing the system to settle after pressurisation.
Isolation delay	A pause after system isolation before measurement.
Key Performance Indicator	A quantifiable measure of performance over an objective.
Link	Connection between variables and the production system.
Machine Learning	The use of data and algorithms to generate insight that improves with iterations.

---

Traditional Manufacturing	Making or processing products from any form of input using equipment or similar means.
Manufacturing Asset	Something of value in manufacturing, such as a component or a device.
Measuring time	Duration of leakage measurement.
Modelling	Capturing and presenting features of a production system.
Module	Structural representation of a configurable object.
Modularity	The number of separate manufacturing aggregates present in the production system
MQTT	A lightweight publish/subscribe messaging protocol designed for low-bandwidth, resource-constrained IoT devices.
Offset compensation	Adjustment for inherent system leaks.
Optimisation	Process of finding the best variable values for a configurable object from among many possibilities that suit an objective.
Product	Something that is produced through a manufacturing activity.
Relationship	This represents the association between configurable objects.
Runtime	It is a layer of abstraction capturing system configuration, functionalities, and constraints.

Self-Configuration	It is the ability of a system to change its configuration (i.e., parameters, calibration, and the connection between different system modules) by installing, updating and (re)formulating to improve or restore system functionality in response to actions and a changing environment.
Services	External components that aid a production system in an objective.
Skill	The ability of the system at the moment to execute the operation.
Stabilisation time	Period for pressure to equalise within the product.
State	The current status of the system awaiting a transition.
Test Pressure	The pressure applied during testing.
Variables	The objects in a configuration that can be configured for an operation. These impact the operation of the configurable object and the production system.
Variable Rule	The constraints that involve configurable object variables and their values.
Vent off delay	A delay before measurement to ensure complete system pressurisation.
Venting time	Duration of pressure release.

---

# Chapter 1

## Introduction

**Contents**

---

1.1	Introduction . . . . .	<b>8</b>
1.2	Problem Statement . . . . .	<b>9</b>
1.3	Research Domain . . . . .	<b>12</b>
1.4	Aims and Objectives . . . . .	<b>14</b>
1.4.1	Aim . . . . .	14
1.4.2	Objectives . . . . .	14
1.5	Thesis Structure . . . . .	<b>15</b>
1.6	Publications . . . . .	<b>18</b>

---

**1.1 Introduction**

The advances in computer engineering, communication and information technologies have significantly impacted the manufacturing environment by finding applications in automation and robotic, data-driven decision-making, industry 4.0, enhanced connectivity and customisation (Karabegović et al. (2020); Kubler et al. (2013); Vaisi (2022); Kuan et al. (2011)). “Intelligent manufacturing” is the term used to describe the next generation of production systems where the production systems have the capability of making intelligent decisions about themselves. It is expected that intelligent manufacturing will be expanding in the years ahead, presented by (Kostal et al. (2019)), as more technologies are integrated and applications explored.

Technologies like machine learning, presented by (Sharp et al. (2018)), reinforcement learning, presented by (Aggour et al. (2019)) and cloud comput-

ing, presented by (Wu et al. (2014)), enable intelligent decision-making on devices and machines based on previous experiences and learning capabilities. Integration of these technologies assists in making effective decisions for machine production systems. Supporting frameworks that integrate these technologies and present a viable approach to deliver smart decisions lead to the current research trends.

Supporting intelligent decisions in manufacturing is accompanied by understanding the input requirements, and output requirements, such as product information, customer requirements, process information and equipment capability. These requirements vary significantly per the product, process, and application criteria in production systems. Understanding these requirements for a dynamic manufacturing environment and developing strategies to address these changing requirements is a growing research direction addressed in this thesis.

## 1.2 Problem Statement

Companies want to expand their business, get more customers, and address the competition. To achieve this they want to make better products, give more value to customers, and meet higher production targets. With ever-changing market constraints, customer requirements and the need for mass customisation for the business edge, the need for dynamic configuration and equipment adaptation has become of paramount importance.

For achieving the business edge in the light of stated goals, it is desirable by the companies that components in a production system demonstrate a modular behaviour i.e., can be easily integrated at different points on the system, and can perform its associated functionality. Additionally, they

should store or utilise data for decision-making for the purpose of quality control, traceability, and other performance criteria. It means that the components must show intelligent behaviour and be able to adapt to the requirements of the product being processed.

The ‘No-Faults-Forwards’ philosophy (Khan et al. (2014)) adopted in the current manufacturing paradigm entails testing to be incorporated in all processes in the operation sequence. This is important so that products may be produced with high accuracy and repeatability. This also helps companies to maintain a business edge as per discussed objectives. Modular components are essential for the ‘No-Faults-Forwards’ philosophy, as these could be used anywhere on the production system, ensuring product accuracy and repeatability. In order to realise such a modular component, this research directs a solution towards incorporating intelligence in the component level of the production system.

Research trends in the domain of intelligent manufacturing is mainly concentrated towards making the whole system intelligent, creating assumptions about the components (i.e. manufacturing assets) that form the building blocks of the production aggregate (Antsaklis et al. (1989); Zhong et al. (2017); Antzoulatos (2017)). These assumptions involve the capabilities of these components in relation to the output and input requirements. An example can be an operation that a manufacturing asset, in the system, can perform but under the consideration of the part clamped into the fixture. This means that the operation is linked with the clamping requirement. Another example could be the manufacturing asset’s parametric or setting limitation.

This research contributes to the conceptualisation and development of self-configuring production systems comprised of intelligent components and



their application or utilisation in the system. These intelligent components must be able to adapt to meet the requirements of the product, customer requirements, and production system. Such is only possible if the limitation of the previously discussed assumption is addressed. In this research, this problem is addressed. Since these components should be able to adapt (i.e. configured) as per requirements, they can be referred to as Configurable Objects (CO).

The configuration settings, for a CO, on the production system are dependent on the features of the product, the characteristic properties of the environment where the equipment is kept, the connected processes, and current business priorities.

The specific research questions can be listed as:

- **Theoretical Foundations:** What theoretical models are needed to underpin the concept of self-configuration in production systems?
- **Adaptation Strategy:** How can a general adaptation strategy be developed to integrate self-configuring COs into production systems?
- **Technology, Tools, and Techniques:** Which technologies, tools, and techniques best facilitate self-configuration and leverage operational data for iterative improvement?
- **Implementation:** How can this approach be implemented in a real-world scenario involving a product within the production system?
- **Business Alignment:** How can business objectives be translated downstream to guide the self-configuration process at the production system level?

## 1.3 Research Domain

The research project is directed at the self-configuration of production systems. The manufacturing process may involve a production system to be operated at different configuration settings, dependent on features, orientation, and topology (i.e. the way the part is introduced into the system). Configuration change in the production system is associated with changes in physical parameters, variations in calibration settings or the use of different system settings. Under conditions of time constraints, any performance criteria or any other production scenario, a product can be operated on different configurations in the same production system.

The majority of the work in this area has been carried out at the configuration change at the system level, but an approach to determine the right configuration settings of the machine by the machine itself in regard to the product and process remains a valid gap.

The focus of this research is towards understanding the domain of self-configuration by modelling the configuration aspect and using data to adapt the configuration as per needs. Technologies that realise this, by mapping the data to value-based decisions, are also researched. The approach to the determination of configuration settings, incorporating internal and external data, is also an investigation focus.

The concept of self-configuration involves the machine's capability to capture and utilise data. The data can be used for the decision-making aspect, thereby extracting 'value' from it. A reasoning approach is of paramount importance that captures and utilises data for decision-making. The reasoning approach should contribute to the objective of adapting the configurable object to the product requirements in a production system.

Relating data to value can give a direction to achieving the objective of self-configuration. Barring (2019) groups data accumulated during production operation into internal and external data streams. The internal stream of data primarily involves that data, contributing to value, that influences the internal state of the production system. The concept of self-configuration is the configuration which the equipment adopts for performing operations on the part or product. On the other side, external data groups those data aspects, contributing to value, that influence the production system. For self-configuration, it could be factors that affect the internal data like volume, priorities, or other key performance indicators (KPIs).

The Research Question (**RQ**) can be framed as;

***How to achieve self-configuration in production systems at the machine level?***

This research presents a **methodology and approach** to address this gap in knowledge, with a focus on testing systems or testing incorporated production systems. A leak testing machine, and a force and vision testing machine, industrial use cases, serve as deployment and evaluation systems for the developed approach. The developed approach that initially targets the testing can be generalised to a wide variety of production systems. The methods can be applied to machines in different application areas in manufacturing, the focus in this research because of its plug-and-play nature is on testing machines. **Tools and techniques** are explored that may work together to achieve complete self-configuration of production systems. **Theoretical foundations and adaptation approach** that basically utilises all the tools and techniques is highlighted.

## 1.4 Aims and Objectives

### 1.4.1 Aim

This research aims to develop a strategy for self-configuring production systems, enabling them to automatically adapt to changing requirements and optimise their operation.

### 1.4.2 Objectives

- **Develop theoretical models:** Create data and conceptual models to capture, store, and effectively utilise configuration information in production systems focused on testing applications.
- **Establish configuration abstraction:** Define relationships between elements of production system configuration, configuration change operation, functionality execution under a configuration, rules governing configuration in systems, and capabilities in a system to guide the self-configuration logic.
- **Identify enabling technologies:** Investigate and select the most suitable technologies, tools, and techniques to achieve system self-configuration in real time.
- **Design system architecture:** Construct a modular and adaptable architecture that facilitates the self-configuration process.
- **Validate implementation strategy:** Develop and deploy an adaptation strategy that allows the production system focused on testing applications to seamlessly reconfigure itself to meet changing internal and external requirements.

## 1.5 Thesis Structure

To capture configuration settings and realise self-configuration, in the thesis the concept of configuration is discussed in detail with reference to production systems in manufacturing. The concept of self-configuration is elaborated with some focus on breaking it down to a set of features that could be mapped to any production system. This is mainly done to make the topic more relatable to the machines on the shop floor and spark an interest towards achieving self-configuring objectives. Leveraging on the developed interest, the building blocks of the system configuration are expanded and the rules of the game are set. The information modelling for the configuration is detailed and the strategy to achieve configuration is presented. Tools and techniques are used to encapsulate the execution of research towards the realisation of a self-configuration strategy. In a simplistic view, the approach that is presented realises self-configuration by capturing information (e.g. constraint, relationship, and variables). This information is manipulated to achieve self-configuration by using an agent system through machine learning (ML). More detail on this is discussed in respective chapters. A brief description can be understood by the figure 1.1.

The thesis structure is as follows;

- **Chapter 1 Introduction** introduces the problem and establishes the research definition and motivation. The domain and scope of research are identified, and a general approach is represented. (**Theoretical Foundations**)
- **Chapter 2 Literature Review** represents some literature and background work on the thesis topic, along with identifying the Gap

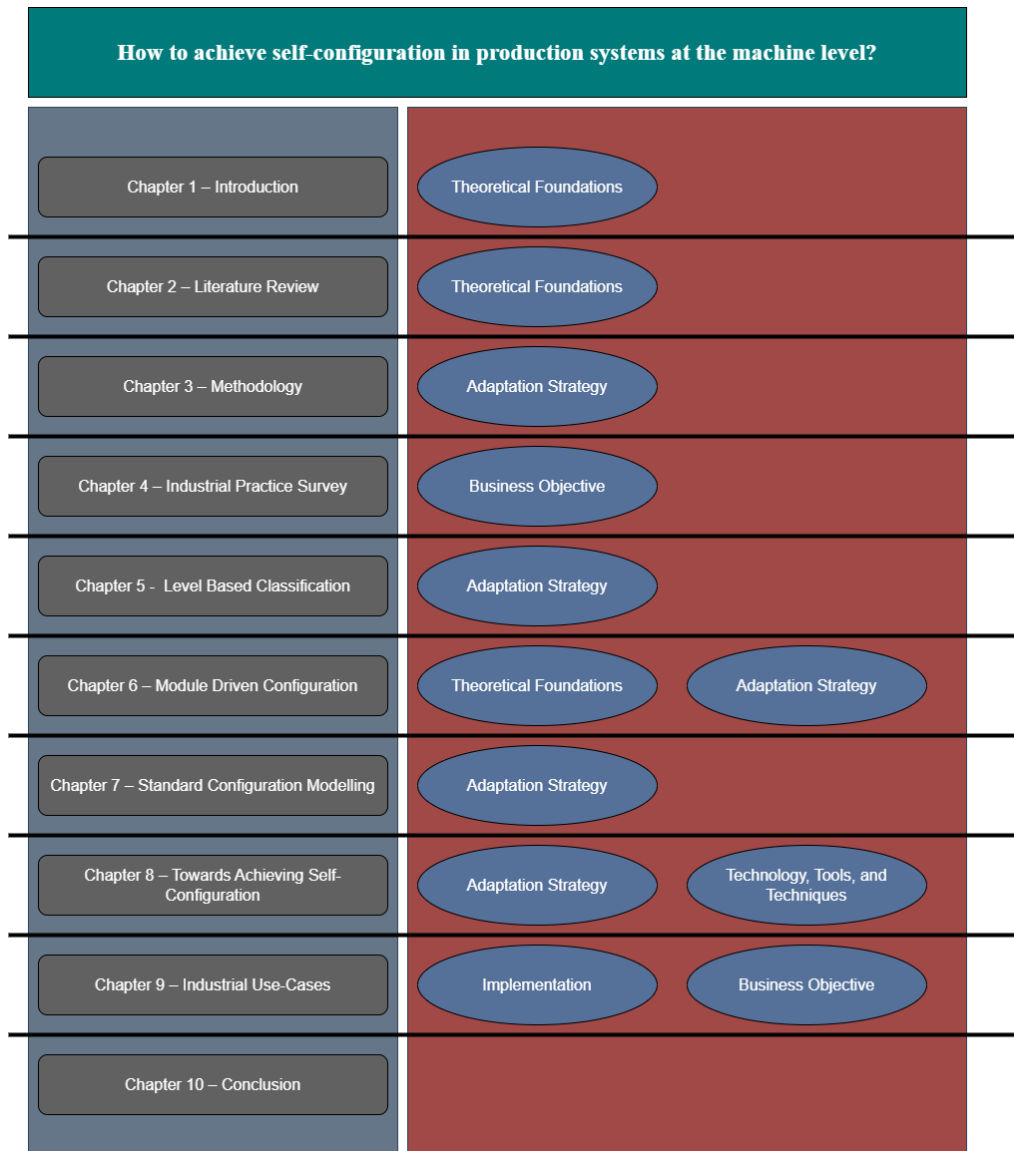


Figure 1.1: Thesis structure, relating chapters to the specific research questions. The chapter-wise mapping of each research question is shown.

in the research area. (**Theoretical Foundations**)

- **Chapter 3 Methodology** discusses the methodology for the thesis, experimental setups and the approaches. Approach for adaptation strategy, data modelling, tools and techniques are elaborated. (**Adaptation Strategy**)
- **Chapter 4 Industrial Practice Survey** highlights the industrial insight on self-configuration and gets the expert reflection on the

impact of self-configuration on production systems. This chapter also serves as a means of validation of the research direction. (**Business Objective**)

- **Chapter 5 - Level Based Classification** expands on the features of self-configuration and maps them to industrial applications. This chapter provides a means of transition towards a higher level of self-configuration in production machines. (**Adaptation Strategy**)
- **Chapter 6 Module Driven Configuration** expands on the building blocks of the production system with regard to configuration. A framework is presented, and a system model on that framework is elaborated. (**Theoretical Foundations**)(**Adaptation Strategy**)
- **Chapter 7 Standard Configuration Modelling** details the modelling of configuration in production systems, with a focus on the framework. A means of capturing the configuration is presented. (**Adaptation Strategy**)
- **Chapter 8 Towards Achieving Self-Configuration** uses the theoretical concepts established in previous chapters to present an adaptation strategy using tools and techniques. (**Adaptation Strategy**)(**Technology, Tools, and Techniques**)
- **Chapter 9 Industrial Use-Cases** Two application cases are illustrated and discussed. The adaptation strategy is applied to achieve self-configuration.(**Implementation**)(**Business Objective**)
- **Chapter 10 Conclusion** presents the thesis outcomes and contributions. The findings are summarised, limitations expanded, and the research reflected on. The significance of the research is elaborated with a focus on a potential industrial use case.

## 1.6 Publications

Table 1.1: The accepted papers from this PhD research.

Research Topic	Journal/Conference	Thesis Chapters
Application of Multi-Agent Systems for Leak Testing (Rehman et al. (2021a))	2021 9th International Conference on Systems and Control (ICSC'2021).	8 and 9
Cloud-Based Decision Making for Multi-Agent Production Systems (Rehman et al. (2021c))	EPIA2021 - 20th EPIA Conference on Artificial.	9
A Framework for Self-Configuration in Manufacturing Production Systems (Rehman et al. (2021b))	12th Advanced Doctoral Conference On Computing, Electrical And Industrial Systems (DOCEIS2021).	8 and 9
Service-based approach to asset administration shell for controlling testing processes in manufacturing (Rehman et al. (2022))	IFAC Manufacturing Modelling, Management and Control 2022.	7,8 and 9
A Modular Artificial Intelligence and Asset Administration Shell Approach to Streamline Testing Processes in Manufacturing Services (Rehman et al. (2024))	Journal of Manufacturing Systems.	7,8 and 9

The papers mentioned in table 1.1 are contributed as first/main author. These research papers are derived from the contents of this thesis from the chapters mentioned in the table. In-progress research papers, from this PhD thesis, in the state of drafts are available and can be provided on request. As a researcher working on the DiManD (Digital Manufacturing and Design) project, the list of research work carried out in collaboration can be accessed through the following link:

<https://scholar.google.com/citations?hl=en&user=QULKUhUAAAAJ>



---

## Chapter 2

# Literature Review

---

## Contents

---

2.1	Introduction . . . . .	21
2.2	Concept of Configuration in Production Systems . . . . .	23
2.2.1	Traditional Configurations in Manufacturing . . . . .	24
2.3	Self-Configuration in Production Systems . . . . .	29
2.3.1	Concept of “ <i>Self-*</i> in manufacturing” . . . . .	29
2.3.2	Discussion on Self-Configuration in Literature . . . . .	33
2.4	Machine Learning Applications . . . . .	35
2.5	Optimisation Algorithms for Parameter Determination . . . . .	38
2.6	Interoperability and Data Integration . . . . .	40
2.7	Applications of Self-Configuration . . . . .	43
2.7.1	Automotive Assembly Line Optimisation . . . . .	43
2.7.2	Smart Energy Grid Management . . . . .	44
2.7.3	Pharmaceutical Manufacturing Flexibility . . . . .	44
2.7.4	Intelligent Warehouse Management . . . . .	45
2.7.5	Agriculture Precision Farming . . . . .	45
2.7.6	Industrial Robot Collaboration . . . . .	46
2.7.7	Smart Healthcare Facilities . . . . .	46
2.7.8	Smart Energy Management . . . . .	47
2.7.9	Autonomous Fleet Management . . . . .	47
2.7.10	Smart Retail Environments . . . . .	48
2.8	Research Gap . . . . .	48
2.8.1	A Case for Granularity and Modularity in Production Systems . . . . .	48
2.8.2	Functional Decomposition in Production Systems . . . . .	52

2.8.3	The Need for a Configuration Abstraction in Production Systems: . . . . .	53
2.8.4	Addressing the Identified Limitations . . . . .	57
2.8.5	Need for Exploring Self-Configuration in Pro- duction Systems . . . . .	60

---

## 2.1 Introduction

In the current manufacturing environment, it is crucial to quickly adapt to different products, complex processes, and customer requirements (El-Maraghy et al. (2021)). The idea of self-configuration in manufacturing has become really important for being adaptable towards changing requirements and improving production efficiency (Chatzigiannakis et al. (2012)). This chapter looks at previous research to explain the concept of self-configuration and how it helps deal with the tricky balance of making products, handling processes, and meeting customer requirements.

A comprehensive review of relevant literature was conducted to understand the current state of self-configuration in production systems, identify theoretical foundations, and pinpoint crucial research gaps (Chapter 2). This review lays the groundwork for the development of the adaptation strategy. Figure 2.1 illustrates the methodology for the literature review.

The goal of this chapter is to explore works carried out that may enable production systems to figure out and use the best settings on their own, finding a good balance between product, processes, and customer requirements. By exploring the idea of self-configuration in literature, this review aims to guide the creation of flexible production systems, from theoretical aspects to practical operational use. The literature review focuses on



Figure 2.1: Adopted methodology for literature review

the concept of configuration in production systems and explores dimensions of self-configuration in various works. Machine learning applications are elaborated along with optimisation algorithms, interoperability, and data integration. It also looks into making different systems work together better, keeping data reliable, and finding practical uses in industries by exploring relevant research works. All of this helps to understand more about the research in this field.

In this chapter, the exploration of the concept of self-configuration lays the foundation for a discussion about the compromise between granularity and modularity. This discussion, explained later, is pivotal to achieving the equilibrium between customisation and efficiency. Increasing customisation and requirements of bespoke products and tailored processes raise questions about precisely setting up machines. So, the level of granularity and modularity here becomes important. This literature review explores

this facet of self-configuration through the lens of adaptation to define, select, operationalise and determine configurations in production systems in terms of mechatronic aggregation (i.e. composition of manufacturing assets).

This literature review also dwells on physical and functional decomposition styles supported by various practical implementation cases in the industry, each utilising an aspect of realising self-configuration capabilities. By scrutinising an array of research works, the chapter seeks to unravel how self-configuration is approached at different levels of granularity and modularity, shaping the spectrum of possibilities for achieving self-configuration in production systems.

## **2.2 Concept of Configuration in Production Systems**

**Conventional Definition of Configuration:** The definition of configuration in the manufacturing industry is generalised as a set of objects and their connections (Berns and Ghosh (2009)).

**Problem with Conventional Definition:** A manufacturing asset is something of value in manufacturing, such as a component or a device Sakurada et al. (2021). In the context of production systems from the above definition, an object may be a manufacturing asset that can be configured or a component in some manufacturing asset. It can be any software or hardware belonging to that manufacturing asset (Borangiu et al. (2020); Liang et al. (2024)). In this research, it is proposed:

*It is important to consider the characteristics of the objects in a system: their extrinsic and intrinsic properties, and the connections between them.*

Connections for digital communication with and between objects are required in order to exploit the opportunities offered by digitalisation in manufacturing (Borangiu et al. (2020); Liang et al. (2024)). These connections may be physical, virtual, or a combination of the two (Monostori et al. (2016); Somers et al. (2023)).

**Requirements for Configuring an Object:** In order to be configurable, an object (representing an asset) must have a set of variables (often referred to as settings) to which values can be assigned to the production system (Strasser et al. (2018)). A particular set of values defines a specific configuration for functionality (Junker (2006); Bordel et al. (2017)). It is possible that certain limitations exist which restrict the range of values that can be assigned to each variable. Some variables may be dependent on other variables or on the settings of connected objects (Radetzky et al. (2019)). Similarly, the connection with and between objects in a production system may be dependent on certain rules and constraints (Aldanondo et al. (1999); Järvenpää et al. (2023)). In some cases, it may simply depend on the choices made by the person in charge of the configuration (Günther et al. (2009); Yang et al. (2023)).

### 2.2.1 Traditional Configurations in Manufacturing

The concept of configuration in production systems is explored in research (Anderson (1994); Bordel et al. (2017)) as the reconfiguring of machines whenever they are restarted. The configuration for machines defines the values for process parameters and is either manually entered or updated at

a storage database where they can be extracted with regular updates (Anderson (1994); Bordel et al. (2017)). These configurations will be different for different operations to be performed.

**Approaches to Configuring Production Systems:** Most production system equipment offers customisation of configuration parameters to a certain degree (Kannisto et al. (2017)). These procedures for configuration are usually only compatible with vendor-specified software on a configurable object and offer little to no integration with the functionality of production systems (Weyer et al. (2015); Givehchi et al. (2017)). The restriction extends towards automation, as the supported procedure to the configuration has limited capability to be integrated with other configurable objects of the production systems at the shop floor (Anderson (1994)). The configuration information is usually (i.e. in most cases) stored on the equipment itself, which means that once the machine is restarted it needs to be entered or loaded again (Anderson (1994); Ferreira and Lohse (2012)). Moreover, if there is some kind of breakdown in the equipment, the process of using and extracting these parameters is greatly limited (Surrucu et al. (2023)). In production systems, the automated configuration is usually realised by following approaches:

1. Configuring by using a central database.
2. Configuring through cloning.

### **Centralised Database**

Development of a centralised database that enables configuration to be validated and enforced by deliberation of policy rules (i.e. from a database

that contains if-else rules), network association (i.e. network environment in which machine operates), and description (i.e. information about process, product, and operation) (Bachula and Zajac (2013)).

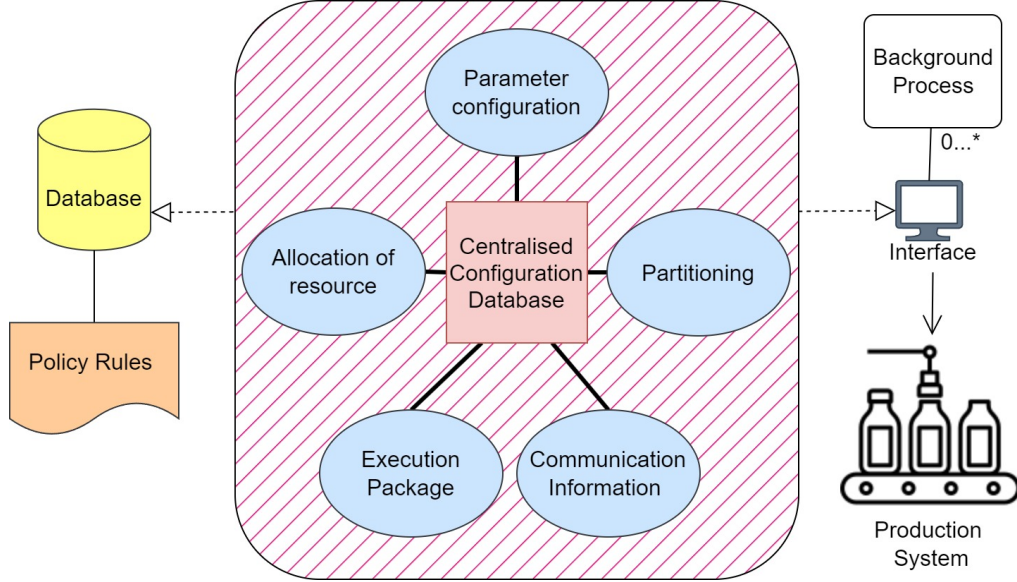


Figure 2.2: Automated Configuration by Policy Rules, Database and Network Association aided by Daemon Processes (Bachula and Zajac (2013)). The configuration database is governed by the associated policy rules. The production systems have a digital interface on which these configurations can be adjusted. Background processes that ensure communication, and connections are also running on the system.

As per figure 2.2, a production system consisting of objects that need to be configured, carries a digital interface for default configuration settings primarily involving parameter configurations, partitioning (i.e. for system modularity) and allocation of space on memory, data storage information, package/libraries to be carried (i.e. dependencies and requirements for operation), address and network information, resource information (e.g. robot, lathe, and others that its connects to), protocols to establish communication, prioritisation, and authorisation information (Anderson (1994)). In addition to these, the production system also has some background processes running that ensure device connection, communication, and synchronisation during operation (such as recording states during PLC task execution etc.) (Lal and Onwubolu (2007); Wan et al. (2013)).



Storage of the configuration parameters in an external database can be considered a suitable alternative to configuration on a device (Shaw et al. (2001); Perez-Leguizamo (2016)).

**Issue On Configuring by Central Database:** Mostly these platforms which offer configuration change in a machine by central database are vendor-specific and offer little to no interoperability (Givehchi et al. (2017)). Small and large production setups utilise different levels of configuration procedures (Anderson (1994); Scanzio et al. (2022)). Small production setups are usually more inclined to vendor-specific procedures as these setups lack well-defined data-oriented communication, transfer, and storage protocols (Opara-Martins et al. (2016)). However, such an approach makes upgrade and installation quite challenging, requiring significant manual involvement. Large production setups on the other hand are more robust in their configuration setups as they drive the procedure to configuration in the machine, thereby influencing the vendor to meet their needs (El-Maraghy et al. (2009)). In the case of large setups where a number of machines are employed, the configuration issue is still prevalent even if provision has been made for automation (Lettner et al. (2013)).

### **Cloning**

Cloning procedures are fundamentally vendor-specific, however, systems have been developed to replicate the procedure with site-specific templates that could easily be used to set up configurations for vendor-specific machines (Jones and Romig (1991)). Figure 2.3 illustrates the concept of configuration setting by using the vendor-specific template. This system, although viable, provides a major barrier to mass customisation where it

is assumed that all machines have identical basic file systems and specific configurations (Jones and Romig (1991); Lettner et al. (2013)). It restricts configuration setups where specific manufacturing assets need to be mapped with the right vendor specifications. Some research (Steiner and Geer (1988); Lettner et al. (2013)) addresses the issue by suggesting system modification, but this kind of modification results in hugely impacting the vendor’s configuration system.

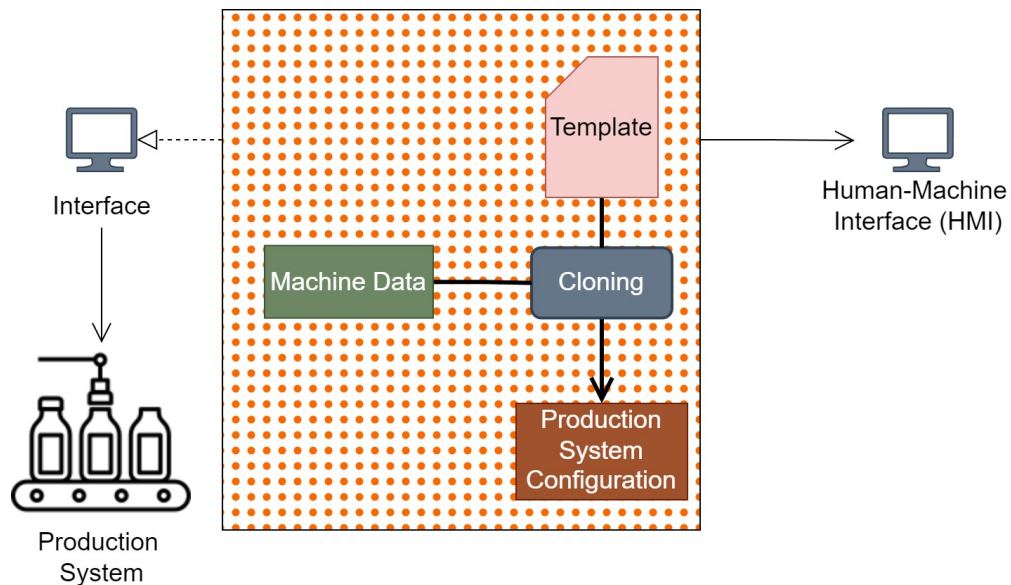


Figure 2.3: Configuration Cloning by Template in Production Environment (Jones and Romig (1991)). The cloning template can be altered through the interface and is displayed on HMI. The machine data is used to configure the production system through the template.

Work has been done to address the machine-specific configuration issue, i.e. changes to the template to meet the machine-specific changes. Earlier work on this includes the development of *typecast* and *mkserv* (Anderson (1994)) schemes that explored application machine-specific changes to the configuration after cloning.

**Issue on Configuring by Cloning:** The underlying assumption in cloning is that the system is static, meaning it is assumed that the machine’s physical configuration remains the same (Feng (2009); Kramer and Magee

(1985)). This system becomes unstable when there is a huge variation in physical configuration, frequent changes or changes in functionality requirements, or new functionality is introduced in the production system (Cannon (2003); Pethig et al. (2017)). Therefore, it becomes infeasible to apply these approaches to current distributed manufacturing paradigms.

The type of configuration applied in such cases may be difficult to determine (Anderson (1994); Kramer and Magee (1985)). The information may depend on the previous state, a combination of prior, likelihood and prediction states, goal parameters, communication, storage formats, and functionality-related information (Wang et al. (2017)). Most of this information may not be present explicitly and may need to be inferred (Bachula and Zajac (2013); Ferreira and Lohse (2012)). The modularity requirements of systems and services, owing to their dynamic nature, also pose obstacles to the adoption of such template-based systems in environments for changing existing configurations (Padayachee and Bright (2012)).

## 2.3 Self-Configuration in Production Systems

In this section, the concept of self-\* is explored in the context of production systems and related to configuration in manufacturing,

### 2.3.1 Concept of “*Self-\* in manufacturing*”

The interaction between physical and data entities has the potential to transform engineering systems (Kao et al. (2015)). This data can be linked to machine functionalities introducing the “self-\*” related aspects such as self-awareness, self-comparison, and self-prediction. These “self-\*” aspects

can further be combined and built up for purposes of self-configuration and self-optimisation in intelligent production systems.

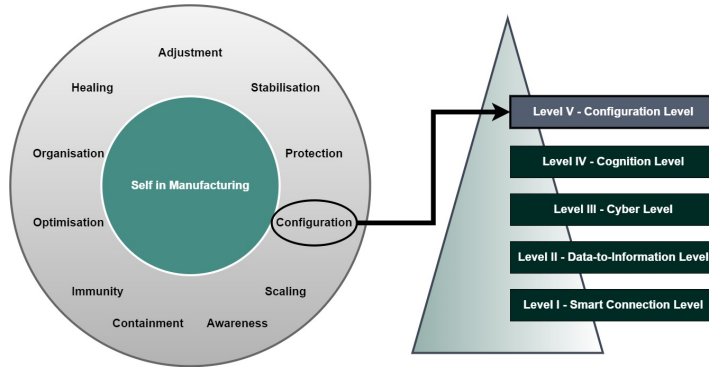


Figure 2.4: Relating Self-Configuration to value extraction levels (Kao et al. (2015)) in Intelligent Systems. As the level increases, more “value ” is obtained from the system. Configuration Level is the highest data and machine interaction level, where the machine can make independent decisions.

A data strategy presented by the UK Government highlights a new data policy addressing the needs of data-driven technologies in industry and steps towards implementing a “data-ecosystem” (UK (2020)). The heterogeneity of industrial data systems demands a cross-collaborative framework for proper interaction. The development of homogeneity in industrial systems is a multi-level approach, with basic steps starting at the lowest connection level. In manufacturing these levels are the smart connection level, data-to-information conversion level, cyber level, cognition level and configuration level (Kao et al. (2015)). Figure 2.4 relates these levels of value-extraction to self-configuration.

The properties of the system that make it capable of accommodating the levels of value extraction are collectively categorised as self-\* properties. Much work on “self-\*” caters to the smart connection and data-to-information levels of data and machine interaction. Common standards and protocols have been significant enablers in achieving connection-level integration (Koronios et al. (2006); Pushkov et al. (2021)). Integration at the connection level remains a challenge in complex factory systems, which

may involve different machines, varied by functionality and supplier, operating at different conditions, time stamps and data formats (Givehchi et al. (2017); Dafflon et al. (2021)).

The data-to-information level is constrained to information acquisition. Significant work has been carried out at this level (Trendafilova and Van Brussel (2001); Liao and Pavel (2012); Li et al. (2022)), which highlighted the need for models that adapt over time in response to the varying information received. Adaptation with respect to parameter variation over time is also necessary to reach the cyber level (Tobon-Mejia et al. (2012); Radetzky et al. (2019)). The cognition level uses decision-making algorithms and reasoning techniques to draw inferences from the cyber level (Iarovyi et al. (2015); Rožanec et al. (2022)).

The variation in scalability and the complexity of distributed systems has witnessed a growth in manufacturing systems that can automatically adjust themselves (Khalgui and Mosbahi (2010); Banerjee et al. (2016)), such as acting on feedback to control the production line and to take independent decisions for the configuration of their current and future states (for e.g. adaptive process control for machining operations (Altıntaş (1994); Gheibi et al. (2021))). Identified self-\* properties include self-healing (Hayes et al. (2008); Ammar et al. (2022)), self-stabilisation (Botygin and Tartakovsky (2014); Binotto et al. (2013)), self-configuration (Scanzio et al. (2022)), self-adaptation (Antzoulatos (2017); Guo et al. (2023)), self-optimisation (Cohen et al. (2008); Löppenbergl and Schwung (2023)), self-protecting (Yan and Vyatkin (2013); Strasser and Froschauer (2012); Casalicchio and Gualandi (2021)), self-organisation (Khalgui and Mosbahi (2010); Olsen et al. (2005); Rodič (2021)), self-scaling (Morgan et al. (2021)), self-management (Sterritt and Bustard (2003); Del Giudice et al. (2021)) and self-immunity (Hofmeyr and Forrest (2000); Ammar et al. (2022)). These system proper-

ties also have interrelationships with each other (Berns and Ghosh (2009)).

Table 2.1: Definitions of Self-Configuration extracted from literature in Chronological Order

Year	Definition
2015	The ability to adapt the system to reach system objectives (Collados et al. (2015)).
2016	The ability to automatically detect and handle changes to the system (Fritze et al. (2016)).
2017	The ability to adapt the system to application domains and be oriented to user requirements (Bordel et al. (2017)).
2018	The ability to implement policies that may adapt the system under influence of a change or disturbance (Bordel et al. (2018)).
2019	The ability to implement and enhance manufacturing responsiveness under changing conditions (Lieberoth-Leden et al. (2019)).
2020	The ability to implement high-level policies on components that demonstrate capacity and capability of adjusting system under objectives (Sakly (2020)).
2021	The ability necessary to enable smart manufacturing (SM) for engineering applications using the inputs generated automatically (Park et al. (2021)).

The concept of self-configuration can be best defined as the capability of a system to adapt to changes by itself. This is elaborated through table 2.1. These changes can be either by re-configuring large complex processes itself, or in adaptability in the architecture or component relationships to maintain and improve performance by achieving desired quality standards in response to changes (Cheng et al. (2004a); Kephart and Chess (2003); Ammar et al. (2022)). It becomes evident by this that self-configuring systems have a direct impact on functionality and maintainability. In this research, “**Self-Configuration**” is defined as follows:

*Self-Configuration is a property that deals with response. It is the ability of a system to change its configuration (i.e., parameters, calibration, and the connection between different system modules) by installing, updating and (re)formulating to improve or restore*

*system functionality in response to actions and a changing environment. In a production system, those components that can be adapted (configured) are referred to as Configurable Objects.*

### **2.3.2 Discussion on Self-Configuration in Literature**

The focus of the literature in this field is to improve flexibility in production systems while promoting high automation levels (Black (2000); Fragapane et al. (2022)). Highly automated production systems ensure rapid production of product variety through autonomous organisation. Flexible Manufacturing Systems (FMS) and Reconfigurable Manufacturing Systems (RMS) (Black (2000); Yelles-Chaouche et al. (2021)) focused on the flexibility offered due to the mechanisation of the production system and enhanced usage of modular elements in production systems. However, these kinds of paradigm shifts need to be linked with production solutions that can accommodate these shifts (Valilai and Houshmand (2013)). These production solutions need to accommodate modularisation and flexible behaviour (Padayachee and Bright (2012); Campos Sabioni et al. (2022)). Self-organisation/self-adaptation in production systems can be achieved by using flexible modular devices (reconfiguration) and utilising innovative techniques to adjust their settings (self-configurations) in order to adapt to requirements (product, process, any other quantifiable objective) (Graessler et al. (2019)).

IBM characterises a self-managed system as one comprising the four adaptation features of self-configuration, self-optimizing, self-healing, and self-protecting. These features may be separated into three levels of hierarchy, according to (Salehie and Tahvildari (2005)). Figure 2.5 illustrates these levels of hierarchy and their corresponding self-properties. It is best to as-

sume that the definition of self-configuration refers to a system’s ability to modify automatically and dynamically (Scanzio et al. (2022)). To maintain and improve performance to the desired quality standards in response to change, these modifications can either be made by reconfiguring large complex processes on their own or by adaptability in architecture or component relationships (Cheng et al. (2004b); Marrella et al. (2017)). This makes it clear that self-configuring systems have an immediate influence on functioning and maintainability (Scheifele et al. (2014)).

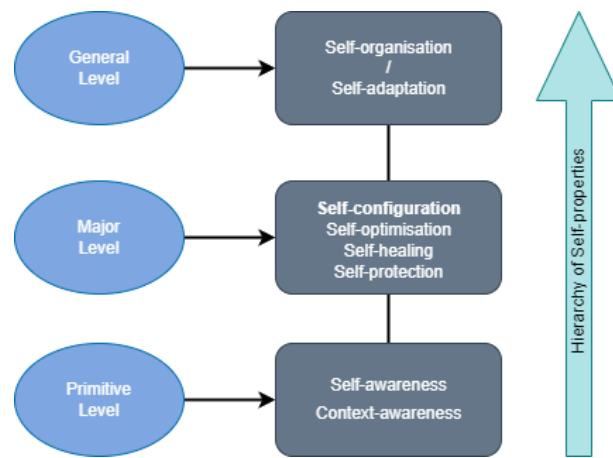


Figure 2.5: Hierarchy of Self-properties. Relating self-configuration to self-organisation/self-adaptation. The hierarchy begins at the base, progresses to the major, and finally reaches the general level. The self-\* properties necessary for achieving self-adaptation in production systems are highlighted at each level with their respective self-\* property.

There are multiple publications that deal with the control of self-organisational behaviour, however, this comes with a huge variety of control solutions depending on the type of flexible production system (Hülsmann and Windt (2007); Qin and Lu (2021)). For instance, (Hees (2017)), offers an approach for selecting a configuration for a production system along with the sequence of operations.

An obstacle to self-configuration is the emphasises on certain aspects like context-awareness (Bannat et al. (2011)), data (Cengiz et al. (2021)), optimisation (Black (2000); Vaisi (2022)) and control (McFarlane et al. (2013)).



However, they do not collectively try to consider them together as a whole and provide a road map for a step-by-step transition.

Machine learning can be used to address this issue, as it can generate insight by learning from accumulated data on all aspects of self-configuration (Radetzky et al. (2019)). It can identify patterns and trends by leveraging data, enabling production systems to make real-time adjustments and adapt to changing scenarios (Andronie et al. (2021)). The next section discusses machine learning in manufacturing applications.

## 2.4 Machine Learning Applications

Machine learning (ML) allows software to learn over time from data and make decisions and predictions that improve over time. ML is being used to improve decision-making, operations, and customer experiences in a wide range of sectors. In the case of self-configuration ML can be used to assist the system in responding and adapting to changes by processing historically accumulated data for the respective system.

Many real-world problems have high complexity and unknown underlying models, which makes them excellent candidates for the application of ML. ML can be applied to various areas of computing to design and program explicit algorithms with high-performance output, such as in the manufacturing industry, robotics (Wang et al. (2019)), e-commerce, medical applications (Wu et al. (2021)), and fault diagnosis (Tang et al. (2020)).

Distributed artificial intelligence (DAI) has attracted research interest because it can solve complex computing problems by breaking them into simpler tasks. DAI algorithms can be divided into three categories: paral-

lel AI, distributed problem-solving (DPS), and multi-agent systems (MAS) (Wooldridge (2009)). Parallel AI involves the development of parallel algorithms, languages, and architectures to improve the efficiency of classic AI algorithms by taking advantage of task parallelism. DPS involves dividing a task into several subtasks, and each subtask is assigned to one of a group of cooperating nodes (called computing entities). Computing entities have shared knowledge or resources and predefined communications with other entities, which limits their flexibility (Bond and Gasser (2014)).

Machine vision and image processing techniques utilised in manufacturing applications are used for integrated inspections to detect defects and improve product quality in the process (Xie (2008)). In many cases, traditional machine learning has made great progress and produced reliable results (Neogi et al. (2014)), but different preprocessing methods are required, including structure-based, statistical-based, filter-based, and model-based techniques. To enhance performance for quality control, these techniques can be combined with expert knowledge to extract representative features (Pernkopf and O’Leary (2002); Wang et al. (2018)) that influence quality.

The integration of cloud technologies in manufacturing applications, exemplified by agent architectures like PROSA (Van Brussel et al. (1998)), extended the goal-based execution models for cloud adaptability supported by ML applications. Manufacturing control and configuration benefit from cloud services (Puliafito et al. (2015)), with task scheduling employing agents (Renna (2011)) and machine learning. Healthcare applications utilize cloud-connected technology with agents for ECG monitoring. Multi-agent reinforcement learning addresses resource-constrained task scheduling, potentially combining cloud-based data-driven decision-making. Cloud technology also supports foresight in job shop scheduling and production control through digital twins (May et al. (2021)).

There is a growing integration of machine learning (Sharp et al. (2018)) and cloud computing (De la Prieta and Corchado (2016), Zhang et al. (2010), Wang and Liu (2012)) within manufacturing processes. While cloud computing frequently serves as a platform for data storage and orchestration (DAniello et al. (2021), Li and Jiang (2021), Tang et al. (2017)), machine learning receives wider applications in areas like predictive maintenance (Sittón and Rodríguez (2017)), process optimisation (Nie et al. (2022)), and intelligent product design (Antzoulatos et al. (2017)). Reinforcement learning has emerged as a technique for scheduling and routing problems (Silva et al. (2019), Zhou et al. (2021), Tang et al. (2017)), while artificial neural networks appear in cyber-physical production system development (Tran et al. (2019), DAniello et al. (2021), Ghadimi et al. (2018)). The increasing use of these technologies indicates a shift towards data-driven (Mourtzis et al. (2016)), self-optimising (Radetzky et al. (2019)) manufacturing environments.

However, it is observed that the objective of production optimisation usually involves the use of ML technology and possibly a cloud-computing technology. The production optimisation can be considered an adaptation requirement, therefore laying the foundation for using ML technologies with cloud computing integration (possibly) to achieve self-configuration objectives (i.e. adaptation objectives) in production systems. Some optimisation approaches to parameter optimisation and determination are discussed in the next section, while some details on achieving interoperability between systems and data integration are discussed in the later section.

## 2.5 Optimisation Algorithms for Parameter Determination

optimisation algorithms, in self-configuring production systems, enable the precise determination of optimal parameters for various components and functionalities. These algorithms, rooted in mathematical principles, are powerful tools to navigate the intricate parameter space and identify values that yield desired outcomes (Sibaliija (2019)). In the context of self-configuration, optimisation algorithms play a crucial role in fine-tuning parameters to achieve seamless adaptability, enhanced performance, and efficient resource utilization (Chen et al. (2017)).

In self-configuring production systems, parameter determination is a critical process that ensures the system operates optimally under varying conditions (Xu et al. (2016)). optimisation algorithms are applied to determine parameters for individual modules, components, and functionalities that collectively form the self-configuring system. For instance, in this research, it is proposed that in a modular production setup, algorithms can be employed to optimise module configurations based on real-time data, production demands, and quality objectives (Gao et al. (2022),Mount (2015)). These algorithms dynamically adjust parameters to align with changing requirements and environmental factors.

The types of optimisation algorithms for this research must be diverse and well-suited to these systems' complex and dynamic nature. Evolutionary algorithms, including Genetic Algorithms and Particle Swarm optimisation (Pierreval and Tautou (1997)), are frequently used in exploring parameter spaces and uncovering configurations that lead to optimal system behaviour. Gradient-based methods, such as Gradient Descent (Zhou et al.

(2022)), adapt parameters incrementally, ensuring the system continuously converges towards improved configurations. Additionally, metaheuristic algorithms like Simulated Annealing (Sibaliija (2018)) provide valuable insights into parameter adjustment by emulating physical processes.

As Industry 4.0 advances, the role of optimisation algorithms in self-configuring or self-adapting production systems is expanding significantly (Vaisi (2022)). The incorporation of artificial intelligence and machine learning techniques Dey (2016) can enhance parameter determination by leveraging historical data (Blum and Schuh (2017)) and predictive analytics (Wang et al. (2022)). The fusion of optimisation algorithms with sensor networks (Zhang et al. (2019)) and edge computing (Shi and Dustdar (2016)) can enable rapid adjustments based on real-time feedback. Moreover, research in hybrid algorithms (Xia and Wu (2005)) (i.e. combining optimisation algorithms) and adaptive optimisation (Kruger et al. (2011)) can address complex scenarios with multiple objectives and constraints.

Agents in manufacturing serve as intelligent software entities that can autonomously make decisions and interact within a production environment (Leitão et al. (2016); Dittrich and Fohlmeister (2020)). Genetic algorithms provide a powerful optimisation tool inspired by natural evolution, allowing agents to find optimal solutions for complex problems like resource scheduling or production line configuration (Abbasi and Houshmand (2011)). The contract net protocol offers a distributed negotiation framework, enabling agents to effectively coordinate tasks, allocate resources, and resolve conflicts (Yeung (2018)). Together, these technologies in this research can facilitate adaptable production systems, improving responsiveness to change and optimising performance.

## 2.6 Interoperability and Data Integration

The integration of diverse production systems demands a robust solution for achieving interoperability and seamless data integration (Estrada-Jimenez et al. (2021)). Asset Administration Shells (AAS) emerge as a critical framework to address this challenge (Tantik and Anderl (2017)).

AAS provides a comprehensive and standardised representation of information on the industrial asset (Beden et al. (2021)). This can be used to develop a whole picture of the asset and assist in adaptation to changes in production systems therefore making it essential to be used for achieving self-configuration (Tantik and Anderl (2017)). AAS enables real-time monitoring, control, and optimisation of production processes (Zheng et al. (2022)). There exist several realisations of AAS standards, some of which are implemented in applications (Deuter and Imort (2021); Cavalieri and Salafia (2021); Cavalieri et al. (2019)).

AAS provides a standardized, vendor-neutral approach to representing production system components, capturing their functionalities, configurations, and relationships (Beden et al. (2021)). AAS has seen significant interest from enterprises, technology providers, and standardisation bodies due to the increasing trend of digitalisation in manufacturing to promote adaptation and responsiveness (Sakurada et al. (2021)).

The AAS is meant to be a digital representation of a real component. As a result, it can be used interchangeably with the term "digital twin" (Wenger et al. (2018)). The major components of an asset administration shell are depicted in figure 2.6 (Wenger et al. (2018)), namely: component manager, manifest, header section, and body section. The detail of these components are as follows:

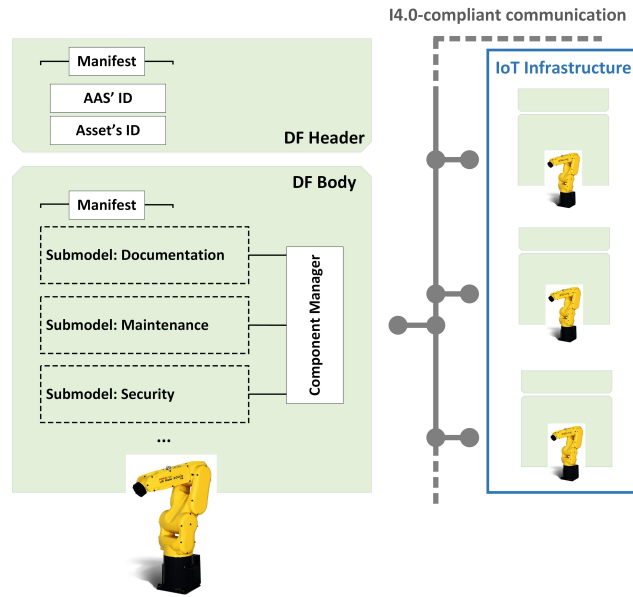


Figure 2.6: Asset Administration Shell (AAS). Representation of an asset with its main components for I4.0 compliant communication with IoT infrastructure within a production facility.

- **The Digital Factory (DF) Header Section** contains the globally unique identifiers for an AAS and its represented asset.
- **The DF Body Section** is composed of multiple submodels, each representing a distinct part of the asset's operation.
- **The Component Manager** links the AAS to a repository of submodels, their description, and their functions. It administers the submodels of the assets. The Component Manager manages and provides access to the Internet of Things (IoT) network of the production facility using a service-oriented architecture.
- **The Manifest** is present in both the header and body sections of AAS. It can be considered as the directory of data content. Specifically, it contains the meta-information serving to provide meaning to the data from AAS.

Submodels depict the various facets of an asset by a given standard. Sub-

models include an Industry's technical/specialised functionality. Among the possible submodels are security, maintenance, drilling, welding etc. More detail on AAS and its implementation is in chapters 7, 8 and 9.

In (Cavaliere and Salafia (2020)), A standard method to realise the description of AAS is shown by the author, who offers an Asset Administration Shell model capable of representing IEC 61131-3 programs and their interactions with Programmable Logic Controllers and each component in the controlled plant. In (Tantik and Anderl (2017)), by providing a framework for connecting existing equipment to external networks, the asset management shell serves as a bridge. to improve network communication by creating new standards and platforms. In (Pethig et al. (2017)), AAS is proposed to integrate the condition monitoring service. In (Wenger et al. (2018)), when considering the possible impact on PLC operation, distributed AAS is utilised to enhance performance. In literature, it is proposed to deal with the problem of organised information sharing and administration in cross-enterprise cooperative engineering. through the use the Data Administration Shell (Löcklin et al. (2021)).

Current AAS has some limitations, mainly the lack of examples of applications in current manufacturing production systems. A lack of standardisation for direct adoption into the manufacturing environment also hinders further propagation. In this research, AAS is used to achieve data interoperability and integration in industrial applications through CAEX standard.

The CAEX standard focuses on representing asset-related data in a vendor-neutral, hierarchical XML format (Berardinelli et al. (2016)). This facilitates data exchange and engineering workflows across the entire lifecycle of a product (Schleipen et al. (2008)). Compared to standards like Automa-



tionML (Drath (2012)), CAEX emphasises a broader scope with less detail on specific automation components (Berardinelli et al. (2016)). It also offers support for simulations like OPC UA (Schleipen (2010)). Where CAEX excels is in its vendor neutrality and its focus on the complete product lifecycle (Berardinelli et al. (2016)). This makes it ideal for long-term data consistency, reducing lock-in to specific vendors, and ensuring interoperability across different engineering tools throughout the product's lifespan (Schleipen et al. (2008)).

## 2.7 Applications of Self-Configuration

This section discusses some applications and potential use cases that demonstrate the practical aspect of self-configuration in different fields. These show the deployment of self-configuration capability or some features of it (explained in another chapter) in specific applications to achieve adaptation and optimisation under changing conditions.

### 2.7.1 Automotive Assembly Line Optimisation

In automotive manufacturing, the challenge of configuring assembly line parameters for different vehicle models and customisation options is widely researched (Li et al. (2011)). Some work addresses this by introducing some features of self-configuration in assembly operations. By integrating sensors (Andronie et al. (2021)), actuators, and AI-driven algorithms, the production line could automatically adjust conveyor speeds, robot trajectories, and workstation setups based on the specific vehicle being assembled (Anthony et al. (2007)). Such optimisation not only reduces setup times but also minimises errors, leading to improved throughput and product quality

(Ebrahimi et al. (2018)). Other research works have explored continuously collecting data on a production system, part availability, and quality metrics (OEE), enabling real-time adjustments to maximise efficiency in the system while accommodating changing production requirements (Ebrahimi et al. (2018)). Some research works explore adaptive process control to improve processes while in production, e.g. thermal error compensation in machine tools (Liu et al. (2019)). This process feedback is used to control a key geometric KPI compensating for anomalies like in the case of tool wear (Vishnu et al. (2023)).

### **2.7.2 Smart Energy Grid Management**

In the energy sector, the integration of some aspects of self-configuration in smart energy grids has allowed for dynamic optimisation of power distribution and resource allocation (Zaidi and Kupzog (2008)). Sensors distributed throughout the grid collect real-time data on electricity consumption, generation, and grid stability (Cerpa and Estrin (2004)). Using AI-driven algorithms, in research (Aguilar et al. (2021)), it is researched that the system can autonomously adjust power flows, reroute energy distribution and manage load balancing to ensure efficient and reliable energy delivery. By utilising the self-configuration capability, the grid can respond to fluctuations in demand and supply, adapting its configuration to minimize power losses and reduce the risk of outages.

### **2.7.3 Pharmaceutical Manufacturing Flexibility**

An increasing trend in pharmaceutical research is towards reconfiguring systems to produce different drugs or dosages (Arden et al. (2021)). A po-

tential research direction explored is integrating self-configuration features in the production system to automatically adjust equipment settings, process parameters, and quality control procedures accommodating changes in product specifications through digital twins (Coito et al. (2022)). This can enable rapid switching between production runs without the need for extensive manual reconfiguration. The self-configuration integration can reduce downtime between production changes, increase overall equipment efficiency, and ensure compliance with stringent quality standards (Reinhardt et al. (2020)).

#### **2.7.4 Intelligent Warehouse Management**

In the logistics sector, self-configuration can be instrumental in optimising warehouse operations. By integrating RFID technology, IoT devices, and AI algorithms, warehouses can dynamically reconfigure storage layouts, shelving heights, and picking routes based on incoming orders and inventory levels (Lee et al. (2019)). Research has been carried out on warehouse systems to continuously analyse real-time data for determining the most efficient arrangement of goods, and minimising travel distances for workers (Guo et al. (2020)). This self-configuration in warehouse management can ensure streamlined operations, faster order fulfilment, and reduced operational costs (Tirkolaei et al. (2019)).

#### **2.7.5 Agriculture Precision Farming**

Self-configuration can find application in precision farming, where it is required that agricultural equipment adjust its settings based on field conditions and crop characteristics (Nayyar and Puri (2016)). Some research

has focused on utilising smart tractors equipped with sensors, GPS, and machine learning algorithms (Burrell et al. (2004)) that can autonomously adapt planting, irrigation, and harvesting techniques to optimise crop yield (Braun et al. (2018)). By analysing data on soil moisture, nutrient levels, and weather patterns, the self-configuration-enabled system can determine the ideal planting density, irrigation frequency, and harvesting timing for each field section (Joseph and Ignatious Monterio (2019)).

### **2.7.6 Industrial Robot Collaboration**

In manufacturing environments, self-configuration can enable safe and efficient collaboration between human workers and industrial robots (Karabegović et al. (2020)). Using advanced computer vision, motion tracking, and real-time communication, robots have been able to dynamically adjust their movements (Tamizi et al. (2023)), force exertion, and interaction patterns based on the presence and actions of human coworkers (Garcia et al. (2019)). By introducing self-configuration features in robot systems a robot can avoid collisions, adjust its speed to match human operators, and adapt its tasks to changing production needs (Jia et al. (2020)).

### **2.7.7 Smart Healthcare Facilities**

Self-configuration can play a vital role in modern healthcare facilities, where patient care and resource allocation need to be agile and adaptable (Lupton (2013)). Hospital rooms can be equipped with smart sensors, IoT devices, and AI-driven algorithms to self-configure based on patient needs (Zhang et al. (2005); Ben Ida et al. (2020)), medical requirements, and infection control protocols. The room environment, lighting, temperature,

and medical equipment can be made to adjust automatically to ensure patient comfort and safety. This self-configuration in healthcare settings can optimise patient care, minimise energy consumption, and contribute to efficient hospital operations (da Silveira et al. (2019)).

### **2.7.8 Smart Energy Management**

Self-configuration is a crucial component towards smart energy management systems that aims to optimise energy consumption and distribution (Shen et al. (2022)). In smart grids, self-configuring devices such as smart meters and sensors can continuously gather data on electricity usage, load demand, and grid stability. Some research work uses AI-driven algorithms, to enable the system to dynamically adjust energy distribution, reroute power flow, and manage peak demand (Orehounig et al. (2015)). This self-configuration feature ensures efficient energy utilisation, reduces grid congestion, and supports the integration of renewable energy sources.

### **2.7.9 Autonomous Fleet Management**

Self-configuration can revolutionise fleet management by enabling autonomous vehicles to optimise their routes, speeds, and driving behaviours (Best et al. (2018)). In transportation and logistics, trucks, drones, and delivery vehicles can utilise aspects of self-configuration by monitoring real-time data on traffic conditions, weather, and package volumes to dynamically adjust their routes and schedules. AI algorithms, in literature, has shown to process this information to optimise fuel efficiency, minimise delivery times, and reduce operational costs (Taylor et al. (2021)).

### 2.7.10 Smart Retail Environments

The retail industry can use self-configuration to create personalised and responsive shopping experiences (Xia et al. (2021)). Smart retail spaces equipped with IoT devices, RFID tags, and AI-powered analytics can dynamically adapt store layouts, product displays, and pricing strategies based on customer behaviour and preferences. By analysing data on foot traffic, purchase history, and inventory levels, the self-configuring retail environment can optimise product placements, offer targeted promotions, and adjust pricing in real-time (Har et al. (2022)).

## 2.8 Research Gap

### 2.8.1 A Case for Granularity and Modularity in Production Systems

Granularity can be defined as: *The size of an individual mechatronic aggregate.* Modularity can be defined as: *The number of separate manufacturing aggregates present in the production system (Chiriac et al. (2011)).*

Granularity can be considered the main dimension for achieving self configuration objectives in production systems. The approach to achieve self-configuration is based on the value-addition factor brought about by the manufacturing equipment (Ribeiro and Björkman (2017)). At a fundamental level, automation in production systems is leveraged around getting throughput performance, therefore promoting the integration of manufacturing assets to form a production system to achieve the particular goal (Ribeiro and Björkman (2017)).

The concept of value-addition in automation for achieving self-configuration revolves around adaptive configuration change under response. This response may originate from requirements (i.e. variables) of functionality in the manufacturing asset as it is introduced in the production system, or from product requirements. In this research, for the self-configuration approach, the granularity is fixed to manufacturing asset functionality. Therefore, in this thesis, it is proposed that granularity in production systems may be assumed to be a mechatronic aggregate that possesses form and functionality at a level not too wide where several functionalities may be represented (i.e. at that aggregate) but not too narrow where individualisation is at such a level that granularity is represented at non-value added components, adding significantly to complexity as the system is programmed.

In Holonic Manufacturing Systems (Leitão and Restivo (2003)) the holon acts as the building block that is simultaneously a part and a whole. Holon, although possessing a concept similar to the granularity concept discussed above, does not have a limit and relies on physical or logical decomposition rather than functionality.

The production system, assumed to be represented by granular manufacturing assets, needs an adaptation architecture that addresses the following;

- The granular level is able to traverse information for a decision as the functionality is executed to other granular components. Simply, information transfer is possible between these granular objects and functionality is dependent on it.
- These granular levels are represented in the form of a structure. This research proposes these in the form of *modules*.

Cyber-physical systems, more especially production systems, as discussed above may be represented using granular components. These granular components must be at a technical level that imparts/adds value on the system. This research proposes that these be represented in the form of modules (Chapter 6). However, there can be concerns about the definition of that level and the number of levels to be considered.

Each granular level is represented by a function that encapsulates requirements (in the form of variables). The granular level is fixed at the functionality offered that adds value to the production system. A production system consists of multiple granular objects, therefore being a combination of functionalities to achieve an objective.

The capture of the granular object is carried out through a module (discussed in Chapter 6). Figure 2.7 presents an illustration of the granularity concept in the module.

In a production system, a granular object offers a functionality represented in a *module*. As these modules, having a form and structure, are removed from the system, the offered functionality is also removed. However, it is not proposed that more modularity is achieved just by raising granularity to a higher level. Production system design must cope with data transfer requirements between modules to perform production operations. This means that two modules may be linked together to perform an operation, where a module depends on data from another module. Combining them to a higher level of granularity does not mean an increased level of modularity, and just increases the complexity of programming. It may also add some incompatibility in data transfer and complexity in dependency among modules at different granularity levels.

Hence, modularity must be defined at those granularity levels that observe



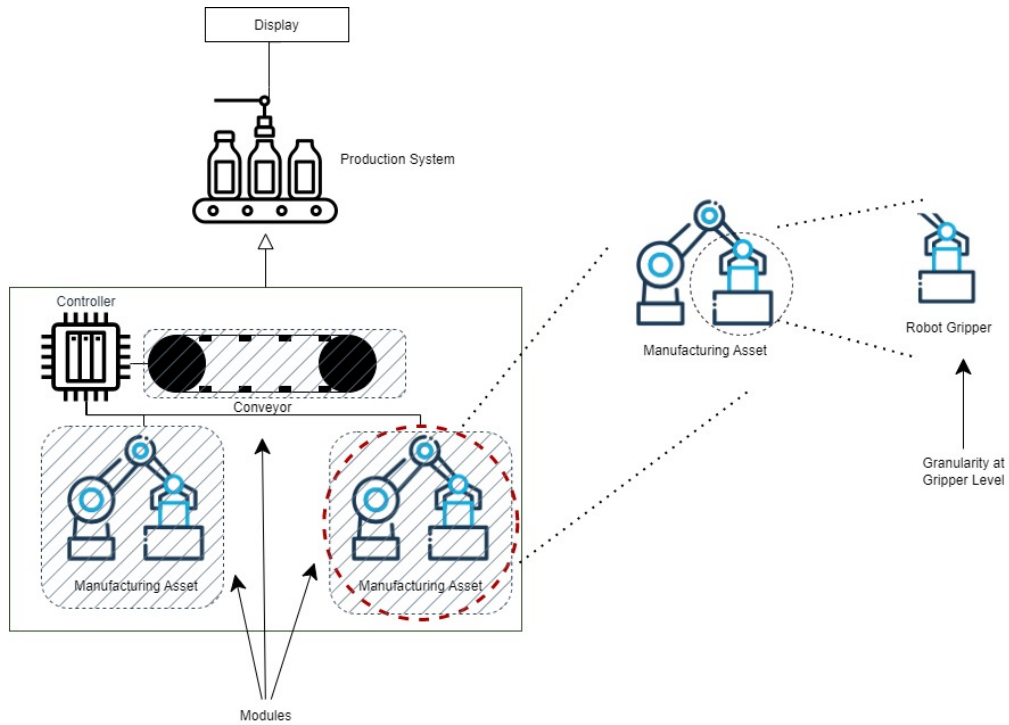


Figure 2.7: A production system consists of multiple granular objects (modules). These modules can have different granular levels. In this illustration, the granularity level is considered at the gripper. Each module adds value, imparting functionality to the system.

value addition from a production perspective. Identifying the modularity level in the production system is contextual and highly dependent on the decomposition style considered. Researchers need to examine different levels of granularity (i.e. how finely a system is broken down) within production systems. This will help determine the points where modularity starts generating the most value. Different ways of breaking down a production system might lead to different optimal modularity levels. Researchers must investigate various decomposition methods to understand how this influences the ideal structure in the production system. Research must contribute to the system's need for flexibility and its optimal modular structure.

## 2.8.2 Functional Decomposition in Production Systems

The modularity level is dependent on the type of decomposition style selected. These styles are of two types;

- **Physical Decomposition** is a production system decomposition style where a one-to-one mapping exists between the cyber part and its physical component. This kind of decomposition style assigns the actuation to the physical boundaries of the production system.
- **Functional Decomposition** is the decomposition style where the module boundaries are adjusted to the system functionality. In a production system, a robot and gripper can be considered a single module as they add value to the system by their functionality (i.e. pick and place).

Both decomposition styles require an assessment of the physical and functional requirements for selection. Granularity levels are defined after this assessment for a system. In a production system, there is no complete physical decomposition, as some functions need to be considered in the cyber component to control the system. An issue may also exist with the functional decomposition as the function may be possible, but the physical barrier of the system may undermine the modularity. This is explored by the functionality and variable rules (elaborated in Chapter 6). Figure 2.8 illustrates the decomposition in the production system concept and the decomposition styles (i.e. physical and functional decomposition). Figure 2.9 illustrates the assessment of developing modules based on physical and functional characteristics.

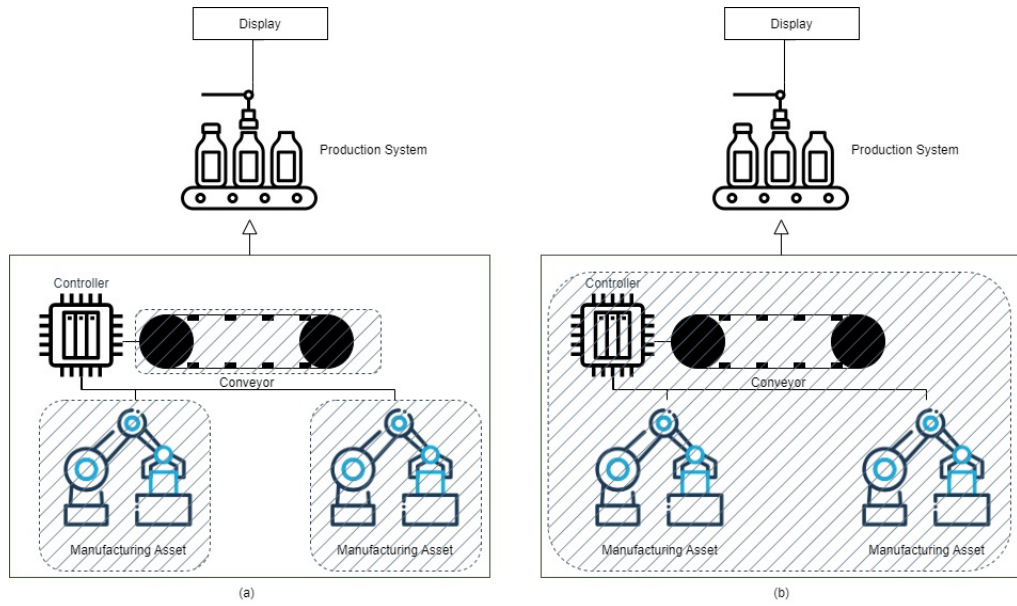


Figure 2.8: Decomposition Style in Production Systems. Modularity is defined at different granularity levels depending on decomposition styles. (a) Physical Decomposition, where granularity is defined at each component in the system. For example, in the picture, granular boundaries are considered at each manufacturing asset. (b) Functional Decomposition, where functionality is defined where the granularity boundary lies. In the picture, the two assets and the conveyor system are considered as a single granular boundary since their functionality adds value.

### 2.8.3 The Need for a Configuration Abstraction in Production Systems:

Existing literature highlights the need for a well-defined configuration abstraction for production systems. This is needed to facilitate the self-configuration in a production system. This abstraction should encompass:

- **Asset Definition:** Clear specification of what constitutes a manufacturing asset.
- **Asset Components:** Explicit outlining of extrinsic and intrinsic elements.
- **Connections:** Elucidation of the relationships between asset components.

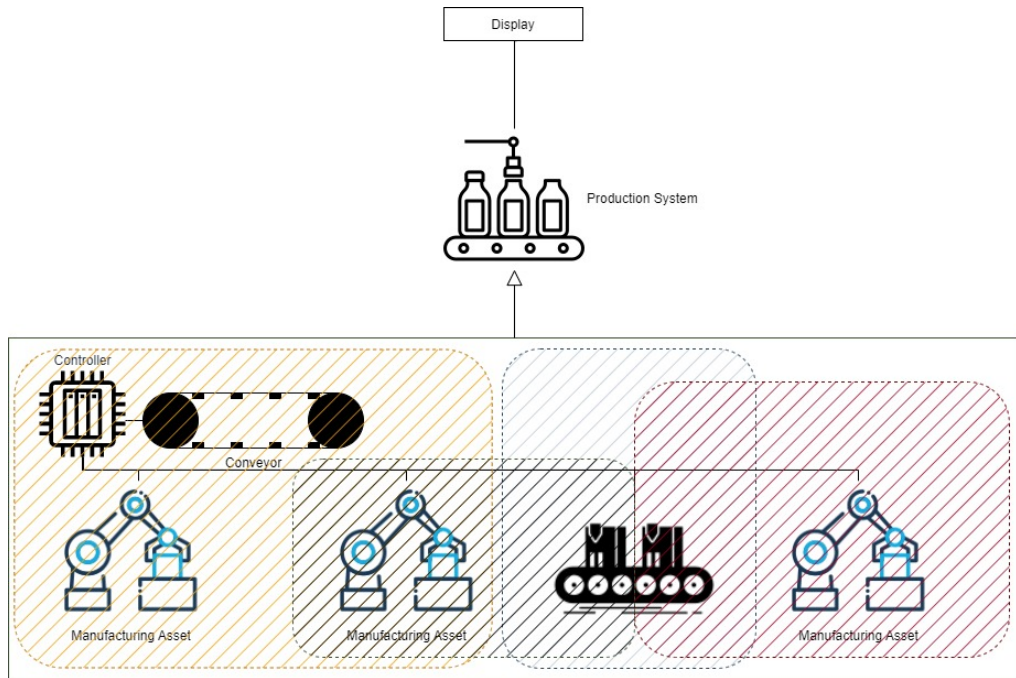


Figure 2.9: Modularity is dependent on a careful assessment of the physical and cyber components of the production system. Granularity is linked with modularity. The granularity level is linked with the collection of mechatronic components responsible for functionalities within the production system. In the picture, there are four modules.

Objects and their associated variables embody a specific domain, implying they can be represented independently. Within production systems, this translates to systems comprised of single or multiple assets, where object variables can be manually assigned or depend on other variables.

**Limitations of Current Configuration Approaches:** In case of configuring by central database, the approach does not take into account the case of equipment where the machine decides for itself what parameters it needs to operate on.

Utilising a centralised database to control the configuration procedure enables dynamic change without localised access and integration of services with the existing manufacturing assets. This reduces the need to validate and examine configuration settings on production systems in order to fig-

ure out the errors. An added security advantage is present as a change in configuration is driven, usually, by a centralised control policy. However, there can be disadvantages in terms of longer start-up times and complex setups for low-level machine configurations.

In configuring a system by cloning the main issue is the method's static nature making it unsuitable for distributed production systems, or situations involving frequent variation in physical configurations, functional requirements, or the introduction of new functionalities.

**Building a Configuration Approach:** Most of the publications, however, do not address the configuration issue, overly relying on some kind of algorithm to give the right settings. This becomes problematic in the case of the shift observed towards mass customisation. Usually, the algorithms fulfil a certain set of conditions and do not interact with the production systems directly (has little knowledge of the changes happening). However, for a production system to be completely self-configurable, it must be capable of identifying changes and responding to them in real-time while using the said algorithms. Therefore, it is necessary to highlight the change in the production system, identifying the elements that contribute to this and making them capable of self-configuration.

The development of a configuration approach, for a production system, that accesses and updates the parameters even when the machine is down, overcomes the limitations of cloning and a centralised configuration database. This is carried out in this research.

The previous section elaborates on the significance of self-configuration in different fields. However, most of the applications where some features or aspects of self-configuration in these fields have been realised are

usually one-offs to the specific field and implementation. They cannot be transferred to another field and not be generalised. These applications are enough to demonstrate the feasibility of self-configuration or self-adaptation but cannot be carried over into another field of application, especially production systems which is the application theme of this PhD research. Moreover, these applications do not follow any standard tool or technology to implement features of self-configuration.

To understand the implementation of features of self-configuration, it is first necessary to align the research with real-world industrial perspectives. This is done to develop an understanding towards industrial requirements for self-configuration and its impact on production systems. To gather this insight, surveys are needed that will attune the research direction to practical industrial requirements.

A framework is needed that can be applied to production systems in different application domains. This framework needs to promote modular architecture deployments and use standard tools and technologies. This thesis aims to address these deficiencies. To generalise for all production systems, detailed knowledge of assets that make up the production system is required. The knowledge of how these assets can be grouped (i.e. granularity) to implement features of self-configuration is also essential.

The key limitations observed are:

1. Most implementations of self-configuration in various fields are specific to those fields and lack transferability.
2. Existing applications lack adherence to standard tools or technologies.
3. A framework applicable across diverse production system domains is

needed.

4. An in-depth knowledge of the constituent assets and their configurations is required for self-configuration. An insight into industrial requirements for self-configuration is needed.
5. Grouping of assets to effectively implement self-configuration features needs to be explored.

#### 2.8.4 Addressing the Identified Limitations

The limitations identified in the research gap can be addressed as;

- **Need for a generalised framework:** There exists a need for a robust framework that accommodates dynamic changes arising from product variations, shifting customer demands, and the integration or removal of production assets in the system. This framework must address the composition of the production system components, organising them into aggregates to ensure adaptability while maintaining a structured system for simplified management. This framework should manage constraints (e.g. KPIs, priorities etc.) while defining interactions among these aggregates and interface with external systems like machine learning models for data-driven insights. This framework should allow for operation-driven system configuration, supporting changes in the system while accommodating changing functionality requirements. This framework becomes important in enabling manufacturing systems to be adaptable and responsive to dynamic shifts in their operational environment.

Gathered from the literature review, the framework's objectives are aligned with the requirements identified in the research and can be

listed as follows:

1. The framework must enable the system to dynamically adapt its behaviour in response to changes in production settings, physical parameters, or calibration settings (Anthony et al. (2007)). This ensures optimal performance under varying conditions.
2. The framework must target modular production systems based on manufacturing aggregates (Padayachee and Bright (2012)). This will allow for the independent configuration of individual modules and enable granular control over functionality.
3. The framework should establish a clear connection between granularity (i.e. level of detail) and modularity (i.e. organisation of aggregates into modules), addressing the interdependence between these concepts (Chiriac et al. (2011)). This facilitates a structured approach to self-configuration.
4. The framework should incorporate mechanisms to handle constraints introduced by variables and functionalities within the system (Strasser et al. (2018)). This ensures that self-configuration settings align with the limitations of the production environment.
5. The framework should support interactions with external systems for enhanced decision-making capability (Andronie et al. (2021)).
6. The framework will not only handle changes in variables and relationships but also adapt to scenarios where new variables are introduced as part of enhanced functionality (Antzoulatos (2017)). This ensures that the framework can adapt to evolvable production systems.



7. The framework must possess a high degree of expressiveness and generality to target a variety of self-configuration scenarios in complex production systems (Guo et al. (2023)).

- **Need for transferability:** A production system comprises interconnected components, each serving a distinct role. A self-configuration strategy is needed, that offers a structured methodology to navigate this complexity. Its primary objective is to attain comprehensive control over the production process in all manufacturing application scenarios. Such a strategy enables precision at various composition levels within the production system. The strategy should allow the independent configuration of individual components, optimising the overall system performance effectively. Such a strategy will ensure adaptability and efficient responses to evolving production requirements.
- **Detailed asset and configuration knowledge:** An approach is needed that focuses on promoting interoperability by capturing the configuration of elements that make up the production system. The approach should define the constituent elements of configuration, providing in-depth knowledge of configuration in the assets that make up the production system. It should offer a structured representation of the configuration and deal with managing configurations for changing requirements. By examining these configuration elements, representing assets and dealing with configuration change activities, the approach will circumvent proprietary protocols, foster compatibility among assets, and enable seamless integration within digital manufacturing, thus addressing standardisation and interoperability challenges. An industrial insight gathered through surveys is needed to develop a practical understanding of assets and their configuration

along with readiness to adopt a strategy for self-configuring these assets.

- **Standard tools and technologies:** Utilising standard tools and technologies for realisation will make the adoption of self-configuration in manufacturing environments easier. The standardised tools and technologies need to be explored for self-configuration. These should be able to capture information on production systems in real-time and work together to optimise the system. The use of these tools and technologies should be compatible with applications across the manufacturing domain.
- **Grouping assets:** The production system needs to be studied to identify the grouping of assets to implement self-configuration. This introduces the inquiry into the principle that should govern the grouping of these assets.

The limitations addressed in this section are discussed in detail in the corresponding chapters.

### 2.8.5 Need for Exploring Self-Configuration in Production Systems

The investigation into self-configuration within production systems gains significance against the backdrop of a manufacturing landscape characterised by intricate customization demands, process intricacies, and evolving customer expectations. At its core, the research aims to navigate the intricate web of variables inherent in production, seeking to strike an optimal balance between product, process, and customer requirements.

To promote interoperability in the production system, there exists a need to model configuration in the production system at a granular level. It is assumed that granularity in a production system is at the manufacturing asset functionality. It is proposed that each granular manufacturing asset introduces a functionality in the production system.

Central to this pursuit is the concept of granularity, which is a key ingredient in achieving the harmony between customisation and efficiency. By dynamically determining the right granularity level, manufacturers can unlock a self-configuring mechanism that seamlessly aligns production configurations with specific product attributes and process intricacies. This self-adaptation is crucial in a world where personalised products are not merely a luxury but an imperative.

The crux of the matter lies in the choice of decomposition style whether to opt for physical or functional decomposition. This choice inherently shapes the granularity levels and, by extension, the adaptability of the production system. The research delves into this the decision-making process, delving into how each decomposition style impacts the precision of self-configuration and the ability to meet the product, process, and customer requirements.

The information in the module must be captured and collected in the form of a standard. The control mechanism in the production system then may use the information for executing tasks.

This research aims to help manufacturers adapt their production systems by understanding the complex dynamics of granularity and decomposition styles. The goal is to develop a systematic framework that enables these systems to autonomously discern and implement the optimal system settings, physical parameters, and calibration settings, driven by KPIs (e.g.

time constraints, priorities, efficiency, production rate etc.), thus transcending the conventional confines of fixed configurations.

---

## Chapter 3

### Methodology

**Contents**

---

3.1	Introduction . . . . .	<b>64</b>
3.2	Requirements of the Approach . . . . .	<b>65</b>
3.3	Detailed Methodology . . . . .	<b>67</b>
3.3.1	Extensive Literature Review . . . . .	67
3.3.2	Industrial Practice Survey . . . . .	68
3.3.3	Level-Based Classification . . . . .	68
3.3.4	Module-Driven Configuration . . . . .	69
3.3.5	Data Modelling for Self-Configuration . . . . .	69
3.3.6	Adaptation Strategy Development . . . . .	71
3.4	Validation Methods . . . . .	<b>75</b>
3.4.1	Industrial Use-Cases . . . . .	75
3.4.2	Research Findings . . . . .	77
3.5	Overview of the Approach . . . . .	<b>77</b>
3.6	Thesis Structure . . . . .	<b>79</b>
3.7	Conclusion . . . . .	<b>80</b>

---

**3.1 Introduction**

This chapter presents a structured approach to addressing the research question, namely to achieve self-configuration in production systems at the machine level.

The research focuses, as established in the research gap (see Chapter 2), on the configuration and the configuration changes associated with production system settings, physical parameters, and calibration settings, driven

by time constraints, shop-floor performance criteria (e.g. efficiency, production rate) and other KPIs. Additionally, the research focus includes a generalised framework for self-configuration, a guiding adaptation strategy that is transferable, a detailed study of the configuration of assets in a production system, and technologies for enabling self-configuration.

In this chapter, the methodology employed to investigate and develop solutions for achieving self-configuration is discussed. The research question is further expanded into several needs, including theoretical aspects, adaptation strategy, technology, tools, and techniques, implementation, and business objectives. These needs provide direction towards the research process. This research adopts a multifaceted, iterative methodology to address the research question of achieving self-configuration in production systems at the machine level.

## 3.2 Requirements of the Approach

As identified in the research gap (Chapter2) the self-configuration solution should possess the following characteristics:

- **Integrability:** The approach must seamlessly integrate with existing production systems and technologies.

**Justification:** The literature review revealed a lack of generalised frameworks that can be easily adapted to various production applications. Existing solutions are tailored to the specific system, limiting their transferability. To address this, the solution should emphasise wider integration through technologies and data standards, for enabling wider adoption.

- **Adaptability:** It should be customisable to accommodate diverse production scenarios and testing requirements.

**Justification:** The literature highlights the need for a self-configuration approach that can handle a wide variety of product changes and changing business objectives. Adaptability is important to ensure that the developed solution remains applicable across different manufacturing contexts.

- **Real-time capability:** The system should make and implement configuration decisions based on real-time data and production conditions.

**Justification:** Existing approaches rely on offline analysis or pre-defined rules, restricting their ability to respond to dynamic changes. Real-time decision-making is critical to ensure system adaptation to unexpected changes.

- **Data-driven:** Decisions should be informed by historical production data, KPIs, and machine learning insights.

**Justification:** A data-driven approach allows for continuous improvement and adaptation based on actual production performance. The literature review shows an increasing shift towards intelligent, data-driven decision-making. Machine learning models can be used to utilise data to respond to dynamic changes.



### 3.3 Detailed Methodology

An illustration of the research approach taken for this thesis is presented in figure 3.1. The methodology approach is detailed in the following subsections.

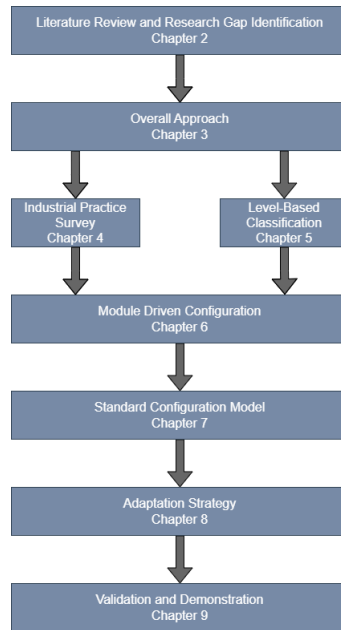


Figure 3.1: The methodology approach taken for this research.

#### 3.3.1 Extensive Literature Review

A comprehensive literature review was conducted to understand the current state of self-configuration in production systems. The methodology for the literature review was detailed in Chapter 2.

This review provided a comprehensive understanding of theoretical foundations, practical implementations, and open research challenges, particularly concerning integrating self-configuration at the machine level within production systems.

### 3.3.2 Industrial Practice Survey

An industry-focused survey was designed and executed to gather insights into real-world needs and current practices relating to self-configuration (Chapter 4). The surveys targeted industry professionals and academics, aiming to understand current self-configuration capabilities, challenges faced in manual configuration processes, and the perceived benefits of automated self-configuration. Survey findings informed the development of the adaptation strategy, ensuring its relevance to industrial practice. These surveys validate the research direction and ensures its practical relevance.

Survey findings revealed that time-consuming manual configuration poses a significant challenge for many production environments. This insight reinforced the need for the adaptation strategy to focus on automated, efficient configuration processes.

### 3.3.3 Level-Based Classification

A novel classification system is established to categorise the features and levels of self-configuration, bridging theoretical concepts with industrial applications (Chapter 5). This classification clarifies the scope of self-configuration and its potential impact.

Building upon insights from the literature review, this Level-Based Classification categorises production systems in terms of their self-configuration features. This classification distinguishes between different levels of autonomy and decision-making capabilities. The classification system facilitates a structured understanding of the current status of self-configuration, potential scope and impact of self-configuration implementation in production systems.

### 3.3.4 Module-Driven Configuration

A detailed system model is developed to represent and understand the configurable elements within production systems (Chapter 6). The framework emphasises modularity, where self-configuring modules encapsulate functionalities and their associated configuration settings. This framework provides the building blocks for understanding configuration and implementing self-configuration and aids in analysing the relationships between configurable elements under constraints.

### 3.3.5 Data Modelling for Self-Configuration

Data modelling involves techniques for information capture, representation using Asset Administration Shells (AAS), state chart usage for functionality coordination, and integration of machine learning (ML) models for configuration updates (Chapter 7 and 8). This addresses interoperability challenges and provides data-driven insights for self-configuration

AAS facilitates interoperability, enabling data exchange between diverse production system modules. State charts are employed for coordinating functionality execution, ensuring the correct sequencing of configuration steps, and transitions. Machine learning (ML) models are integrated to analyse data and provide system configuration. These models will be trained on historical production data, KPIs, and sensor readings (i.e. pressure sensor, camera vision).

Figure 3.2 illustrates a simple data flow methodology for the research. The details of this approach is expanded in the later overview section.

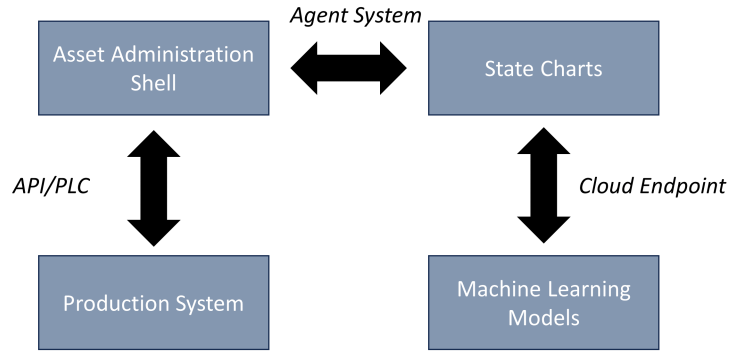


Figure 3.2: Simple data flow diagram of the research. Data from the production system is stored in its asset administration shell. A change in production conditions or performance requirements results in configuration setting change through the interaction of agent systems and machine learning model.

### Enabler for Interoperability

In this research, AAS is employed as the means to represent the digital footprint of assets in the production system. This ensures compatibility and effective interfacing of submodels representing functionalities of different assets within the AAS.

A generalised approach to capture data/information about the asset involves the identification of components that constitute the production system, its pertaining information and their sub-model elements. A brief overview of this generalised data model is presented in figure 3.3. Effectively the encapsulation of relationship, capability, constraints, and operations is carried out through this generalised modelling.

A description of all these components is presented in table 3.1. This implementation is provided in Use-As basis and can be adapted to serve the need of implementation. This data modelling is used for capturing information on the production system (Chapter 7) and using that information in the adaptation strategy (Chapter 8) to realise self-configuration.

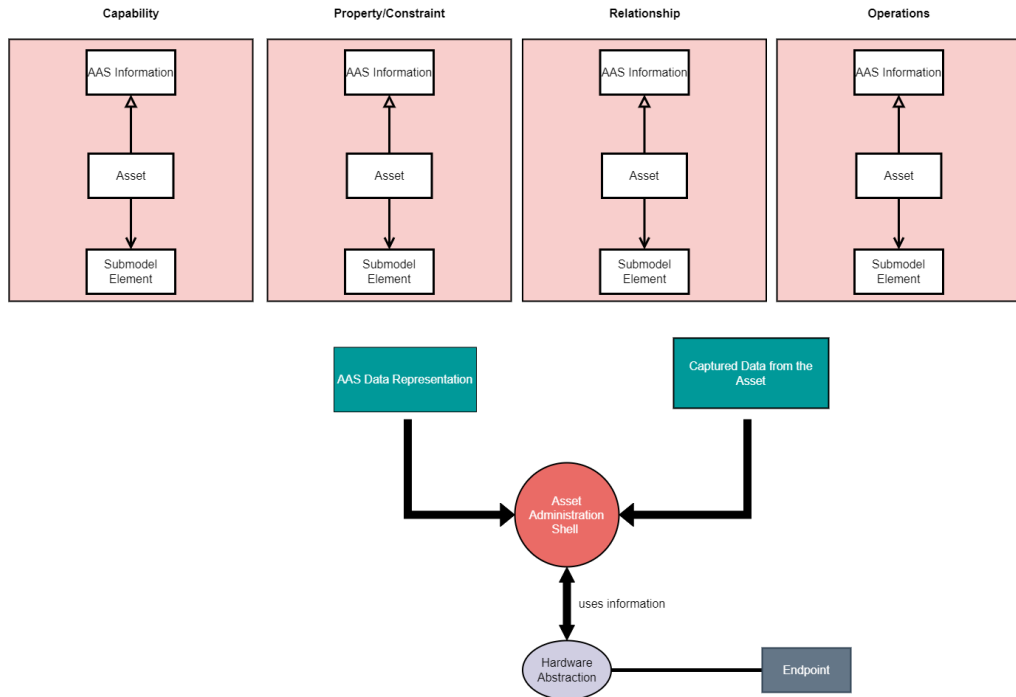


Figure 3.3: Data modelling generalisation for capturing capability, relationship, constraint, and operation information for self-configuration. Information interchange happens between hardware abstraction and AAS for self-configuration by querying from endpoints during the execution of functionality.

Table 3.1: Components of generalised data model for self-configuration.

Components	Description
AAS Information	Contains information about the production system, the manufacturing asset belongs to. Any relevant identification and reference.
Asset Information	Contains information on the manufacturing asset. Identifies the asset from other assets within the system.
Submodel Element	Contains information about the functionality of the manufacturing asset in the system. An asset can have one or more Submodel elements. These functionalities can store information on skills, settings, calibration, results or any other specific to the asset.

### 3.3.6 Adaptation Strategy Development

The adaptation strategy is the central contribution of this research, integrating theoretical insights, industrial insights, and data modelling (Chapter 8). The strategy provides a systematic approach for achieving self-

configuration within production systems, addressing changes in system settings, physical parameters, and calibration settings driven by performance criteria and KPIs. The strategy integrates AAS for asset representation, state charts for functionality orchestration (ensuring correct sequencing of steps and transitions), and ML-based decision-making. The ML models analyse real-time data and historical performance metrics to suggest optimal configurations within production constraints. The strategy requires a proof of concept about the technologies being used. Therefore, each technology must be conceptualised for the application, developed and tested.

#### **Requirements for Tools and Techniques**

The tools and techniques that need to be developed should consist of the following:

- **Coordinating Functionality Execution:** There is a requirement for coordinating functionality in the production system to achieve the stated aims. This tool has the following requirements:
  - All transitions involved in the execution of functionality must be accurately captured. For e.g. connecting, parameter update and execution etc.
  - Functionalities of a wide variety of manufacturing applications should be able to be represented using the tool.
- **Controlling Functionality:** There is a requirement for orienting and combining actions with behaviours. Provides a means of control for functionality execution. This tool has the following requirements:
  - The control technology must be able to coordinate with connected tools to perform operations while taking into account

information from these tools.

- Can be easily integrated and is scalable to meet different manufacturing application requirements.

• **Information Capture:** There is a requirement to capture production system information in real-time. Also, it should be able to interact with other layers of the adaptation strategy for information interchange. This tool has the following requirements:

- Must be able to capture complete information on the production system and its assets acting as a digital twin (i.e. digital representation).
- Can be updated in real time.
- Should follow a standard so that can easily be applied to wider manufacturing applications.

• **Querying Information:** There is a requirement to query and update information. This guides the configuration change. This tool has the following requirements:

- Must be able to interface with the information capture and coordination tool.
- Control tool can access this for querying information for real-time decisions.
- Can update information in the information tool in real-time.

• **Interaction Between Layers:** There is a requirement to interact between different layers of the adaptation strategy. This tool has the following requirements:

- Can integrate all layers of the adaptation strategy.

- Can work with all tools employed in all layers of adaptation strategy.
- **Hardware Abstraction:** There is a requirement to interact with the physical system for executing functionality. This tool has the following requirements:
  - Can interact with the coordination tool.
  - Has the capability to extract information from the system, update information and execute functionality.

The techniques in place are then integrated to serve the purpose of self-configuration. Figure 3.4 presents a step-by-step technological development for the architecture.

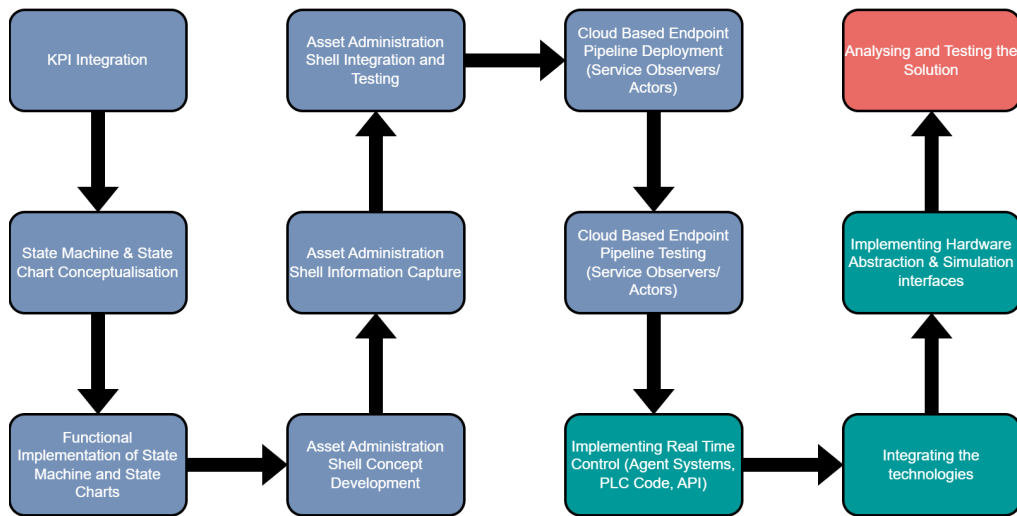


Figure 3.4: Technological development for the self-configuration research. Each stage behind the development of the adaptation strategy is shown.

The technological development enlisted can be evaluated by accessing them against the enlisted needs:

- KPI Integration is contributing towards **Querying Information**.  
(*Data Component*)



- State Charts and State Machines is contributing towards **Coordinating Functionality Execution**. (*Control System*)
- Asset Administration Shell is contributing towards **Information Capture**. (*Asset Representation*)
- Cloud-Based ML Pipeline is contributing towards **Interaction Between Layers**. (*Parameter Optimisation*)
- Agent System/PLC Code/API is contributing towards **Controlling Functionality** and **Hardware Abstraction**. (*Control System*)

The manner in which these tools contribute to the research objectives is listed in the implementation chapter (Chapter 9).

## 3.4 Validation Methods

### 3.4.1 Industrial Use-Cases

The research presents industrial use cases demonstrating the self-configuration approach in leak-testing and force-testing applications (Chapter 9). These use cases validate the adaptation strategy and its potential impact in real-world production environments. The leak-testing application aims to find the right configuration setting for product volume and test pressure. The force-testing application determines test configuration guided by image recognition of the fixture. The setups for the industrial use-cases is illustrated in figure 3.5. Table 3.2 provides the mapping of the specific research questions established in Chapter 1 to the implementation objectives of industrial use cases. This mapping of the implementation objectives enables validating this research against the objectives.

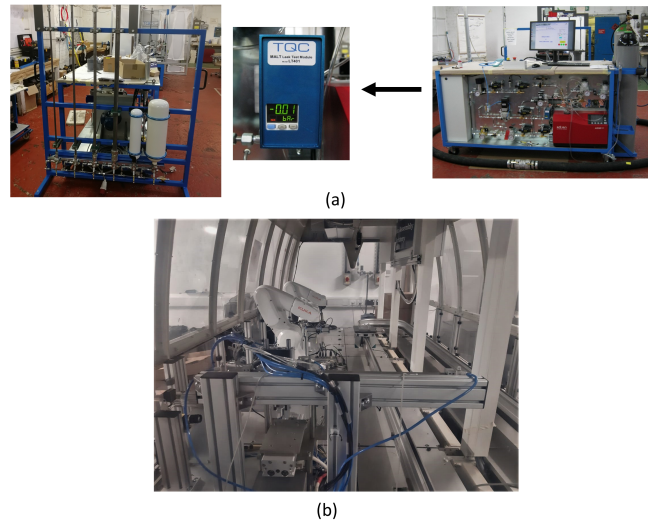


Figure 3.5: The leak testing setup: (a) the cylinder volumes under test and MALT test system being a part of the test bench for general leak testing (b) The PRIME force testing setup involving a force test station, two robots and a shuttle on the rail.

Table 3.2: Mapping the research questions to the industrial use-cases.

Research Questions	Implementation Objective
What theoretical models are needed to underpin the concept of self-configuration in production systems?	Capture information on industrial use case functionality.
How can a general adaptation strategy be developed to integrate self-configuring COs into production systems?	Classify industrial use cases in terms of features, model their configuration and define the granularity.
Which technologies, tools, and techniques best facilitate self-configuration and leverage operational data for iterative improvement?	Utilise the tools and techniques developed in this research to achieve self-configuration in industrial use cases.
How can this approach be implemented in a real-world scenario involving a product within the production system?	Demonstrate functionality execution after self-configuration through adaptation strategy.
How can business objectives be translated downstream to guide the self-configuration process at the production system level?	Integrating business objectives to industrial use cases to determine the best setting for functionality.

### 3.4.2 Research Findings

Key findings from this research contribute to the field of smart manufacturing (Chapter 9). These include the developed adaptation strategy for self-configuration, the classification system for production systems, exploration into agent systems for manufacturing control, integration techniques through interoperability, and a framework for understanding configuration in production systems.

## 3.5 Overview of the Approach

The approach to self-configuration consists of multiple constituent components. The approach is detailed in figure 3.6. The components are listed as follows;

- **Asset Representation:** This gives digital representation to the physical manufacturing asset. A manufacturing asset is an aggregate responsible for performing operations (i.e. functionality). In this research, AAS provides digital representations of physical assets, their capabilities, and configuration settings.
- **Control System:** The control system executes functionality, by communicating with asset representation. This is a tool that is used to achieve the objective.

In this research, the developed component orchestrates functionality execution, communicates with asset representations, and implements configuration changes in the physical system.

- **Data Component:** Internal and external data elements capture data for initial machine settings and business objectives. These will

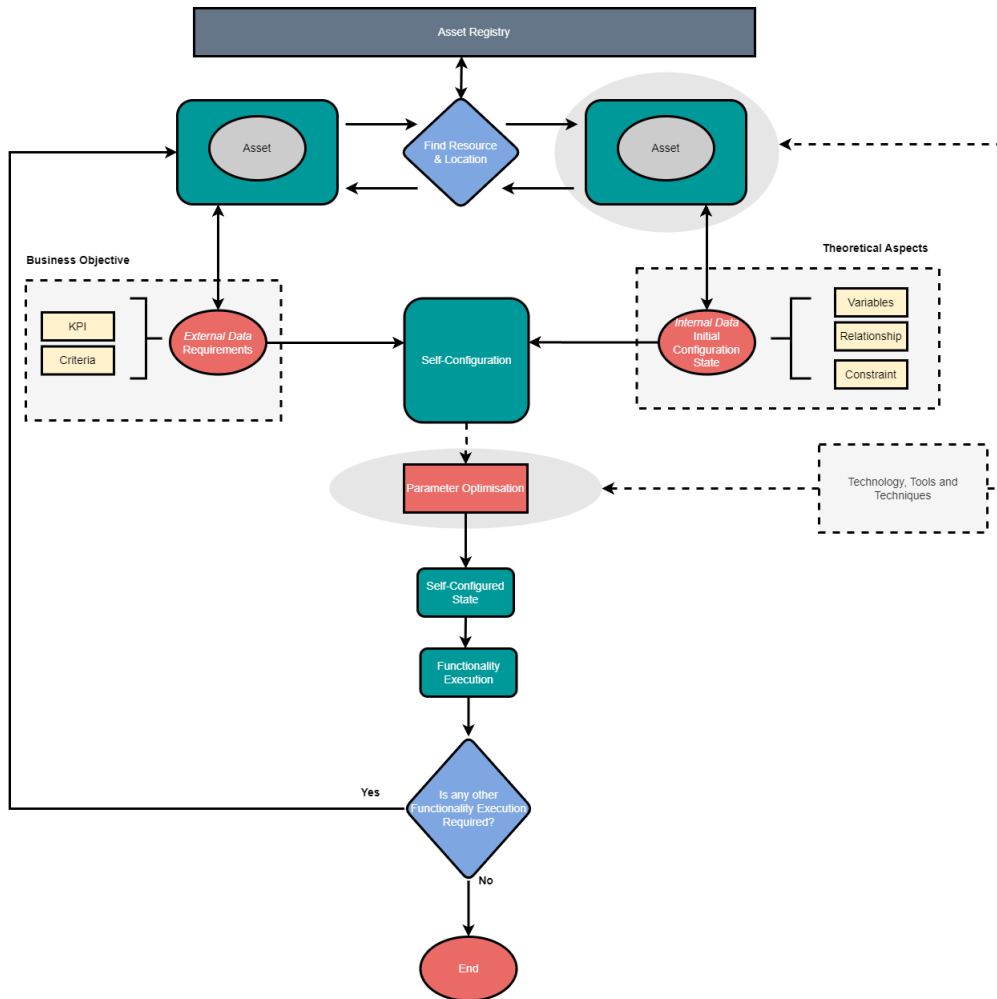


Figure 3.6: Proposed Self-Configuration Approach for the research. Integrating Asset Representation, Agent System, and Parameter Optimisation. The diagram shows component mapping with research problem objectives. These components work together to assist in the realisation of the self-configuration adaptation strategy

guide the self-configuration. A tool that supports information capture in the production system must be utilised.

In this research, a system must be developed for capturing real-time production data, KPIs, and other relevant information.

- **Parameter Optimisation:** The parameter optimisation is realised within the adaptation strategy.

In this research, machine learning models analyse data to suggest optimal configurations aligned with KPIs.

**Self-Configuration Adaptation Strategy:** This strategy provides guidance for achieving self-configuration in production systems. The information gathering tool, asset representation, control system and parameter optimisation are used with theoretical insights to develop an approach to the realisation of self-configuration in production systems. It provides a systematic framework for achieving self-configuration.

### 3.6 Thesis Structure

The thesis structure is illustrated in figure 3.7. Potential contributions are divided into thesis chapters and related to the needs of the RQ.

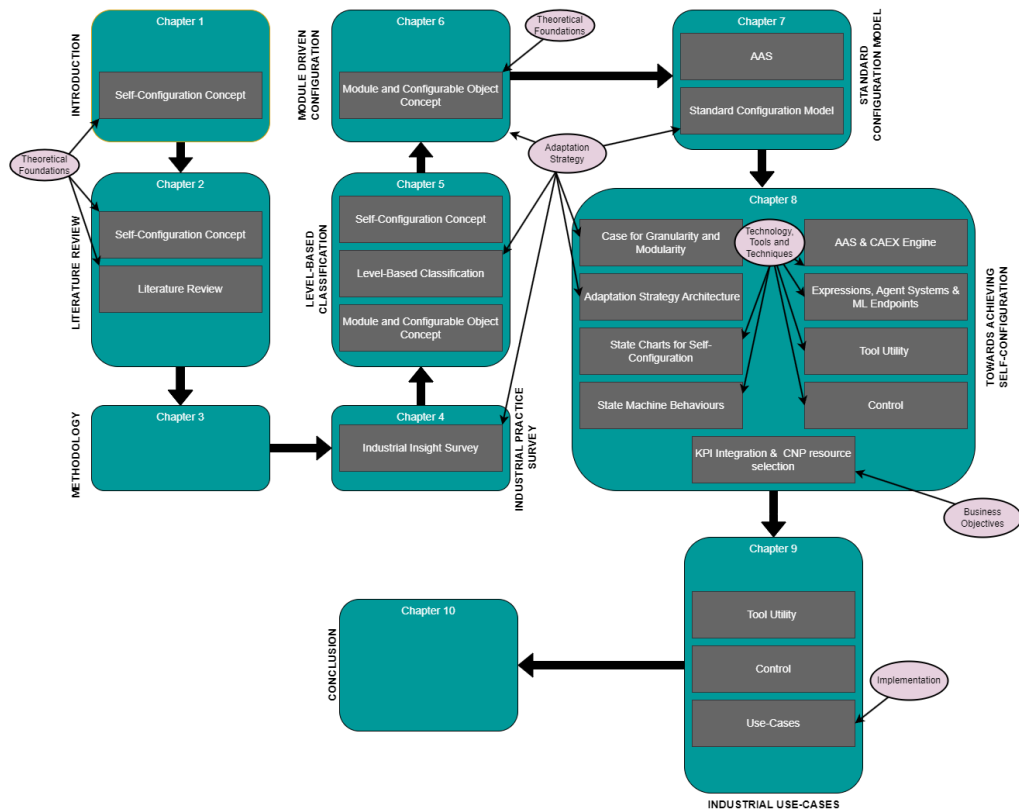


Figure 3.7: Ph.D. Thesis structure, relating the contributions to needs and dividing them into chapters.

## 3.7 Conclusion

This methodology chapter outlines a comprehensive and structured approach to addressing the research question. The combination of a thorough literature review, practical insights from the Industrial Practice Survey, and a novel classification system provides a strong foundation for the development of the adaptation strategy. The emphasis on data modelling, interoperability, and the integration of state charts and ML techniques ensures a robust and data-driven solution. Validation through industrial use cases and the identification of key research findings further solidify the potential impact of this work.

---

## Chapter 4

# Industrial Practice Survey For Adoption Of Self-Configuring Production Systems

---

## Contents

---

4.1	Introduction . . . . .	<b>83</b>
4.2	Survey Description . . . . .	<b>85</b>
4.2.1	Objectives and Research Questions . . . . .	87
4.2.2	Target of Survey . . . . .	89
4.2.3	Survey Format . . . . .	90
4.3	Analysis And Result . . . . .	<b>91</b>
4.3.1	Industrial Challenges in Adopting Intelligent Production Systems . . . . .	91
4.3.2	Data in Manufacturing . . . . .	96
4.3.3	Self-Configuring Production Systems in Manu- facturing . . . . .	99
4.3.4	Survey Discussion: . . . . .	102
4.4	Threats to Validity . . . . .	<b>103</b>
4.4.1	Internal Validity . . . . .	103
4.4.2	External Validity . . . . .	104
4.5	Reflection on Impact . . . . .	<b>104</b>
4.5.1	Reference Market/Stakeholders . . . . .	106
4.5.2	Maturity Level . . . . .	106
4.5.3	Value Proposition Matrix . . . . .	108
4.5.4	Integrated Business Model (IBM) - GAP Matrix	108
4.5.5	Status for Innovation . . . . .	109
4.5.6	Potential Value Utilisation . . . . .	109
4.6	Conclusion . . . . .	<b>110</b>

---



## 4.1 Introduction

In manufacturing industries, the configuration in production systems is set at the start of the production operation, dependent on the product, process, and customer requirements (Rehman et al. (2021b)). These configurations mainly rely on previous settings for a similar product, the production restrictions (i.e. on and by the system), the engineer's expertise, or converging from an arbitrary configuration to a setting for the product. This takes a lot of time to achieve the correct configuration, as several cycles of corrections need to be done due to multiple dependencies. With the growing demand in customisation requirements, this increases time to market and costs associated with the production operation (Tuck and Hague (2006)).

Capturing configuration in production systems is previously discussed in the literature chapter. There have been strategies such as cloning or loading through a database where a suitable configuration can be set for a production system, making the configuration operation faster (Bachula and Zajac (2013); Lee et al. (1997); Jones and Romig (1991)). These existing strategies cannot be applied to the current shift in the manufacturing paradigm, due to the mass customisation of the product. Previously, it was acknowledged that the process must remain independent of the product, which means that the production equipment performs the functionality without having any relevance to the product (e.g. test station just performs the operation and does not take into account that the product needs to be in a fixture).

In Chapters 2 and 3, the definition of self-configuration is proposed to address the instated configuration capture issue in production systems involving configurable objects. This is mainly done to promote the flexibility of automation and the ability to respond to changes in production systems.

By achieving self-configuration, time to market can be reduced, as less time is needed to overhaul the system to specific product needs.

As seen in figure 4.1 setting configuration for a production system is a collaborative endeavour taking into account various dependencies like the product, process (functionality), customer requirements and expertise. It also requires constant verification to preserve the functionality aspects of the production system. A strategy for self-configuring production systems is proposed in this thesis.

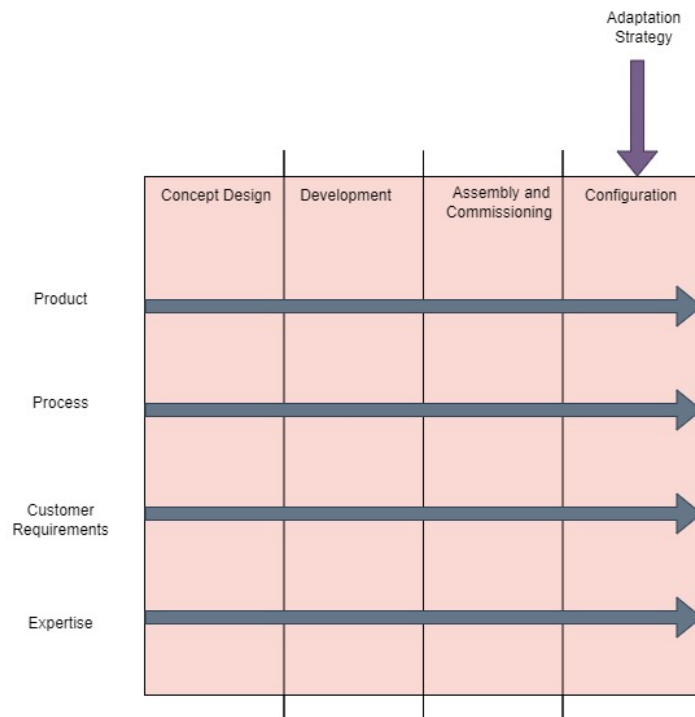


Figure 4.1: Collaboration related to production system development. The adaptation strategy targets the last configuration stage of the production system development.

Despite the advantage, there is a significant lack of interest in self-configuration in industrial practices. This chapter focuses on the determination of industrial readiness and the barriers that manufacturing organisations face when implementing such a strategy. There have been studies that focus on the modularisation of equipment to achieve some kind of reconfiguration in terms of the physical arrangement, but integration is often taken

as an assumption. Simply, it is typically understood that as long as the physical arrangement is possible, then the equipment modules are able to work/adapt. This work addresses the strategy to adjust settings for a production system (i.e. self-configuration) on existing production systems. The work in this chapter illustrates the data infrastructure capability in existing production systems and organisations. The barriers to the adoption of smart production systems and their adaptation for self-configuration are surveyed.

Therefore, this study presents an empirical survey to assess the current industrial readiness, needs and challenges that the industries face towards adaptation of Self-Configuring production systems.

## 4.2 Survey Description

The survey was carried out in three parts, following the methodology presented in Fernández-Sáez et al. (2015). This methodology is briefly stated as follows:

1. **Problem Identification and Goal Setting:** The research sought to understand the challenges hindering the adoption of intelligent and self-configuring production systems within the manufacturing industry, particularly for SMEs. The goals of these surveys are:
  - Identify barriers to adoption.
  - Understand the state of data infrastructure and value extraction in manufacturing.
  - Gauge industry readiness and requirements for self-configuring production systems.

2. **Research Design:** The study employed a primarily descriptive survey-based approach to gathering insights from industry professionals. Three interconnected surveys were designed to progressively explore the core research areas. This allowed for triangulation of data to gain a more comprehensive understanding.
3. **Sampling:** Experienced manufacturing professionals (e.g. practitioners, researchers, academics) with a focus on SMEs. A combination of purposive sampling (i.e. targeting specific groups relevant to the study) and convenience sampling (i.e. using accessible networks like NMN and the DiManD project) was used in these surveys.
4. **Data Collection and Analysis:** Online surveys were the primary mode of data collection. Three carefully designed survey questionnaires were piloted and refined before wider distribution. The surveys used a mix of closed-ended (e.g., multiple-choice) and open-ended questions to capture both quantitative and qualitative data. The collected data likely underwent both qualitative and quantitative analysis. Statistical analysis was done of closed-ended questions for identifying trends, percentages, and potential correlations. Thematic analysis of open-ended responses was done to uncover deeper insights, opinions, and potential challenges or requirements.
5. **Interpretation and Reporting:** The analysis was used to conclude the research questions, addressing barriers, data infrastructure, and the potential for self-configuring manufacturing systems. The findings are illustrated in this chapter.

### 4.2.1 Objectives and Research Questions

The survey attempts to address the barriers and challenges faced by the manufacturing industry towards adopting intelligent production systems, where intelligence is regarded as the ability to act autonomously and make decisions. The survey is also used to gain insight into data infrastructure in the manufacturing industry with a focus on value extracted from the gathered data. These surveys are combined with another series of questions to get industry professionals' opinions on self-configuring production system requirements. The result of the survey is analysed and a comparison of self-configuration requirements to the data infrastructure and barriers/challenges is provided.

The objective of these surveys is to get awareness of the barriers/challenges and data infrastructure present in manufacturing towards achieving self-configuring production systems. These surveys will also assist in the identification of existing gaps and further research directions on this matter. The Questions (Q) in each survey are mapped to the current industrial insights with the motivation for self-configuration.

In the first survey "Industrial Challenges in Adopting Intelligent Production Systems", the focus is on industrial understanding of intelligent production systems and the major challenges and barriers faced in adoption. This survey also probes the experience of industry professionals with existing intelligent systems, their opinions on the most beneficial characteristics for productivity, and where intelligence should reside within a system. This helps in assessing industrial readiness towards ever-increasing complexity in manufacturing that requires the distribution of intelligence among production system components, challenges of adoption of such components and the potential timeline for overcoming these challenges.

In the second survey “Data in Manufacturing”, the value of data gathered in manufacturing is ascertained. The series of questions helps in understanding the relationship between gathered data and value through identification, filtration, and processing. The value attained from the data can be linked to economic benefits (e.g. increase in productivity/quality). To acquire value, through this survey, the ‘useful’ aspect of gathered data, ‘useful’ filtering and analysis, ‘useful’ transformation and ‘good’ decisions are to be understood. The questions also assess the frequency of data-driven decision-making, challenges in data analysis, the types of data analysis techniques used (i.e. descriptive, diagnostic, predictive, prescriptive), and the industrial network protocols employed within the organisation.

In the third survey “Self-Configuring Production Systems in Manufacturing”, the decision-making capability of production systems concerning specifically setting up parameters, optimisation and executing functionality, i.e. self-configuration, is explored. Through this survey, the industrial readiness level for this decision-making in current production systems is determined. It examines current practices for setting up machines, opinions on the benefits of machines that adjust their settings automatically, the necessary technological upgrades, and the most advantageous applications. The questions also delve into the perceived challenges, potential solutions for common manufacturing issues, and strategies for adopting such machines.

### 4.2.2 Target of Survey

#### **Target Population:**

The target population for the survey were experienced industry practitioners, researchers, and academicians. The focus was on people working or associated with SMEs. The target population were involved in roles such as engineers, production managers, quality technicians, and researchers specialising in manufacturing.

To reach this population, a combination of purposive and convenience sampling was used. The University of Nottingham and TQC Ltd. partnered to identify relevant participants.

#### **Sampling Technique:**

A combination of purposive sampling (i.e. targeting specific groups relevant to the study) and convenience sampling (i.e. using accessible networks like NMN and the DiManD project) was used for the surveys.

The survey was carried out in Nottingham, United Kingdom with the assistance of the University of Nottingham and TQC Ltd. The initial set of actions consisted of an assessment of answers to the pilot surveys dispatched at TQC Ltd. The assessment was carried out to determine the quality of the answers received and to ascertain if additional questions need to be added. These surveys were then dispatched to Nottinghamshire Manufacturing Network (NMN). Finally, the surveys were carried out among early-stage researchers and beneficiaries of the DiManD Horizon 2020 project to get a wide range of responses.

Participants were primarily located in Europe.

### 4.2.3 Survey Format

Three surveys (3) were carried out covering challenges/barriers, data infrastructure and self-configuring production systems. Each survey was divided into logical sections that build on the survey theme incrementally, going into more detail with later questions. Figure 4.2 illustrates this logical divide into sections giving the format of the survey. The challenges/barrier survey and data in the manufacturing survey are related to the self-configuring production system survey. Figure 4.3 details the link between these surveys with a focus on self-configuration.

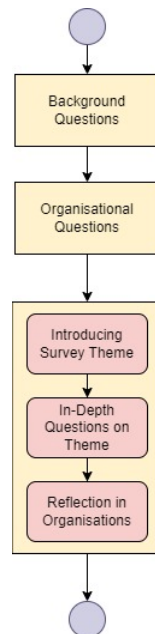


Figure 4.2: Structure of the questions in the surveys.

- The first section consists of background questions on the participant. They involve questions on residents' country, occupation, job position and work experience.
- The second section consists of organisational questions that target the main theme of the respective survey. The technical questions deal in more detail with the theme of the survey, asking in-depth questions



Focus	Number	Question	Relation	Objective
Self-Configuration	1	Benefit of machines that automatically adjust their settings.	Data in manufacturing	To get an opinion on how significance of automatically adjusting machines in production environment.
	2	Addition in your existing technological infrastructure.	Data in manufacturing	To look for potential gaps, needing to be filled for such machines.
	3	Selection of the settings of a machine.	Data in manufacturing	To get an idea on what factors decide the configuration.
	4	Application of machines that can automatically self- adapt	Challenges	To get an insight on expanding applications of such machines
Challenges	5	The challenges in using machines that automatically adjust their settings.	Challenges	To observe the Industrial insight about perceived challenges.
	6	Relating challenges to manufacturing process problems.	Challenges	To relate perceived problems to production problems. A way to integrate solution to these during strategy development.
	7	Frequency of problems relating to configuration in production life-cycle.	Challenges	This would help to further converge the problems to respective phases in life-cycle.
Strategy	8	Best strategy for adaptation of self-configuring machines.		Insight on major strategy phases.
	9	Evaluation tool that assists in determining the industry 4.0 competency of machines.		A model checking tool for determining the state of self-configuration of system, also linking to adaptation strategy.

Figure 4.3: Focus of the surveys carried out. Planning is done to gain insight from professionals on self-configuring aspects of production systems.

on the topic of the survey theme. These questions aim to address practical aspects of the theme, asking the target population about challenges/barriers, data and self-configuration.

### 4.3 Analysis And Result

Figure 4.4 illustrates a brief overview of the surveys. This figure shows an overview of the main findings of each survey. The surveys carried out highlight a need for addressing challenges in data security, demand, and adaptation. A solution to the lack of skills in machine settings and the shortage of field experts is also desired.

#### 4.3.1 Industrial Challenges in Adopting Intelligent Production Systems

This survey aimed to gather insights from a variety of participants, including academia, non-profit organizations, government entities, and industry professionals. The survey included questions about the characteristics required to make a production system ‘intelligent’, the criteria of intelligence

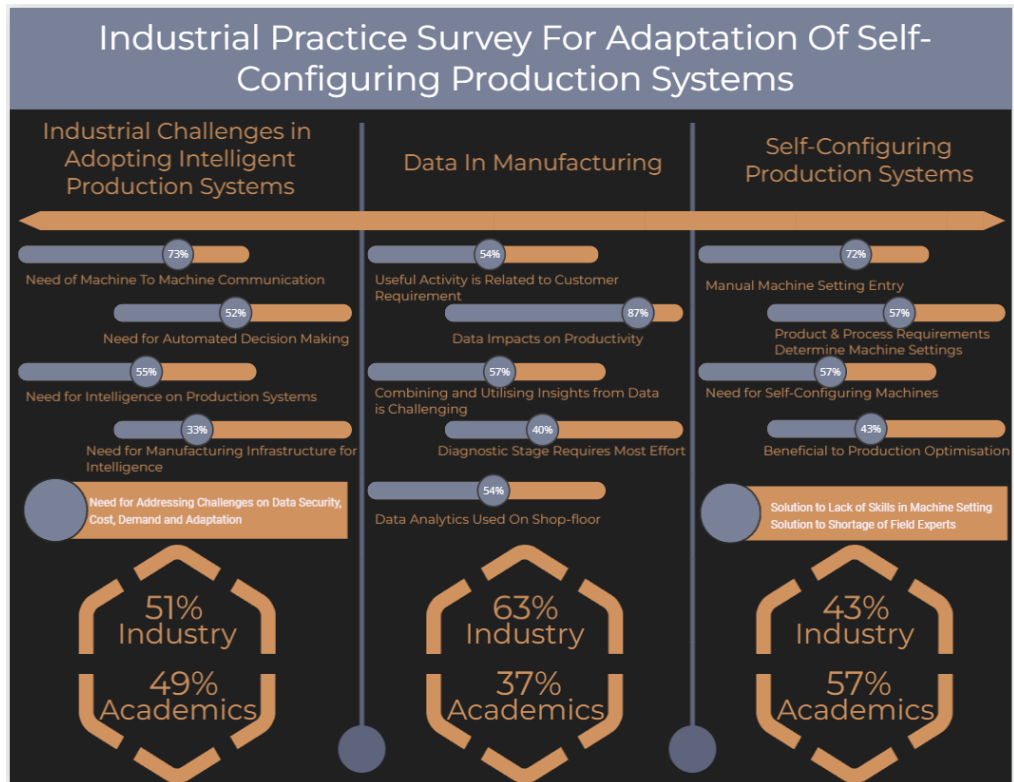


Figure 4.4: An infographic of the Surveys. The main findings of the survey are presented. Relation between industrial challenges, data in manufacturing, and self-configuring systems explored. A need exists for addressing challenges in data security, demand, and adaptation. Self-configuring systems will provide a solution to the lack of skills in machine settings and the shortage of field experts.

in production systems observed at manufacturing facilities, and the best criteria for increasing productivity without excessive complexity or cost. Additionally, it addressed where intelligence should be located in production systems, the capability of current manufacturing infrastructures to support intelligent production systems, and challenges faced in adopting intelligent production systems, along with estimated timeframes for overcoming these challenges.

#### Survey Participants:

- A total of 33 participants responded.

- The majority of respondents (51.5%) were from industry (SMEs).
- Participants had diverse backgrounds, with engineers (30%) and researchers (20%) being the most prevalent. The rest were (17 %) Project managers, (3%) innovation managers, (3%) technology managers, (7%) software projects, (10%) directors, (7%) production engineers, (3%) R&D managers.
- A significant proportion of respondents (44%) had more than 20 years of experience. This was followed by 7% participants with 10 to 20 years of experience. The survey also received responses from professionals with 5 to 9 years of experience (16%), and less than 5 years of experience (19%).

**Key Survey Findings:**

1. Characteristics for an Intelligent Production System.
  - The majority of industry professionals defined intelligence in production systems as having machine-to-machine communication capability (72.7%).
  - A significant percentage believed that intelligent systems should be capable of making decisions and analysing data themselves (69.6%).
  - Cooperative relationships between intelligent machines and prediction capabilities were also considered essential (60% and 51%, respectively).
2. Criteria of Intelligent Production Systems at Manufacturing Facilities. The criteria is illustrated in figure 4.5.

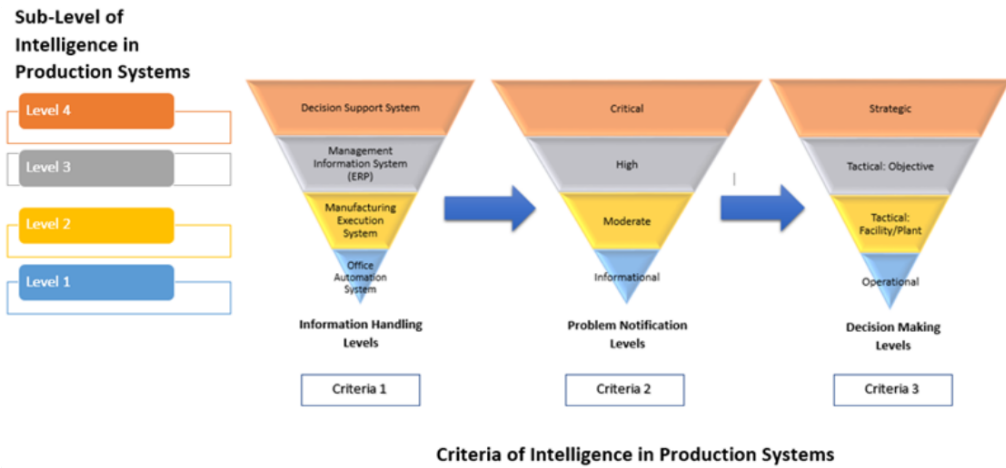


Figure 4.5: Criteria of intelligence in production system. Each criterion of intelligence in the production system in the industry is shown and participants are asked to rank their current system on it.

- For information handling, most production systems were identified as being at Level-1 and Level-2 (36.4%).
- Problem identification was typically at Level-2 (39.4%).
- Decision-making in production systems was most commonly at Level-3 (33.3%).

### 3. Preferred Criteria for Increasing Productivity.

- Respondents favored a shift to Level-3 for information handling (36.4%).
- Problem identification should ideally be at Level-3 (48.5%).
- Current Level-3 was considered the best for decision-making in production systems (51.5%).

### 4. Location of Intelligence.

- A majority believed that intelligence should reside on the production system itself (54.5%).
- Some respondents (33.3%) suggested shared intelligence on a network connected to the production system.

- A portion (18.2%) considered intelligence on both the production system and a shared network.

#### 5. Manufacturing Infrastructure.

- About a third of industry respondents (33%) didn't think they had the right manufacturing infrastructure for incorporating intelligence.
- A smaller percentage (24%) believed they had the appropriate infrastructure.

#### 6. Challenges in Adoption of Intelligent Production Systems.

- Challenges related to data security, bias/resistance to change, high cost/investment, business process re-engineering, uncertainty in demand, and adapting to changing conditions were highlighted.
- Responses were distributed across these challenges.

#### 7. Timeframes for Overcoming Challenges.

- Most challenges could be addressed within 2-3 years, with some progress possible in 9-12 months.
- This indicates that the industry is not far from accepting intelligent production systems, and challenges can be tackled on a priority basis.

These findings provide valuable insights into how industry professionals perceive and approach the adoption of intelligence in production systems, including their definitions, criteria, challenges, and the state of their infrastructure. It's evident that there's a growing interest in making production systems more intelligent, with a focus on machine-to-machine communication and decision-making capabilities.

### 4.3.2 Data in Manufacturing

This survey aimed to collect information from a diverse group of participants from different sectors, including industry, academia, government, and other areas. It focused on the role of data in manufacturing, particularly in the context of production systems. Here are some key findings:

#### Survey Participants:

- There were 30 participants who responded to the survey.
- The majority of respondents (63%) were from the industry, particularly SMEs.
- Various professional backgrounds were represented, including engineers (21%), researchers (21%), directors (11%), (7%) engineering manager, (7%) project engineer, (7%) production engineer, (4%) operations director, (4%) supplier development, (4%) head of department, (4%) finance director, (4%) consultant and (6%) others.
- Most participants (60%) had over 20 years of experience. This was followed by (17%) with 10 to 20 years of experience. The survey also had responses from professionals with 5 to 9 years of experience (7%), and less than 5 years of experience (17%).

#### Key Survey Findings:

##### 1. Relevant Manufacturing Activities.

- The most relevant and “useful” activity in manufacturing, according to a majority of respondents (53.3%), is related to ful-

filling customer requirements. This suggests a strong customer-centric focus.

- Activities that reduce risks, defects, and improve quality were considered important by 23.3% of respondents.

#### 2. Data Categorization.

- Historical data about observed processes was the most significant method for categorizing “useful” data, according to 30% of respondents.
- Data source-based and mathematical inferences were also considered significant (26.6% each).

#### 3. Impact of Data at Different Stages.

- Customer data was seen as most important during requirement gathering.
- Guidelines were considered vital during conceptual design.
- Material data and guidelines played a key role in various design phases.
- Process data was crucial for production planning and manufacturing.
- Customer and lifecycle data were influential during dispatching.

#### 4. Data Impact on Productivity.

- A majority (86.7%) believed that data impacts decision-making on the shop floor, underscoring the importance of data in real-time operations.

#### 5. Challenges in Data Analytics.

- Combining and utilizing insight from analysed data were rated as the most challenging aspects of data analytics (56.7% each).
- Collecting data from various sources was also seen as challenging (56.7%).

6. Time and Effort in Data Analytics Stages.

- The diagnostic stage required the most effort (39.7%), while the prescription stage required the least effort and time (26.7%).

7. Use of Data Analysis Techniques.

- The majority of respondents had some knowledge of data analysis techniques at their organisations.
- For descriptive data analysis, statistical reporting was the most commonly used technique (26.7%).
- For diagnostic data analysis, root cause analysis and FMEA were the leading techniques (13.3%).
- Data visualization was also commonly used (13.3%).

8. Use of Predictive Data Analysis Techniques

- A majority (40%) had some knowledge of predictive data analysis techniques.
- For predictive data analysis, trend analysis, what-if analysis and simulation were the most commonly used techniques.

9. Use of Prescriptive Data Analysis Technique.

- A significant number (33%) had some knowledge of prescriptive data analysis techniques.
- Discrete choice modelling and linear programming were commonly used techniques (6%).



10. Industrial Network/Automation Protocols.

- EtherNet I/P was the most commonly used industrial network/automation protocol (56.6%).
- Manual data gathering (46.7%) was also prevalent.
- Other protocols, like WLAN (33.3%), EtherCAT (26.7%), and OPC UA (20%), were also in use.

These findings provide valuable insights into the role of data in manufacturing, data categorization, data impact, and the utilisation of data analysis techniques. The survey indicates that data plays a significant role in decision-making and product development across different manufacturing stages. EtherNet I/P is a widely used protocol, suggesting a need for data communication and automation in manufacturing settings.

### 4.3.3 Self-Configuring Production Systems in Manufacturing

In a survey about self-configuring production systems in manufacturing, industry professionals and research academics provided insights:

#### Survey Participants:

1. **Survey Demographics:** The survey received responses from 43% industry professionals and 57% academics. The participants had various backgrounds, with 57% being researchers, 14% directors, 14% engineers, and 15% involved in production operations. Experience levels varied, with 14% having over 20 years of experience, 14% with

10 to 20 years, 57% with 5 to 9 years, and 14% with less than 5 years of experience.

**Key Survey Findings:**

1. **Entering Machine Settings:** When asked how settings in a machine were typically entered before operation, 72% preferred manual entry, 14% used digital interfaces, and 14% used a combination of both.
2. **Determining Machine Settings:** For determining machine settings, 57% relied on product and process requirements, 29% on customer requirements, and 14% on in-house developed machine settings.
3. **Benefits of Self-Adapting Machines:** A majority (57%) believed that machines automatically adjusting their settings would significantly reduce time and costs. Some (29%) thought it would reduce costs and increase productivity, while others (14%) considered it could decrease process time.
4. **Technological Infrastructure for Self-Adaptation:** Respondents (43%) felt that a data connection between systems was required for enabling self-configuring machines. Others (29%) believed a standardized data format, along with data connection, was needed. Some (28%) thought no change in their technological infrastructure was necessary.
5. **Beneficial Applications:** Respondents saw the most beneficial applications for self-adapting machines in production optimization (43%) and in setup and changeover processes before operation (43%).

6. **Domains of Application:** Respondents believed self-adapting machines would be most beneficial in the domains of factory automation, machining, assembly, and machine efficiency. These machines could also impact downtime minimization and promote process integration.
7. **Adoption Strategies:** There was no clear consensus on the best strategy for adopting self-configuring production systems. The options included incremental adoption, big bang adoption, parallel adoption, and incremental parallel adoption, with varying preferences (28-29%).
8. **Evaluating Self-Adapting Capability:** All respondents (100%) agreed that a tool for evaluating their machines' self-adapting capability would be helpful.
9. **Challenges in Implementation:** Respondents identified the major challenges in implementing self-adapting machines, including a lack of knowledge about product/process and production systems (rated as a major challenge by the majority). Other challenges included the lack of process benchmarks, infeasibility of deployed solutions, and compatibility issues.
10. **Addressing Manufacturing Issues:** Respondents believed that self-configuring machines would significantly address common manufacturing issues, particularly overcoming the lack of skill in setting up machines, high implementation costs, and compatibility issues.
11. **Solving Operational Challenges:** For operational challenges, respondents believed self-configuring machines could significantly address problems related to experience requirements, predicting the right settings under constraints, and reacting to market demand.

These insights reflect the views of industry professionals and academics on the potential benefits, challenges, and strategies related to self-configuring production systems in manufacturing.

#### **4.3.4 Survey Discussion:**

The findings across the three surveys provide a holistic view of the current landscape regarding intelligent and adaptive manufacturing:

##### **Desire for Intelligence:**

A clear understanding and desire exists for intelligent systems that utilise machine-to-machine communication and data-driven decision-making capabilities. It is noted, however, that the current state of infrastructure in manufacturing facilities may not be able to realise this. There are prevalent concerns about data security, cost of systems, demand variation and result adaptation. Skill gaps are apparent, especially where machines need to be configured to account for these hurdles.

##### **Utilising Data:**

Data is recognised to impact decision-making, yet most data analytics efforts are concentrated in the diagnostic phase. The potential power of predictive and prescriptive analytics is recognised, but adoption is lower. This presents an opportunity for improved production efficiency and foresight. The use of a common communication protocol between manufacturing assets shows a positive trend towards automation. In the same way, there exists an ongoing challenge of integrating data sources from diverse systems

with potentially incompatible protocols.

#### **Self configuration Challenge:**

The findings show that self-configuring systems will reduce time, cost, and complexity. This presents a strong case to reduce the reliance on manual settings and make production lines more responsive. The findings indicate that there is no single dominant adaptation strategy, hinting at the need for customised approaches depending on the individual manufacturing setup. The core challenge, as demonstrated by the findings, remains the field expertise and skill gap to set up the machines. It requires an understanding of the complex relationships between products, processes, and the machines themselves.

## **4.4 Threats to Validity**

### **4.4.1 Internal Validity**

A threat to the internal validity of the survey can be the sample size of the respondents. A diverse range of participants from different backgrounds and experience levels are included in the surveys to overcome this. Responses were received from participants belonging to different sectors, but mostly from manufacturing automation, being the main focus of the surveys.

Another threat to the internal validity faced in these surveys that can arise is that the respondents do not understand the questions. Each survey theme is introduced and contextualised at the beginning of each survey to

overcome this. Each question has been explained in much detail and for clarity has also been related to industrial applications if possible. Pilot surveys were conducted to ensure survey questions were delivered accurately and mitigate ambiguity. After receiving feedback from the pilot surveys, certain questions were clarified and reformatted.

#### **4.4.2 External Validity**

The external threats to validity may be related to the demographics of the respondents. The survey was mainly limited to the manufacturing organisations in the United Kingdom. Responses were also received from other countries. However, most responses originate from the UK. Therefore, the results cannot be generalised for all manufacturing companies. The survey represents responses from manufacturing organisations well enough to represent a trend for self-configuration.

### **4.5 Reflection on Impact**

Based on the surveys, it is observed that the data infrastructure for the realisation of self-configuration is present. Most of the configuration operation is carried out manually or automatically on existing production systems. There are some requirements on configuration settings on the product and process, along with customer requirements.

As per responses gathered from the three surveys, it is observed that there exists a need for self-configuration in production systems. There are challenges and barriers to the adoption of this ‘smarter’ production system, but most of the data infrastructure is in place. There exists a realisation

in manufacturing industries on increasing productivity by leveraging data-driven decision-making advantage. Through the surveys, it is observed that most professionals realise the limits of their organisational capability and also have some ideas in place for overcoming their restrictions. For the research in this thesis, this input was crucial in shaping the configuration concepts, interoperability approach, and adaptation strategy.

One interesting insight observed from the surveys is the need for a tool that assists in evaluating the current capability of production systems for self-configuration ability. This will be quite useful as a means of identifying potential areas of improvement and providing direction towards self-configuration.

Configuration in a production system is dependent on data and on multiple aspects during the data processing stages. The data that impacts the production system can be categorised, and each category can be important at different stages of the production life cycle. In a manufacturing organisation, there are multiple techniques that are effective in developing insight at different stages of production. These techniques have data requirements and other prerequisites. The adaptation strategy that makes the production system automatically adjust its settings must take into account these factors in all scenarios that might impact production.

The survey response (figure 4.4) outlines that there exists a need for self-configuring production systems. In this section, the value that such an application can add to a business in terms of a business model is explored. The reference market for such an application is discussed, along with an evaluation of the maturity level of such markets. Going into more depth, the value proposition at the production system lifecycle is analysed. After identifying the value, it is related to the business model and value

dimensions about strategic, customer, market, and creation components are explored. The status of innovation is highlighted and supported by illustration and validation.

#### **4.5.1 Reference Market/Stakeholders**

The self-configuration adaptation strategy for production systems can be applied to specialised applications that require product, process, and experience-oriented setup of configuration settings. These reference markets may be in automotive, aerospace, testing, and pharmaceuticals among others. The reference markets may fall within any area of application that requires a highly customised production system set up to meet varying demands.

#### **4.5.2 Maturity Level**

Enabling reference markets to adopt self-configuring production systems, is essential to determine the maturity of the current production systems. Chapter 5 discusses this, where a method to classify production systems for self-configuration is provided. Table 4.1 discusses the self-configuration in respect to three dimensions: creation, proposition and capturing.

---



Creation	<p>Production systems should be able to capture and handle data requirements for the configuration change strategy. The following must be in place;</p> <ul style="list-style-type: none"> <li>• Data management capability (collection, storage, conversion, and transmission) in the production system.</li> <li>• Control system for the production process.</li> <li>• Optimisation capability.</li> <li>• Operational identification and context awareness through the controller (hardware and software).</li> <li>• Monitoring capability through production system-specific interfaces.</li> <li>• Self-capability in the production system (i.e., has the capability to infer meaning from the state).</li> </ul>
Proposition	<p>The configuration change strategy can give autonomy to the production system to make decisions based on product, process, and business goal needs. Such production systems focus on business-specific objectives rather than spending effort on changing settings to accommodate those needs.</p>
Capturing	<p>The configuration change strategy saves time, i.e. the machine takes the decision itself. Also, less reliance on experience-oriented personnel to change settings. More compliance with changing day-to-day business needs.</p>

Table 4.1: Maturity level of Production Systems for Enabling Self-Configuration

### 4.5.3 Value Proposition Matrix

The value proposition matrix expands the value to the production system life-cycle. The new value brought by realising self-configuring adaptation strategy in existing production systems is traced to the domain (of realisation) and life-cycle stages (Table 4.2). In the business model, the new value can be taken advantage of by the owners of the domain at each life-cycle stage.

Value Proposition	Lifecycle				
	Creation	Purchase	Use	Renewal	Transfer
Control	New Value			New Value	Customer
Skill	Manufacturer	New Value			
Workflow	Production System				
Platform	Manufacturer	Supplier		Manufacturer	Customer
WorkStation		Customer			
Module	New Value	Manufacturer		New Value	

Table 4.2: Value Proposition Matrix for enabling Self-Configuration, linking stakeholders to lifecycle.

### 4.5.4 Integrated Business Model (IBM) - GAP Matrix

The business model is further expanded along with the value gained by a focal company (FC) that uses the potential self-configuration strategy. An effective way to demonstrate this is by GAP Matrix using strategic components, customer, and market components, & value and creation components. This matrix helps to rethink the whole business model while factoring in the newly added value from the self-configuration adaptation strategy. Figure 4.6 illustrates the GAP matrix of a focal company that uses the self-configuration strategy.

Strategic Components	Strategic Model	Resource Model	Network Model
	<p><b>1.Where focal company (FC) compete?</b> Production systems development and deployment.</p> <p><b>2.How the FC compete?</b> Integration and development of tools and technologies to make production systems as per customer requirements.</p>	<p><b>1.The core assets and competencies of the FC?</b> Integration and customised designing.</p> <p><b>2.The complementary assets and competences of the FC?</b> Setting the developed system to be in-line with shop-floor. Providing test services. Repairing and adding functionalities.</p>	<p><b>1.The structure of and relationship between supplier/partner? (need of vertical/horizontal integration)</b> Direct relationship with supplier and customers for system development.</p> <p><b>2.The relation with other stakeholders? (new/old, government, law)</b> Designs have legal binding under contracts.</p>
Customer and Market Components	Customer Model	Market Offer Model	Revenue Model
	<p><b>1.The current target group of the FC?</b> Automotive, Government, Nuclear, Aerospace etc.</p> <p><b>2.The way FC reaches the customers? (interfacing economic activity)</b> Sales, networking through events, word of mouth, website.</p>	<p><b>1.The relation between the FC and the competitors? (competitor analysis)</b> No direct interacting relation.</p> <p><b>2.The FC offering? (product + service)</b> Smarter production system capable of making decisions about themselves without assistance.</p>	<p><b>1.The FC's revenue streams in terms of number, type, relevance?</b> The revenue stream is reliant on the number of production units sold. The innovation can bring added monetary stream in form of giving a service that makes existing production systems smarter.</p> <p><b>2.Pricing method?</b> Possible incentive for added functionality in production system. Subscription based for optimization and Process Identification.</p>
Value and Creation Components	Manufacturing Model	Procurement Model	Financial Model
	<p><b>1.The current FC's manufacturing strategy? (strategy and competitive priorities)</b> Make specialised production systems and provide configuration change strategy as service to the customers on a subscription basis. They also need to pay for additional infrastructure deployment that enables self-configuration.</p> <p><b>2.The current FC's manufacturing operations?</b> Procurement, Designing, Fabrication, Assembly, Testing, Dispatch</p>	<p><b>1.The FC's choices in term of make or buy?</b> Buy some components that require high level production facility. Make components that can be made in-house as much as possible.</p> <p><b>2.The FC's information management (key resources, HR, logistics)?</b> Mostly done through meetings, management software, project managers and databases (purpose specific).</p>	<p><b>1.Capital acquisition (D+E) and balance?</b> Customer pays in percentage during whole project. Most cost is at the start of the project for design and initial system development. The next phases are executed as the review processes are passed and customer fulfils monetary obligations. Configuration change strategy needs an initial deployment cost and then subscription cost for operation.</p> <p><b>2.Cost structure (direct/indirect, fixed/variable, emerging and sunk cost in transactions)</b> Direct, fixed cost mostly for the production system. Variable pricing model for configuration change management strategy depending on need, and technical requirements</p>

Figure 4.6: Conceptual business model indicating in detail the strategic components, customer and market components, & value and creation components for self-configuring production systems.

### 4.5.5 Status for Innovation

The realisation of a self-configuring adaptation strategy enables a shift in existing business models and all of its associated sub-models. In figure 4.7 the initial status of the business is mapped without the offer of self-configuration capability for the focal company. The desired status is the envisioned target that the focal company can aim for as motivation for adopting the self-configuration strategy.

### 4.5.6 Potential Value Utilisation

Figure 4.8 simplifies the value capture to the respective domains and highlights the basic essentials that focal company must contribute to at a very basic level. This also presents an opportunity for market and value capture

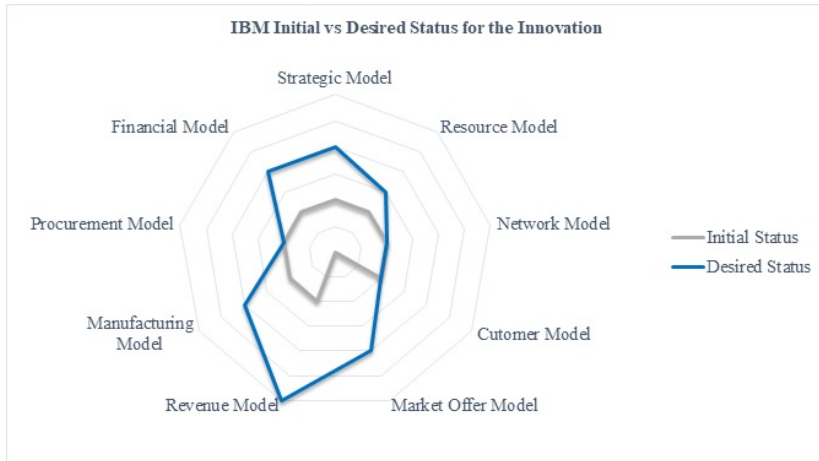


Figure 4.7: The status for innovation; A reflection on the initial and desired business model through the introduction of self-configuration in production systems.

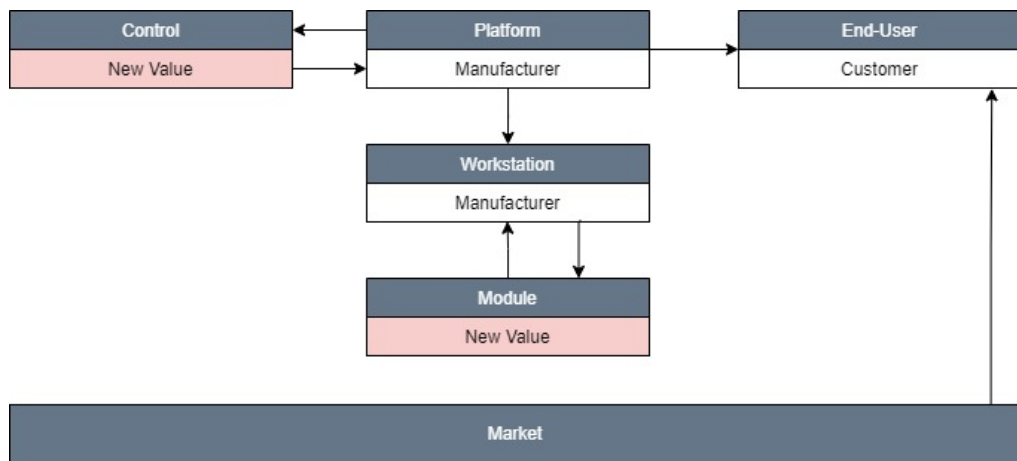


Figure 4.8: An illustration for potential value utilisation/realisation by self-configuration strategy in production systems. New value generated by the solution can serve as direction for business model.

by offering the module and control for self-configuring capability in existing production systems.

## 4.6 Conclusion

A survey to capture industrial insight on self-configuration was carried out. A potential business model taking advantage of the introduction of self-configuration capability in existing systems was presented. The discussion

on the surveys affirms the research gap and validates it through industrial insight. These findings are:

1. Industrial professionals acknowledge the need to address challenges associated with data security, cost of smart production systems, variation in demand and result adaptation required.
2. Most industrial practitioners feel that self-configuring production systems will solve the problem related to the skill shortage of field experts and skills required for machine setting.
3. The surveys highlighted a lack of machine-to-machine communication, lack of automated decision-making, lack of incorporating intelligence and lack of updates in manufacturing infrastructure as main challenges towards the adoption of intelligent production systems.
4. Interestingly, from the surveys it is observed that companies and institutes are moving in the direction of integrating more data-driven decision-making.
5. Machine setting is still mainly a manual task, and dependent on product and process requirements.

These findings clearly highlight a need for self-configuring production systems, presenting new opportunities (i.e. new value) for system integrators, researchers and associated companies. To understand the self-configuration in production systems, the next chapter evaluates the current status of the systems on the shop floor in terms of identified features of self-configuration. Chapter 6 maps the configuration of manufacturing assets that constitute the production systems, and Chapter 8 and Chapter 9 attempt to realise self-configuration in systems to gain the new perceived value.

---

## Chapter 5

# A Level-Based Classification of Technical Enablers in Production Systems

## Contents

---

5.1	Introduction . . . . .	<b>113</b>
5.2	A Level-Based Classification for Self-Configuration . . . . .	<b>114</b>
5.2.1	Methodology . . . . .	116
5.2.2	Classification Requirements . . . . .	117
5.2.3	Features of Self-Configuration . . . . .	118
5.2.4	Stages of Self-Configuration . . . . .	120
5.3	Evaluating Enablers of Self-Configuration with the Level- Based Classification . . . . .	<b>125</b>
5.3.1	Evaluation of Enablers of Self-configuration . . . . .	126
5.3.2	Industrial Application of Level-Based Classifi- cation of Production Systems . . . . .	132
5.3.3	Challenges to the Application of Self-Configuration . . . . .	133
5.3.4	Limitations of the Level-Based Classification Method . . . . .	136
5.4	Conclusion . . . . .	<b>137</b>

---

## 5.1 Introduction

A level-based classification system was developed that breaks down the concept of self-configuration to its features that must be individually realised. The novelty in this research is identifying and relating these features into an integrated classification, providing a methodological approach to imple-

ment self-configuration in practical applications (i.e. production systems). The possible enablers, challenges and future direction is also discussed.

The chapter is divided into four sections, with the establishment of the concept and details of the proposed classification system in the first two sections and insight into self-configuration in production systems along with future trends in the next three. Section 1 deals with the introduction and the context of the chapter. Section 2 presents the novel level-based classification for production systems to achieve self-configuration, and proposes stage-wise transitions to navigate up the levels in practice. Section 3 examines enablers and application of self-configuration in manufacturing. The key approaches currently in use are discussed, while major challenges to self-configuration implementation are identified. Finally, Section 4 concludes the chapter.

## 5.2 A Level-Based Classification for Self-Configuration

To implement self-configuration in a dynamic setting, it is necessary to identify the “when” and “how” conditions of the event that causes configuration change in order to minimise perturbation and maximise outcomes. A learning mechanism can aid in robust self-organisation by service-orchestration, making the system more effective, efficient, adaptive, responsive and advantageous (Leitão et al. (2013); Ribeiro et al. (2008); De Souza et al. (2008)). One of the approaches to identifying “how” and “when” self-configuration should occur was studied by (Rodrigues et al. (2015)) where templates were used from a top-down perspective, and artificial intelligence from a bottom-up perspective to change the structural adaptation of the



system. A system for self-configuration should be capable of adapting, and be constantly looking for new improved configurations during operation. This expands the system functionality for adaptation at the operation level (EIMaraghy and Urbanic (2004)).

Studies dealing with the incorporation of self-configuration (self-adaptation) into manufacturing in terms of key enablers, approaches and challenges, from the literature present a clear idea that self-configuration is composed of multiple “**features**” acting together. This observation is used to develop a level-based classification for production systems, proposed in this thesis, based on criteria for self-configuration realisation. The classification levels are captured in figure 5.1, and described in more detail in the following sections. Each feature can be detailed by identifying the relevant stage and level where it lies for the production system. This, in a sense, presents a road-map on how to proceed with self-configuration for the respective production system.

The levels, features, and stages are distinguished into two degrees of adoption for self-configuration in manufacturing systems: System Readiness and System Execution.

**System Readiness:** The levels that a system has to fulfil to make it capable of self-configuration.

**System Execution:** The levels that a system has to fulfil to accomplish the effect of self-configuration and improve on the effect.

## 5.2. A LEVEL-BASED CLASSIFICATION FOR SELF-CONFIGURATION

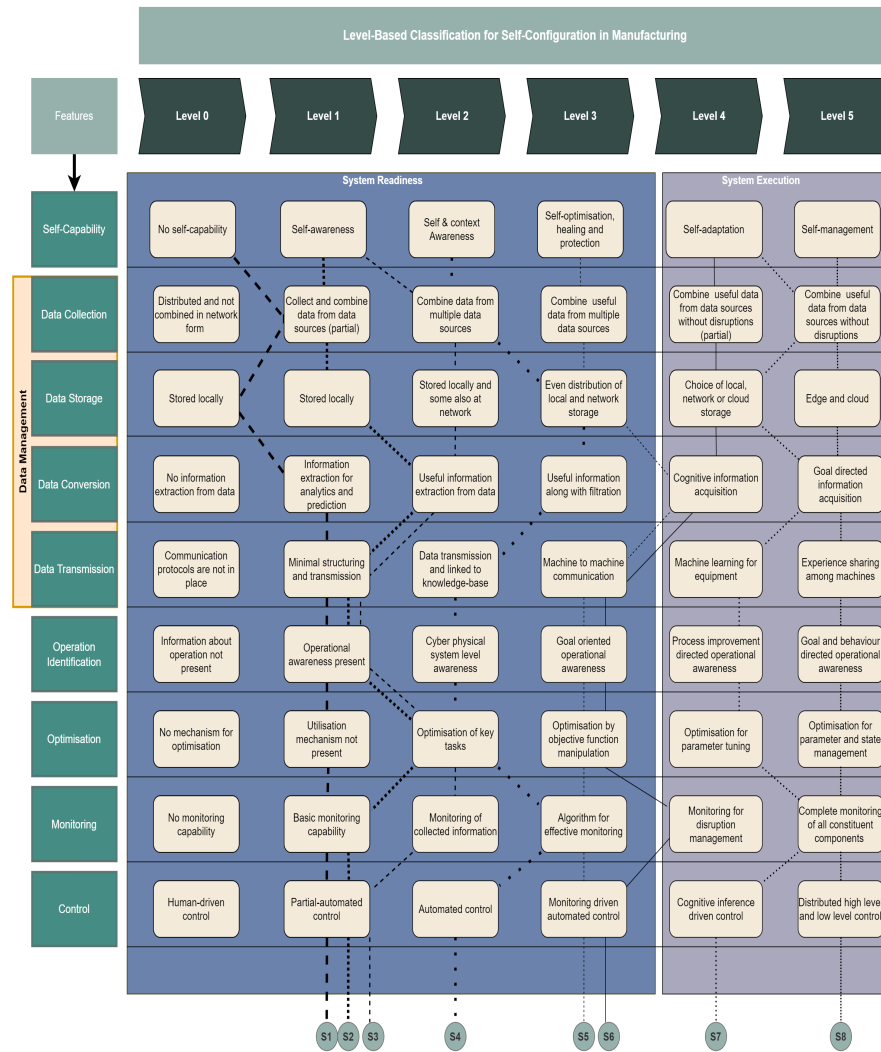


Figure 5.1: Level Based Classification for Self-Configuration. The degree of the capability in production system is outlined with dividing the levels into two aspects of System Readiness and System Execution. The System Readiness is from Level-0 to Level-3 and System Execution is from Level-4 to Level-5. The Stage-Wise Transition (S1-S8) presents the transformation of production system for incorporating self-configuration across levels.

### 5.2.1 Methodology

A four-tiered developmental process highlighted in (Baetge (1974)) is used as the method for creation of the level-based classification. This is carried out to promote a systematic classification development for the self-configuration target. Since, the classification is of a practical nature, as an initial step the production system it is to be applied on, must be defined

and delimited. Secondly, the elements of the classification must be instated. Third, the relationship between the elements should be established in the wider scope of application. Finally, the evaluation and validation must be carried out and examined.

### 5.2.2 Classification Requirements

There exists two categories of requirements that the classification must adhere to. The first one is the *General Requirements* aimed at modelling the classification. This encompasses application direction and description, adaptability relative to degree of realisation, utilisation for application, low effort to apply and infer findings by the application of classification (Neuhausen (2001)).

The other requirement is towards complimenting the classification for self-configuring applications, therefore limiting the scope of classification. This requires that the production system with self-configuration must be understood. In case of *Specific Requirements* first, the classification should have a general validity. It means it must be possible to generalise the classification over a wide array of production systems. Secondly, it must be possible to analyse production system at holistic, superordinate and qualitative level using the classification. Third, the classification must be described with the aid of its features and the relevant levels, making it independent of technological implementation. Fourth, the stages of the classification must be able to be differentiated based on their properties.

### 5.2.3 Features of Self-Configuration

The features of self-configuration are deduced based on analysis carried out for self-configuring and self-organising production systems. The features that enable self-configuration are self-capability, data management (collection, combination, storage, conversion, and transmission), operation identification, optimisation, monitoring, and control (Antzoulatos (2017); Sanchez et al. (2020); Leitão and Restivo (2003)). A brief description of these features is as follows:

- **Self-capability** encompasses the ability of a system to realise its state and potential, be aware of its function and environment, and set up necessary mechanisms for self-management.
- **Data management** encapsulates the capability for data collection, storage, conversion, and transmission dealing with the complete data life-cycle in production system
  - **Data Collection:** The system must be capable of collecting and combining data streams from multiple data sources. The cognitive/learning inferences should enable the system to extract useful information from the data collected, along with mitigating noise in the accumulated data.
  - **Data Storage:** Storage should be dynamic with the system being capable of determining data location, and the value of data as per latency and performance requirements for either processing at edge or in the cloud.
  - **Data Conversion:** The system must be capable of filtering useful information for cognitive inferences and accumulating information for goal-directed behaviour.

- **Data Transmission:** Protocols and standards must promote interoperability between machines/systems and must be adaptable. Experience across the system could be leveraged for self-configuration to make the system more adaptable to changes.
- **Operational identification** for self-configuration is moving towards the objective like maximum efficiency, the best performance and lowest cost, and ensuring correct behaviour execution as per that objective. To manipulate the system for this, the process improvement should focus on operational awareness and execution directed to goals and desired behaviour.
- **Optimisation** for self-configuration is directed by the objective function (i.e. function that defines the criteria for optimisation for the system). A system that realises self-configuration embeds tuning of system parameters and state-transition management in order to optimise the objective function.
- **Monitoring** of the constituent components that make up the system and overcome disruptions is essential for self-configuring capability. The monitoring must be connected to automated and cognitive inference driven control.
- **Control** through utilising insight generated by the system to direct self-configuration behaviour. Event triggers can be used to offer high and low-level control in distributed manufacturing systems. Self-configuration uses effective monitoring to offer control capability to the target system.

## Levels of Self-Configuration

The levels (see figure 5.1) in the classification vary from Level-0 to Level-5 representing a scale of self-configuration capability, with Level-5 signifying complete self-configuration. The features of self-configuration under discussion are split into multiple levels of capability, and maturity for each feature is achieved by transitioning from Level-0 to Level-5. This transition through levels is done in stages to make it practical to be deployed in industrial applications. This is achieved through stage-wise transition from one stage to the next. This kind of approach allows the completion of the third and fourth requirements of classification development.

### 5.2.4 Stages of Self-Configuration

Self-configuration is achieved via a stage-wise transition, in which each stage represents a combination of features at certain levels of maturity, as illustrated in the classification diagram (fig 5.1). Using existing models of Bauer et al. (2019) for autonomous production systems, the stages are adapted to define the stages of self-configuring production systems.

Determination of where a feature of the system lies within the classification is carried out by the identification of the maturity of the feature along the levels. A stage is associated with a feature based on the maturity along the levels where most of the characteristics of the features in that application lie.

There are eight (8) proposed stages for the transition from no self-configuration ability to fully implemented self-configuration. Stages 1 to Stage 4 fall within the System Readiness degree, while Stage 5 and Stage 6 are transitional stages that make the system ready for the System Execution degree.

Stage 7 and Stage 8 are responsible for producing and maintaining the effect of self-configuration in the System Execution degree. This stage-wise transition assists in associating the current self-configuration capability of a system with a direction of progression to higher levels of self-configuration.

Stages of level transition for self-configuration can be related to cost and complexity. The stages, increasingly, add to the complexity of the system and also the associated cost. A trade-off exists, depending on the system's requirements, on the level that must be reached by the system in terms of capability, associated cost, and complexity. These stages can be stated as:

- **Stage-1 Information Stage:** The system gains the partial capability for accumulating and combining data from data sources along with processing the data for analytics. Transmission capability is limited to accumulated data transfer only. Operational awareness is present, but no optimisation capability is available. Basic monitoring of the system state is carried out with partial human-driven control. Example: A production system that can gather data for limited processing, but cannot act on the accumulated data, i.e. make decisions itself. It can record information in a storage medium, that can be extracted through a human operator. Context awareness is present, meaning that it can record and map information during an operation.
- **Stage-2 Extraction Stage:** The system becomes capable of combining data from sources with data stored locally. The system has more capability to extract useful portions of data along with optimisation of key tasks.

Example: A production system that can gather all relevant data for processing, and can combine data for analysis. The storage in this production system is still done locally. Optimisation (e.g. time,

value, or cost) conditions can be fed to the production system and some settings can be made to reach the optimisation objective.

- **Stage-3 Awareness Stage:** The system develops a sense of awareness about its condition. Data capability is increased with the system capable of combining data from multiple sources, and storage is possible locally and on a network. Monitoring capability becomes enhanced with more focused monitoring of information.

Example: The production system has the ability to collect data for processing, store that data locally in a storage medium, and transfer it over a network onto a server. Each operation in the system can be monitored, along with changes in state identified during operations.

- **Stage-4 Context Stage:** The system develops an awareness of its environment (i.e., becomes a cyber-physical system). The system effectively filters useful information from the gathered data. Data transmission capability is increased with more sharing of information across devices and linked with a common knowledge base. Automated control is present, guided by effective monitoring algorithms.

Example: In addition to the example of Stage 3, useful information like time or a value can be extracted from the production system. It also has context awareness, i.e. relates the change in state to an outcome (i.e. programmed) like a pass/fail decision based on differential pressure feedback from the machine.

- **Stage-5 Self Stage:** The system becomes ready for the initial stages of self-configuration. The capability of system for self-optimisation, healing and protecting are introduced. Information useful for decision-making is gathered. The system develops data transmission capability across machine platforms. Objective functions are used for op-



timisation as per specified goals, with the system monitoring and driving the control strategy.

Example: The production system has a protective mechanism built-in that observes and identifies undesired behaviours and assists the system to act as per objectives. The user can provide an objective function, accounting for cost, time or any other, that the production system can utilise during operation.

- **Stage-6 Adaptation Stage:** The system is able to adapt to changes without assistance. Data collection could be carried out without disruptions and self adjusts to eliminate disrupting factors. The system could store data in a network or cloud storage. A disruption management mechanism is present.

Example: The production system becomes more reactive, i.e. if there is a disturbance/disruption then the system takes steps to overcome it (i.e. by means of control logic or otherwise). Also, the system is capable of adapting the production system to meet the requirements set (product, process, or KPIs).

- **Stage-7 Partial Self-Management Stage:** The system is capable of combining data without disruption, and data is collected depending on the goals. Machine learning could be incorporated for smart decision-making with data transmission to control machine configuration. Process improvement is the main focus at this stage, with optimisation directed towards parameter tuning. The system is capable of gathering and monitoring information about all its constituent components. Based on these cognitive inferences, control of the system can be driven.

Example: The production system becomes smarter with links to ma-

chine learning pipelines, that assist in decision-making to reach objectives. The system is capable of gathering useful data that contributes to the goal. Parameter tuning is supported by machine learning pipelines. Automated control is mainly relied on.

- **Stage-8 Self-Management Stage:** The system gains complete control over its configuration, as it becomes completely self-managed. The system can process portions of data at the Edge (i.e. near the data source), while other portions could be dispatched to the cloud for analytics. Experience sharing can be enabled in the system, making the configuration more effective with subsequent iterations. Event-based triggers are available that provide enhanced control functionality.

Example: The production system has the capability to learn over time in regard to objectives. High-computational resources may be linked to the system for making smarter decisions. The production system operates on an event-trigger mechanism and gathers relevant data to support the event to optimise as per the objective.

These stages show how progression towards higher levels of self-configuration emerges from progression in maturity levels of the contributing features. This level-based system can be used to understand a system's readiness for self-configuration capability and depending on the requirements achieve the balance of functionality with cost and complexity. The stage-wise transition is necessary to make sure that self-configuration can be realised in earnest by evaluating the current state of production system of self-configuration readiness. Only once the system reaches Stage-5 it can realise initial effects of self-configuration. Progressing from Stage-5 also leads to progression on self-configuration realisation.

## **5.3 Evaluating Enablers of Self-Configuration with the Level-Based Classification**

The level-based classification of self-configuration in the production system can be used in two ways. Firstly, it can be used to evaluate a manufacturing system, to determine the stage of self-configuration it is capable of by breaking down the domain into features and evaluating the maturity level of each of those features.

Alternatively, the classification can be used to evaluate possible contributing technologies. Technologies applied to a manufacturing system can help increase the maturity level of specific features. By understanding which features and levels they contribute to, technologies can be identified to fill in gaps a system may have to reach higher stages of self-configuration capability.

This section discusses four potential enablers of self-configuration and evaluates them with the level-based classification. This serves as a study into the applicability of these technologies to self-configuration, and also a demonstration and validation of the classification.

The features of self-configuration discussed in the model presented in Section 3 can be influenced by the technology or technologies utilised. The technologies that enable the realisation of self-configuration can highlight the features targeted and areas that may be lacking. In this section, possible enablers of self-configuration are examined with their respective levels and stages. Challenges are presented that demonstrate the aspects of features that must be improved with each technology for complete self-configuring production systems.

### 5.3.1 Evaluation of Enablers of Self-configuration

Self-configuration focuses on the operational aspect of the production system, acting on extracted information to make decisions and take actions. This level enables machine to machine communication, experience sharing, optimisation and monitoring along with dealing with parameter tuning and design improvement. To obtain this, it is necessary to utilise those technologies that achieve such or similar behaviour.

In manufacturing intelligence literature, many technologies are proposed to aid in fulfilment of self-configuration objective, four are examined as they provide a good spread of options for implementation (Table: 5.1). Some of their details and approaches to achieve self-configuration or similar effect are described in the following subsections. The following sections detail a brief description of each technology followed by the evaluation.

#### **Multi-Agent Systems**

Self-configuration can be accomplished by agent-based control and communication in a production system environment. “Agent” in the context of a multi-agent system refers to an intelligent software agent which monitors and controls the production equipment. A combination of passive, active and cognitive agents make up the multi-agent system environment, with increasing levels of intelligence, autonomy, and capability. Multi-agent systems use multiple individually simple agents to collectively achieve complex control paradigms for self-configuration, and show complex behaviour emerging from simplistic individual processes.

Bachula et al. (Bachula and Zajac (2013)) have described a three-stage framework for agent-based control in a discrete manufacturing system envi-

Table 5.1: Enablers of Self-Configuration

Enablers	Description
Multi-Agent Systems	Consists of independent software agents responsible for a resource or portion of a resource. Agents co-ordinate among themselves to form emergent behaviours from simple individual behaviours to form a distributed control paradigm.
Grid Computing	Consists of multiple computers, often distributed geographically, networked together to accomplish joint tasks. Requires systems to maintain sense of state. Grid computing has the capability to combine distributed resources and co-ordinate among them for stateful execution to target goal.
Control Theory	Uses system controllers for controlling behaviour in a designed system. Consists of target system and controller. The controller implements a control strategy and produces signals that adapt target systems as per the desired goal.
Component Based Development	An architecture based on system functionality components and connectors that govern interaction between such components. This prompts structural self-awareness in the system, important for self-configuration.

ronment. The technology employed in their research focuses on cooperative agents, resulting in a plug and play control system. This plug and play control gives the capability to the distributed production system of being able to configure itself based on agent behaviour. Multi-agent systems have been applied in decentralised manufacturing environments where their capability for intelligent decision-making, agent-to-agent communication and cooperation are relied on for production control (Leitão et al. (2016)). These capabilities make a firm case for their application to self-configuration and self-organisation of such systems.

### **Grid Computing**

The autonomic principles of grid computing address the ability of grid systems to adapt to changes for maintaining system performance and recovering from any perturbation (Messig and Goscinski (2005)). Grid systems maintain a sense of state that helps them to identify and manipulate themselves as per desired criteria. Works on grid computing have used configuration broker for web services to maintain reliable web service and stateful resource access (Messig and Goscinski (2005)). Other works use similar approaches in telecommunication (Yan et al. (2007)) and wireless applications (Ahuja and Myers (2006)). However, grid computing has seen little acceptance in manufacturing.

### **Control Theory**

Control theory drives the production system by using the manufacturing system controllers. The central production system controller implements a control strategy through manufacturing controllers. This strategy is based on the desired goal provided for the production system. Control driven approaches have been extensively applied to introduce self-\* behaviours in manufacturing. (Priego et al. (2014)) presents a supervisor architecture for enabling system availability in spite of failure, therefore maintaining the system state at the control level. (Council and others (1998)) identified self configuration as a key challenge for automation systems. The work done by (Priego et al. (2014)) looked to propose a configuration mechanism coupled with a model driven development technique to ensure control is maintained in the event of a system failure. This was accomplished by restoring functionality of a controller to a backup when failure occurs. The research argued that the recovery of the functionality is a function of the

execution point in which the system has failed.

With the rigidity of the 61131-3 standard (Elements (2018)), PLC controller configuration could be achieved only when the system is not running (although hot-swapping of portions of code may still be possible while within the same CPU cycle without stopping the manufacturing process). The swapping approach is viable but causes significant downtime (Zoitl (2009)). For this to be possible the system control needs to stop running, the new complete program needs to be loaded, and finally, execution resumed. A functional block approach that uses the IEC61499 standard for PLC control due to its distributed architecture is most useful to achieve self-configuration behaviour.

Other research works targeting dynamic reconfiguration like that described in (Botygin and Tartakovsky (2014)) used a dynamic ranking table for optimising control system workload. Similarly, (Binotto et al. (2013)) used the order arrival time and controller workload to set configuration balance across controllers. Energy optimisation (Guo et al. (2009)), fault control (Merz et al. (2012)) and network failure control (Streit et al. (2014)) all have served as drivers for self-configuration based on objective.

### **Component Based Development**

Component-based development deals with the encapsulation of reusable functionality and self-sufficiency for self-configured production systems at the framework levels. Autonomic components provide the ability to export profiles that contain information about functional, operational, and control aspects. This information includes knowledge about their behaviour, adaptability to other systems, resource prerequisites, performance and interaction (Chirn and McFarlane (2000)). In this methodology, aspects of

composition formalism, smart components, and hot swapping may serve as enablers for self-configuration. Composition formalism can help in generating configurations for the equipment under constraints (Cheng et al. (2004a)). Smart components can adapt to changes serving as building blocks for production systems (Jann et al. (2003)) and hot swapping (Appavoo et al. (2003)) can help in self-configuration by code replacement, code swapping or module change.

### **Level-Based Classification of Enablers of Self-configuration**

The level-based classification is applied to the aforementioned technological enablers. Figure 5.1 can be seen as a framework for incorporating level-based classification on a production system. The technological aspects that are collectively responsible for self-configuration can be divided by the capability of each constituent feature. Each feature can then be detailed by identifying the relevant stage and level where it lies. This, in a sense, presents a road-map on how to proceed, i.e., what stage-wise transition to be taken for increased self-configuration capability.

Figure 5.2 also clearly illustrates that the enablers alone are currently insufficient to realise higher stages and levels of self-configuration. They can only be achieved when these enablers are linked with different technologies. In this way, these enablers act as a bridge between different technologies to achieve the purpose of self-configuration.

By understanding the target stage of self-configuration (i.e. which will be a trade-off between functionality and the cost/complexity of implementation), the use of the classification enables the identification of the most relevant technologies, but also potential gaps. For example, the enabler multi-agent systems have most of the features at Level 4, and this cor-



5.3. EVALUATING ENABLERS OF SELF-CONFIGURATION WITH THE LEVEL-BASED CLASSIFICATION

	Multi-Agent Systems	Grid Computing	Control Theory	Component Based Development
Self-Capability	Self-adaptation S6 L4	Self & Context Awareness S3 L2	Self-optimisation, healing and protection S5 L3	Self & Context Awareness S3 L2
Data Collection	combine useful data from data sources without disruptions (partial) S6 L4	combine data from multiple data sources S4 L2	combine useful data from multiple data sources S5 L3	combine data from multiple data sources S4 L2
Data Storage	Choice of local, network or cloud storage S6 L4	Choice of local, network or cloud storage S6 L4	Stored locally and some also at network S3 L2	Stored locally and some also at network S3 L2
Data Conversion	Useful information along with filtration S4 L3	Cognitive information acquisition S5 L4	Useful information extraction from data S3 L2	Useful information extraction from data S3 L2
Data Transmission	Data transmission and linked to knowledge-base S4 L2	Machine Learning for Equipment S7 L4	Machine to machine communication S5 L3	Minimal structuring and transmission S2 L1
Operation Identification	Goal oriented operational awareness S6 L3	Cyber Physical System level awareness S4 L2	Cyber Physical System level awareness S4 L2	Cyber Physical System level awareness S4 L2
Optimisation	Optimisation by objective function manipulation S6 L3	Optimisation of key tasks S4 L2	Optimisation of key tasks S4 L2	Optimisation by objective function manipulation S6 L3
Monitoring	Monitoring for disruption management S6 L4	Algorithm for effective monitoring S5 L3	Monitoring of collected information S3 L2	Monitoring of collected information S3 L2
Control	Monitoring driven Automated Control S6 L3	Cognitive Inference driven control S7 L4	Automated Control S4 L2	Partial-Automated Control S3 L1

Figure 5.2: Level Based Classification for Self-Configuration applied to Enablers along with Stage-Wise transition Identification. Note that there is only one occurrence of Stage 7 in the diagram and none of Level-5.

responds generally to self-configuration stage 6. However, to achieve full stage 6 self-configuration, limitations in the data transmission and data conversion features would need to be overcome, requiring the integration of additional technologies machine-to-machine communication standards for example.

### 5.3.2 Industrial Application of Level-Based Classification of Production Systems

Level-based classification can be applied to industrial production systems to evaluate their fitness for self-configuration as per their constituent features. The capability of features in respective industrial applications is identified along with its stage and the level where it is present. This assists in setting up a target level for features and provides a direction to achieve the desired self-configuration capability. To demonstrate the applicability of the novel level-based classification, three related industrial production systems are considered: basic differential pressure Leak Tester, Multi-Application Leak Tester (MALT) and then MALT integrated with a multi-agent system (JADE). Figure 5.3 shows these systems for demonstrating the application of level-based classification.

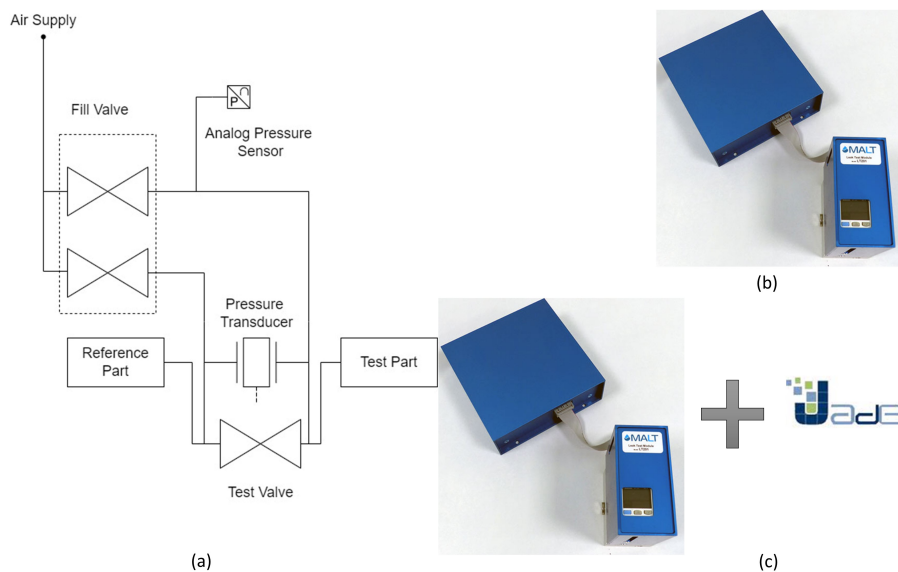


Figure 5.3: Industrial production systems for demonstrating the applicability of level-based classification: (a) Basic differential pressure Leak Tester; (b) Multi-Application Leak Tester (MALT); (c) Multi-Agent integrated with MALT System

Differential pressure decay leak testing is used to sensitively detect leaks in production parts and is one of the most commonly applied industrial leak

test methods. An absolute sensor is used to relate the difference between test pressure and atmospheric pressure, and a pressure transducer is used to measure leaks during the testing process. Functionality involves filling the test part and reference part with air, stabilising and measuring the relative change in pressure between the two parts. Applying the level-based classification on this basic industrial production system yields most of its features at Level-0 or Level-1. The production system is at the very basic degree of System Readiness for self-configuration. Details about the features, their respective levels and stages for this system can be seen in figure 5.4.

The application of level-based classification on Multi-Application Leak Tester (MALT) and Multi-Agent integrated with MALT System is detailed in Chapter 9.

### 5.3.3 Challenges to the Application of Self-Configuration

The technologies proposed for enabling self-configuration in manufacturing present opportunities and challenges in their applications. Figure 5.2 evaluates the technological approaches for self-configuration in manufacturing system by relating their features with the developed classification. This presents a clear indication of the level and stage where each of the technologies lie in terms of contribution to the features of self-configuration. Based on this analysis, the challenges for self-configuration are identified.

**Control Challenge:** There exists a need for effective control that not only provides functionality to operations, but also the measurement for occurred failures. The self-configuration process could involve modification to

5.3. EVALUATING ENABLERS OF SELF-CONFIGURATION WITH THE LEVEL-BASED CLASSIFICATION

		Basic Differential Pressure Leak Tester	
Data Management	Self-Capability	No self-capability S1   L0	Basic functionality is present. The system does not have the ability to realise its state, and potential for self-mangement
	Data Collection	Collect and combine data from data sources (partial) S1   L1	The system is able to collect values from test and reference volumes to determine leaks during the test phase.
	Data Storage	Stored locally S2   L1	The data values are stored locally in individual system components like differential pressure transducer, pressure gauge
	Data Conversion	No information extraction from data S-   L0	The data obtained by differential pressure transducer and pressure gauge is given directly to the operator without extracting information of value.
	Data Transmission	Minimal structuring and transmission S1   L1	The data available follows the basic structuring/transmission. The data can only be displayed on the measuring components and is not able transmitted.
	Operation Identification	Information about operation not present S-   L0	The leak tester only displays sensor and transducer values. It does not contain any information about operation being carried out.
	Optimisation	No mechanism for optimisation S-   L0	No ability is present in the system to tweak its parameters and states as per any desired objective.
	Monitoring	Basic monitoring capability S1   L1	The system lacks basic monitoring at the level of the individual components of the leak tester. No capability to overcome disruptions is present.
	Control	Human-driven control S-   L1	Human controls the functionality of the system.

Figure 5.4: Level Based Classification for Self-Configuration applied to basic differential pressure Leak Tester

one or more parameters, a sequence change, variation in information flow, communication paths and channels, or a modification to a physical component's functionality. For this, a dynamic control mechanism is required along with robust monitoring. This allows for variations in one component without influencing other components as a change is detected. This maintains the stability of the system (Zoitl (2009)). One assumed property of self-configuration is that the self-configuration process does not disturb the control process, but is aided by it. Moreover, preconditions need to be executed for actions to take place. Therefore, in control applications,

real-time constraints must be met for safe execution. However, the implementation of such an enabling architecture that supports such stochastic environments is a major challenge (Brennan et al. (2002)).

**Knowledge and Framework Challenge:** For self-configuration, there is a lack of knowledge about the technical infrastructure required for such a capability to be present in the equipment. This comes in addition to the general lack of information on how to implement self-configuration in production systems, the constituent feature components, and the transitional approach to reach complete self-configuration. The level-based classification introduced in this chapter contributes to this aspect, supplemented by the following chapters that add to this.

**Trust and Interoperability Challenge:** For equipment to transfer and co-relate data between different data streams, free movement of data is required. Increasing concerns over data sensitivity, intellectual property, data ownership and management, require a strong data governance policy that could be linked with the production systems (Raptis et al. (2019)). However, due to the lack of interoperability between protocols, systems, products, and devices such policies rarely exist. Due to vendor specific standards of many system components, interoperability requirements fail to match the desired infrastructure requirements for self-configuration. The capability to handle data growth and the security of acquired data act as a challenge for data management and realisation of self-\* abilities (Raptis et al. (2019)).

### 5.3.4 Limitations of the Level-Based Classification

#### Method

The level-based classification system offers a valuable tool for evaluating a manufacturing system's potential for self-configuration. However, some limitations exist, particularly the lack of extensive testing across various cases.

- **Limited Validation:** The system has not been thoroughly tested and validated across a wide range of manufacturing systems with diverse functionalities and complexities. The inability to fully evaluate a system's capability to achieve self-configuration through its features is limited by this constraint. This research contributes to this by presenting a framework that enables to understand the building blocks of configuration in production system so that the concepts discussed in this chapter may be carried over to other manufacturing applications.
- **Subjectivity in Feature Evaluation:** Assigning maturity levels to a system's features can involve some subjective judgment. There might be variations in how different users evaluate the same features.
- **Focus on Enablers:** The classification system focuses on identifying technologies that can contribute to self-configuration (i.e. enablers) rather than providing a detailed roadmap for implementation. Additional steps would be required to translate the classification results into practical actions for achieving self-configuration in a specific system. This research contributes towards providing this roadmap by presenting an Adaptation Strategy.

Reasons why the level-based classification system was not subjected to

testing across various cases:

- **Complexity of Manufacturing Systems:** Manufacturing systems vary in size, automation, and complexity, requiring testing across a wide range of facilities and equipment. This would be a significant undertaking beyond the scope of project.

### Future Developments

Despite the limitations, the level-based classification system offers a foundation for further development. Some future developments by which it could be improved:

- **Standardisation of Feature Evaluation:** Develop more objective criteria for assigning maturity levels to a system's features, reducing the potential for subjectivity.
- **Case Study Development:** Apply the classification system to a wider range of real-world manufacturing scenarios, documenting the results and using them to refine the classification.
- **Integration with Implementation Tools:** Combine the classification system with practical tools or guidelines to help manufacturers translate the classification results into concrete steps for achieving self-configuration. This research works on it.

## 5.4 Conclusion

This chapter presents a novel model for evaluating production systems and enabling technologies with respect to their maturity or application to

self-configuration. This enables users to understand where technologies could be used, where gaps may exist in their system, and where developments should be prioritised. The concepts of “self” and “configuration” are defined and coupled with the idea of “self-\*” in manufacturing. This is used to establish the definition of self-configuration in manufacturing. A level-based system for classifying self-configuration capability in production systems is presented, with stage-wise transition to higher degrees of self-configuration. The stage-wise transition is related to the balance between functionality versus the cost and complexity of implementation in production environment. An industrial use-case of a test process based on manufacturing system (MALT) is demonstrated on a developed framework to achieve such self-configuration by mapping to the proposed level-based classification features.



---

## Chapter 6

# Module Driven Configuration

## Contents

---

6.1	Introduction . . . . .	140
6.2	Overview of Framework . . . . .	141
6.2.1	Need for a Framework . . . . .	141
6.2.2	Design Principles . . . . .	142
6.2.3	Framework Overview and Key Concepts . . . . .	146
6.3	Module Driven Configuration System Model . . . . .	151
6.3.1	Framework Architecture and Abstraction Levels . . . . .	151
6.3.2	Overview of the Architecture . . . . .	154
6.3.3	Configurable Objects . . . . .	159
6.3.4	Modules . . . . .	162
6.3.5	Module Based Production System . . . . .	169
6.3.6	Module Based System Application . . . . .	175
6.4	Conclusion . . . . .	182

---

## 6.1 Introduction

This research introduces an integrated framework to achieve self-configuration. This direction aims to achieve some modularity and separation between production system functionality and self-adaptive behaviour. In this research, the framework forms the building blocks towards understanding the self-configuration concept. Therefore, the proposed framework maintains a strict separation of concern between components of the production system

and the behaviour required to achieve an objective (i.e., self-configuration). This is pivotal in understanding the system's self-adaptive behaviour separate from its functionalities. This also makes the production system more maintainable. This framework will undergo testing and validation in subsequent chapters.

The conceptual model is defined and the underlying principles for the proposed generalised framework are elaborated. Next, the architectural elements of the framework are introduced that can be used to construct a production system model and then to achieve self-configuration through their application.

## **6.2 Overview of Framework**

This section elaborates on the proposed framework. The need for the framework is first introduced followed by the design principles behind the framework. Next, the conceptual model is developed from the proposed framework and related to a generalised model for a self-adaptive production system (self-configuration). Finally, the adaptation loop is discussed which uses the framework on the components of the production system to control functionalities.

### **6.2.1 Need for a Framework**

In a manufacturing environment, changes can occur due to factors like product variations, changing customer demands, and the addition or removal of production modules. To effectively manage such dynamic behaviour and ensure smooth operations by adapting configurations, a framework is re-

quired. This is further emphasised by the increasing need for flexibility and scalability. Usually, expanding manufacturing facilities involves introducing systems for added functionalities. Incorporating these add-ons into the existing production systems requires a framework that can handle changes seamlessly.

The framework should address the level of detail of the components in the system that bring functionality, and the organisation of these components into aggregates that can be configured. This ensures that each of these aggregates can adapt to the specific functionality while maintaining some structure, making the whole system easier to manage and update.

These dynamic systems will have some constraints on functionality and the associated variables. The constraints ensure the safe and efficient working of the system. The framework must manage these constraints while defining interactions between these aggregates. Mass customisation and data-driven decision-making propagate a more effective adaptability requirement in systems. Therefore, the required framework should also offer interaction with external systems like machine learning models, providing insights and parameter values based on accumulated data.

The framework must be modular to enable the application to functionality-based system configuration. This supports changes in functionality variables and relationships, but also attains a configuration as new variables are introduced as part of upgraded functionality.

### 6.2.2 Design Principles

The proposed framework aims to cover most aspects of the self-configuration needs for production systems. Each principle serves as a guiding concept

to achieve the objectives (discussed in the section 2.8.4) effectively within the framework:

- **Generality:** To make the framework applicable across various manufacturing applications and to simplify modelling complexities in production systems. This aim drives the design principle of “generality”.
  - **Principle Definition:** Generality refers to the framework’s ability to address a wide range of manufacturing applications without requiring domain-specific expertise or experts.
  - **Framework-specific Meaning:** In the context of this framework in this research, “generality” means that the design focus for the framework is towards developing architectural abstractions, such as modular components (i.e. aggregates) and constraints, their interactions, to model the production system.
  - **Enabling the Aim:** The principle of “generality” enables the aim by ensuring that system engineers, regardless of their domain expertise, can use this framework to design self-configuring production systems for any manufacturing application.
  - **Objectives Targeted:** 2,3 and 8.
- **Exactness:** To provide the basis for implementing and analysing self-configuration in production systems. This aim leads to the design principle of “exactness”.
  - **Principle Definition:** Exactness implies that the framework has structured operational semantics for module-based design.
  - **Framework-specific Meaning:** Within this framework, “exactness” signifies that there are well-defined rules and procedures, as explained in later sections, for module-based design,

ensuring exact self-configuration adaptation in applications.

- **Enabling the Aim:** The principle of “exactness” enables the aim by providing a structured approach that allows for the accurate implementation and analysis of self-configuration in production systems.
- **Objectives Targeted:** 5,6 and 7.
- **Separation:** To separate the functionality of the production system from its self-adaptive (i.e. self-configuration) behaviour, reducing complexity in integrating these aspects. This aim leads to the design principle of “separation”.
  - **Principle Definition:** Separation involves distinguishing the production system’s functionality from its self-configuration requirements, simplifying the integration process.
  - **Framework-specific Meaning:** In the context of this framework in this research, “separation” means that specific rules address the system’s functionality and adaptive behaviour separately.
  - **Enabling the Aim:** The principle of “separation” enables the aim by streamlining the integration process and minimising complexities that can arise from combining functionality and self-configuration requirements.
  - **Objectives Targeted:** 1,5 and 7.
- **Runtime:** To maintain a runtime abstraction layer that assists in monitoring, reasoning, and adapting the production system while it is operational. This aim leads to the design principle of “runtime”.
  - **Principle Definition:** Runtime involves having a digital twin

abstraction layer that operates alongside the production system, facilitating real-time monitoring and adaptation.

- **Framework-specific Meaning:** Within this framework, “runtime” means the existence of three levels of abstraction configuration, functionality, and constraints that facilitate monitoring and adaptation while the production system operates.
  - **Enabling the Aim:** The principle of “runtime” enables the aim by providing a digital twin layer that allows for real-time changes and adaptations in the production system.
  - **Objectives Targeted:** 5,6 and 8.
- **Enforcement:** To implement self-configuration in a production system by enforcing constraints. This aim drives the design principle of “enforcement.”
    - **Principle Definition:** Enforcement implies that the framework can impose specific behaviours in a production system.
    - **Framework-specific Meaning:** In this framework, “enforcement” signifies the capability to define configurations through variables and constraints, managed by a runtime digital twin.
    - **Enabling the Aim:** The principle of “enforcement” enables the aim by allowing the framework to exert control over a production system’s functionality.
    - **Objectives Targeted:** 4, 6 and 7.

These design principles are defined to address the objectives established in the previous section (6.2.1) and enable the framework to effectively tackle self-configuration challenges in production systems.

### 6.2.3 Framework Overview and Key Concepts

In this section, the building blocks of the framework are defined and the relations that constitute the proposed framework are elaborated. The concept and the terminologies behind the self-configuration approach modelled in the proposed framework are discussed.

#### Introduction to Framework Concepts

It is proposed that there are two main concepts in the framework; the **Production System Model** and the **Adaptation Strategy**, as depicted in figure 6.1.

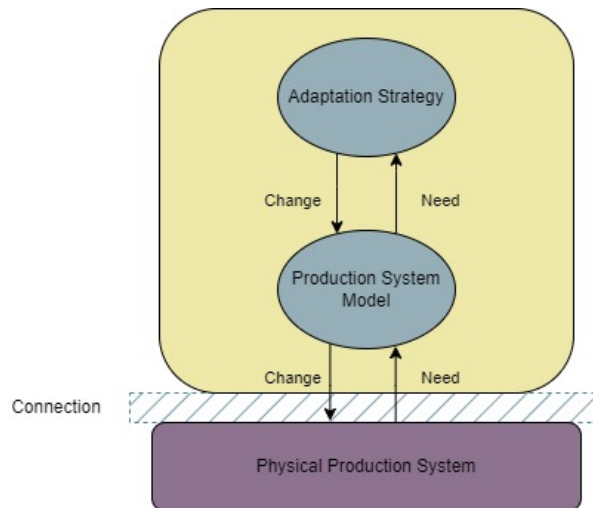


Figure 6.1: Conceptual illustration of the framework, showing aspects of the self-configuring production system. Production System Model presents a runtime of the Physical Production System acting as its digital twin. Adaptation Strategy monitors and controls the Production System Model in response to changes. Production System Model updates the physical system under guidance from Adaptation Strategy.

- **Production System Model:** The production system model is the representation of the physical production system. This model presents a runtime of the physical production system, providing a digital twin



of the system and enabling its adaptation as per the adaptation strategy.

Since an accurate representation of the physical production system in real-time for adaptation strategy is needed, it is required that the production system model be a true abstraction of the production system. In this research, it is proposed that the framework has an abstraction layer with three levels. These capture the system configuration, functionality, and constraints.

1. **Configuration, Functionality & Constraints:** In this research, it is proposed that the framework captures the complete architecture of the physical production system in the production system model while ensuring that there exists a mapping between the representation and the physical production system. This mapping will then ensure that changes in one can implement a reflective change in the other. This mapping should be updated regularly to maintain consistency between the production system model and the production system.
2. **Modularity:** The proposed framework promotes the addition and removal of configuration modules based on production system functionality, bringing a sense of modularity. Internally, the relationships between variables in a configuration module are subjected to rules defined at design. This separation of functionality rules and variable rules is present to establish the formation of configuration topologies. These rules are dependent on the fundamental architectural design of the production system itself. For example variable rule can be a certain parameter value restriction for a functionality in a manufacturing asset, functionality rule can be the interdependence of two components in

different manufacturing assets for configuring the parameters.

- **Connection:** The production system model is connected to the production system bi-directionally. This ensures that any change carried out in one affects the other. Such kind of conformance between both is necessary to ensure consistency for the adaptation strategy.

This can be achieved by getting a representation of the production system model and updating the information (i.e. about the difference) in the physical production system. Similarly, reflective changes can be identified in the production system, and they can be updated in the production system model. Now, this is possible but an inefficient approach as it requires integrating of approach (i.e. something in between) that identifies the change and updates both ways.

In this research, it is proposed that the framework captures the complete architecture of the physical production system in the production system model while ensuring that there exists a mapping between the representation and the physical production system. This mapping will then ensure that changes in one can implement a reflective change in the other. This mapping should be updated regularly to maintain consistency between the production system model and the production system.

- **Adaptation Strategy:** This ensures the monitoring and controlling of the production system model. The strategy works to sense the need for a configuration change and under set requirements carries it out. This is decided based on functionality rules and variable rules. The step change is decided (i.e. for configuration change) and the effective decision is updated in the production system model.

### **Elaboration on the Framework Concepts**

The concept presents a holistic view of self-configuring production systems. This framework, which is built upon this conceptual model, adheres to several design principles (6.2.2). These principles underpin the framework's implementation, emphasizing two key aspects: the Production System Model and the physical production system. The core strength of this framework lies in its ability to represent the production system within the Production System Model.

### **Configuration Change in Action**

To illustrate the practical application of the framework, consider this scenario: Suppose a manufacturing system in a production system is experiencing a high number of parts to process, then the production system must adapt by delegating the incoming parts to a similar manufacturing system. This can be modelled by routing and a quick configuration change to meet part and customer requirements using this framework. Therefore, it is necessary to study the production system and the components (i.e. manufacturing assets) in it to capture it completely so that the framework can be used to respond to the needs.

### **Adaptation Strategy - A Closer Look**

The adaptation strategy is composed of the steps involved in achieving self-configuration in production systems. The figure 6.2 illustrates the adaptation loop involving the elements of the self-configuration in the adaptation strategy.

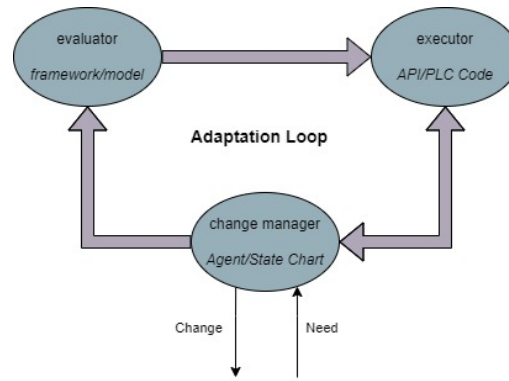


Figure 6.2: Adaptation Loop for the process in the framework to achieve self-configuration in production systems. *Evaluator*, evaluates the rules for the configuration change, *change manager*, guides the configuration change for functionality and *executor* changes the configuration on the system.

In the Adaptation Strategy, there is an *evaluator* that evaluates the functional rule and variable rule for a configuration change. After evaluation, the configuration change can be loaded into the production system, complementing the product and process requirements. These changes are relayed to the system responsible for the change on the production system (e.g. can be agents, a script etc.). The change-inducing system utilises an *executor* for changing the configuration.

The production system model is linked to the evaluator so that real-time constraints and rules can be applied to the production system. In summary, the evaluator, executor, and change manager are the fundamental elements of the adaptation strategy. This means that the proposed framework relies on offline programming with real-time applications to achieve self-configuration.

## 6.3 Module Driven Configuration System Model

### 6.3.1 Framework Architecture and Abstraction Levels

This section details the basic architecture of the proposed framework and the accompanying elements. This section aims to clarify the concepts introduced in the previous section.

#### Introduction to Framework Architecture

The proposed framework represents the production system by maintaining a **runtime**. The runtime, in a production system model, is a layer of abstraction capturing system configuration, functionalities, and constraints. The production system model can then be used to achieve self-configuration by utilising the adaptation strategy.

**Abstraction Layers in the Production System Model** In the previous section, it is proposed that the **Production System Model**, in runtime, is a layer of abstraction, this is further detailed in the figure 6.3. Production System Model is structured into three distinct levels:

- **Functionality (First Level):** The first level captures the functionality of the production system. This includes representing the commands necessary to initiate specific behaviours, alongside client-server system functionalities. This level captures the behaviour of the production system through a standardised data representation (i.e.

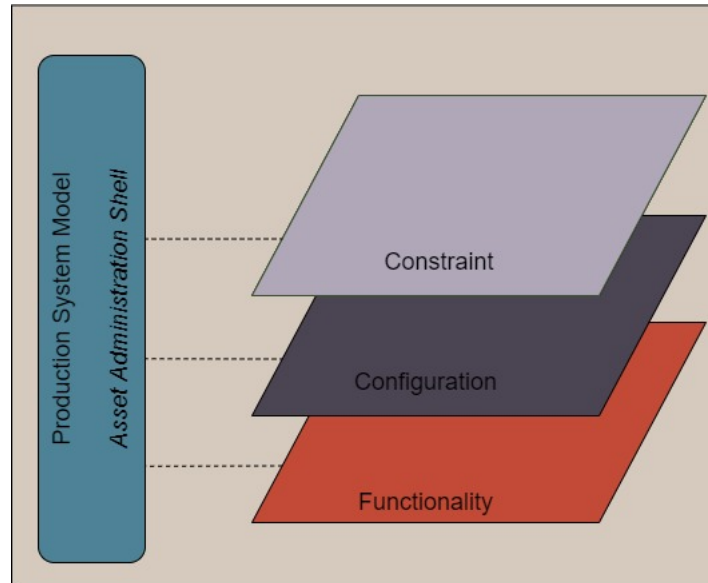


Figure 6.3: The production system model is represented at three levels of abstraction namely: *Functionality*, *Configuration* and *Constraint*.

ontologies and semantics) in conjunction with the production system interfaces.

- ***Configuration (Second Level):*** The second level of the abstraction layer captures the system's configuration in terms of **variables**. These variables are derived from functionality requirements. In this context, variables are considered as representations of **Configurable Objects (CO)**. COs encompass all elements within a production system, both physical and cyber, that are responsive to configuration change. Relationships between variables are established and governed by rule constraints. For example, settings within a production system can be represented using parameters, with relationships specifying connections between variables. Variable rules define the necessary constraints on these variables and are employed by an **evaluator** to explore the rules between variables and COs.
- ***Constraints (Third Level):*** The third level, which forms the abstraction layer, captures the constraints acting on the variables in

a configuration. A configuration is represented by the variables, their relationships, and the constraints binding them. A change produced in either of these elements of configurable objects results in a new configuration. This level handles the changes that can happen if a variable is changed and its impact on relationships and constraints. For example, in a production system, a product requirement is received that demands the replacement of a configurable object. The proposed framework allows for the change in configuration through clearly defined functionality rules. These are called **functionality rules**, first to establish separation from **variable rules**, and second to be associated with production system functionality (as they directly impact these). E.g. for a leak test system functionality rule can be related to a change in the pneumatic module raising the pressure testing capability to 50KPa for a product variant and the variable rule can limit a parameter value based on system design. More detail on these rules is presented in section 6.3.4.

**Configuration Changes in the Framework** The change in relationships between variables can also be handled by the proposed framework. For example, a production system requires a change in a configurable object that increases the parametric limit of a variable, previously if that parameter went beyond a value there existed a relationship to trigger an alarm. The relationship can now be handled by the variable rule change in the constraint level of the proposed framework. Therefore, the effect of such configuration change can be seen in action once the functionality is executed with the changed configurable object. As the changed configurable object is introduced, the relationship is changed implicitly according to variable rules.

Therefore, in the proposed framework the change in configuration can be handled for both change in variables, and change in relationships (by functionality rules). These two types of layers and their associated rules (i.e. functionality and variable) capture the self-configuration approach by mapping the configuration space.

### 6.3.2 Overview of the Architecture

The self-configuration approach requires that the three levels of abstraction be captured. In this section, the *Structural Module* as the fundamental concept is presented and considered as a basic unit for self-configuring production systems for capturing these three levels. This section discusses how structural modules encapsulate key elements, such as functionality, variable rules, and functionality rules, and explores their role in adapting to configuration changes.

**Introduction to Structural Modules** Each Structural Module contains:

1. **Functionality:** Representing behaviour in form of capabilities of configurable object.
2. **Variable Rules:** Governing limitations on the configurable object variables and their values.
3. **Functionality Rules:** Effects relationships between configurable objects and also the effect of changing variables in configurable objects behaviour.



**Interplay of Multiple Modules** In this research, it is considered that there may exist multiple modules in a production system, which in the case of distributed (i.e. evolvable) manufacturing may also evolve over time. A production system contains a superposition of several of these modules, representing sharing its configurable objects. In the proposed framework, it is assumed that these configurable objects can be added and removed, so the framework has the capability to add and remove connections in modules. This is carried out through functional and variable rules. This change in configuration as a means of reasoning through variable and functionality rules enforces accurate behaviour in production systems.

**Illustrating Module Changes** To illustrate this concept, consider figure 6.4, which provides a visual representation of module structure and its evolution within the framework.

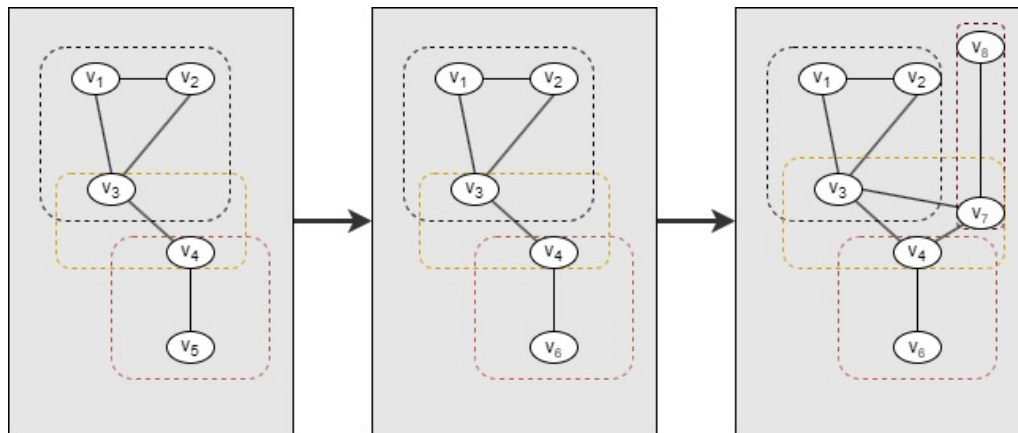


Figure 6.4: Module Evolution in the Configuration Change Process. The illustration shows changes in module w.r.t variable change (*middle*)( $v_5$  to  $v_6$ ) and (*right*) new module introduction.

Let's consider that there are three configurable objects that are defined in the form of three modules (left), Separation between these interacting modules is shown using dotted lines. As visible, the upper module consists of variables  $v_1$ ,  $v_2$  and  $v_3$ , the central module consists of variables  $v_3$  and  $v_4$ , and finally the lower module consists of variables  $v_4$  and  $v_5$ . The variables in

a configuration housed in a structural module have relationships, defining the connection between them. Also, some variables are shared between modules, showing dependency, relation, and constraints.

Lets' assume that a configuration change happens that induces a change in the variable from  $v_5$  to  $v_6$ . As per the variable rules defined in the module, the  $v_5$  is disconnected from the module and  $v_6$  connected. This update does not in any way affect the central and upper modules in terms of functionality.

Another configuration change happens, and a new configurable object is introduced in the system (right) having variables  $v_7$  and  $v_8$ . Functionality rules, impact the central module and a new relationship is established. As a result, the central module relationships are modified and the central module houses more variables. In addition, a new module that houses the two new variables is introduced.

**Module Types and Structure** Module types capture the type of configuration, so the type of variables, along with the interaction between variables and the corresponding functionality rules. A production system is represented by the superposition of a number of modules capturing configuration in configurable objects. Utilising this, the architecture can capture configurations of configurable objects and the functionality of those objects. More detail on this is presented in section 6.3.4.

**Structural Module Representation** The structural module is represented by the following figure 6.5 in the proposed framework. Modules can be best expressed as the structural representation that deals with the deployment of variables in a configuration for a configurable object in a

logical link. *Links* are graph-like structures that consist of possible connections between variables and production systems. *Allocation* can be used to relate the variables with data values, therefore the configuration, to the connection in links.

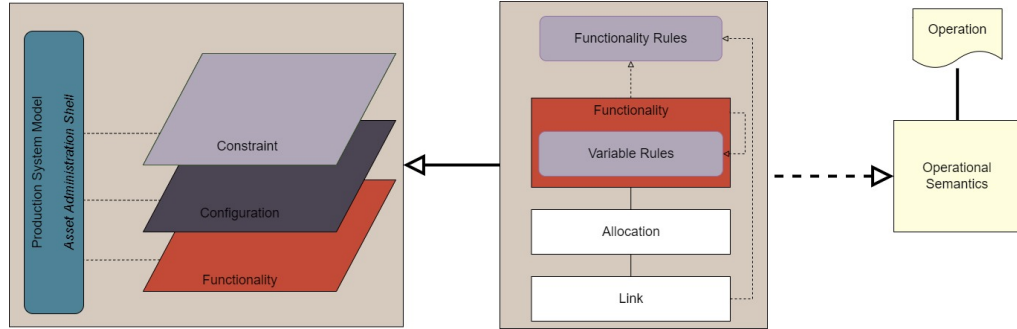


Figure 6.5: Representation of a Structural Module. Functionality, configuration, and constraints are encapsulated in this Structural Module defined through operational semantics. A structural module consists of variables, rules (functionality rules and variable rules), allocation and link.

Along with these, the definition of the module encompasses the variable rules and functionality rules. The variable rules define the interaction between variables in configuration for a configurable object present in a module. Functionality rules define the rules that can be used to govern the configurable objects, the links, and their allocations.

**A Module Example** Figure 6.6 illustrates a module example to understand the module structure with links and allocation and its connection to elements of AAS. It is assumed that the variables  $v_1, v_2$ , and  $v_3$  are linked with a value through an allocation. Assuming the allocation to be “A”, then the links “L” can be established as updating the allocated values to the production system.

If such logic is followed then the variable, their relationships, and their constraints can be represented using a module-based proposed framework, as shown in the figure.

### 6.3. MODULE DRIVEN CONFIGURATION SYSTEM MODEL

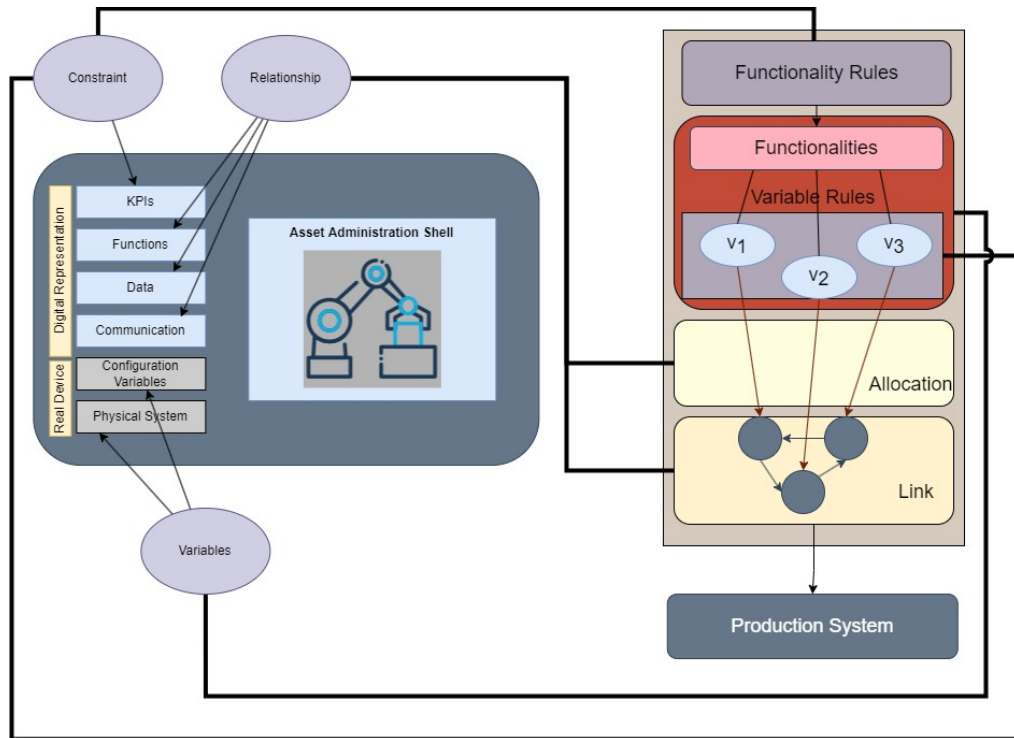


Figure 6.6: Module Structure connected with the elements of AAS. Variable rules govern variables. Functionality is executed subjected to constraints defined by rules (functionality and variable). Relationships are encapsulated in allocation and link.

The use of links and allocation within modules is highlighted as an effective means of structuring complex production system interactions. These concepts are deemed straightforward to comprehend and manipulate, yet they can represent intricate connections within production systems. The proposal underscores the advantage of separating links and deployment from system functionality, enabling independent manipulation and adaptability to a broader range of production systems. Each module in module-based production systems encapsulates its functionality and interactions. It's suggested that a module's structure, concerning configuration-related interactions, remains unchanged until a functionality rule action is initiated. Consequently, in typical scenarios, the production system maintains a static architecture. While the execution of functionality doesn't alter the module structure, functionality rules can introduce architectural changes as needed.

Notably, these changes in module structure do not impact configurable objects but only affect their variables. Hence, during execution, configurable objects maintain their configuration state, albeit with potential changes in variable values, while links and allocation remain constant.

Summarising, the proposed framework relies on a structural representation known as module as the basic unit of architecture. These modules can be superimposed in a production system to represent its configurable objects. Now, the configurable objects are presented with their instances before proceeding to module types and their instances.

### 6.3.3 Configurable Objects

A Configurable Object (CO) can be precisely characterized as follows:

In a production system, a Configurable Object is a representation of a configurable object type denoted as  $CO^t$ . This representation takes the form of a labelled transition system  $(CC, I, V, T)$ , where:

- $CC$  is a set of configurations consisting of variables, relationships, and constraints that fulfil certain production objective criteria. Each configuration, often referred to as setting, is responsible for physical system operation behaviour.

As a simple example, consider configurations comprising two discrete variables  $X$  and  $Y$  shown in figure 6.7.

$$X \times Y = CC \cup CC' \tag{6.1}$$

$$CC \subseteq X \times Y \tag{6.2}$$

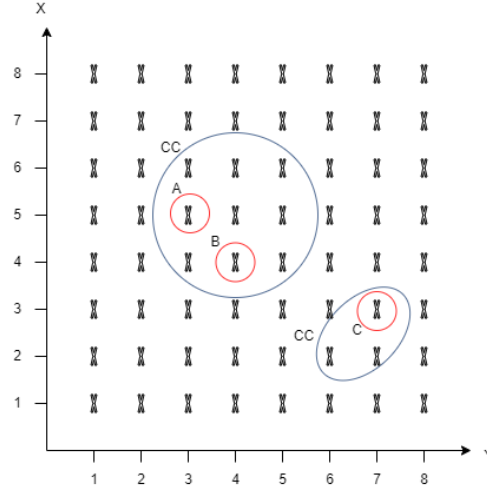


Figure 6.7: Configuration space  $X \times Y$  of variables  $X$  and  $Y$ .  $CC$  are the candidate configurations satisfying relationship and constraints.  $A, B$  and  $C$  are those candidate configurations that satisfy the production system requirements.

$$CC \models Relationship \times Constraints \quad (6.3)$$

The candidate configurations  $CC$  is a subset of configurations that satisfy relationships and constraints on the variables but not necessarily the requirements (e.g. KPIs) of the production system. Considering that  $A, B$  and  $C$  are the configurations that meet these requirements.

$$\{A, B, C\} \subseteq CC \quad (6.4)$$

$$\{A, B, C\} \models Relationship \times Constraints \times Requirements \quad (6.5)$$

- $I$  is a finite set of interfaces that provide a connection to the configurable objects. These provide links to the configurable object, through which changes can be made.
- $V$  is a finite set of variables and their associated values.
- $T \subseteq CC \times I \times (G(V) \times F(V)) \times CC$  is a set of labelled transitions, with  $Update(V) = G(V) \times F(V)$  where  $G(V)$  is the guard on  $V$  and

$F(V)$  is the corresponding update function acting over  $V$ .

$$\text{Variables} \times \text{Values} \rightarrow V \quad (6.6)$$

$$F : V \rightarrow V \quad (6.7)$$

$$G : V \rightarrow \text{Boolean} \quad (6.8)$$

By the above definition statement, every transition represented as  $\tau = (CC_i, i, \text{update}, CC_j) \in T$  can also be instated as  $\tau = CC_i \xrightarrow{i \text{ update}} CC_j \in T$ . It is assumed that a configurable object has a variable mapping  $V$  available through an interface  $i \in I$  that is available for interaction (i.e. to be configured).

The *semantics* of the configuration object type  $CO^t$  is illustrated as  $(S, \sum, \rightarrow)$  where

- $S$  represents a set of states that are a result of the realisation of  $V$ . An update on  $V$  will give a different set of states  $S'$ . The states represent the behaviour that the machine follows during functionality execution as it takes up a configuration setting.
- $\sum$  is a set of labels.
- $\rightarrow$  are transitions given by rule;

$$\tau := \frac{CC_i \xrightarrow{i \text{ update}} CC_j \in T \quad G(v) \quad v' = F_\tau(v) \quad i \in I}{S \xrightarrow{i \text{ update}} S'}$$

$S'$  is achieved through the transition from  $S$  under interface  $i$ , through *update* with accessible variable under interface  $i$  iff (1)  $\tau = CC_i \xrightarrow{i \text{ update}} CC_j$  is a possible transition of  $T$ , (2)  $G(v)$  is the guard imposed on variable mapping  $v$ , (3) Update occurs as  $F_\tau(v)$  is possible, and finally (4) accessible

variables in configurable object exist through interface  $i$  such that  $i \in I$ . If such a transition is possible, then a change in configuration is possible in a configurable object resulting in a set of states for functionality execution.

**Applying the Concept:** In manufacturing, a Configurable Object (CO) is a manufacturing aggregate in a production system that can be configured. This aggregate has different settings (i.e. Candidate Configurations) to perform functionality (i.e. Variables, relationships and constraints). Interfaces are the different ways to connect the machine to the production line. Labelled Transitions are the step-by-step instructions for changing the machine's settings or connections.

The “semantics” of a CO are like understanding what happens when a recipe is followed and functionality executed. States are snapshots of the machine in different configurations, labels are like tags on each step, and transitions are the actual steps.

So, using a CO means starting with a machine configuration (i.e. a set of states representing the behaviour of functionality as it executes) and following the recipe's steps (i.e. transitions) to execute functionality. This flexibility allows adapting manufacturing systems for different tasks without building entirely new machines. It's like having a versatile recipe for factory machines, each configuration giving a separate set of states.

### 6.3.4 Modules

Modules can be considered as dynamic entities that constitute interacting elements. The modules can be described as an embodiment of three concepts, namely; functionality, link, and allocation (see figure 6.6).



The functionality consists of the operation of the production system like pick and place, leak test etc. These functionalities have certain variable requirements. The link is used to establish the association between the functionality-specific variable requirements and provide a link to the production system. The allocation maps the data values to the variable and is then updated on the production system through a link.

The variables, forming the configuration, present within the module influence the states during functionality execution. These variables in the module, based on functionality, are dependent on variable rules. This means that if these rules governing the variables in the module change then it affects the variables. However, this change must be accompanied by some constraints. This change results in modifying the module for different candidate configurations.

**Applying the Concept:** In the context of manufacturing, modules can be considered as flexible building blocks that comprise three fundamental elements: functionality, links, and allocation. Functionality defines what a particular module does in a manufacturing process, such as a welding station that requires specific variables like temperature and pressure. Links connect and update different modules, so the welding station can get the temperature data from another part of the production system. Allocation is similar to equipping each module with the correct tools and information it needs to perform its role effectively.

These modules are not static; they can adapt to changing production requirements. For example, if a car manufacturer introduces a new type of seat, the welding module's requirements might change. However, these changes must comply with predefined rules and constraints to maintain

the overall efficiency of the production system. Manufacturing modules are adaptable components that follow guidelines for an efficient production process.

### Links and Allocation

The link and allocation are abstract representations that are used to assign and update values to the module variables. Links are used to demonstrate connections between the variables through instances of generalised link types for the production system. A link type  $L^t$  is represented by a tuple characterised by  $(N(L^t), P(L^t), Cn(L^t))$ , where,

- $N(L^t)$  is the collection of possible links (i.e. interfaces) that a link type can take, with the production system.
- $P(L^t)$  is the primitives that actuate the link types (i.e. necessities for establishing connection).
- $Cn(L^t)$  represents the constraint on link types (e.g. subnet, ports etc.). They are a subset of the constraints imposed by functionality, related to connection.

The link types are dependent on the production systems. These links are related to variables through an allocation. These allocations  $A$  are mapping of data values to the variables accessed by the link types through  $N(L^t)$  satisfying  $P(L^t)$  under subjected  $Cn(L^t)$ .

**Applying the Concept:** In manufacturing, links and allocation function as the communication pathways and parameter provisions that facilitate the operation of production modules. Think of a car assembly line where

different modules handle tasks like welding and painting. Links represent how crucial information, like temperature requirements from the painting module, is transmitted to others, ensuring each module knows what to do. Allocation ensures that each module has the data to perform its task, like equipping the welding module with the necessary settings. If the car manufacturer decides to introduce a new paint colour, this would require adjustments in links (i.e. as per the new colour requirement) and allocation (i.e. to provide the right paint settings and other data).

However, these changes must complement predefined rules and constraints to maintain efficient production.

### Module Types

A module type  $M^t$  is represented by a tuple  $((V, A, L), Cn)$  where,

- The triple  $(V, A, L)$  are the components that capture the configuration of a production system. These components correspond to a set of variables  $V$  representing configuration  $C$  for functionality, an allocation  $A$  that assigns data values to these variables, and the link  $L$  that updates these variables on the physical system.
- The constraint  $Cn$  is basically of two types, one specific to variables derived through functionality requirements and the other belonging to functionality-specific constraints. The **variable constraints** are given by the form  $(v, Cn_v, i_v, g_v, f_v)$  where  $v$  is the variable from the variable set,  $Cn_v$  is the rule on that variable,  $g_v$  corresponds to the variable guard, and  $f_v$  corresponds to the data update function applied to that variable. The **functionality constraints** is given by the form  $(v_f, Cn_{v_f}, i_{v_f}, g_{v_f}, A_{v_f})$  where  $v_f$  is the variable from the variable

set derived from functionality,  $Cn_{v_f}$  is the rule on that variable from functionality,  $g_{v_f}$  corresponds to the variable guard against functionality, and  $A_{v_f}$  corresponds to the allocation related constraints from functionality, like one variable in a module is linked to another variable in another module therefore needs to be configured in relation to each other.

The structure of the module type may be dynamic but remains static during functionality execution. The dynamic nature is due to an update in triple  $(V, A, L)$  and a change in rules pertaining to it.

**Applying the Concept:** In manufacturing, a module type ( $M^t$ ) serves as a blueprint for configuring a production process. It consists of three key elements: variables, which represent the process's settings or conditions (e.g., conveyor belt speed); allocation, which specifies how settings are assigned to these variables (like setting a pressure value among possible pressure values); and links, which facilitate information flow to and between different systems (e.g., conveying data to physical production systems). These elements ensure a production system operates as intended. Additionally, module types include constraints, such as variable rules (e.g., speed limits for safety) and functionality constraints (e.g., adjusting resource allocation based on machine speed). While module types can adapt over time, they remain fixed during production. Essentially, module types offer a dynamic framework for configuring and managing manufacturing processes with rules and constraints defining their behaviour.

For example, in an automobile assembly line, consisting of modules of a module type, variables could include parameters like welding machine temperature, allocation might determine what variable values are assigned, and

links would ensure that information is communicated to the assembly line module. Variable constraints could restrict machine temperature within safe limits, and functionality constraints might dictate that if a welding machine operates at a higher temperature, then an alarm can be raised.

### Functionality and Variable Constraints

The variable and functionality constraints affect the module structure, which individually are defined by forms  $(v, Cn_v, i_v, g_v, f_v)$  and  $(v_f, Cn_{v_f}, i_{v_f}, g_{v_f}, A_{v_f})$  respectively. In this section, the constraints  $Cn$  component for both are elaborated.

More precisely, the constraints may be defined as:

$$Cn ::= Cn_V \mid Cn_A \mid Cn_L \mid Cn_i \wedge Cn_j \mid \neg Cn \quad (6.9)$$

Constraint  $Cn_V$  represents the rules applied to the parametric values of variables, it can be a range or a fixed value.  $Cn_A$  represents the constraint on allocation, and  $Cn_L$  is the constraint on linking with the production system.

The production system constraints are evaluated based on the constraint  $Cn$  and the module structural components  $V$ ,  $A$ , and  $L$ . Variable constraints are evaluated by confirming that the proposed configuration  $CC$  fulfils variable constraints.

**Functionality Rules** Functionality rules are used to govern the behaviour of the production system as the functionality is executed. These rules impact the behaviour of the functionality, affecting the configuration in terms of provided variables, the allocation of data values to these

variables and the link to the production system. They are represented as updates to variables by module structure components  $V, A, L$ .

The assignments of guards and actions can be used to apply functionality rules on a module to provide variables for self-configuration and the relevant changes for links and allocations. The actions are as follows;

$$\text{Action} : R_f := \text{Expressions} \quad (6.10)$$

The action is carried out in the form of expressions established in detail in Chapter 8.

**Variable Rules** Variable rules define the rules concerning the variables in a module structure. They deal with all interactions happening implicitly among variables for functionality execution. These involve limiting a parameter value, restricting it to a range or imposing a dependency of parameter value on others within the same module.

For actions, and expressions governing the variable rules are given by;

$$\text{Action} : R_v := \text{Expressions} \quad (6.11)$$

The variable rules are derived from functionality and environment in a production system. The restrictions on functionality by these rules are imposed until they are satisfied. The expression governs the manner of change or the kind of change possible. The expressions are established in detail in Chapter 8.

**Applying the Concept:** In the context of manufacturing, module design entails setting rules and constraints that govern the behaviour of a production system. These rules impact the structure of modules, which consist of variables representing various production parameters (e.g., machine speeds, temperature) and functionality-derived variables related to specific production functions. Constraints come in different forms, such as limiting variable values, regulating module allocations, and defining connections with the production system. These constraints are applied to the production system's configuration to ensure that it complies with variable, allocation, and link constraints.

Functionality rules play a crucial role in defining how the production system behaves and how modules configure themselves. For instance, a functionality rule might dictate that when a certain condition is met, a specific action occurs, like creating a new variable, deleting an existing one, or updating an allocation or link. Variable rules further define interactions within the module by specifying how variables are accessed and manipulated. Overall, in a manufacturing scenario, these rules, and constraints ensure that modules adapt and configure themselves appropriately during production. For example, in an automotive assembly line, functionality rules can adjust the speed of robotic arms based on quality control feedback, while variable rules can specify what limit these parameters can have.

### 6.3.5 Module Based Production System

In the proposed framework, the production system is defined in the form of modules driven by functionalities. The configuration can be adapted based on functionality requirements and subjected to functionality and variable constraints. A production system may have multiple modules depending

on functionalities, and variables  $V$  of these functionalities will depend on rules. Therefore, it can be argued that each module presents a functionality, i.e. adds value to the production system, bringing with it the required set of variables, hence the configuration. There are variable rules that confront the variable logic in the respective module, and functionality rules that represent the rule logic for relationships and other constraints on variables. The rules can also be used to observe some kind of constraint among different modules applied to functionality variables collectively.

In a production system, there can be multiple functionalities belonging to multiple configurable objects (i.e. manufacturing assets) put together to perform production system operations. Considering that a production system consists of two modules for two functionalities, the module-based system application can be best illustrated in figure 6.8. The rule applied to both modules shows a kind of coordination between variables in a configuration. This section introduces this kind of coordination in detail and elaborates on the operational semantics of the module-based production system.

The production system consists of a finite set of modules, consisting of knowledge defined from previous sections and module instances  $m$  of type  $M^t$ .

### Rules Constraints

The rule constraint that deals with module-based production system applications is similar to individual variable and functionality rule constraints discussed for single modules in previous sections. The tuple is defined in the form of  $(V, Cn, I, G, R)$ . Here, the constraints  $Cn$  acting on the system are defined by;



6.3. MODULE DRIVEN CONFIGURATION SYSTEM MODEL

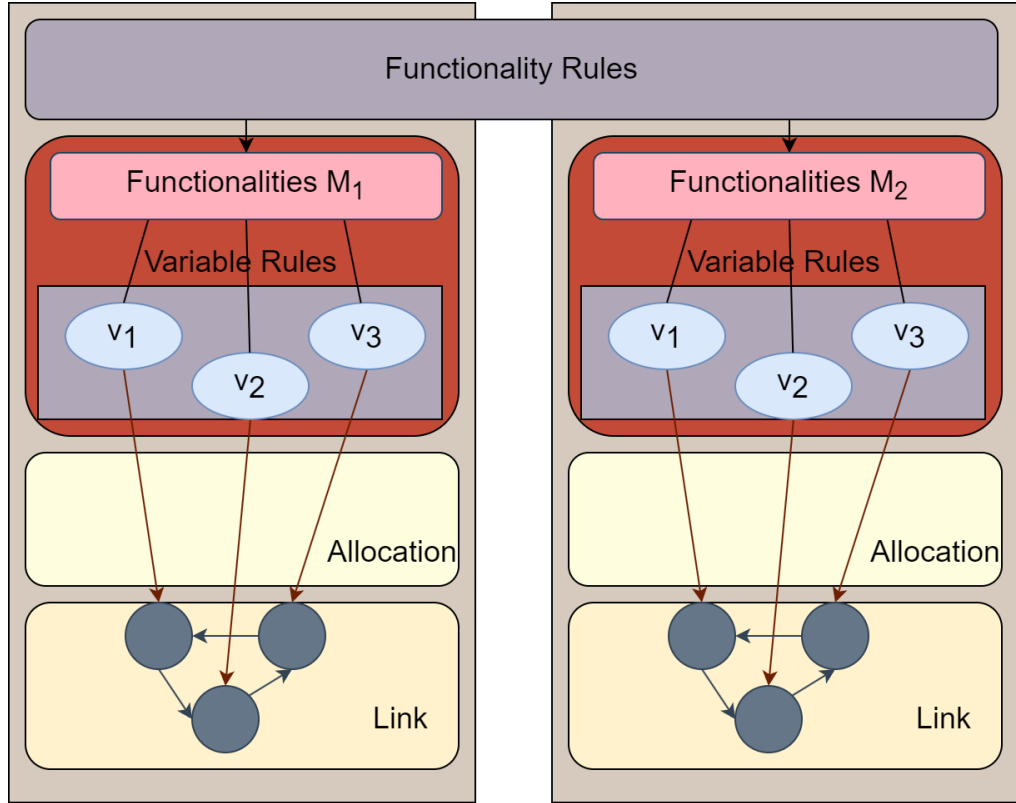


Figure 6.8: An illustration of a Module-based system having two modules hosting functionalities  $M_1$  and  $M_2$ . Functionality rules can be shared between modules, meaning they can govern the functionality of both modules and also define a dependency between them.

$$Cn ::= Cn_V \mid Cn_A \mid Cn_L \mid Cn_i \wedge Cn_j \mid \neg Cn \quad (6.12)$$

In the above definition, the constraint can be on the specific configuration value  $Cn_V$ , or a constraint on allocation on the production system  $Cn_A$ , or a constraint on link with production system  $Cn_L$ . There can be a constraint applied through the compositional logical operation of rule constraints along with negated constraints.

These constraints are applied to module configurations on its context parameters, which can be represented as  $(M, m)$  where  $M$  is a set of module instances and the module instance  $m$  given by  $m = (m \rightarrow (V, A, L) \mid m \in M)$ .

### 6.3. MODULE DRIVEN CONFIGURATION SYSTEM MODEL

The module instance  $m$  belonging to set  $M$  models constraints based on components  $V, A, L$ . The constraints on the module are evaluated by confirming that the proposed configuration  $C$  belonging to module instance  $m$  fulfils all the constraints on  $V, A, L$  compositional and negated constraints. For a module-based system, the guards (using constraints), and actions can be represented as:

$$guard : G := G_I \mid G_V \mid G_F \quad (6.13)$$

$$Action : R := Expression$$

Simply, the guard in the case of a module-based production system may be a guard on variables  $V$ , on the interface  $I$  or on the functionality  $F$ . The action is carried out in the form of expressions established in detail in Chapter 8.

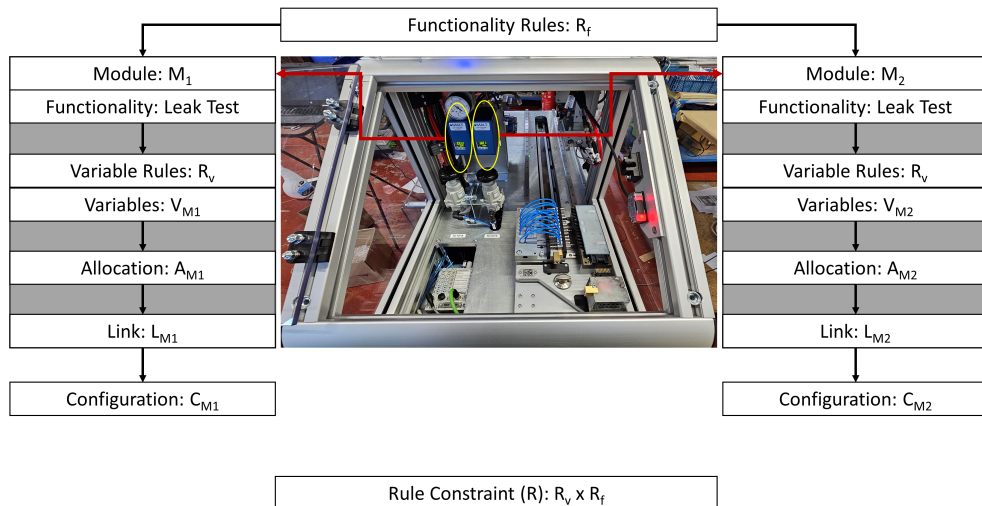


Figure 6.9: An example of a Module-based system having two modules  $M_1$  and  $M_2$  hosting leak test functionality. Both modules can perform leak test functionality subjected to rules ( $R$ ). Variable rules ( $R_v$ ) are governing individual module variables and functionality rules ( $R_f$ ) govern the functionality behaviour of both modules.

### Operational Semantics

A module-based production system  $MPS$  (figure 6.9) is represented by:

$$MPS \in \{CC \times M^t \times R_v\} \times R_f \quad (6.14)$$

$$MPS \in \{M\} \times R_f \quad (6.15)$$

This definition consists of a set of candidate configurations  $CC$ , a set of module types  $M^t$ , and a set of variable and functionality rules governing the modules  $R$ .

A module-based production system executes through functionality operation and requires a change in configuration within a module as new modules are introduced in the system or a requirement is presented. The operational semantics of a module-based production system are presented as a labelled transition system  $[MPS] = (CC, \sum, \rightarrow)$ . Here:

- the set  $CC$  consists of candidate configuration.
- the set of labels  $\sum$  corresponds to change on configuration, under functionality operations actions or effect of variable and functionality rules.
- the set of transitions  $\rightarrow = \xrightarrow{R_v} \cup \xrightarrow{R_f}$ , where transitions are defined by rules in a module and rules between modules in a module based system.

The rules and concepts are used to capture operational semantics-related information in the coming chapter and also govern the expression-based execution of the process.

**Applying the Concept:** In this framework, consider a manufacturing facility that produces gas cylinders and one of its crucial processes is leak testing. The production system consists of distinct modules, each resembling a specialized production unit. In this case, the modules represent various stages in the leak testing process, such as initial inspection, pressure testing, and final inspection. All these modules are incorporated into a machine.

These modules are driven by specific functionalities: the initial inspection module checks for any visible defects, the pressure testing module performs the leak test, and the final inspection module ensures the quality of the tested cylinders. Each module has its set of variables, such as testing pressure levels, testing duration, and inspection criteria, which are vital for configuring the leak testing process.

Variable rules set constraints on these variables, ensuring that testing parameters stay within safe and efficient limits. Functionality rules dictate the system's behaviour; for example, if the pressure testing module detects an abnormality in a cylinder, a functionality rule might trigger an immediate rejection of the cylinder. This coordination ensures that the leak testing process remains effective and safe. The system's operational semantics involve dynamic transitions between configurations, allowing the plant to adapt in real-time. If the production demand increases, a functionality rule could add a new pressure testing module, maintaining flexibility and efficiency in response to production needs.

### 6.3.6 Module Based System Application

The proposed framework facilitates the self-configuration deployment in production systems through a module-based approach. In the module-based system, each module representing a configurable object consists of components that interact to achieve self-configuration through interfaces (i.e. each variable has an interface) and data points (i.e. representing values in configuration space). Each module, representing a configurable object, has variables that need to be configured under certain constraints known as variable and functionality rules.

There can be multimodule interactions between configurable objects, as the rules can be shared between modules. If the rule permits, the variable (i.e. in their values) can be the same in modules belonging to different configurable objects. It is assumed that in this approach the self-configuration only executes interactions in the module during the starting of an operation but remains static during the operation itself. Also, the variables, desired by functionality, to be configured are set at design time and do not change at production system configuration.

The question arises of how the static configuration can be used to have some kind of dynamic configuration in a machine. The functionality is linked with a module, representing a configurable object, therefore making it easier to move toward dynamic configuration as the configurable object is changed physically in a production system. Since functionality introduces variables that need to be configured, therefore a new variable is introduced every time a module is changed. The only restriction here is that the system remains static during functionality execution, however, the system can again be configured before starting another functionality execution.

### Configurable Object-Based System

In this framework, the production system is composed of modules that are governed by rules for self-configuration. The interaction between variables between different modules may be permitted subject to constraint rules, i.e. functionality rules. An interaction between variables happens when such a rule is enabled and an interface exists between two variables. So it can be said that if the interaction is enabled between two or more variables present in different modules through rule constraints, then completion of the interaction consists of the variable taking a value somewhat related to the manner in which constraints bind the variables. A more formal definition is as follows;

A production system of configurable objects  $Pr(CO)$  is a tuple given by  $(CO, Pr)$  where,

- $CO = \{co_1, \dots, co_n\}$  is a set of finite configurable objects in a production system, and
- $Pr$  can be best understood as an interaction between the configurable objects to execute functionality for production operation. Interaction  $pr$  is a triple  $(I_{pr}, G_{pr}, F_{pr})$  where
  - $G_{pr}$  is a guard on variables in configurable objects pertaining to interaction  $pr$  through the provided interface  $I_{pr}$ , and
  - $F_{pr}$  is an update function on the variable, through which the change happens on the variable, and finally
  - $I_{pr}$  is the interface for interaction  $pr$  through which configurable object interacts.

The semantics of a production system of configurable objects based on a

module based system can be given as  $(C, \Sigma, \rightarrow)$ .

By controlling the guard and update function, the manner of interaction between variables between configurable objects can be controlled. The new values acquired by this for each configurable object become the configuration of the configurable object for the production operation.

The semantics provide the implementation basis for the proposed framework, coordinating interaction between modules representing configurable objects. The change to the configuration of the configurable object happens in terms of its variables under constraints that remain static during production operation. Therefore, for the configurable object-based system, the interaction happens in steps:

1. Each module represents functionality, a production system can have several of these modules. The register information about the module must contain information about its variables, the constraints (e.g. relationship, KPIs etc.) and the information on how the variables between configurable objects are linked. These are defined in variable and functionality constraints.
2. At the start of production operation, the variables are configured as per the allocation and link components (module-based systems).
3. The interaction between configurable objects happens through the transition rule.
4. The functionality of the production system is carried out.

An illustrative example is presented in figure 6.10 representing the update of variables from one configuration to another in a Configurable Object Based System.

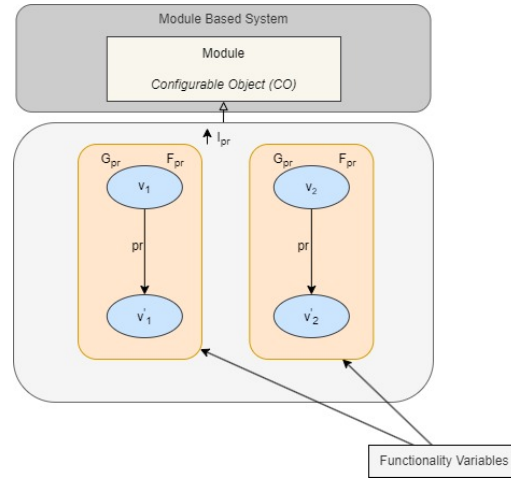


Figure 6.10: An illustrative example of static configuration through the framework in a Configurable Object Based System. The  $G_{pr}$  and  $F_{pr}$  act on variables to update them through interface  $I_{pr}$ .

### Towards Dynamic Configuration

This framework has been introduced to provide a solution to address dynamic behaviour in the production system. In the module-based production system, the dynamic behaviour is due to changes in functionality as modules are added or removed. A production system that implements a module-based system consists of multiple modules, each responsible for a functionality, composed of variables forming a configuration. In that sense, a module based system implementation connects granularity with modularity represented in the form of modules providing multiple functionalities and interfaces. This provides a direction towards adaptation of a self-configuring production system where each module is responsible for its own functionality accompanying its own set of variables, relationships, and constraints.

For dynamic self-configuration, the framework introduces the variable and functionality constraints that address the changing configuration variable requirements. A constraint dictates on how the variables interact with themselves (i.e. value/range) and among each other (i.e. relationship). The



framework elaborates on the components necessary to make self-configuration possible in production systems. These constraints act on configuration instances from the candidate configuration space.

The framework proposes a structured set of interactions as a module with its components. The components can be shared between different modules to account for changing needs in production systems. A production system may have multiple modules that account for different functionalities in the production system. These functionalities will bring a set of variables that are subjected to certain constraints known as variable rules and functionality rules. These variables can then be allocated a set of values and linked to the production system to execute the functionality.

The allocation allows interaction with external systems like machine learning models so that the system learns from historical data and take decisions (gets values) from insights. The module-based system implementation in the system keeps the whole structure in the production system modular. This modular nature allows that the proposed framework can be applied to achieve dynamic configuration. In this framework, the allocation component makes the allocation of values to variables possible. These variables in our proposed framework can be interacted to through an interface, for allocation and linking.

The proposed framework contains instances of modules of a module type. Therefore, the module-based system is quite generic, targeting various functionality-based dynamic configurations in production systems. This framework encompasses the essence of self-configuration, as it allows not only dynamic change between variables (i.e. relationships) pertaining to constraints but also self-configuration change as new variables are introduced as part of upgraded functionality. Hence, by this framework, expres-

siveness is provided to all degrees of self-configuration.

## A Manufacturing Example

Let's apply the representation to a practical manufacturing application example involving a single workstation module within a self-configuring modular manufacturing system.

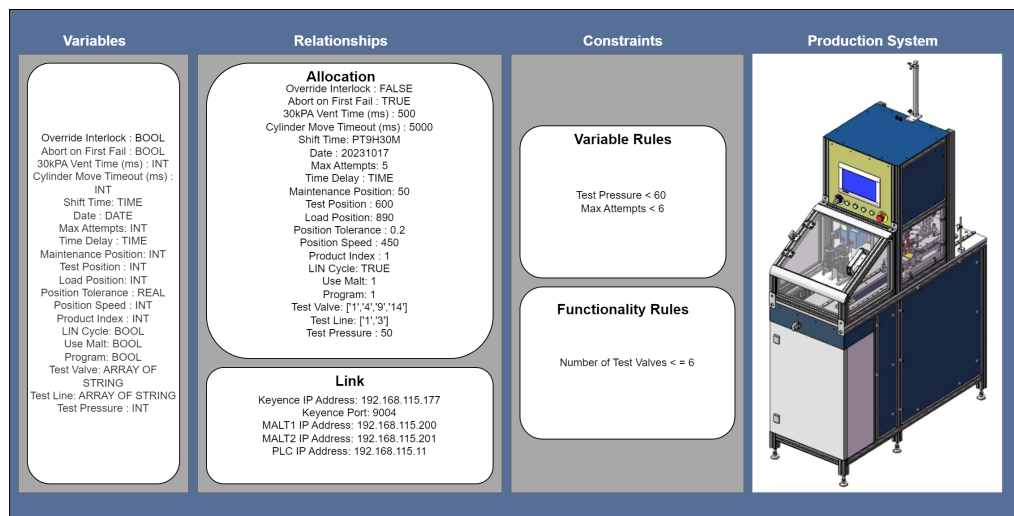


Figure 6.11: Module-based system application on a production system used to leak test products developed at an SME (TQC Ltd.). The configuration in terms of variables, relationships, and constraints is identified. The variable rules and functionality rules are presented.

**Scenario** In this example, a single workstation, developed at an SME (TQC Ltd.), is used to demonstrate the application of a module-based system. The test system as seen in figure 6.11 is used to test various products for an automobile customer. For the “leak test” functionality, the variables are identified. The values for the functionality and the product are allocated through relationships and updated to the production system and its inherent assets through links.

The MALT which is a leak test equipment inside the system is only capable of, in this machine, a test pressure of a maximum of 60kPa. There are

communication requirements for the product under test. The machine is capable of attempting to communicate with the product up to 5 times. The functionality rule is on the number of test valves to operate on at a single time. The test valves on the product need to be opened up in combinations to perform a leak test. Although the machine is capable of actuating more than 6 valves at a time, it would damage the product and therefore such a functionality rule is imposed.

**Key Insights from the Example** In the manufacturing example, the pressure limit of the testing equipment (60kPa) is a constraint that must be respected during configuration. Similarly, the number of valves that can be opened simultaneously is also a constraint. These limitations influence the possible configurations of the workstation to ensure its proper operation (i.e. leak testing).

The manufacturing example reinforces the concept of dynamic configuration. Since each module (MALT unit) represents a specific functionality like leak testing, adding or removing modules alters the capabilities of the system as a whole. In this case, adding more MALT modules will enhance the system's capability to test more complex volumes, as both MALTs can work together to test a complex product in a shorter time. This change necessitates updating the configuration to adapt to the new functionality requirements. Constraints play a role in ensuring the system's configuration leads to valid outcomes as per requirements.

The insights from the example can be instated as:

- **Constraints:** The manufacturing example highlights the importance of clearly defined constraints, both variable (e.g., pressure limit of the testing equipment) and functionality (e.g., the maximum num-

ber of valves to operate simultaneously) constraints.

- **Relationships:** The example shows how specific functionality (i.e. leak testing) dictates the required variables (e.g. test pressure, valve combinations).
- **Complexity:** Even a seemingly simple scenario like a single workstation leak testing demonstrates the complexity of configuration setting and the rules that govern it.

## 6.4 Conclusion

Module-based systems, presented in this research, represent a structured way to design intricate and adaptable systems. In this paradigm, systems are divided into smaller modules, each with its own set of rules, functionality, and variables. These modules can be customized, connected, and tweaked to meet specific needs, making them highly agile in the face of change.

This approach simplifies system design, upkeep, and expansion. Modules can be built, tested, and improved separately, which not only saves time but also cuts down on costs. Additionally, their modular nature makes it easy to incorporate third-party components, fostering innovation.

The power to configure modules dynamically and set rules for their interactions empowers module-based systems to be self-configuring, self-optimizing, and self-repairing. They can adapt to varying workloads, component glitches, or shifting goals, ensuring their dependability and efficiency. Module-based system will undergo testing and validation in subsequent chapters.

However, successfully implementing module-based systems requires care-

ful planning, taking into account module types, configuration parameters, variables, rules, and interconnections. The design process should align with the unique needs of the application. Moreover, ensuring smooth communication and synchronisation between modules is vital for achieving the desired system behaviour. A way of capturing configuration information in modules is represented in the next chapter.

---

## Chapter 7

# Standard Configuration Model for Production Systems in Manufacturing

**Contents**

---

7.1	Introduction . . . . .	<b>185</b>
7.2	Concept of Configuration in Production Systems . . . . .	<b>186</b>
7.3	Configuration Model . . . . .	<b>186</b>
7.3.1	Formulation of Configuration . . . . .	190
7.3.2	Configuration Operation and Instance . . . . .	193
7.4	Impact on Configuration . . . . .	<b>197</b>
7.4.1	Product and Functionality . . . . .	197
7.4.2	Capability . . . . .	199
7.4.3	Service . . . . .	200
7.5	Towards Capturing Configuration for Production Sys- tems . . . . .	<b>201</b>

---

**7.1 Introduction**

The work is divided into sections; Section 2 discusses applying the driven Configuration approach, established in the previous chapter, to the configuration of production systems. Conventional means of configuring production systems are explained in detail in the literature review chapter. Using the module-based approach, configuration in production systems is captured through a proposed configuration model. The configuration model is discussed in Section 3 with the impacts on the configuration being explored in Section 4. Section 5 deals with the application of the configuration model. This chapter is aimed at establishing theoretical aspects regarding configuration that may be manipulated to achieve self-configuration.

## 7.2 Concept of Configuration in Production Systems

Applying the Module Driven Configuration approach, a module is established where the configuration brings about functionality. This configuration needs to be modelled for the production system. To capture information on the configuration, per requirements of the module-based approach, is referred to as “Configuration Modelling”. Figure 7.1 illustrates the configuration model for production control in production systems derived from elements of module-driven configuration. This gives insight into the **variables** on which the configuration relies, the possible rules and constraints. The “Model Mapping” is the **allocation** and **linking** of the variables to the manufacturing asset (i.e. configurable object). The Model Mapping also encapsulates the variable and functionality rules that map the variables to product, functionality, capability, and services. The “Production Control” is the utilisation of the configuration to execute functionality. It may be carried out through PLC, agent system or any other controller.

## 7.3 Configuration Model

The production system is composed of entities or components that work together, executing functionality according to a directive (PLC event trigger etc.) to achieve a goal. These entities or components can be fixed, i.e., used as-is basis, or configured to meet the requirement of the process and product. These components are called Configurable Objects. Figure 7.2 establishes the scope of the configuration modelling in production systems consisting of configurable objects. It is assumed that configurable objects



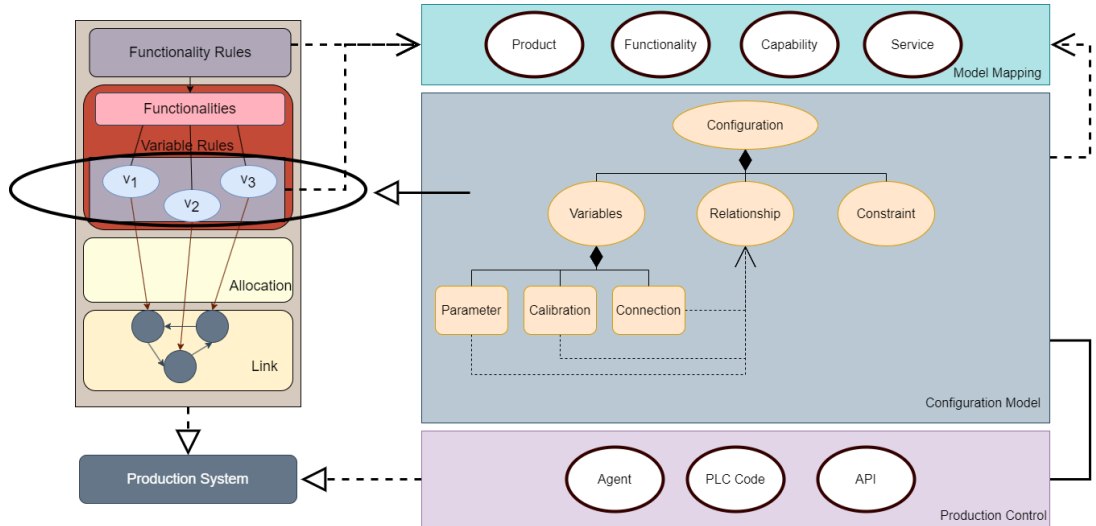


Figure 7.1: Envisioning the standard configuration model for production control in Production System. Three layers model is established (1) Configuration modelling layer that capture the information, (2) Mapping layer that maps the model with other submodels ; product, functionality, capability and services, (3) Production control layer that executes functionality in the production system with the configuration.

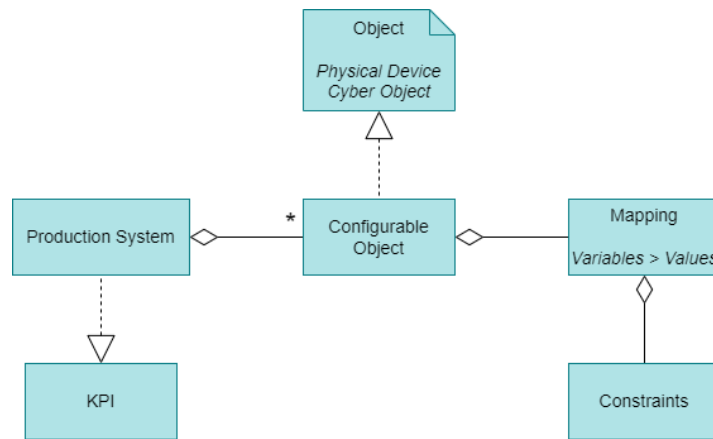


Figure 7.2: Establishing the scope of the configuration modelling in production systems. The production system consists of configurable objects (physical or cyber objects), and there exists a mapping of variables to values in those configurable objects subjected to constraints.

always introduce variables that need to be configured, dependent on functionality. All of these variables can be configured but may be restricted in a manner through variable and functionality rules.

A production system can have multiple configurable objects (i.e. in the form of modules) operating together, collectively forming production sys-

tem functionality. These configurable objects are accompanied by a mapping of variables to values, i.e. “Allocation”. These variables represent the aspects of the configurable objects that can be changed, i.e., can take suitable values.

The information model conceptualisation of configuration in a module is a representation of the variables that make up configuration, their relationships, constraints, operations, and expressions governing those and associated rules. The information model is used to capture data on configuration in the production system and give meaning to individual components. The general representation for configuration in the production system must provide both horizontal and vertical integration.

The configuration consists of three main ingredients;

1. A production system consists of configurable objects representing manufacturing assets, where each asset introduces a functionality. A manufacturing aggregate that has the capability to be configured in a production system is known as a Configurable Object represented by a module.
2. Relationship that exists between these configurable objects, represented by functionality rules.
3. Constraint or group of constraints acting on the configurable objects that make up the production system, represented by variable rules.

The manufacturing aggregate that is able to adapt (i.e. configured) must be defined as an independent entity, i.e. modular, with a unique identifier. These objects derive an abstract representation for the complexity of the whole production system and define a sense of granularity. The relationships capture the manner in which these configurable objects are linked

to each other. Constraints encode the structural and functional rules that govern the configurable objects, their relationships, and the execution of functionality in a production system. The relationships, rules, and constraints are captured through variable and functional rules (Chapter 6).

**Configurable Objects:** A group of components or entities come together to achieve a purpose of production. These groups of components are referred to as configurable objects, bringing about functionality. These components may be able to adapt as per product and process requirements through rules, therefore being Configurable Objects (G). The configuration settings, for a Configurable Object, on the production system are dependent on the features of the product, the characteristic properties of the environment where the equipment is kept, the connected processes, and current business priorities. The configuration model for the configurable object contains a description of all information necessary to define the object, that is the logical and structural representation of the data that is necessary to define each component.

**Relationship:** In production systems, relationships represent the linkages between configurable objects. This is captured in the functionality rules. For e.g., a relationship exists between a test equipment and the fixture.

**Constraint:** These are the limitations imposed on the configurable objects and their associated relationships. These can be product limitations, process criteria or business objective. These directly affect the performance of the production system. Constraints on the variables are provided by variable rules.

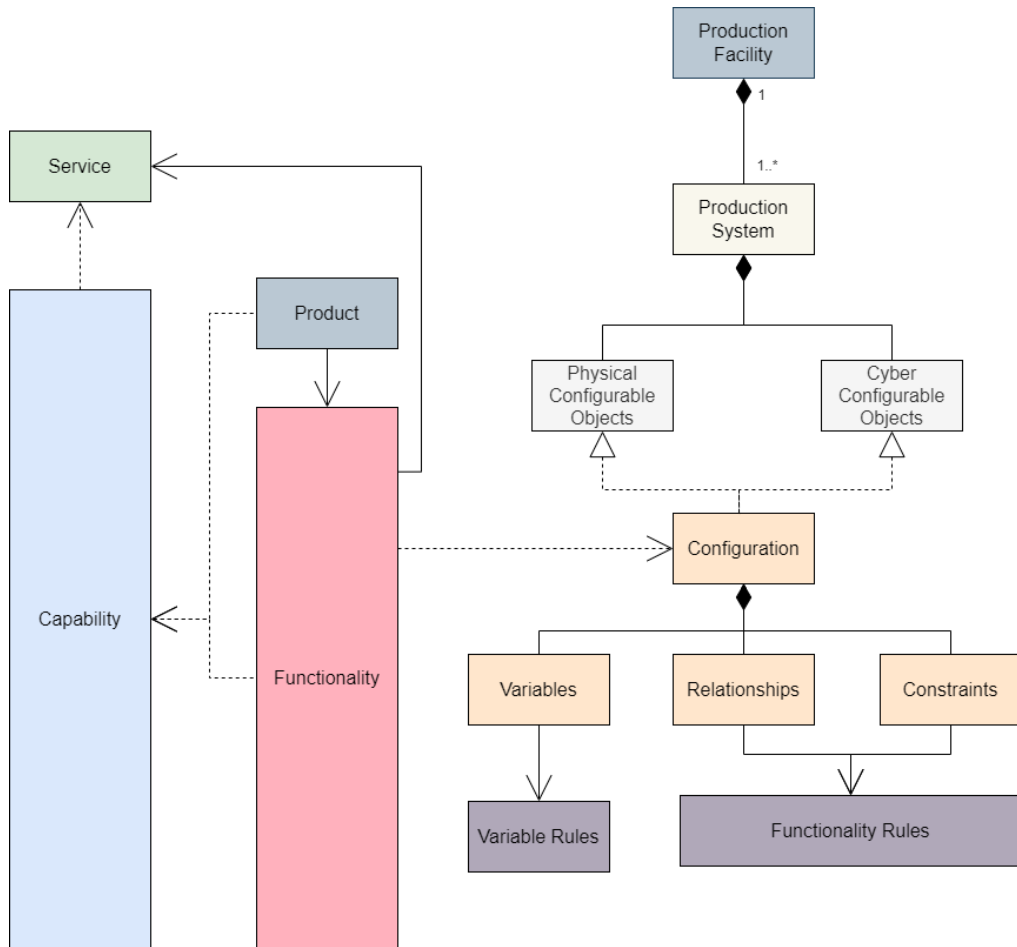


Figure 7.3: The UML representation of Configuration in Production system. Functionality presents configuration requirements, depending on capability, product and service requirements. Variable and Functionality rules impact the configuration. (An explanation of the arrows in the UML diagram can be found in figure 7.4)

The following sections elaborate on the model of elements in figure 7.3.

The purpose is to understand configuration in production systems and use

that to automate the configuration process.

### 7.3.1 Formulation of Configuration

Configuration, in a production system, is composed of configurable objects, represented in the form of modules that include functionality-dependent variables, the relationship between these variables and their associated con-

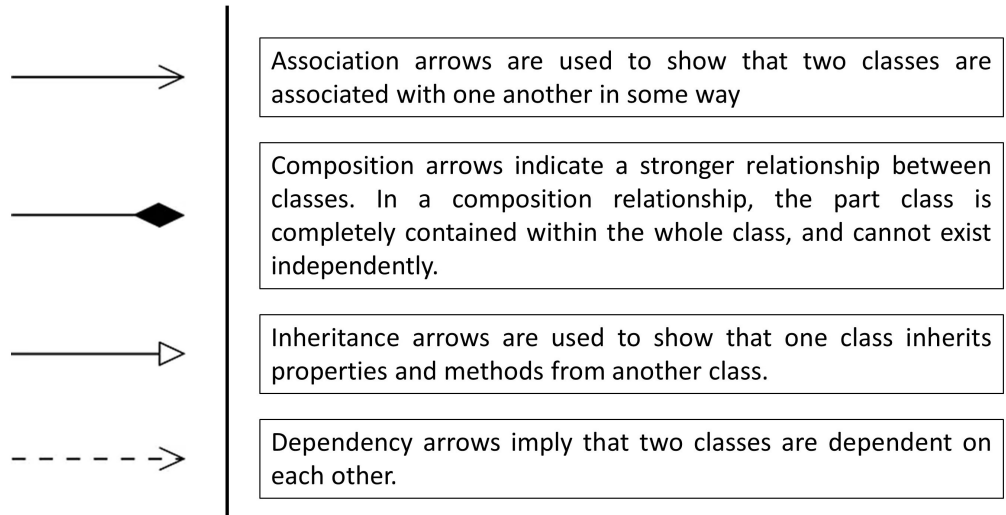


Figure 7.4: The arrows in UML represent relationships between classes. An explanation of UML arrows in the research is illustrated.

straints these are subjected to. The set of Configurable Objects (CO) is given as;

$$CO = \{CO_1, CO_2, \dots, CO_n\} \quad (7.1)$$

The configurable object can be defined based on variables.

$$V_{CO_i} = \{V_1, V_2, \dots, V_k\} \quad (7.2)$$

Relations exist in the form of ordered pairs, of the form  $(CO_i, CO_j)$  where for  $CO_j$  is only possible if  $CO_i$  is present. This presents a condition of dependency between configurable objects, to be more elaborative in their variables. Therefore, for the  $CO_i$  and  $CO_j$  case, the relationship between variables in configurable objects can be written as  $(V_{CO_i}, V_{CO_j})$ . This kind of dependency is mentioned in the functionality rules of the module. These ordered pairs are not only limited to unitary dependence, which means there can be multiple constitutive configurable object dependencies. This can be represented by  $(V_{CO_i}, V_{CO_{j_1}}, V_{CO_{j_2}})$ . The relationship between configurable objects can be written as ;

$$Rel_{V_{CO}} = (V_{CO_i}, V_{CO_j}) \quad (7.3)$$

Constraints are defined by the user to the configuration for the functionality. Constraints may be influenced by process, product, or objective. In a configuration in the production system, the constraints act on variables and their relationship, limiting them in values. The capability of the system can also impact the constraints, which in turn limit the variables and their relationships.

Configuration ( $C$ ) in a production system is a combination of the configurable objects, their relationships, and the constraints.

$$C \in CO \times Rel_{CO} \times Constraint \quad (7.4)$$

Since configurable objects consist of variables that are dependent on functionality having relationships and constraints, then the expression can be expanded;

$$C \in PV_{CO} \times Rel_{V_{CO}} \times Constraint_{V_{CO}} \quad (7.5)$$

Constraints are driven by expressions and can be presented in variable and functionality rules. Configuration has a dynamic nature because a change in any product, process, capability, or service may cause an impact on configuration (i.e. variables, relationships, and constraints).

### 7.3.2 Configuration Operation and Instance

Operations are those representative activities that are used to describe the configuration, i.e. setting up the configuration for the functionality and all aspects that deal with executing a functionality with that configuration in a production system. Since configuration in production systems is being modelled, this section deals with operation in relation to configuration change, called Configuration Operation.

Such Operations can be divided into Abstract and Physical Operations. At the CO level for configuration change, the operations that do not involve physical resources like cloud service-matching and part change etc. are **Abstract Operations**. **Physical Operations** are related to the behaviour of the CO like pressure setting, stabilisation setting etc.

In a leak test system for testing leakages in cylinder volumes, the operation involved in setting up test parameters like test pressure and measure time are Physical Operations. The leak test system will also be configured to store the test data and the results on a local server. Such operations that do not impact the behaviour of CO (leak test system) and will be involved in enabling this communication for the system are Abstract Operations.

$$OP_{CO} = OP_{Abs} \mid OP_{Phy} \quad (7.6)$$

The Physical Operations can be further classified into internal and external physical operations based on their roles. **Internal physical operations** are related to the preparation action for physical execution (i.e. functionality), and **external physical operations** are those operations constituting

actions that impact on state (i.e. execute the functionality).

$$OP_{Phy} = OP_{PhyI} \mid OP_{PhyE} \quad (7.7)$$

Figure 7.5 gives a brief overview of the configuration operation and its types. A suitable description of these operations and details is presented in the following sections. This is used in the Module Driven Configuration (see Chapter 6) approach to capture information for variable and functionality rules. A mechanism for formulating expressions is defined as a means of combining them for developing complex expressions. While utilising the established terminologies, the reserved keyword “abstract” is used to describe resource-independent configuration operations. Whereas, “physical” operations are for resource-dependent configuration operations.

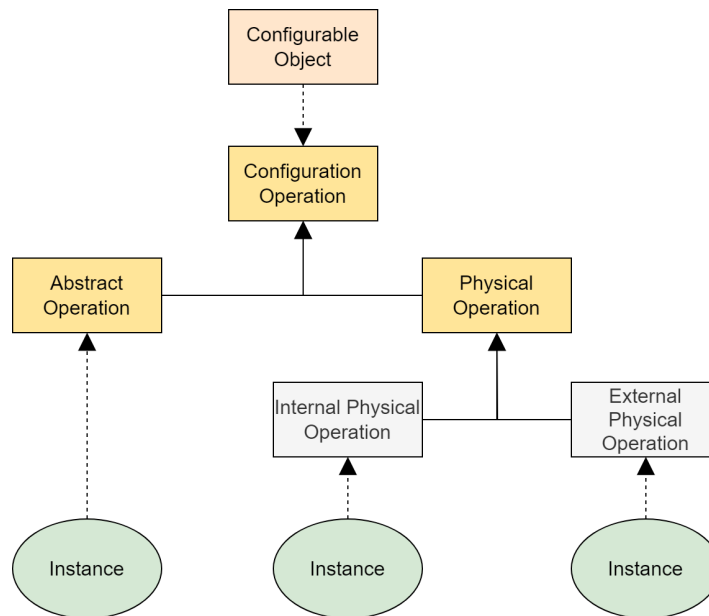


Figure 7.5: The hierarchy of configuration operation. The dotted lines represent the instances which carry out the specific type of operation. Abstract instances carry out those abstract operations governed by a production plan, while internal physical and external physical operation are carried out by the production system instance. For a successful configuration operation abstract operation, internal physical operation and external physical operation should have matching capabilities and requirements.



#### Configuration Instance

As established in the previous sections, configuration modelling is based on variables of the configurable objects in a production system, their relationships, and the constraints that impact them. Configuration in production systems can be best understood as;

*Configuration in production systems is a representation of a fixed encapsulation of configurable objects, and by this context available variables, that the system could take depending on relationships and constraints.*

Modelling each individual variable that is configurable, gives the configuration for the production system. In order to satisfy the relationship and constraint requirements for the configuration, the following conditions must be met:

1. Capability is present in configurable object (i.e. manufacturing asset) to realise the configuration. For e.g. capability to operate on a certain pressure value.
2. Realisation must be present in the configurable object. It means that the production system should be context-aware, and able to realise a change in variables as instructed.
3. A configuration could be formed and maintained. Once the variables are changed, they must be updated on the production system. This sets the requirement for allocation and linkage in the system.

Configuration modelling is only possible for those variables that possess the capability to be changed. Along with this, the configurable object must possess the ability to realise changes, i.e. if a certain parametric

value is changed in the settings, the configurable object must be able to adapt as per that value.

A configurable object must be synchronous with the changes happening in itself. Effectively, the configurable object must be able to detect and maintain changes. For instance, an additional component like a fixture is substituted on the production system, so the configurable object must be able to identify a combination with the fixture and maintain it. In the same aspect, the configurable object must be able to realise the arrangement with other such configurable objects and link up with them to execute a functionality.

A configuration operation involves multiple steps to achieve the desired configuration for executing functionality. The abstract, internal physical and external physical operations define the input to the configurable object, any aspect added/removed, and the output from the configurable object. Through this, the configuration operation defines the associated activities that deal with changes in configuration.

#### **Transitioning to Configuration**

From the above, the configuration operation guides through configuring a system. Effective distributed control in the production system (Mcfarlane and Giannikas (2013)) is possible when the configuration operations are executed for the configuration change in a defined manner.

Let's assume that  $P_{\text{sin}}$  are the activities involved with input,  $P_{\text{add}}$  represents any add-on to the configuration operation and  $P_{\text{sout}}$  gives the output activities. For functionality to be executed on a production system, the variables must be configured. This is called transition to a configuration.

$$(P_{sin}, P_{add}, P_{sout}) \rightarrow V \quad (7.8)$$

The above equation represents a mapping of variables to values  $V$  in a configurable object. This represents the relationship for the variable as per the definition of configuration in production systems. It can be a single value or a range of values specified.

## 7.4 Impact on Configuration

The configuration for any configurable object holds under a set of conditions/constraints. These constraints impact the configuration of the object, resulting in a change in configuration settings. Product, functionality, capability, and service are the main influencers that impact configuration.

These aspects are related and discussed in detail in this section, with a focus on their relative impact on configuration. Abstraction is the condition in which they impact and a generalised representation is developed and presented.

### 7.4.1 Product and Functionality

In terms of product and functionality in a production system, certain requirements must be fulfilled, as proposed in this research, so that they can be linked up with the configuration. These involve the complete information required to represent the product, including its bill of materials, its sub-assemblies and parts. These requirements for a production system are stated as follows:

1. The production system should contain a specified sequence of functionalities that need to be performed from the start to the end of the cycle. For e.g. the whole test sequence for a product.
2. Requirement for each functionality must be defined for each product. The dependency from the production system on product configuration must be accurately laid out.
3. The module that performs functionality in a production system must contain information about identification. For e.g., a product must be tested at a certain pressure than it can only be processed on the module that can offer functionality at that pressure.
4. The production system must have the ability to integrate and act on some objective during the functionality sequence. This is required to distinguish among modules like a test station module but be different from a loading module in identification.

Similarly, the requirements for the functionality can be elaborated as:

1. Machine information should be consistent (i.e. remains static) for the duration of execution of the functionality.
2. Contextual awareness of the functionality should be present in a production system. The system has the capability to infer meaning from context.
3. Functionality should be consistent with production system requirements.

Functionality  $F$  expressions for product guides and influence the configu-

ration. They can lead towards different configurations through transition.

$$C_i \xrightarrow{E_F} C_j \tag{7.9}$$

This representation of functionality captures the transition of a configuration from one to another, guided by an expression. This description can be used to relate functionality to configuration and criteria.

$$F \in PF_C \tag{7.10}$$

Figure 7.6 provides a simple representation of the product.

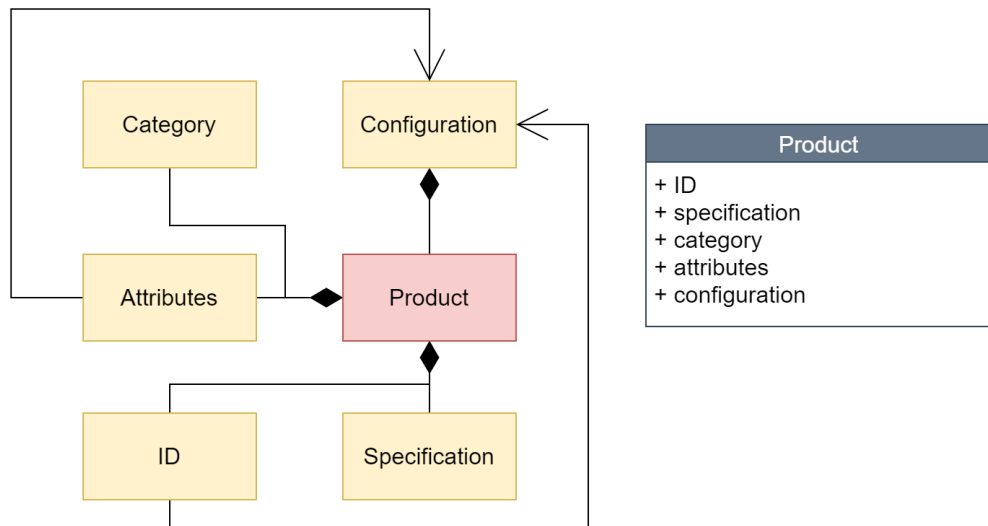


Figure 7.6: The product contains an ID, specification, and configuration. The attributes can possess values for a property. For classifying the product, categorisation, and specification description may be used.

### 7.4.2 Capability

Capability is best understood as the skill of the equipment to execute functionality. Capability is a generalised representation of the tools, resources,

and skills required to execute a certain functionality. Capability is guided by product specifications and machine requirements. Skill is dynamic for a resource in a production system, and is dependent on the state of the system, dependency on product and functionality, and transition from one product state to another. Figure 7.7 gives a more detailed view of the skill and capability in the production system. So, in this aspect, a resource having a capability may not have the skills to execute functionality under certain conditions. E.g., the capability may be present in the system to pressure test a product but not the skill to execute it as the product features are not fulfilling the fixture requirement currently in which the product may be clamped .

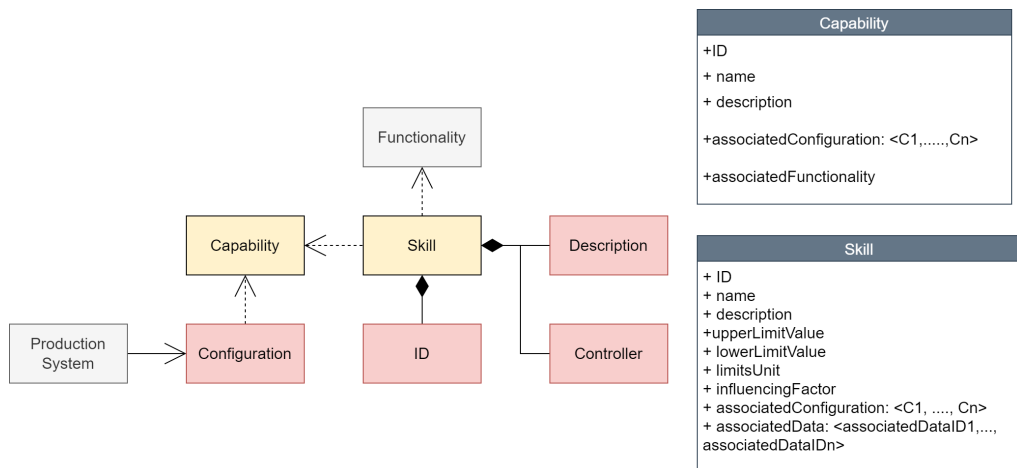


Figure 7.7: A simple capability and skill representation . The capability is based on skill and drives its properties from configuration. Skill is a more descriptive concept containing unique identification, description and assigned controller. Functionality influences the capability directly as functionality generates certain dependencies that must be fulfilled for it to execute.

### 7.4.3 Service

Services can be integrated with the configuration model as they directly impact configuration. However, for this to be possible, it is necessary that these services can also be configured. These services are mainly linked

to product and functionality requirements. Another aspect of this is the requirement of configuration to be dynamic, as various services may be required during functionality execution. Figure 7.8 shows a brief overview of the service and its link with product and functionality.

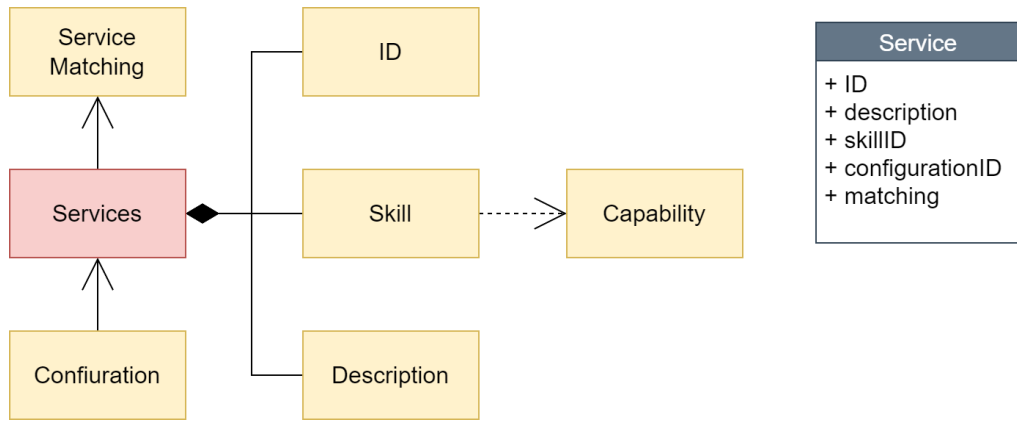


Figure 7.8: A simple service representation. The service requirement is dependent on configuration. Service matching is necessary to determine the compatibility of the service with requirements. Each offered service has an Id, Skill (i.e. encapsulating capability), and a service description.

## 7.5 Towards Capturing Configuration for Production Systems

In summary, this chapter explores the concept of configuration in production systems, breaks it down to a certain level of granularity, and discusses the representation for capturing configuration in production systems. Established means of configuration in manufacturing are explored and the need for a standard model for it is established.

Configuration in a production system consists of configurable objects, their relationships and their associated constraints (see section 7.3). Configuration operations are defined as activities used to describe configurations and aspects that deal with changes in configuration within configurable objects

## 7.5. TOWARDS CAPTURING CONFIGURATION FOR PRODUCTION SYSTEMS

(see section 7.3.2). The abstract and physical configuration operation concept is presented. Configuration transitions are defined in a configurable object in the production system.

The impact of product, functionality, capability, skill, and service on the configuration of a production system in manufacturing is also discussed. This forms the building block for capturing configuration for any production system for the Module Driven Configuration Model. A representation of configuration in production systems with details of individual components is illustrated in figure 7.9.

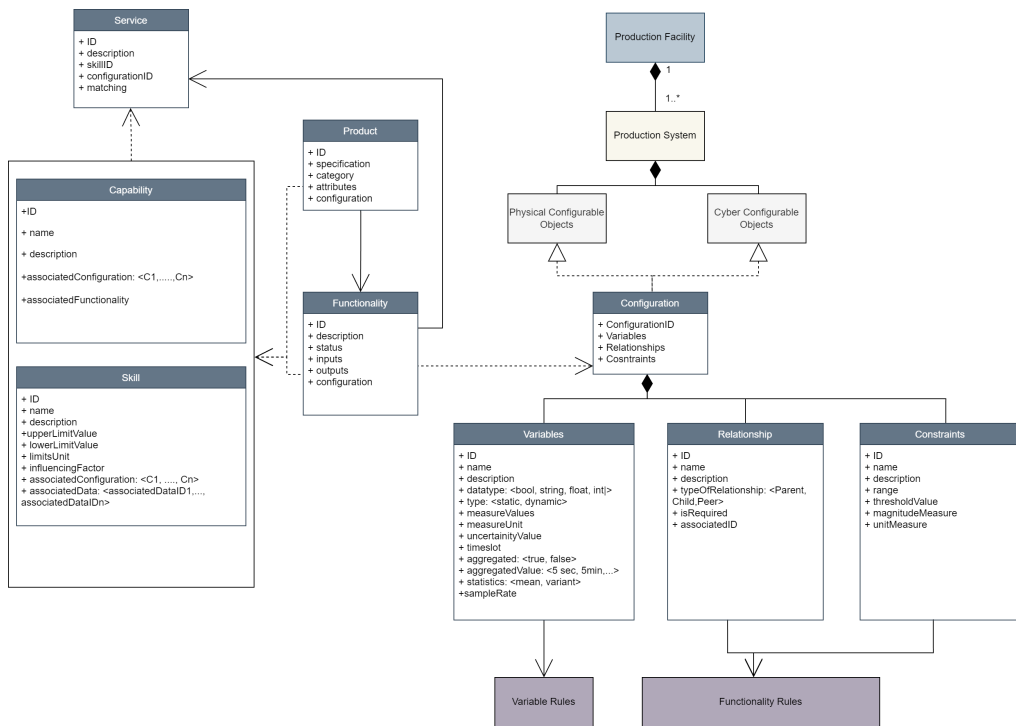


Figure 7.9: The UML representation of Configuration in Production system. UML representation of individual components is detailed, which is established in this chapter.



---

## Chapter 8

### Self-configuring Production

### System: An Adaptation

### Strategy

---

## Contents

---

8.1	Introduction . . . . .	<b>205</b>
8.2	Overview of the Adaptation Strategy . . . . .	<b>206</b>
8.2.1	Implementation Requirements . . . . .	209
8.2.2	Structure of Adaptation Strategy . . . . .	212
8.3	Capturing Configuration Module Information . . . . .	<b>215</b>
8.4	Deployment of Engine for Capturing Information . . . . .	<b>216</b>
8.4.1	Connecting Module-Driven Configuration with the CAEX Engine . . . . .	216
8.4.2	Linking to Asset Administration Shells (AAS) . . . . .	217
8.4.3	Dynamic Updating of Information . . . . .	218
8.4.4	Facilitating Communication with Lower Layers . . . . .	218
8.4.5	Understanding Asset Administration Shells (AAS) . . . . .	219
8.5	State Chart to State Machine for Functionality Rep- resentation . . . . .	<b>220</b>
8.5.1	State Charts to State Machine for Self-configuration . . . . .	220
8.5.2	Achieving Self-Configuration for Functionality Stat Machines defined as State Machine Be- haviours . . . . .	221
8.5.3	Configuring State-Specific Actions: . . . . .	222
8.5.4	Tracking State Changes: . . . . .	222
8.6	Machine Learning for Self-Configuration . . . . .	<b>223</b>
8.6.1	Enhancing Machine Learning in the Produc- tion System for Production Systems: . . . . .	223
8.6.2	Workflows for Streamlined Machine Learning Integration: . . . . .	223

8.6.3	Proposed Workflow Overview: . . . . .	224
8.6.4	Predicting Configuration Values: . . . . .	224
8.7	Multi-Agent System Integration . . . . .	<b>227</b>
8.7.1	Realisation of Self-configuration in Production Systems . . . . .	228
8.7.2	Using Expressions for Self-Configuration in Pro- duction System . . . . .	231
8.7.3	Industrial Application for Agents . . . . .	234
8.8	Real Time Control . . . . .	<b>235</b>
8.8.1	Enhancing Adaptation Strategy with Real-Time Control and Hardware Abstraction: . . . . .	235
8.8.2	Hardware Abstraction for Self-Configuration: .	236
8.8.3	Real-Time Control and Hardware Abstraction Interaction: . . . . .	236
8.8.4	Service Actors and Their Role: . . . . .	237
8.9	Simulation . . . . .	<b>238</b>
8.9.1	Simulation for Self-Configuration Module: . . .	238
8.9.2	Detailed Simulation Process: . . . . .	239
8.9.3	Intercommunication within the Self-Configuration Module: . . . . .	239
8.10	Conclusion . . . . .	<b>240</b>

---

## 8.1 Introduction

It is assumed that a production system consists of multiple configurable objects (Chapter 6), in this sense a way of representing such information

for all of these objects is needed using the information model (chapter7) discussed. As this information is captured, an approach to achieve the best configuration under conditions (e.g. KPIs, priority) using machine learning pipelines is presented in this chapter.

The structure and principles of the Adaptation Strategy are laid out. The information-capturing mechanism in the strategy is explained with a link to the configuration module defined. Functionality representation by state machine through state chart is elaborated. Machine Learning for self-configuration is generalised. Achieving self-configuration through agent-based system integration is demonstrated. Real-time control and simulation of the production system through the strategy are expanded.

Section 2 covers the adaptation strategy. Section 3 and Section 4 introduce the engine for capturing information and discuss the deployment of the engine. Section 5 elaborates on functionality representation. Section 6 expands on Machine Learning for Self-Configuration. Section 7 deals with enabling self-configuration through multi-agent system integration. Sections 8 and 9 expand on real-time control and simulation integration.

## **8.2 Overview of the Adaptation Strategy**

In modern manufacturing, the ability to adapt rapidly to changing production needs and conditions is important to meet business objectives. Achieving such adaptability, often termed “self-configuration”, requires a systematic strategy. This section introduces a self-configuration adaptation strategy for production systems.

**The Need for a Self-Configuration Strategy** A production system is a complex web of interconnected components, each serving a specific purpose. To achieve a high level of adaptability, it is essential to have a self-configuration strategy. It provides a structured approach, using previously established module and module-driven configuration concepts.

The purpose of this strategy is to achieve complete control over the production process. In other words, it allows for a high level of control at different granularity levels in the configuration of production systems. This means that each individual component can be configured independently to optimize the overall system's performance. For instance, changes in the temperature can affect how the camera is calibrated.

**Three Layers of Adaptation Strategy** The self-configuration adaptation strategy, in this research, consists of three fundamental layers:

- **Production System Coordination Layer:** The layer that captures the module functionality and can be tailored to capture different functionalities of different modules.
- **Production System Runtime:** This layer provides the core features derived from module-based system and configuration modelling, as well as maintaining a runtime during deployment (Chapter 6).
- **Production System Drivers and Control:** The layer where the functionality is executed as per changed configuration.

**Relating Adaptation Strategy to Level Based Classification** A mapping of the level-based classification of the production system is carried out and shown in this figure 8.1. It presents the components of the

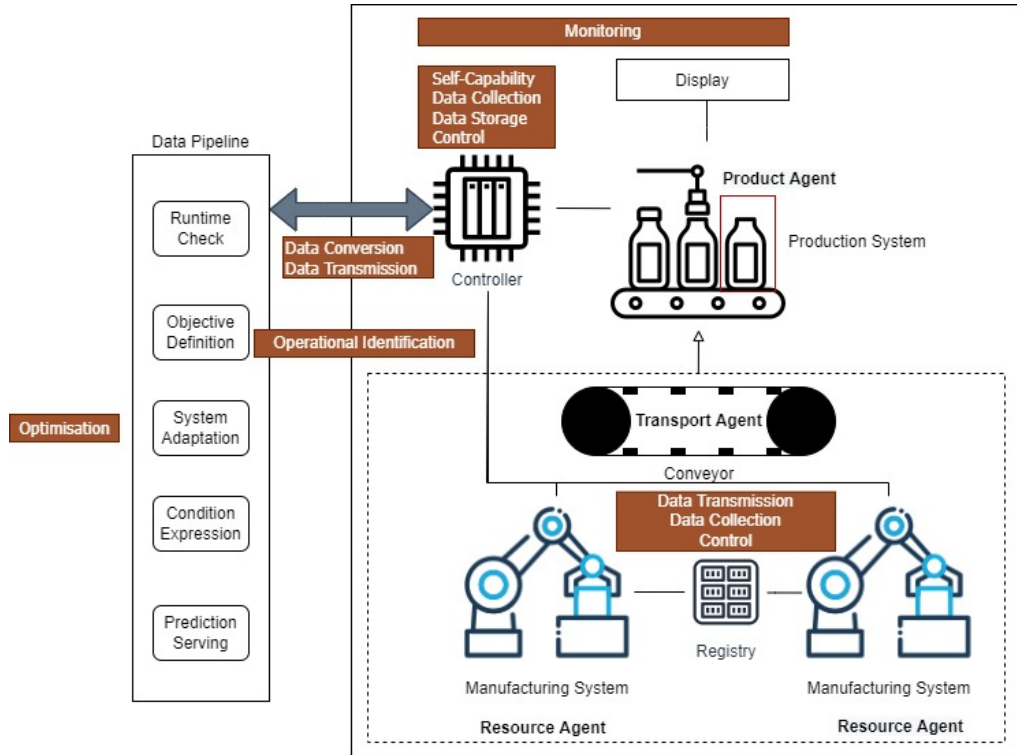


Figure 8.1: Mapping Level Based Classification for Self-Configuration to Production system components. The classification features are mapped to the system to provide the focus for manufacturers for improvement.

self-configuration framework and their interactions. The area where each feature of the level-based classification must be targeted to achieve complete self-configuration is presented.

The controller of the production system hosts most of these features such as self-capability, data collection, data storage, and control. The controller is the gateway to the production system, therefore, for self-configuration, most of the features should target this component. For self-configuration, a means of converting and transmitting the collected data is needed, so the feature of data conversion and data transmission is conceptualised to work between the controller and data pipeline. Operational identification for real-time operational awareness and direction targets the data pipeline and controller component of the framework. The components of the data pipeline are detailed in table 8.1.

Going into detail of production system, by definition established before, will consist of manufacturing assets and services that coordinate to produce products. At the system level, the features of data transmission, data collection and control need to be present to ensure functionality with the controller of the production system.

Table 8.1: Data Pipeline Component Description for Production System

Component	Description
Runtime Check	Check the manufacturing system against the relationship, constraint, requirements (product and process), and capability conditions. This gives additional depth to operational execution. It provides a check on the current state of the production system.
Objective Definition	A target corresponding to cost, time or any performance parameter in terms of guiding parameter is set for the product being produced.
System Adaptation	The system is configured as per the target guiding parameter by coordinating with the prediction serving component.
Operational and Conditional Expression	Define the operation skill requirements and the manner in which these skills need to be executed. More detail on this and the approach to formulate is presented in work by Rehman et al. (2021a).
Prediction Serving	This component of the data pipeline acts as a gateway for the ML model deployed (previously trained). This is used to serve the requests by the system adaptation component for self-configuration.

### 8.2.1 Implementation Requirements

The adaptation strategy, which forms the foundation of the strategy to achieve self-configuration in module-driven production systems, is guided by a set of implementation requirements that map to design principles of module-driven configuration (Chapter 6) and features of level-based classification (Chapter 5). Figure 8.2 illustrates the design principles detailed

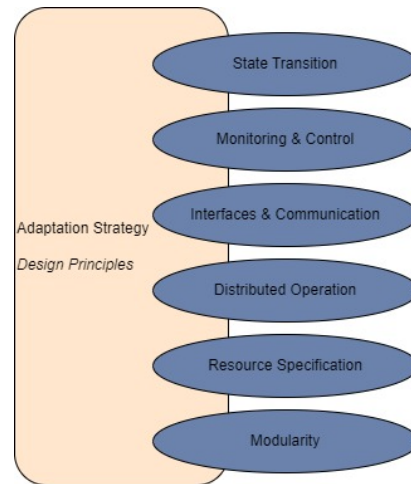


Figure 8.2: Design Principles of Adaptation Strategy.

as:

- **State Transition:** The ability to capture and manage state information is a fundamental requirement. Transitioning from one state to another represents module functionality. In the proposed architecture, this is carried out through State Charts. The State Chart are captured into State Machine behaviour. These behaviours can be directed through Agent-based or real-time control.

Related to the design principles of *Exactness* and *Enforcement* of module-driven configuration approach and features of *Operational identification* and *Control* of level-based classification.

- **Monitoring and Control:** A key component of the adaptation strategy is the incorporation of monitoring and control mechanisms. Self-configuration hinges on the system's context awareness, which necessitates robust monitoring services. Monitoring services must be integrated with the strategy and ensure proper control through control services.

Related to the design principle of *Runtime* of module-driven configuration approach and features of *Monitoring* and *Control* of level-based



classification.

- **Interfaces and Communication:** Interfaces must be present for each component in the adaptation strategy. This will assist in communicating between different components of each layer, and also provide a means of communication between layers themselves.

Related to the design principle of *Runtime* of module-driven configuration approach and features of *self-capability* and *data management* of level-based classification.

- **Distributed Operation:** The adaptation strategy should promote distributed operations. The proposed strategy works well in this regard, as it can be deployed at the module level in the production system.

Related to the design principle of *Runtime* of module-driven configuration approach and features of *Optimisation* and *Operational identification* of level-based classification.

- **Resource Specification:** Resources for the production operation need to be completely defined and linked to the Adaptation strategy. Resource linking is proposed at the low-level layer, where state charts at the top level and engines in the middle layer can affect the resource. Related to the design principles of *Exactness* of module-driven configuration approach and features of *Operational identification* and *Data management* of level-based classification.

- **Modularity:** The components of each layer should be kept modular. This assists in the possibility of applying the strategy to a wide range of production applications.

Related to the design principles of *Separation* and *Generality* of module-driven configuration approach and features of *Operational*

*identification* and *Self-capability* of level-based classification.

### 8.2.2 Structure of Adaptation Strategy

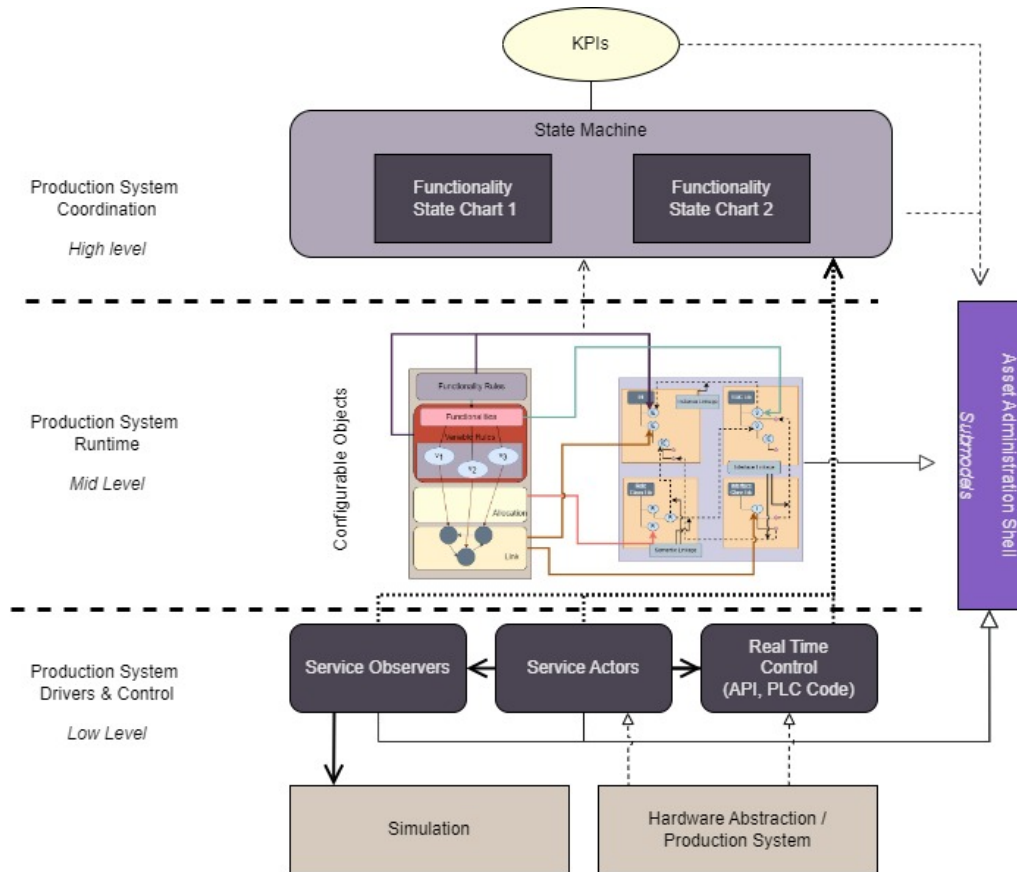


Figure 8.3: The proposed Adaptation Strategy integrated with CAEX Engine. CAEX stands for Computer Aided Engineering eXchange, a standard used to describe the structure of manufacturing data. The CAEX Engine aids the management and exchange of engineering data in a standardized format (more explained in a later section). It consists of three layers: Production System Coordination (High-Level), Production System Runtime (Mid-Level), and Production System Drivers & Control (Low Level).

The structure of the adaptation strategy is illustrated in figure 8.3 by integrating three key layers: the Production System Coordination (High-Level), Production System Runtime (Mid-Level), and Production System Drivers & Control (Low-Level).

### The Three Layers of the Adaptation Strategy

1. **Production System Runtime Layer:** This layer consists of information on the runtime of the production system. The runtime embodies a practical way of monitoring and verifying of production system state. This means that runtime captures all information about production system configuration, operation, state, and functionality. It is proposed that in this research, the runtime layer possess the following characteristics;

- **Heterogeneity and Mobility:** This layer should be able to represent different types of information and provide a management interface.
- **Relationship and Dependency:** Relationships exist between different information components. This layer must capture these relationships to ensure the correct behaviour of applications.
- **Contextualisation:** Complete information of the production system state must be captured and managed by linked management systems.
- **Information Quality:** The layer should be able to maintain information at the same consistency to support reasoning, i.e. the same information quality should be available throughout the operation.
- **Reasoning:** The runtime layer should have the capability to support the execution of functionality. Also, the layer should be able to reason while taking into account relationships and contextual information.
- **Modelling Formalism:** The layer is able to translate information from real-world concepts to modelling constructs, and

contextual information.

- **Efficient Access:** The layer must be able to provide easy and efficient access to production system information through access to data objects.

This layer based on the characteristics captures information about the configuration of the production the configuration operation, the state, and the functionality. This assists in the correct execution of functionality and improves the execution by identification and prediction of changes, constraints, or errors.

In the context of the Adaptation Structure, the ‘runtime’ is the layer that embodies the real-time operation and monitoring of the production system. It’s a dynamic layer that ensures real-time awareness of the system’s status.

2. **Production System Coordination Layer:** This layer encapsulates the manner of functionality execution in the form of state charts. Each state chart represents a system’s functionality. These state charts can be encapsulated into the behaviours of a state machine. This layer deals with the coordination of the other two layers (i.e. low and mid-level) via these state machines. Planning is carried out at this level, as behaviours in the state machine can be manipulated to match customer and functionality needs (e.g. KPIs).
3. **Production System Drivers & Control Layer:** This layer deals with hardware drivers, hardware control and hardware abstraction. The layer incorporates real-time control through a production system-specific API or PLC event trigger. The RTC can trigger functionalities through a linked production system. This layer also contains Service Actors and Service Observers that provide an interface with

third-party components (e.g. cloud services). The layer also interacts with the simulation service through state observers and changes the production system functionality according to simulation goals.

## 8.3 Capturing Configuration Module Information

The module and operational semantics for production systems in manufacturing can be represented by a model that captures the information dynamically. The model must be vendor-neutral to represent data and elements in a production system, i.e., the manufacturing system and its resources.

The information, represented in the model, follows an object-oriented paradigm, making it feasible to define the module as an object. So, the information model may contain objects that encapsulate both physical and digital components containing different aspects as data objects.

Objects represented in this model may form a hierarchy and shows composition and aggregation. Also, each object contains attributes that may describe properties related to geometry, operations, and behaviour among others.

The CAEX (Computer Aided Engineering Exchange) standard has been used for modelling information about production systems in manufacturing (Drath (2012)). The information on the standard is in the appendix.

## 8.4 Deployment of Engine for Capturing Information

This section details the relationship of Module Driven Configuration with the CAEX engine. Then the CAEX engine elements are mapped to the asset administration shell providing a means of integrating AAS with the concept of module driven configuration.

### 8.4.1 Connecting Module-Driven Configuration with the CAEX Engine

The module-driven configuration approach can be related to the developed CAEX Engine for capturing configuration in production systems. The individual components of a module can be mapped to the CAEX engine components. As detailed in the appendix, the CAEX engine has four main components *Instance Hierarchy (IH)*, *System Unit Class (SUC)*, *Role Class Library (R)* and *Interface Class Library (I)*. Figure 8.4 illustrates the mapping between a module and the engine components.

As per figure 8.4, the configuration module is mapped to the engine components. The functionality of the configurable object, represented in terms of variables, is housed in the Instance Hierarchy in the form of internal elements(IE). Each variable is represented in one IE in the instance hierarchy (IH). IE contains the variable name, its description, its value, and all other properties necessary for detailing the variable. The IE is instantiated as an instance of the system unit class object.

The configurable object is represented in the system unit class library and then instantiated as an IE in the instance hierarchy. The allocation in

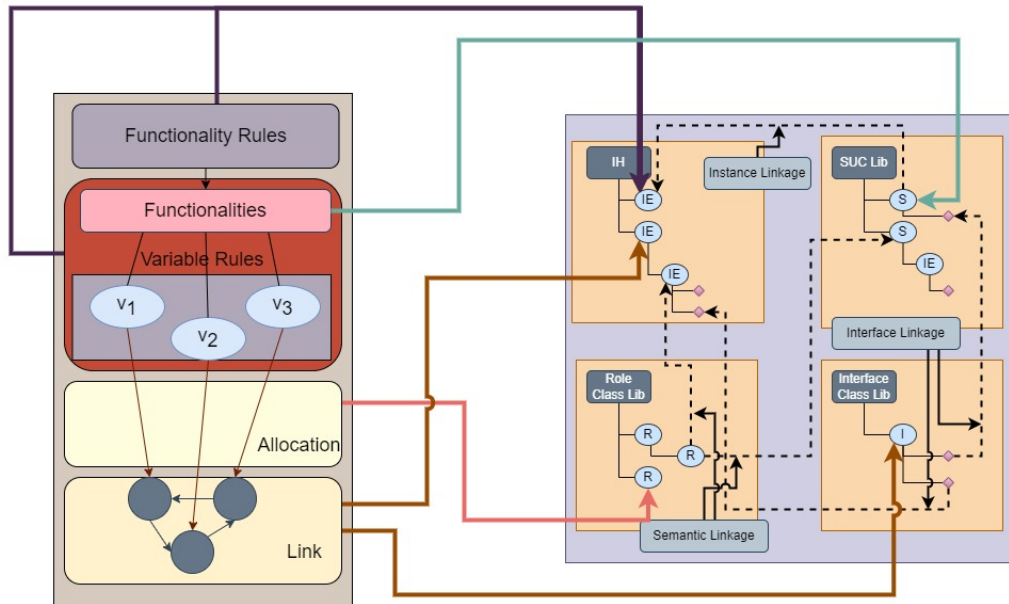


Figure 8.4: A configurable object module is mapped to CAEX engine components. The functionalities are mapped to the system unit class object instantiated in terms of variables as internal elements within an instance hierarchy with the corresponding rules. The allocation and links are mapped to role class library and interface class library respectively.

the module where the values are assigned to the variables is mapped to role class library elements. The information about linking the configured variables to the production system is stored in the interface class library and the IE for each variable.

### 8.4.2 Linking to Asset Administration Shells (AAS)

Previously, an information capture model was presented for the configuration module (Chapter 6). The AAS for a configurable module consists of variables, their relationships, and constraints. The relationships are represented in the module using “Allocation” and “Link”. The variables in the AAS are connected in terms of functionality. The constraints in the AAS are represented in the functionality and variable rules. Since the AAS and configuration module are linked together, in the same manner, the engine assists in deploying an AAS representation.

### 8.4.3 Dynamic Updating of Information

As the AAS is updated, the configuration module is updated and is translated to the update in engine elements. The capturing of the configuration of a configurable object in this manner becomes dynamic, as a change in AAS is realised with a change in the engine elements. Figure 8.5 gives an example of the change happening in AAS and the way it affects the engine elements through its mapping.

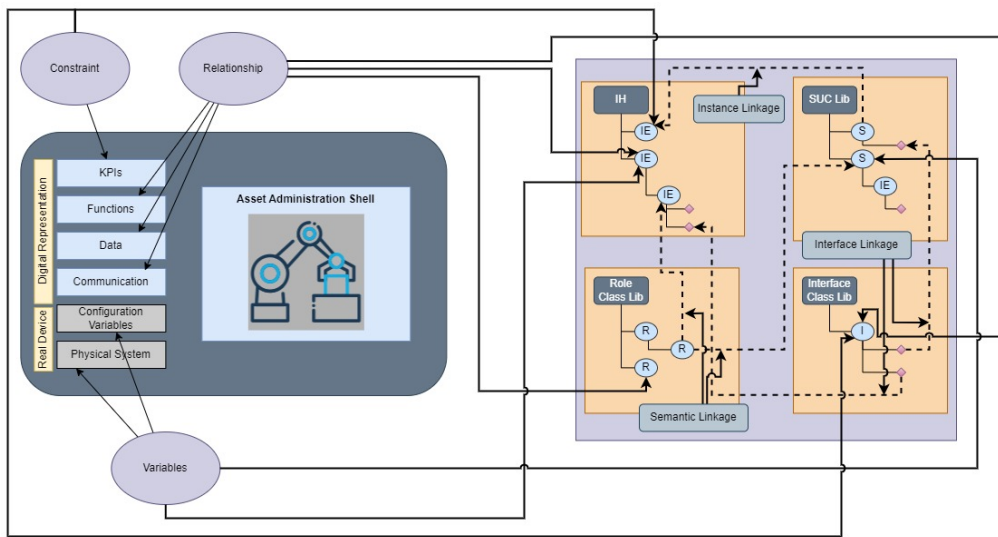


Figure 8.5: An illustration linking elements of an AAS to the CAEX Engine. Variables are housed as IEs, initiated as an instance of system unit class (SUC). The relationships are captured in IEs, interface instances (I) and role classes (R). Constraint information is captured in IEs and Is of CAEX engine.

### 8.4.4 Facilitating Communication with Lower Layers

The captured information in the engine can be used to interact with the low-level architecture components. This can be to a real-time control component or service actors/observers. For instance, in the case of real-time control agent systems may take this information to ascertain configuration before execution of functionality.



### 8.4.5 Understanding Asset Administration Shells (AAS)

As depicted in Figure 8.6, an asset administrative shell is composed of four major components (Wenger et al. (2018)), namely: component manager, manifest, header section and body section.

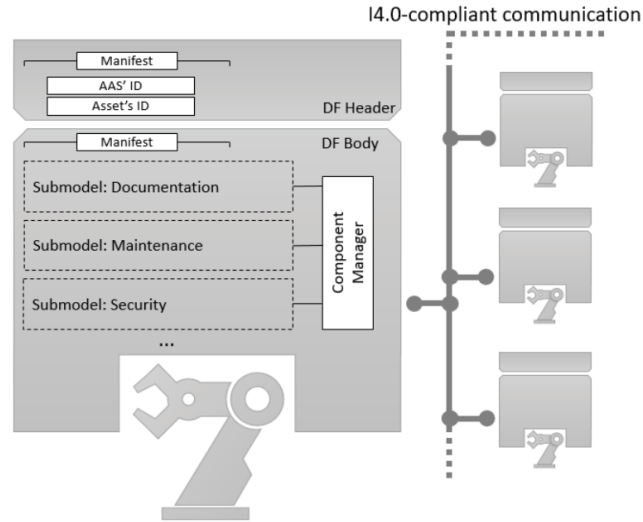


Figure 8.6: Asset Administration Shell representation of an asset with its main components for I4.0 compliant communication with IoT infrastructure within a production facility.

- **The DF (digital factory) Header Section** contains the globally unique identifiers for an AAS and its represented asset.
- **The DF Body Section** is composed of multiple submodels, each representing a distinct part of the asset's operation./
- **Component Manager** links the administration shell to a repository of submodels, their description and their functions. It administers the submodels of the assets. The Component Manager manages and provides access to the IoT network of the production facility using a service-oriented architecture.
- **Manifest** is present in both the header and body sections of AAS. It can be considered as the directory of data content. Specifically,

it contains the meta-information serving to provide meaning to the data from AAS.

## 8.5 State Chart to State Machine for Functionality Representation

As discussed in the architecture of the Adaptation Strategy the high-level layer consists of State Machine that encapsulates state charts representing functionality. In this section, explanations and directions for representing functionality through statecharts is detailed. The overview of state charts and state machines are discussed in the appendix section.

### 8.5.1 State Charts to State Machine for Self-configuration

In this approach, implementation has been carried out that encapsulates state charts that represent functionality in state machines. Figure 8.7 provides an overview of ISA-88 state machine and its encapsulation of state charts into valid behaviours. The state machine guarantees that only ‘valid’ transitions can be executed. It is based on ISA-88 standard (Hawkins et al. (2010)) for batch and continuous production.

The states and transitions that can be triggered on particular states are defined by a state machine. Here are a few instances:

- Only when the state machine is currently in the “idle” state will a “start”-transition cause it to enter the “starting” state.
- The state machine will change its present state to “complete” whenever the production of an order has been finished. Only in this state

can it be restarted.

- Nothing happens in case of a “stop”-transition while in the “stopped” state. No illegitimate transitions are allowed to be fired.

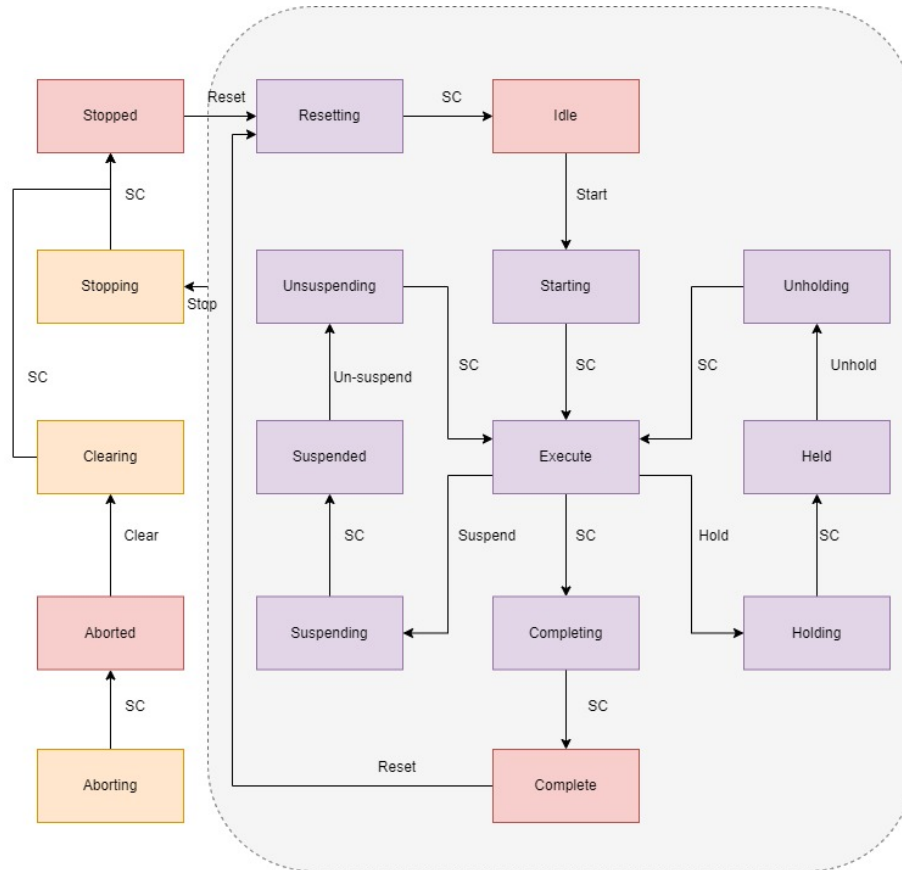


Figure 8.7: State Machine encapsulating functionality state charts as behaviours. It provides valid transition control mechanism for the production system to follow.

### 8.5.2 Achieving Self-Configuration for Functionality Stat Machines defined as State Machine Behaviours

The state machine, as per ISA88 standards, offers the capability to execute specific actions within the various active states. These active states are an integral part of the state machine and are Starting, Execute, Holding,

Unholding, Suspending, Unsuspending, Completing, Resetting, Stopping, Aborting, and Clearing.

In the context of module-driven configuration, self-configuration is facilitated through the use of state machines. These state machines adhere to the ISA88 standards and can execute predefined actions in specific active states, ensuring that the production system can dynamically adjust and configure itself (i.e. self-configure).

### **8.5.3 Configuring State-Specific Actions:**

To implement self-configuration, custom actions are created to define how the production system should behave in various states. These states, such as ‘starting,’ ‘executing,’ ‘suspending,’ and others, are used to apply a module-driven configuration approach. To create a behaviour (i.e. how the functionality performs at a state), logic is implemented as the production system enters a particular state. These behaviours include anything from initialising movements to configuring hardware or software settings.

Self-configuration occurs seamlessly within the module-driven configuration approach as a part of functionality execution, as the system is configured during state transitions.

### **8.5.4 Tracking State Changes:**

To make the system capable of self-configuration, it’s essential that the state changes are monitored in real-time. This is where state change observers come into play, responding to state changes as they happen. This behaviour in functionality records the effect of each state transition.

This integrated approach allows system to achieve self-configuration by employing module-driven configuration principles. As the system transitions between states, custom actions and observers work together to ensure that the system adapts and configures itself seamlessly to meet the specific requirements of each state.

## **8.6 Machine Learning for Self-Configuration**

### **8.6.1 Enhancing Machine Learning in the Production System for Production Systems:**

In the context of production systems, machine learning (ML) integration enables self-configuration by predicting parameter values that can be updated in the production system. In this section, a generalized workflow for incorporating ML capabilities to predict variable values within a configurable object is presented. This approach aims to streamline self-configuration and produce results that match the requirements (e.g. cost, time or other KPIs).

### **8.6.2 Workflows for Streamlined Machine Learning Integration:**

Functionality presents requirements of variables that need to be configured. A generalised workflow for providing machine learning capability for predicting the values of these variables in a configurable object is presented. These workflows can be used to apply machine learning to the production system use cases. The workflows are important as they can be used to effi-

ciently get the results from a set of inputs for self-configuration purposes. They are, in essence, like fingerprints, unique to the specific requirements of each production system.

### 8.6.3 Proposed Workflow Overview:

These workflows are provided as a basis for possible integration directions with machine learning services. Figure 8.8 shows the proposed workflows for the deployment of machine learning in production systems. The workflow is based on tabular data having structured data in rows and columns. The motivation to provide these workflows is to integrate ML components, keeping the implementation modular instead of focusing on ML component specifications.

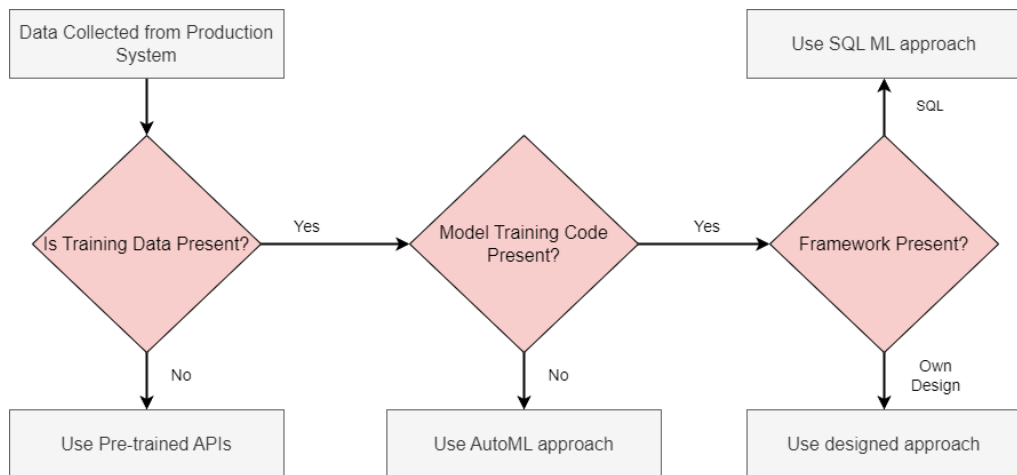


Figure 8.8: Proposed workflow for machine learning in the production system. The workflow defines the approach to use dependent on data and requirements.

### 8.6.4 Predicting Configuration Values:

To predict configuration values, several paths are available within this approach as illustrated in figure 8.9:

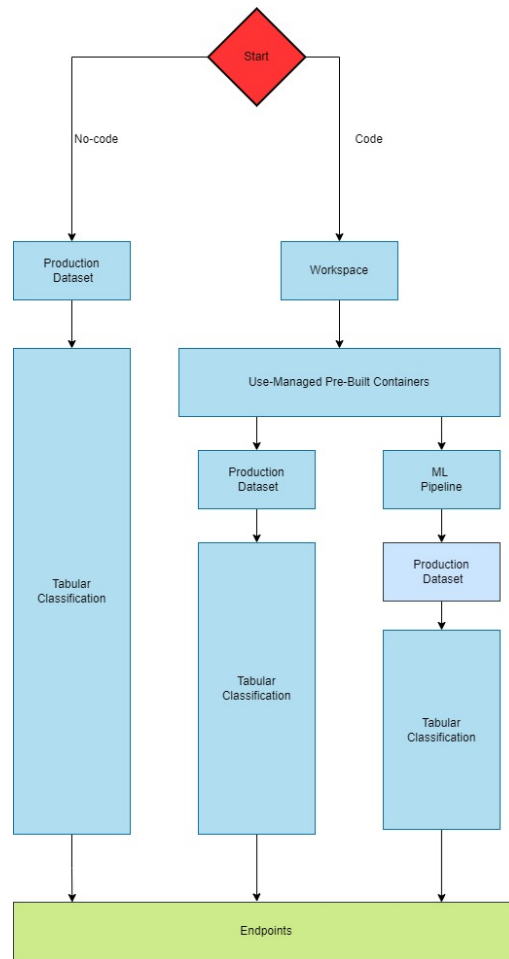


Figure 8.9: Illustration of the mechanism for the prediction of configuration values using the AutoML approach.

1. **AutoML Approach:** When training code is unavailable, AutoML can be employed. The data collected from the production system is utilized for tabular classification or forecasting. This approach is code-oriented or code-less. In a code-less setup, the production system dataset is hosted in a data warehouse service, and an AutoML service is used to derive configuration values. In the code-oriented approach, custom code is crafted for AutoML-based prediction. In either case, the ML model can be deployed on an endpoint, allowing production systems to request real-time configuration values.
2. **Off-The-Shelf Approach:** In scenarios where training code is available, off-the-shelf standard approaches like SQL ML can be harnessed.

These approaches are employed for classification and forecasting using tabular data. A custom-designed approach can also be used, incorporating techniques such as hyperparameter tuning and TensorFlow/Keras. The data is processed with code developed by the user, and the model is regularly evaluated and updated for precise configuration value predictions. Endpoint deployment enables real-time processing and utilization by production systems.

3. **Custom-Designed Approach:** In the case of the own-designed approach, figure 8.10 illustrates the generalised approach. The training code is deployed based on the training data. Techniques like hyperparameter tuning and TensorFlow/Keras can be used for predicting configuration values. This generalised approach can be used to integrate and model a variety of production system models based on custom designs. These models can then be deployed on endpoints to serve in real-time.

Figure 8.10 details an elaborative approach for predicting configuration based on the proposed workflows. Through these detailed workflows and approaches, the integration of machine learning into the self-configuration process for production systems becomes accessible and efficient. The choice between AutoML, standard approaches, or custom designs depends on the availability of training code and specific system requirements. Regardless of the method chosen, the outcome is streamlined self-configuration, ensuring production systems adapt effectively to their operational needs.





Figure 8.10: The generalised elaborative approach for predicting configuration values for configurable objects in production systems.

## 8.7 Multi-Agent System Integration

An approach to the realisation of self-configuration in production systems is presented in this section based on adaptation strategy. While this approach incorporates various components, including agents and expressions,

it's important to clarify that this section focuses on a particular aspect of the self-configuration framework. Agents are selected as enablers of self-configuration, as established in Chapter 4.

### **8.7.1 Realisation of Self-configuration in Production Systems**

#### **Agent driven Self-configuration in Production Systems**

A production system comprises numerous assets (i.e. configurable objects), that perform manufacturing processes, with services to produce products as per customer requirements. The use of agents in this research is limited to a single configurable device, i.e., a single manufacturing system, that can be expanded to include multiple manufacturing systems and their interconnection to form the production system as a whole.

#### **Agents Involved in Self-Configuration**

The application involves six actors namely the operator, the manufacturing system, the Transport Agent (TA), the Product Agent (PA), the Resource Agent (RA), and a data management pipeline. The pipeline consists of services involved in managing the data, using the accumulated data and coordinating with other services and agents to achieve self-configuration. The service description, agents (Table: 8.2), pipeline, and their connection for self-configuration are elaborated on in the next section.

Table 8.2: Agent Description for Production System

Agent	Description
Transport Agent (TA)	TA is responsible for transporting the product between manufacturing systems in a production system.
Product Agent (PA)	PA is instantiated to represent one product as it is entered into the production system. A product, represented by PA, has a name along with functionality requirements to produce it.
Resource Agent (RA)	RA is responsible for handling resources. One RA is responsible for one resource represented by a name, functionality, and location. This resource is a asset that executes a process on the product

### Sequence of Agents Framework for Self-Configuration

Configuration in a manufacturing system comprises process parameters, communication and connectivity settings, and calibration settings. The movement of data in the research for self-configuration is achieved through agent technology that controls operations occurring during execution. The agent technology communicates with the data pipeline components to perform self-configuration. The interactions between agents, the production system, the data pipeline, and the operator is shown in the sequence diagram (figure 8.11);

1. At the start of the production cycle all the manufacturing system registers on a Directory Facilitator (DF) Service. It has an associated name, location, and a functionality (process it can execute) offered. For the purpose of simplicity, each manufacturing system executes one functionality. Each asset is represented by a RA.
2. The PA requests DF service for the functionality and finds the respective RA that is able to execute the desired functionality.
3. If multiple RAs are able to execute functionality, then the best RA is

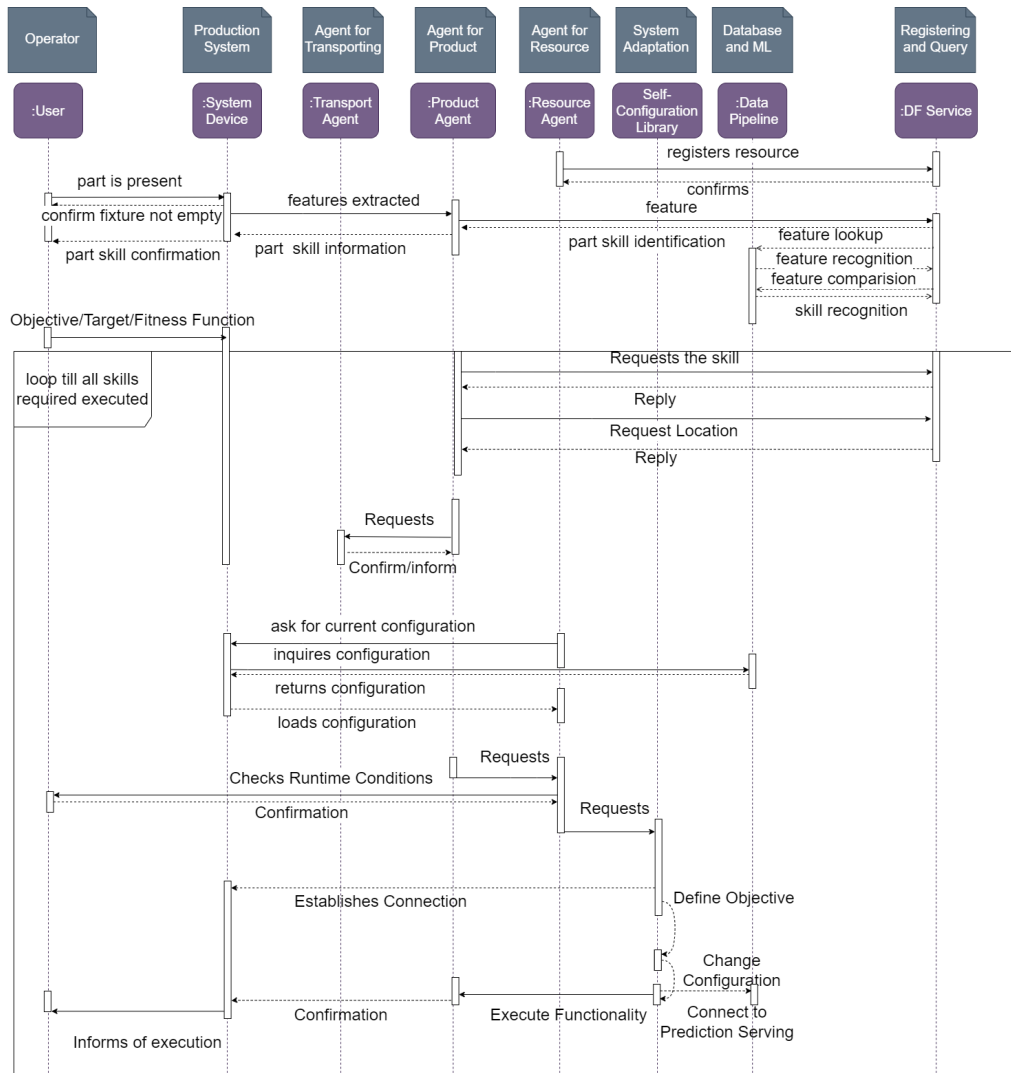


Figure 8.11: Sequence diagram for functionality execution using agent application in self-configuring production systems.

selected through negotiations based on Contract Net Protocol (CNP).

4. As the RA is selected, the TA transports the product to the asset (represented by the chosen RA).
5. The data pipeline connects with the asset. It configures the system to meet the configuration as per product requirements. For this to occur, it checks for the runtime conditions for the resource in regard to the product. If all conditions are satisfied, then the functionality is ready to be executed.

6. The asset needs to be configured as per requirements. The ML services connected to the data pipeline configure the production system and functionality is executed.
7. Once the functionality is performed, the RA returns with a confirmation and PA moves instructs the TA to move to the next location to perform the next functionality.
8. If no more functionality is required, then PA requests the TA to move the product out of the system.

The run-time conditions assist in checking the current state of the system and matches it with the desired future states. Let's consider that there is a need for a part to be picked up by a robot. The robot may possess the capability to pick up the part but if it is already holding a component the check against the run-time condition will return as negative. Apart from capability, the run-time conditions can be verified against relationships, constraints, and specific (e.g. product and process) requirements. The Key Performance Indicator (KPI) serves as an anchor for self-configuration, i.e., the self-configuration in the asset is driven by KPI as a guiding parameter. As stated before, the system adapts to changes to improve functionality. The guiding parameter provided as KPIs is used by the self-configuring component of the data pipeline, optimising the system to the best possible setting for that parameter.

### **8.7.2 Using Expressions for Self-Configuration in Production System**

Expressions drive the functionality of the production system by influencing the agent system. Expressions are of two types; “conditional” and “oper-

ational”. Conditional expressions govern the behaviour of the agent when executing a functionality while on the other hand operational expressions are concerned with the behaviour of the production system itself.

Operational expressions, involving the execution or performance of the operation, are formulated in a logic-based language that can be understood by agent systems. These expressions are responsible for the manner in which the production system carries out its execution for any part.

Let ‘T’ be an operational expression that constitutes of a set of functionalities to be executed. These functionalities are executed sequentially, in parallel or in an interleaved manner. The operational expression ‘T’ can therefore be considered as a set of functionalities and their associated conditions;

$$T = \{C_1S_1, C_2S_2, C_3S_3, \dots, C_nS_n\} \quad (8.1)$$

The ‘T’ operational expressions are comprised of functionality ‘S’ under condition ‘C’ executed in the mentioned manner. The behaviours in which these functionalities under conditions are executed are represented in expressions given as follows;

$$T_{sequential} = C_iS_i.C_jS_j \quad (8.2)$$

$$T_{choice} = C_iS_i \oplus C_jS_j \quad (8.3)$$

$$T_{parallel} = C_iS_i \parallel C_jS_j \quad (8.4)$$

$$T_{interleaved} = C_iS_i | C_jS_j \quad (8.5)$$

These expressions can be formulated either to demonstrate sequential, choice, parallel or interleaved behaviour individually or in combination.

The interleaved behaviour represents a case of functionality synchronisation where one functionality waits till other functionality is ready to be executed. The rules that guide the behaviour, based on Aceto et al. (2005), along with its general abstraction, are represented by the following expressions with  $i, j, k$  representing functionalities and condition instances :

$$T = \frac{Premise_1, Premise_2, \dots, Premise_n}{Conclusion}.Condition \quad (8.6)$$

$$T_{choice} = \frac{C_i S_i . C_j S_j \xrightarrow{C_i S_i} C_j S_j}{C_i S_i . C_j S_j \oplus C_k S_k \xrightarrow{C_i S_i} C_j S_j}.C_i \quad (8.7)$$

$$T_{sequential} = \frac{C_i S_i}{C_i S_i . C_j S_j \xrightarrow{C_i S_i} C_j S_j}.C_i \quad (8.8)$$

$$T_{parallel} = \frac{C_i S_i \xrightarrow{a} C'_i S'_i \quad C_j S_j \xrightarrow{a} C'_j S'_j}{C_i S_i \parallel C_j S_j \xrightarrow{a} C'_i S'_i \parallel C'_j S'_j}.(a \in \theta = \sum (C_i S_i) \cap (C_j S_j)) \quad (8.9)$$

$$T_{interleaved} = \frac{C_i S_i \xrightarrow{a} C'_i S'_i}{C_i S_i \parallel C_j S_j \xrightarrow{a} C'_i S'_i \parallel C_j S_j}.a \notin \theta \quad (8.10)$$

Here  $a =$  action and  $\theta =$  Set of Intersecting Actions

Such kind of representation of operational expression governs the execution of operations specific to the part being operated on. As the part being operated on is identified its expression is loaded , the agent uses that expression to perform the test.

Operational and conditional expressions are responsible for agent functionality execution for any testing operation carried out by the production system. An operational expression is accompanied by multiple conditional expressions for each functionality. These coupled expressions can be loaded into the agent responsible for the execution, as the part is identified. These expressions can be housed in a storage component on a scalable, extensi-

ble database service that is queried by the agent system to load the best possible expressions for the identified parts.

### 8.7.3 Industrial Application for Agents

The industrial application for agents is explored in this section, where routing is designed to serve as a simulation for production systems. This designed application compliments the proposed sequence diagram and expressions. This realisation connects the assets in the production system to the data pipeline components. A smart controller is necessary for the realisation of the proposed approach, i.e., establishing a connection and communicating with pipeline components. Data collected by the controller is stored locally or externally, converted, and transmitted to the pipeline where it is processed by the pipeline components. As per the sequence diagram, runtime conditions are checked and the system is adapted as per objective definition through the prediction serving component. The agents control the functionality of the manufacturing system.

The pipeline automation is achieved through event-driven trigger functions that listen to an event occurrence. These fetch events happening in real-time and instantiate the data pipeline components.

The functionalities and the location of the manufacturing system are entered and assigned in the layout. Each asset in the production system corresponds to one RA. The requirement of functionality is entered through the interface instantiated by a PA. The PA instructs TA, after negotiation through CNP, to transport the product to the asset location. As the product reaches the location, the system coordinates with the data pipeline components to adapt the asset, configuring as be defined objective and



achieving self-configuration.

The primary focus on agent-driven application is towards self-configuration in production systems and coordinating functionality through the corresponding expression concepts. While this approach demonstrates a feasible method for achieving self-configuration, the complete adaptation strategy may encompass other components such as CAEX implementation, state charts, and more, depending on the specific application.

The application can serve as a foundational concept for more extensive self-adaptation strategies, providing a deeper level of insight into the dynamics of production systems. Furthermore, in a complete implementation, different elements like CAEX, state charts, and other mechanisms could be used in conjunction with agents and expressions to enable comprehensive self-configuration and adaptation of production systems.

## **8.8 Real Time Control**

### **8.8.1 Enhancing Adaptation Strategy with Real-Time Control and Hardware Abstraction:**

The adaptation strategy proposes the usage of Real-Time Control (RTC) linked to hardware abstraction. The purpose of separation is to keep the RTC independent of the hardware component. This provides the ability to use hardware-specific API or PLC event triggers for controlling production system functionality guided by RTC.

### 8.8.2 Hardware Abstraction for Self-Configuration:

Hardware abstraction, for self-configuration, is concerned with 2 parts;

1. **Communication with Configurable Objects:** Hardware abstraction offers a means to communicate with the configurable objects within a production system.
2. **Resource Management:** It facilitates resource addressing on the configurable object, enabling both configuration and the execution of functionality.

This level of abstraction can deal with any PLC or API for a configurable object. As guided by the RTC the hardware abstraction creates an instance of connection with the hardware. Depending on the type of connection, resources can be addressed to configure and execute functionality. The resources can be addressed through specific commands (API Calls, PLC function inputs) and configuring the configurable object, the commands can then be used to send a signal to the production system to execute functionality.

### 8.8.3 Real-Time Control and Hardware Abstraction Interaction:

Figure 8.12 illustrates in detail the RTC and its interaction with hardware abstraction. The user application, for self-configuration, interacts with mid and top-level layers of the adaptation strategy and through hardware, abstraction configures the manufacturing asset.

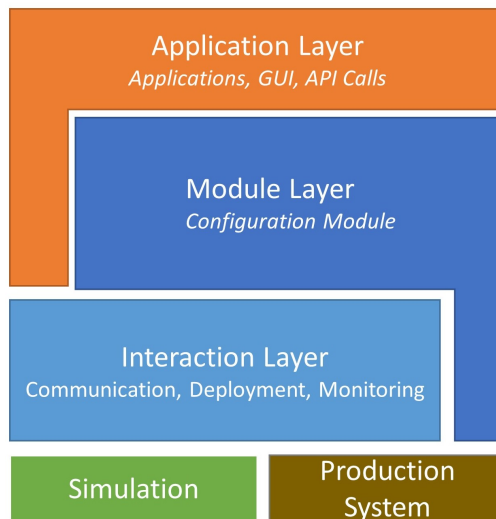


Figure 8.12: RTC and the interaction of its components with hardware abstraction for module-driven self-configuration. The application layer interacts with the configuration module in a production system through its interaction layer. Adaptation strategy acts on the configuration module.

#### 8.8.4 Service Actors and Their Role:

Service Actors can also link and instantiate commands to hardware abstraction. These service actors are third-party services like (e.g. MQTT, AMPQ etc.) that instantiate commands on hardware after detecting changes in Service Observers. These service actors and service observers can interact with the simulation environment to gather the best configuration for the configurable object.

Incorporating RTC and hardware abstraction within the adaptation strategy enriches the self-configuration process. By decoupling RTC from hardware specifics, it enables a more flexible and efficient approach to configuring and controlling production systems.

## 8.9 Simulation

### 8.9.1 Simulation for Self-Configuration Module:

The simulation component acts as a digital representation of the physical asset. The simulation environment also contains data flow, that feeds data from the physical asset to the simulation environment. In the approach, data is represented using CAEX engine previously elaborated. This engine captures the production system information in an AAS. More detail on AAS is detailed in the previous section.

The simulation environment is capable of handling multiple data representations to account to different configurable objects in production systems. The simulation environment supports optimisation through software libraries processing data from the digital representation. The simulation environment can act on the provided KPIs (e.g. time, cost etc.) to infer the best possible configuration.

The significance is that the optimization algorithms are designed to be modular and distinct from the simulation environment. This modularity allows flexibility in addressing various optimization targets within specific constraints. Depending on different KPIs or combinations thereof, distinct optimization algorithms may be preferred. The execution of these optimization procedures can occur locally or through Cloud-Based Services, adding an extra layer of adaptability to the system.

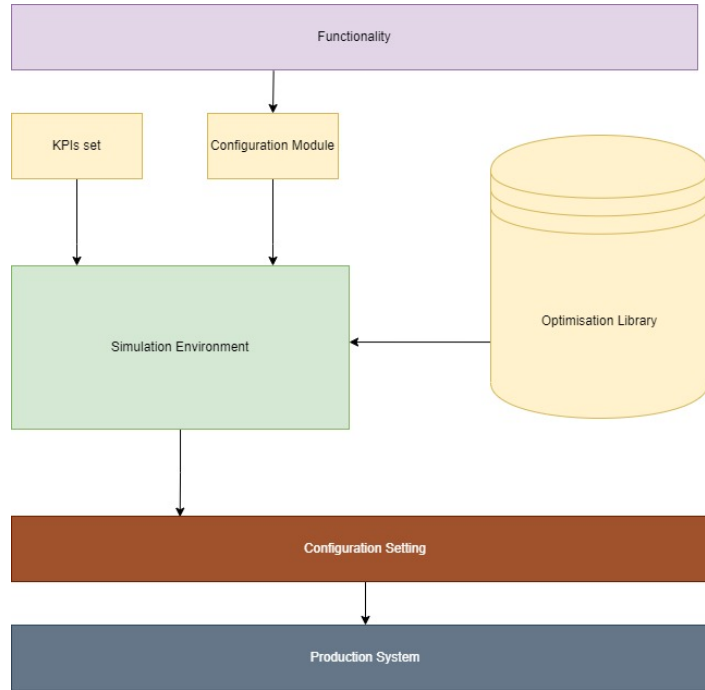


Figure 8.13: Illustration of simulation for configuration module.

### 8.9.2 Detailed Simulation Process:

Figure 8.13 illustrates in detail the simulation carried out. The KPI is set for the simulation environment based on which the optimisation algorithm is selected. The simulation environment using the optimisation library selects the best possible configuration setting for a functionality. The output in form of the optimisation configuration is received and loaded into the production system. The functionality can then be executed.

### 8.9.3 Intercommunication within the Self-Configuration Module:

An essential aspect of this module is communication, which occurs between the physical representation of the asset, its digital counterpart in the simulation, and the optimization library. Various communication methods can be employed depending on the specific application domain. These methods

encompass OPC UA, MQTT, TCP/IP, and others, each chosen to best suit the requirements of the self-configuration system.

The simulation component, within the larger self-configuration strategy, acts as a critical bridge between the physical asset and the digital self-configuration process, enabling the system to dynamically adapt and optimize its functionality as conditions change.

## 8.10 Conclusion

This chapter contributes towards providing the structure and principle of the adaptation strategy. Overview of the strategy involves detailing its three layers *Production System Coordination Layer*, *Production System Runtime* and *Production System Drivers and Control* based on implementation requirements. A way of capturing information to aid the adaptation strategy towards achieving self-configuration is explored through CAEX standard.

This chapter provides details of the implementation process while utilising the adaptation strategy. The standard tools that can be used to self-configure systems while using the strategy are explored, showcasing the research effort. The information captured through a CAEX standard is detailed for manufacturing assets in systems. A means of capturing functionality in terms of state chart and state machine behaviours is elaborated. Multi-Agent system for coordinating functionality and control serves as an enabler for realising self-configuration through the proposed self-configuration strategy. Real-Time control is provided through APIs and PLC event triggers, along with service actors and observers through the interaction of the adaptation strategy with the hardware. Simulation aspects

are explained to support routing and determining the most suitable configuration. This chapter serves as the basis for industrial implementation of the strategy in the next chapter.

---

## Chapter 9

Industrial Use-Case:

Validating the Adaptation

Strategy for Self-Configuring

Production Systems



## Contents

---

9.1	Introduction . . . . .	<b>243</b>
9.2	Methodology . . . . .	<b>244</b>
9.2.1	Production System Coordination . . . . .	245
9.2.2	Production System Runtime . . . . .	246
9.2.3	Production System Drivers & Control . . . . .	247
9.3	Implementation . . . . .	<b>248</b>
9.3.1	Mapping Level-Based Classification features to the Industrial Application . . . . .	248
9.3.2	State Chart and State Machine Representation for Functionalities . . . . .	252
9.3.3	Multi-Agent Integration for Self-Configuration .	260
9.3.4	Deploying Machine Learning Pipelines of Cloud for Self-Configuration . . . . .	263
9.3.5	Real-Time Control of Test System . . . . .	268
9.4	Experiments . . . . .	<b>271</b>
9.4.1	MALT . . . . .	271
9.4.2	PRIME . . . . .	295
9.5	Conclusion . . . . .	<b>317</b>
9.5.1	Potential Cost Saving by Adaptation Strategy:	319

---

## 9.1 Introduction

This chapter deals with implementing the adaptation strategy established in the previous chapter. The guidance provided by the adaptation strategy

is used to realise self-configuration capability in two industrial use cases. This chapter aims to validate of the research carried out in this PhD thesis. The implementation carried out through this adaptation strategy is related to the objectives of the research established in Chapter 3.

The validation of the research is carried out on MALT and PRIME test station. MALT( Multi-Application Leak Test) is a leak test module developed by an SME (TQC Ltd.) that provides the ability to test leaks in a part under pressure. Variations in part features, functionality conditions or any constraints can cause the test configuration to change. This self-configuration application aims to assist in the adaptation of test configurations due to change. There can be different levels of granularity as MALT can be provided either as a part of a large production system or stand-alone. PRIME test-station is used to test a part under force for deformation. Like the MALT use-case variations in part features, functionality conditions or any constraints can cause the test configuration to change. The PRIME test station is one station in the PRIME assembly system. This self-configuration application aims to automatically determine the force test configuration setting for the part.

## **9.2 Methodology**

The Adaptation Strategy established in Chapter 8 drives the methodology for applying the developed approach. The solution to the self-configuration problem is realised through a three-level approach: Production System Coordination, Production System Runtime and Production System Driver & Control. Each of these layers requires a separate realisation (i.e. implementation), and then integration to achieve self-configuration. Each layer

of the Adaptation Strategy and the respective realisation is discussed in the following sections:

### 9.2.1 Production System Coordination

This layer encapsulates the functionality execution in the form of state charts. These state charts are incorporated into State Machine behaviours.

The functionality execution is based on customer and functionality needs, therefore the KPIs. This High-Level layer directly interacts with the other two layers to execute functionality.

For both the leak testing system (MALT) and the force test in PRIME, state charts are developed to capture functionality and customer needs. Parts are identified based on fixture identification, and settings for each test are determined accordingly. The state charts coordinate the functionality in the MALT and PRIME test station from part identification to execution. The transitions carried out in the state charts are as follows;

- Activating the state chart transition for self-configuration for specific functionality in the production system (MALT or PRIME test station).
- Generating part requirements for the functionality execution. These can be a picture, a parameter requirement, a certain customer-imposed condition or a functionality restriction.
- The KPIs are set for the functionality, dependent on the part requirement previously generated. These KPIs can also be dependent on customer or functionality needs.

- These KPIs are related to variables for the test that need to be updated (configured).
- The values of these variables are determined through external interfaces (e.g. machine learning pipelines or a pipeline hosted on Google Cloud).
- The updated values are then applied at the respective test station, and the functionality is executed.

These state charts for the leak test production system (MALT) or PRIME test station are incorporated as behaviour in the state machine. The state machine uses the tool utility developed to coordinate the functionality.

### 9.2.2 Production System Runtime

Information about the state and configuration of each test system is stored in the Runtime layer. This layer represents a means through which the services, APIs or PLC can verify and monitor the current condition of the test system. The runtime layer captures:

- Information about the production system, the identifiers, operations, functionalities, and all relevant properties in a standardised format.
- Relationships between parameters relevant to the specific test, such as leak rate, volume, test pressures (for MALT) or maximum force, force time, and camera images (for PRIME).
- The captured information is related to the state of the respective test system, the fixture in place, and the product being tested. Therefore, this layer captures all information necessary for the linked management system to make that inference. In the respective test case, the

state charts coordinate the functionality execution, so this layer interacts with the production coordination layer to determine the state of the test system.

- This layer captures information in the form of submodels. There are multiple submodels in the test system representing functionality, documentation, and API Calls among others. These submodels are accessed through the relevant interfaces.
- Submodels can be interfaced in combination so that reasoning can be established through relationship query and contextualization. For instance, fixture information can be used to determine force configuration, while fixture information, configuration variables and image information are stored in different submodels.

The developed CAEX Engine captures this information in the form of an Asset Administration Shell (AAS) of each test system. The submodels of the test system in this research represent the functionalities, documentation, settings, and configuration.

### 9.2.3 Production System Drivers & Control

This layer deals with the PLC control of the respective test stations through event triggers from the Coordination layer. Real-time control (RTC) is encapsulated in this layer through linkage with runtime information capture. The RTC executes PLC calls (PRIME) or MALT API Calls depending on updated parameters, executing functionality (e.g. force test or leak test), and modifying configurations.

A demonstration is carried out for leak testing and force test use-cases to execute the functionality with an updated configuration. This layer is

coordinated through the production system coordination layer to execute functionality after updating setting values as per functionality and customer requirements.

## 9.3 Implementation

This section goes in detail on each aspect of the three-level Adaptation Strategy realisation. State Chart transitions are detailed with encapsulation into State Machines for MALT and PRIME test station functionalities. Information capture through CAEX is elaborated in connection with Asset Administration Shell. Tool Utility for coordination of self-configuration process is expanded. The ML Endpoint deployment, multi-agent integration, expression development are explained. Real-Time Control is carried out to achieve self-configuration.

### 9.3.1 Mapping Level-Based Classification features to the Industrial Application

#### **MALT Leak Test Application:**

The level-based classification features are taken as a means to model self-configuration ability for the industrial use case. The controller for the MALT serves as the main focal point of feature deployment. For this research, a simulation environment was modelled so that an example routing may be set up. Figure 9.1 illustrates this simulated routing. The agents (i.e. enablers of self-configuration) move the product to the leak test system (MALT) for the functionality to execute. The monitoring feature of the proposed classification covers the complete monitoring of the whole system,

including all changes happening in it. A MALT interface and simulation interface is developed that monitors the product as it moves through to the test system and testing is carried out. The simulation interface also monitors the product information, the routing through the system, the number of products present in the system and the mean time to produce the product. This gives a complete holistic view of the production system, giving highest Level-5 to the feature of “monitoring”.

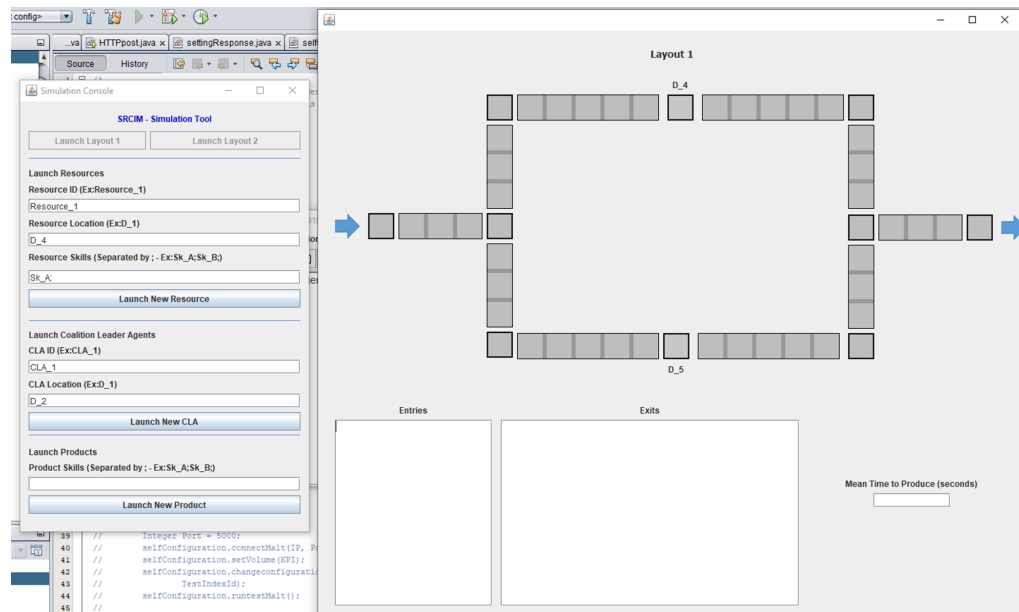


Figure 9.1: Routing design with the interface for self-configuration demonstration.

MALT controller is the main focus for implementing the features for self-configuration. MALT controller is developed so that it can record the changes that happen to its pneumatic module, can be calibrated, connect with other systems and record information that can be processed. It is able to realise its state, and display graphs to represent them along with the pass/fail outcomes inferred from the results. Therefore, a “self-management” feature was developed for achieving self-configuration on the MALT. Data recording capability on the controller can store data from tests, and information about connecting to the environment, along with different configuration settings for the system (MALT) giving it features of

“data collection” and “data storage”.

The controller in MALT is capable of communicating with other systems by TCP/IP communication for “data transmission” and is able to send/receive data in a compatible format, i.e., supports “data transmission”. To account for “operational identification” the MALT is integrated with an agent system that coordinates with the data pipeline to achieve self-configuration (as per requirements). The data pipeline handles the “optimisation” functionality for the MALT system for test purposes. Here in the data pipeline, the runtime conditions are checked, objectives defined, conditions formulated and the system adapted through agents by prediction serving components (endpoints).

The application of the self-configuration framework on MALT is carried out once these features are implemented and the controller is integrated with data pipeline and agent system (JADE). Previous Chapter 5 talks about level-based classification in detail. For the purpose of research, the data pipeline is hosted in a cloud environment. Google Cloud Platform (GCP) is used for this application.

#### **PRIME Force Test Application:**

The developed level-based classification can be applied to the PRIME test case. The PRIME test station is used to test parts under force. In this context, the Beckhoff PLC serves as the central focal point for deploying the self-configuration features. A simulation environment is created, setting up an example routing for the product being tested and the agent systems deployed. Agents, in our approach, acting as the enablers of self-configuration, move the product to the test station where the functionality is to be executed. The proposed classification system, through its features,



ensures complete monitoring of the system, including all changes happening in it. The simulation interface, along with a Beckhoff HMI PLC interface, monitors the product's movement through the system, the number of products present, and the mean time to produce the product.

The Beckhoff PLC is designed to record any changes that occur in the attached pneumatic modules and can be calibrated to connect with other systems (i.e. as per granular decompositions), allowing it to record information that can be processed. The mapped features of the self-configuration enable the PRIME test station to achieve self-management, data collection and data storage capabilities, through identification of the components. The controller (Beckhoff PLC) can be used to achieve this, by mapping the features. Furthermore, the Beckhoff PLC can communicate with other systems using OPC UA for efficient data transmission.

To achieve operational identification, the Beckhoff PLC is integrated with an agent system that coordinates with the data pipeline to facilitate self-configuration. The data pipeline handles the optimisation functionality for the Beckhoff PLC system, which includes checking runtime conditions, defining objectives, and formulating conditions. These tasks are accomplished through the agents by prediction-serving components (ML endpoints). In the current deployment, the ML endpoint hosts the images of the fixture that are analysed by Convolutional Neural Networks (CNN) to determine the best force configuration setting for testing.

As the features are mapped with the PRIME test station and integrated with the agent system (JADE) along with the data pipeline, the self-configuration can be applied to the production system through the Adaptation strategy. This approach provides a complete view of the production system, with the highest level, level 5, attributed to the monitoring feature.

The data pipeline is hosted in a cloud environment, and the application is executed using the Google Cloud Platform (GCP).

### 9.3.2 State Chart and State Machine Representation for Functionalities

#### **For the MALT Leak Test Application:**

In this application, the Adaptation Strategy is deployed to enable the self-configuration process for the MALT. The state charts and state machine representations encapsulate the production system's functionality using event-driven mechanisms. This event-driven approach forms the high-level layer of the Adaptation Strategy, responsible for coordinating the production system in the specific use case of leak testing.

**Application of State Charts and State Machine to MALT:** For the supporting case study, which is the leak testing operation, Yakindu Statechart tool is utilised. This tool is instrumental in capturing all the events that can trigger self-configuration within the production system for leak testing operations. Yakindu Statechart employs compositional modelling formalisms to represent the internal behaviour of production systems, essentially creating a logical structure from a collection of simpler modules.

The self-configuration state chart essentially involves triggering the self-configuration process either by introducing a part into the production system or by requesting the functionality to be executed on a part (figure 9.2). Events such as "Identify Part" are responsible for analysing introduced parts in terms of their features.

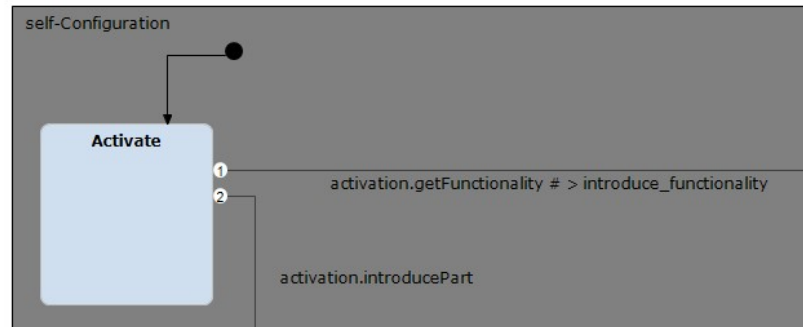


Figure 9.2: Activating the state chart transition for self-configuration

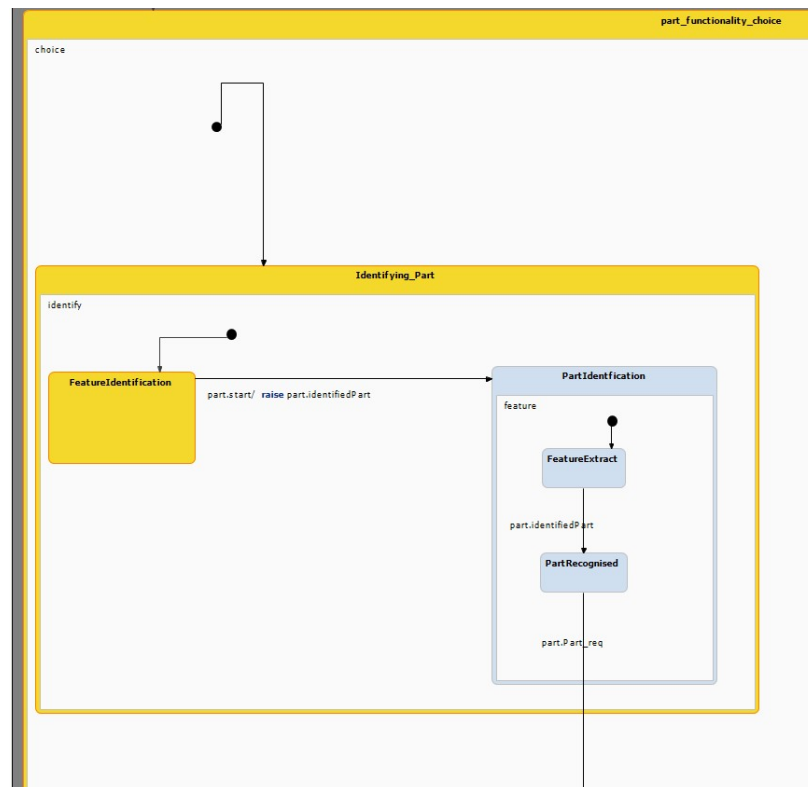


Figure 9.3: Generating part Requirement for the Introduced Part

The identified part generates a list of part requirements, which are prerequisites for executing the desired functionality. These Part Requirements are then matched with the configurable variables of the system, for the self-configuration process (figure 9.3).

The settings required for the functionality are determined through a composite state called "Setting Requirement." Customer and process requirements are identified and set, thereby establishing the parameters necessary

to configure the production system for the desired operation (figure 9.4).

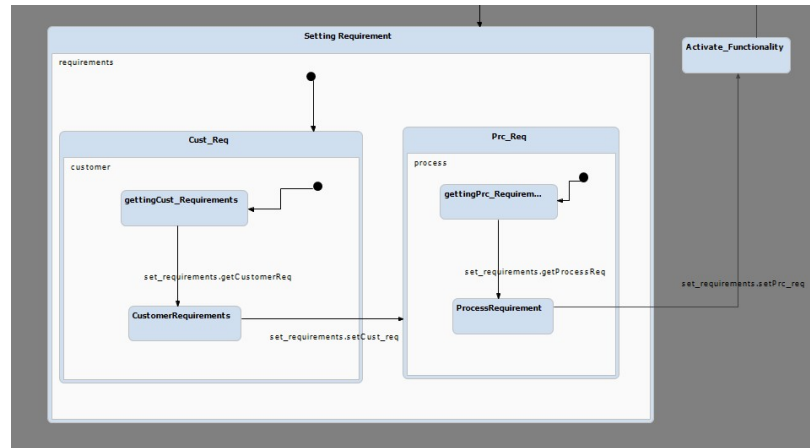


Figure 9.4: Setting Requirements for the Functionality

The activation of functionality, in this case, the "Basic Leak Test," depends on the configuration of variables that need to be changed. These variables, crucial for the functionality, are determined through state transitions and endpoints (figure 9.5).

Once the variable values are established through endpoints, they are allocated to the production system, ensuring that the functionality can be executed with the correct configuration (figure 9.6). The allocation of variables in the production system is established through linkage, and the configuration changes are carried out on the test system (figure 9.7).

The state charts developed can be encapsulated within an ISA-88 state machine, a standard for batch control. This encapsulation involves multiple process stages, including initiation, execution, holding, unholding, suspending, unsuspending, completing, resetting, stopping, aborting, and clearing.

**Capturing Information in Asset Administration Shell:** The explained state chart and state machine representations above form the *High-Level* Production System Coordination layer. The information encapsu-

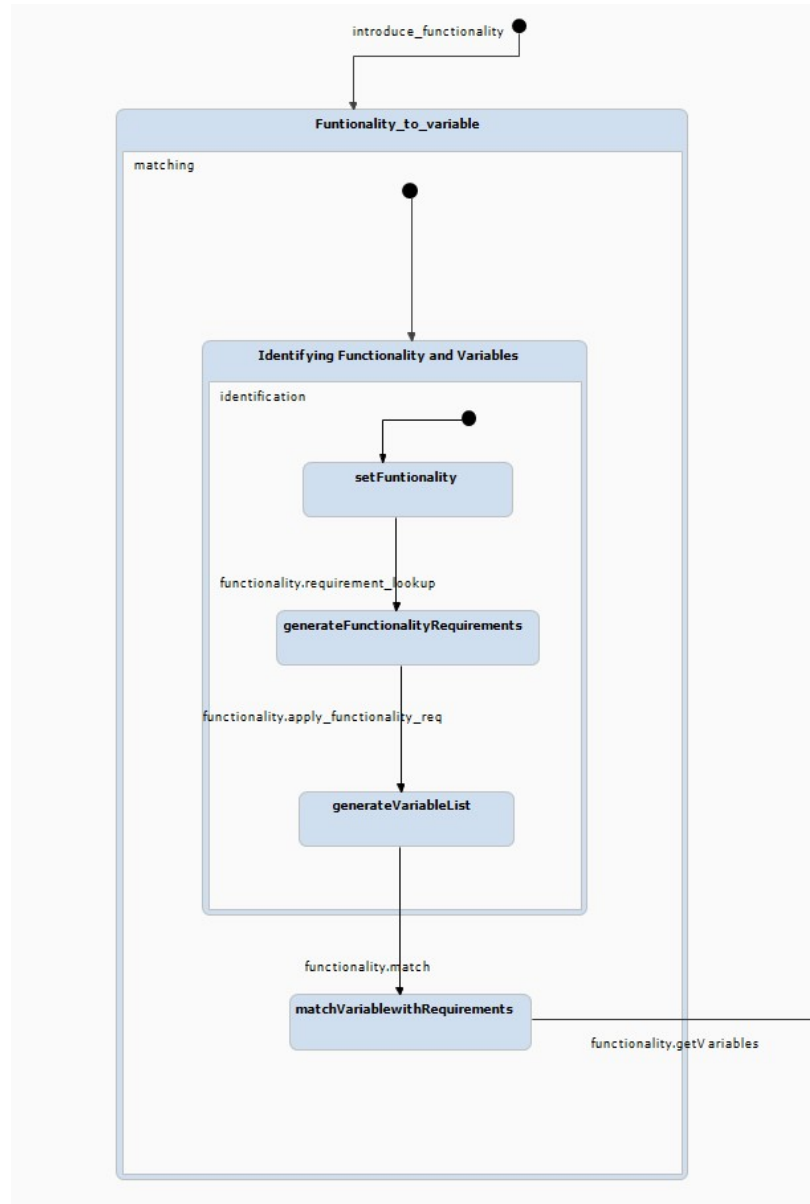


Figure 9.5: Matching functionality with requirements

lated through these state transition assist in executing the functionality. The high-level layer uses the data captured in the Production System Runtime (Mid-level) layer. The mid-level layer, as discussed in the previous chapter, contains information on the runtime of the production system.

In this research, a Configurable Object (CO) is considered as the production system component (i.e. cyber or physical) that may be configured. The CO object entity, as discussed in previous chapters, is a superposition of

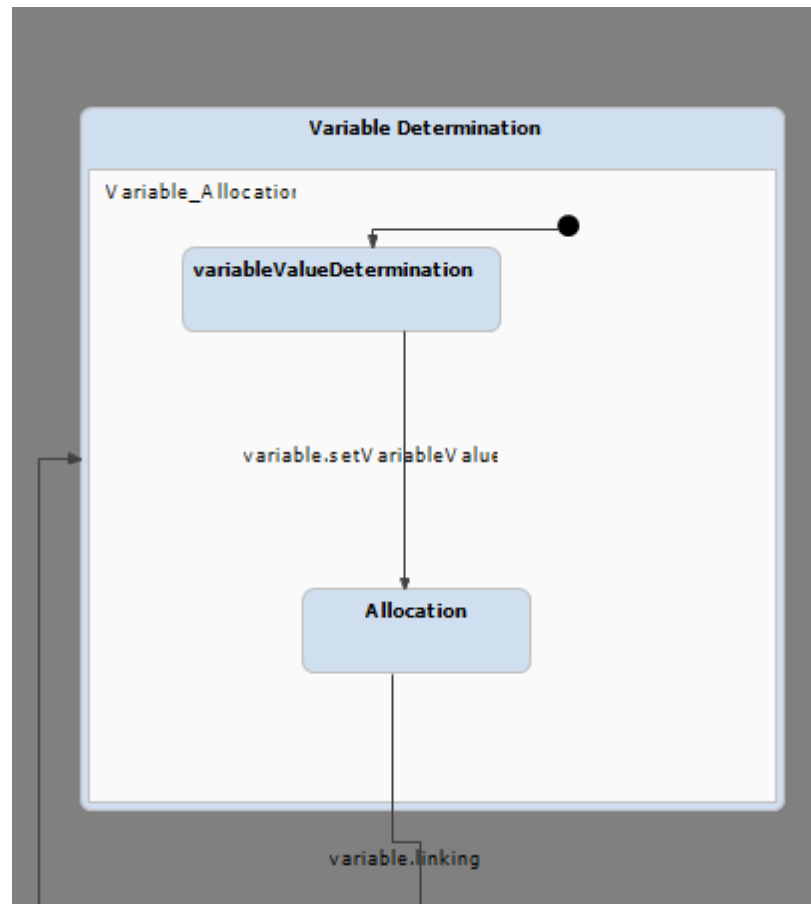


Figure 9.6: Variable Value Determination for Functionalities

a number of modules. A change in module structure represents a change in functionality driven by variables, but not a change in CO. During the start of the production operation, the variables are configured, according to allocation and linkages. The interaction between all COs in a production system is carried out through transition rules mapped in the state chart high-level layer.

In order to represent the production system for the MALT leak test use case, all information pertaining to executing the leak test functionality needs to be captured. To support this a model is needed that captures this information, stores it and is dynamic in nature. The model explained in previous chapters promotes a vendor-neutral mode of capturing information.

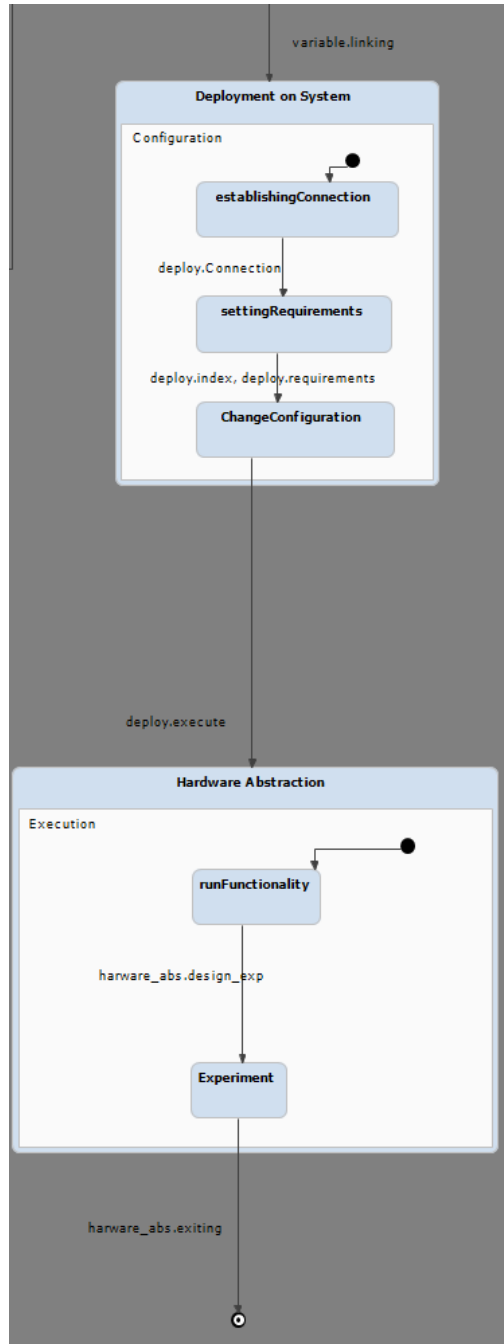


Figure 9.7: Deploying variables on production system and executing functionalities

**Tool Utility for Assisting Coordination of Functionality Execution:** A Tool Utility is developed that aids in coordinating functionality execution in the MALT. This tool interacts with the Asset Administration Shell (AAS) to establish contextualization and relationships for the Production System Coordination layer. The high-level layer executes func-

tionality based on this coordination. The tool utility is depicted in figure 9.8.

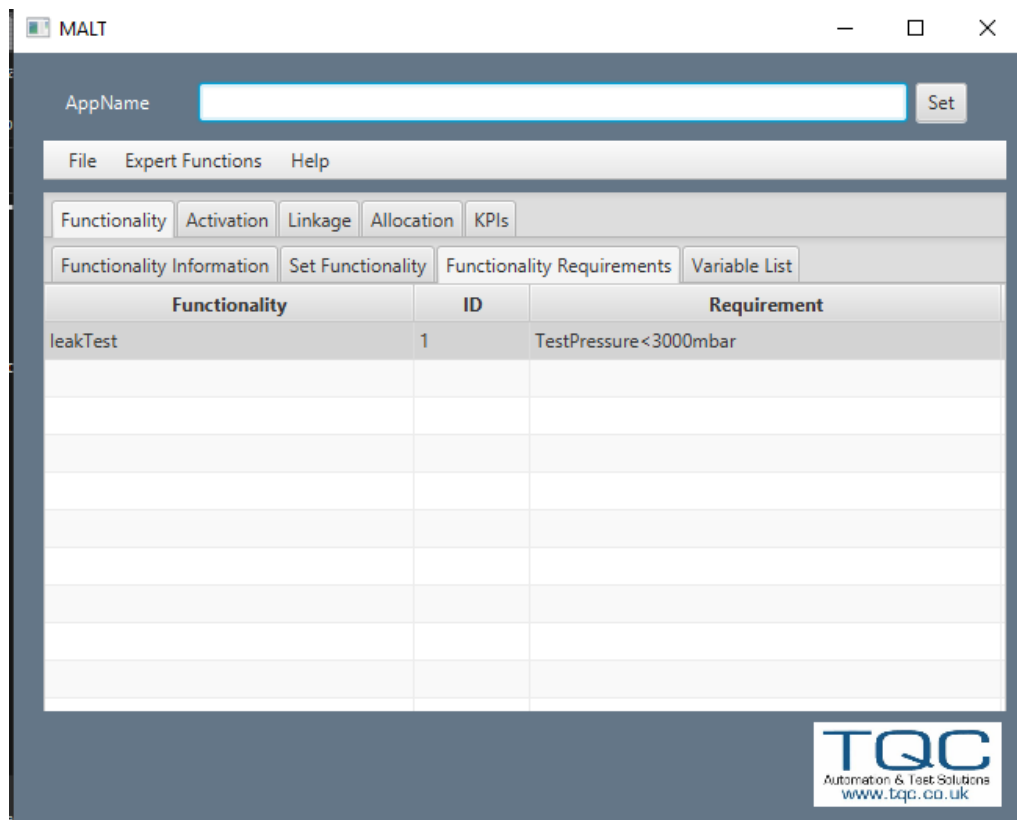


Figure 9.8: A simple glimpse of the Tool Utility developed to aid the Production System Coordination Layer

The tool captures information through part activation, feature extraction, and part recognition. It relates recognised parts to customer and process requirements. In the case of MALT, the “Basic leak test” functionality requirements include part volume, test pressure, and leak rate/differential pressure. These requirements are linked to variables for configuration.

#### **For the PRIME Force Test Application:**

**Application of State Charts and State Machine to PRIME:** In the PRIME force test application, state charts and state machines are used to encapsulate the “force test” functionality of the production system. These



state charts and state machines operate at the high-level coordination layer and play a crucial role in orchestrating the execution of the force test in the PRIME test station.

The self-configuration state chart defines the activation of the self-configuration process. It can be initiated by introducing a part into the production system, which is identified based on a fixture, or by requesting the execution of the functionality on a specific part.

Events such as “Identify Part” and “Part req” generate lists of requirements necessary for the part to undergo the functionality. These requirements are related to customer and functionality specifications.

The “Setting Requirement” is a composite state responsible for configuring the customer requirements and process requirements, which in turn define the functionality. Customer requirements are established through events such as “get Customer Requirements” and “set Customer Requirements”. Similarly, events like “get Process Requirement” and “set Process Requirement” relate the customer requirements with the functionality requirements, forming the Process Requirement composite state.

The “Activate Functionality” state triggers the execution of the force test functionality on the PRIME test station. This state governs the identification of variables that need to be configured based on the requirements generated earlier. The values of these variables are determined through the event “set variable value” and are obtained via endpoints, often hosted on Google Cloud.

The encapsulation of these state charts and state machines within an ISA-88 state machine defines the self-configuration process’s different stages, which include starting, executing, holding, unholding, suspending, unsus-

pending, completing, resetting, stopping, aborting, and clearing.

**Capturing Information in Asset Administration Shell:** The Configurable Object (CO) model is used for representing the “force test” functionality in the system, specifically in the context of a force test equipment module (i.e. PRIME test station). The self-configuration adaptation strategy based on a configurable object model captures information on variables, relationships, and constraints using an AAS representation connected through an interface.

The CO model enables the high-level coordination of the production system through state transition rules, using data captured in the mid-level runtime layer. The PRIME force test module consists of digital and physical objects encapsulated within a module representing the force test functionality.

**Tool Utility for Assisting Coordination of Functionality Execution:** The Tool Utility captures information through part activation, feature extraction, and part recognition, which is linked to customer and process requirements for the part. The functionality of the production system depends on its configuration, consisting of variables, their relationships, and constraints.

### 9.3.3 Multi-Agent Integration for Self-Configuration

#### Agent System Deployment on MALT:

The digital information about the physical asset is deployed in the form of Asset Administration Shell. The presented strategy is practically applied to a distributed manufacturing environment. The simulated routing for the

distributed manufacturing environment is shown in the figure 9.9.

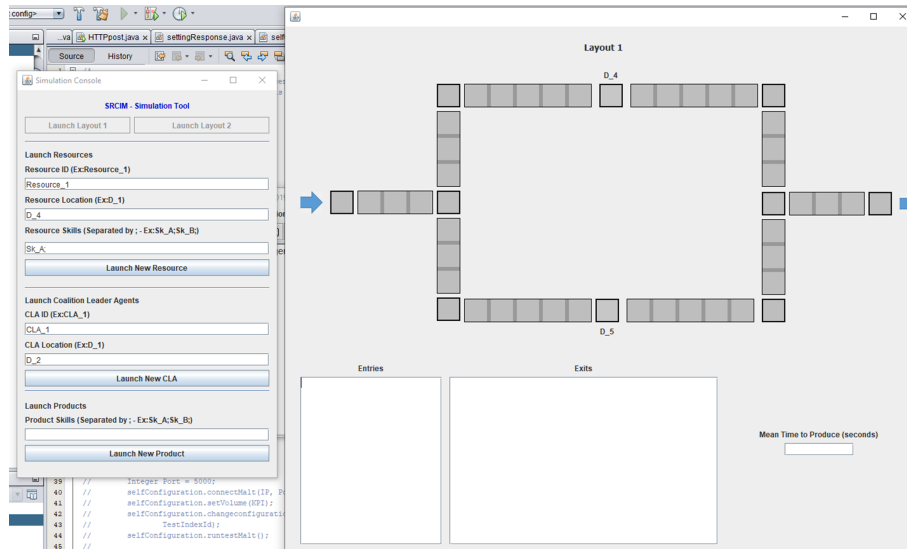


Figure 9.9: Routing design for MALT with an interface for self-configuration demonstration.

The functionality of this layout is linked with multi-agent system execution environment. Initially, resources are allocated to a location through the interface in the simulated layouts. There are three type of agents in the environment namely; Transport Agent (TA), Product Agent (PA) and Resource Agent (RA). Resources, with their skill, name and location, as they are entered through the interface are handled by one RA. There exists one RA per resource. PA is instantiated to represent the product as it is entered into the system. A product is accompanied by a name and the required skills necessary to produce it. Each PA represents one product.

### Agent System Deployment on PRIME Test Station:

A corresponding layout was designed to demonstrate the approach (figure 9.10).

The functionality of the layout is linked with a multi-agent system execution environment. Initially, resources are allocated to a location through

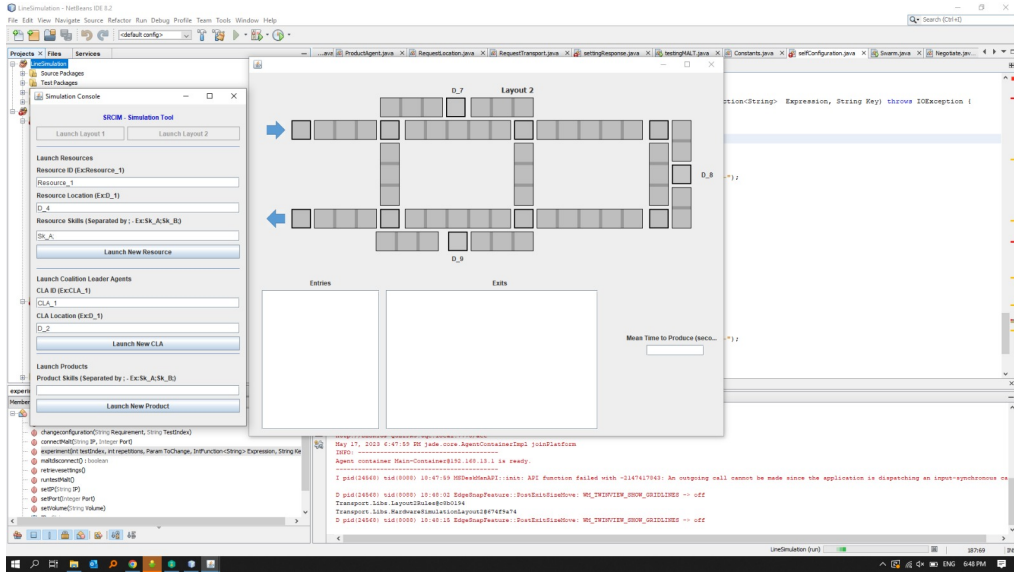


Figure 9.10: Routing design for PRIME with an interface for self-configuration demonstration.

the interface in the simulated layouts. The implementation requires a set of agents with defined responsibilities that are each instantiated when required. Some agents may each have a physical asset associated with it and provides an interface to the virtualized "skills" performed by the physical asset. The list of agents and their functionalities are the same as those defined for the previous use-case.

Agent programming is implemented with the JADE (Java Agent Development Environment) platform like the previous use-case (MALT). The agents deployed in JADE use cloud based classification input generated by vision model trained on the dataset (for fixture identification), as input leading to further actions. All the actors in the production environment are controlled by means of agents.

In the current use case, the agent execution and simulation is performed after the deployment of six agents: PA (Product Agent) , MA (Monitoring Agent), TA1 (Transport Agent), TA2, TA3 and RA (Resource Agent). RA and TA represent the set of stations and their required transportation,

respectively. TA and RA were not deployed individually, rather resources 1, 2 and 3 are represented by agent RA and conveyors by the agent TA, resources being each robot in the setup and a test station.

The simulation is performed within two variants, based on cloud based decision process, set by the MA for the variation in hinge assembly.

### 9.3.4 Deploying Machine Learning Pipelines of Cloud for Self-Configuration

#### Machine Learning Deployment for MALT:

Self-Configuration is essentially system adaptation under changes to restore or improve functionality. The system adaptation component of the self-configuration library takes the guiding parameter and optimises the system device to the best possible settings for that parameter (Algorithm 1). For this to happen it needs to query from established endpoints as the nature of self-configuration is dynamic. These endpoints can be deployed locally to take in configuration settings from the server deployed or may send requests to inquire about those values from endpoints deployed elsewhere (i.e. cloud) (Algorithm 2). These endpoints load these configuration settings after processing on ML/GA pipelines. A simplistic view of algorithm execution is presented (figure 9.11).

**Algorithm 1:** Algorithm for Self-Configuration

**Result:** Test Result

Initialise IP, Port, KPI;

Connect to Equipment(IP, Port);

Set KPI;

Change Configuration;

**while** *Configuration Not Changed* **do**

    Confirm connection;

    Set Response Variable;

    Execute setting.Response(KPI) from Endpoint;

    Send Request and get Response;

    Get Configuration ;

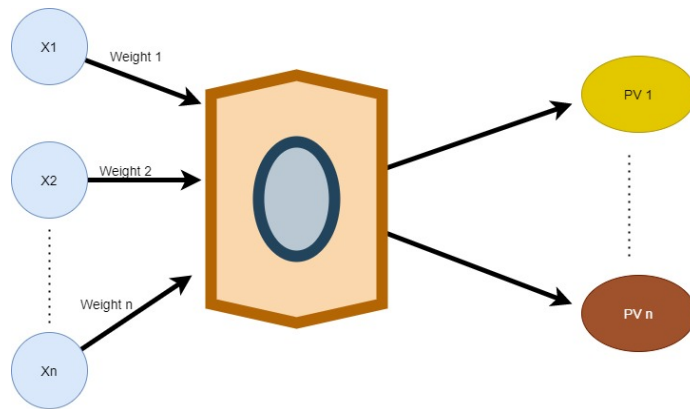
    Load Configuration ;

    Set Parameters ;

    Confirm Configuration as Response;

**end**

Run test;



Input KPI: X1, X2, .. Xn || Weights: Weight 1, Weight 2, .... Weight n || Predicted Value: PV 1 or PV 2 or ... PV n

Figure 9.11: An illustration of the logistic regression algorithm.

---

**Algorithm 2:** ML Endpoint for Self-Configuration

---

**Result:** Predict: String

Initialise Global model;

Initialise Bucket Name, Project Id, GCS Model File;

Initialise Client;

Create Bucket Storage;

Create Blob;

Download file to destination in function Download ModelFile();

**if** *not model* **then**

Download ModelFile() from Storage ;

Assign model = Load File;

**else**

Load File as model

**end**

Initialise Params;

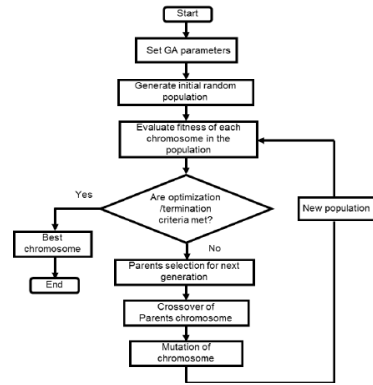
Get Request ;

Generate mode Predict as Predict;

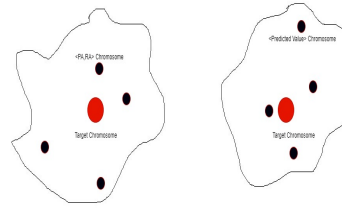
Return Predict;

---

The genetic algorithm (GA) is taken for both selecting both the best possible resource in the Contract Net Protocol (CNP) aspect implementation of the strategy and also selecting the best possible configuration setting value. Figure 9.12 provides an overview of GA. The approach taken here basically makes a Target (Best) Chromosome by combining KPIs in a tuple. Each PA and RA generate their own represented chromosome tuple. The best chromosome is determined based on the aspect of its closeness with the target solution. In the case of configuration setting, the configuration settings have their own chromosome representation and the best chromosome selected gives the configuration setting that is to be loaded. This endpoint approach is useful for selecting a configuration when no historical data is



(a) An overview of the genetic algorithm



(b) Solution Space representation for GA. The algorithm searches for solution chromosomes near to the target chromosome. The best selection is with minimum number of generations to reach the target chromosome

Figure 9.12: An illustration of Genetic Algorithm

present.

The usage of GA and ML pipelines will depend on the product and process requirements, as well as the presence of historical data. If no historical data is present, then a configuration may be selected through the GA pipeline and improved with iterations. In manufacturing setups, if historical data is accumulated, then ML pipelines can be used to query the optimum configuration. The advantage of ML pipeline over GA pipeline lies in giving the optimum configuration that can be used to execute functionality, while in the case of GA it will be an approximate configuration that needs some iterations to reach the optimum. GA pipeline could be very beneficial in evaluating the most suitable resource (i.e. asset) to execute functionality in the system with changing needs (e.g. priority, cost, buffer or other KPIs). Once the suitable resource is approximated, ML pipeline could determine the best optimum configuration for the product. The suitability of the resource is evaluated and improved over time with iterations.



**Machine Learning Deployment for PRIME test station:**

Cloud computing is used to enhance the capability of the adaptation strategy by bridging it with service-oriented architecture. The agent looks for certain events that act as a trigger for it to execute its functionality. ML pipeline housed in the cloud can be instantiated by means of event trigger functions of the PLC through agents. The agent and the cloud platform rely on a gateway to realize their functionality. The event trigger functions are used by the MA to activate the capture of images of the data matrix on the fixture and send them to cloud storage. This population of images in cloud storage triggers additional functionality and decision-making by the ML pipeline. The insight generated by the ML pipeline, here the configuration settings, based on the captured images in the cloud platform, is sent to the station which then uses it to execute an operation.

The algorithm for ML processing used for image detection and classification is Neural Architecture Search (NAS) where a dataset and task (image detection and classification) are provided. This is used to find the design of a machine learning model, that performs best among all other models for a given task as the model is trained under the provided dataset. NAS uses a search strategy to find the best model from all possible models that maximise performance. Figure 9.13 illustrates the NAS algorithm.



Figure 9.13: NAS algorithm for determination of image detection and classification ML Model

The three constituents of NAS include search space, search strategy and performance estimation. Search space defines the neural architecture selec-

tion basis like chain or multi-branch network, micro/macro-search or cell-search (Elsken et al. (2019)). Search strategy and performance estimation employ multiple methods selected on the search space selected previously (Elsken et al. (2019)) such as random search, reinforcement learning, and evolutionary algorithms. The model derived from this approach can be used directly for the purpose. Google Cloud Platform (GCP) based its AutoML service on a novel architecture NASNET that uses NAS for image classification. NASNET redesigns the search space, so the best layer can be found and stacked multiple times in a flexible manner for the final network. This network was used to perform a search strategy on image datasets and best best-learned architecture was selected for image detection and classification. More detail on the work can be found in the work done by the Google Research Team (Zoph et al. (2017)).

#### 9.3.5 Real-Time Control of Test System

##### **Control of MALT:**

Realisation of the developed framework is carried out through an implementation that connects the manufacturing system to the data pipeline components. A smart controller is necessary for realisation of the proposed approach, i.e., establishing a connection and communicating with pipeline components. Data collected by the controller is stored locally or externally, converted, and transmitted to the pipeline where it is processed by the pipeline components. As per the sequence diagram, runtime conditions are checked and system adapted as per objective definition through prediction serving component. The agents control the functionality of the manufacturing system.

A data pipeline was set up that connected the components to the controller of an industrial leak testing rig TAMI (Test bench for leakage identification on aircraft fluid mechanical installations) and dry-air leak testing device MALT (Micro Application Leak Test). Agent systems were utilised based on JADE (Java Agent Development Framework) that control individual skills.

#### **Control of PRIME Test Station:**

The developed solution was implemented by connecting the PRIME test station to the data pipeline components, which involved a Beckhoff PLC. The PLC established a connection and facilitated communication with the pipeline components. The data collected consisted of images of the data matrix of part fixtures stored on the GCP cloud environment. These images acted with the ML models to deliver insights to the agents (on PLC) for configuration settings. A prediction serving component checked runtime conditions and adapted the system (i.e. changed configuration) based on objectives and part requirements.

In the PRIME force test case, a data pipeline was set up, connecting the PLC control to the decision-making ML endpoint through agent systems. Agent systems based on JADE were utilized to control individual skills. In this case the skill, or functionality, of the force-test is used.

Force test, in PRIME test station, involves the application of a force of a certain value maximum on the hinge assembly. The force is maintained for a certain time period, i.e. the hinge is exercised. The deformation/deflection is measured under force application. The resulting deformation/deflection result indicates the pass/fail status of the product.

The test station skill “force test” and the location was entered and assigned in the layout. Each manufacturing system corresponded to one RA (Resource Agent). The product skill requirement, such as force-test, was entered through an interface instantiated by a PA (Product Agent). The PA instructed the TA (Transport Agent) through CNP (Contract Net Protocol) to transport the product to the manufacturing system location. As the product reached the location, the system coordinated with the data pipeline components to adapt the manufacturing system and achieve self-configuration. The experimental setup and agent system execution can be visualized through the provided figures.

The PLC controller played a crucial role in implementing the self-configuration features. It recorded changes on its IO module and sensors (camera and force), could be calibrated, connected with other systems, and stored data for processing. The controller’s capabilities included data collection, data storage, data transmission through TCP/IP communication with other systems, and operational identification. The PLC controller was integrated with an agent system that coordinated with the data pipeline to achieve self-configuration goals. The data pipeline handled optimization functionality for the test system, considering runtime conditions, objectives, conditions, and system adaptation through agents and prediction serving components (ML Endpoints).

## 9.4 Experiments

### 9.4.1 MALT

#### Introduction

This section demonstrates the effectiveness of the developed self-configuration strategy for the MALT leak testing system. The experiment is designed to validate the system's ability to dynamically adapt the leak testing process in response to the following challenges:

1. **Variability in Parts:** This experiment tests the data and conceptual models developed in this research (Objective 1). By adapting testing parameters to parts of different volumes, the experiment will assess the ability of the adaptation strategy to capture and utilise configuration information. It will also examine how well the system can dynamically identify and adjust to varying part features, validating the configuration abstraction of the system elements (Objective 2).
2. **Dynamic Production Requirements:** The experiment addresses the system architecture objective (Objective 4). By evaluating the system's ability to maintain optimal configuration in the face of changing priorities, the experiment will assess the modularity and adaptability of the architecture.
3. **Reduction of Expert Dependency:** This experiment is designed to validate the implementation strategy (Objective 5). By minimising the need for specialised knowledge in leak testing, the experiment will test the system's ability to seamlessly adapt the system settings

based on the defined configuration rules (Objective 2). This will demonstrate the practical application of the identified enabling technologies (Objective 3).

#### **Application of Level-Based Classification on MALT:**

MALT (Multi-Application Leak Test) is a leak test module developed by an SME (TQC Ltd.) that provides the ability to test leaks in a part under pressure. Level-based classification is applied to identify the features that can be improved through stage-wise transition for achieving self-configuration. figure 9.14 illustrates the application of level-based classification.

#### **Capturing Information on MALT:**

MALT, as a leak test production system, is represented as a module that has the functionality of a leak test. The MALT leak test equipment module consists of objects that encapsulate digital and physical aspects. The CAEX Engine developed can capture configuration pertaining to the functionality of the leak test for MALT.

The AAS for the MALT captures the configuration for the functionality leak test in terms of variables, their relationships, and constraints. For the MALT, as a configurable module, variables capture the configuration settings, and relationships are represented using “Allocation” and “Link”. These variables are connected in terms of the leak test functionality. The constraints on the functionality are represented in the functionality and variable rules.

The CAEX engine is connected to AAS representation for the leak test functionality. The AAS is connected to the MALT through an interface.

		MALT System	
Data Management	Self-Capability	Self & context Awareness S4   L2	MALT is capable of determining its state, function and components (fixture) attached. It has necessary mechanism in-place for self-management.
	Data Collection	Combine useful data from multiple data sources S5   L3	MALT can combine data sources to give an outcome i.e. pass/fail criteria based on objective.
	Data Storage	Stored locally and some also at network S3   L2	Configuration data is stored locally in memory and also test data is populated on server
	Data Conversion	Useful information extraction from data S3   L2	The system extracts useful information from data to develop inference for outcome.
	Data Transmission	Data transmission and linked to knowledge-base S4   L2	Data is available in structured and transmissible format with other devices or even cloud-based services.
	Operation Identification	Cyber physical system level awareness S4   L2	MALT has complete cyber-physical awareness along with operation identification based on sequence entered and configuration selected for execution.
	Optimisation	Utilisation mechanism not present S1   L1	Currently optimisation functionality does not take into account the objective. Optimisation depends on the operator.
	Monitoring	Monitoring of collected information S3   L2	MALT is able to monitor accurately the configuration and test data.
	Control	Partial-automated control S3   L1	The system can carry out the test but human input is required for selecting the configuration for execution and also starting the test. Also, the operator has to acknowledge the result.

Figure 9.14: Level Based Classification for Self-Configuration applied to Multi-Application Leak Tester (MALT). The identified features need to be improved to enable MALT to achieve self-configuration. This happens through Stage-Wise Transition explained before.

Therefore, the CAEX engine assists in deploying AAS for the production system and configuring it as per functionality requirements. Figure 9.15 illustrates the captured information through CAEX Engine for MALT system components.

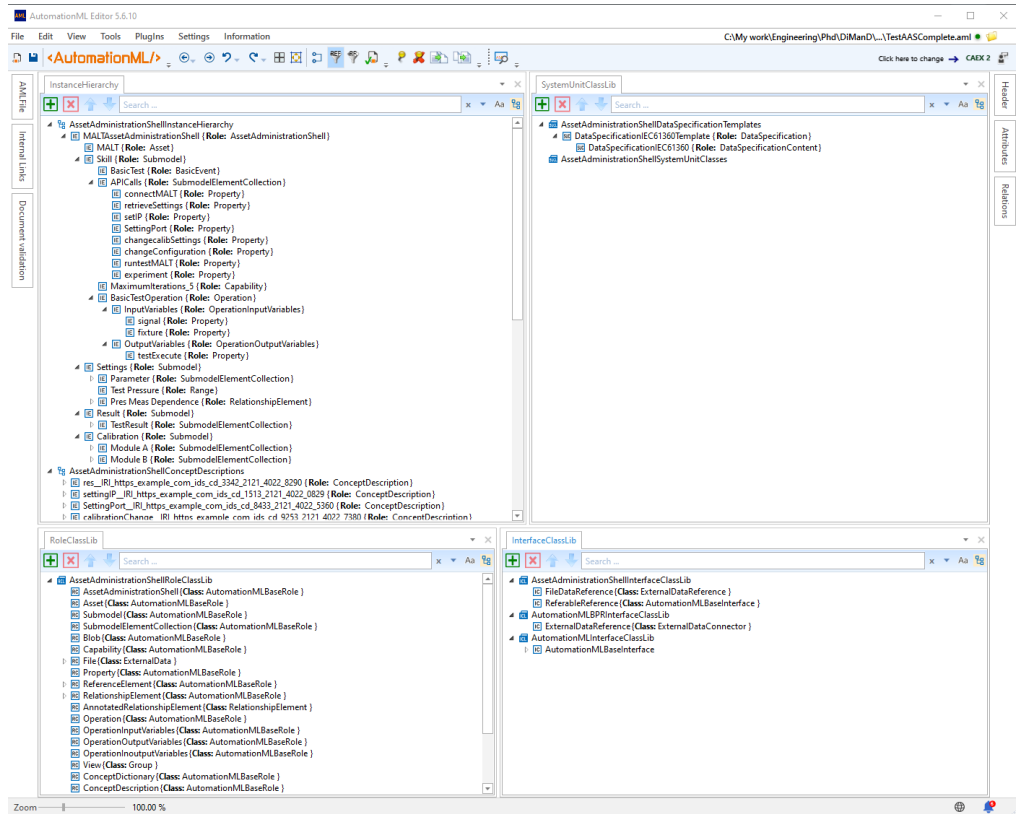


Figure 9.15: Capturing MALT system information in CAEX Engine.

### Representing Information in AAS:

As the AAS for MALT is updated through the CAEX Engine, the configurable module connected is also updated through the interface. The dynamic configuration change is established as a change instantiated by the CAEX engine in the AAS elements and is translated to the MALT system.

Figure 9.16 shows the AAS representation for MALT. The AAS consists of submodels “Skill”, “Settings”, “Result” and “Calibration”. The submodel “Settings ” contains information about the setting parameters. These parameters need to be configured for executing functionality. CAEX engine configures these parameters based on requirements captured through state charts.



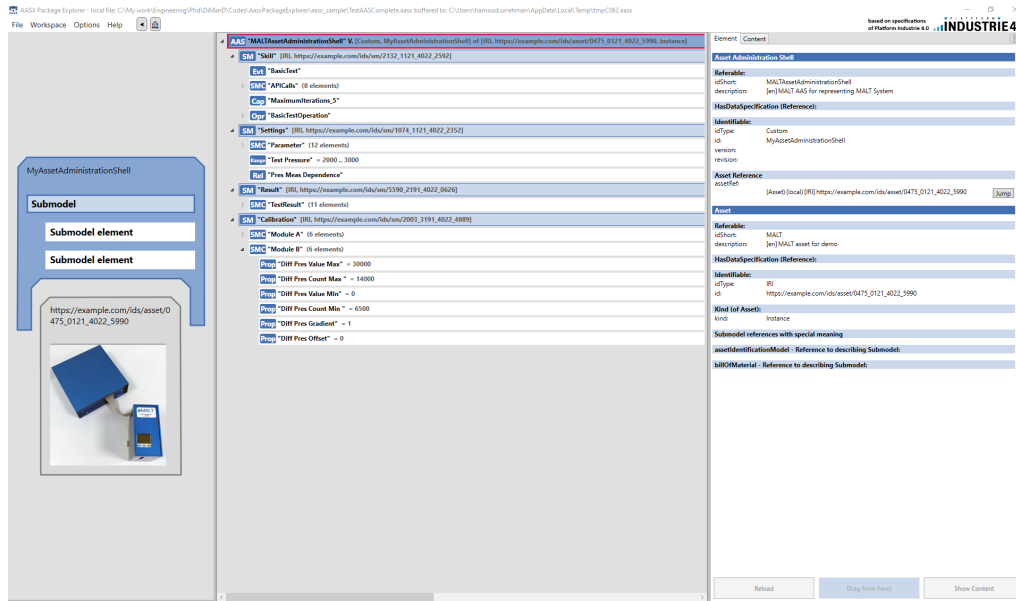


Figure 9.16: AAS for MALT

This submodel also contains information on the range of certain parameters (e.g. pressure) in the system, along with the dependency relationship. The submodel can be updated in the AAS through CAEX engine and interact with the MALT system through submodel json file. This means that the interface in the MALT is capable of updating its configuration through each submodel, and does not entail that the whole AAS be provided for updating the configuration. However, this also means that there exists a strict structural requirement on the submodel that is provided to the interface. Figure 9.17 represents the structure of the submodel that must be provided.

Figure 9.18 details the components of the submodels. The submodel “Calibration” contains information on the calibration data for the sensors on the production system. In the case of MALT, the calibration data corresponds to the two pneumatic modules present in the equipment capable of performing tests. The submodel “Results” captures the result of each functionality executed (i.e. test carried out) in a submodel collection. The “Skill” submodel houses the information related to the leak test function-

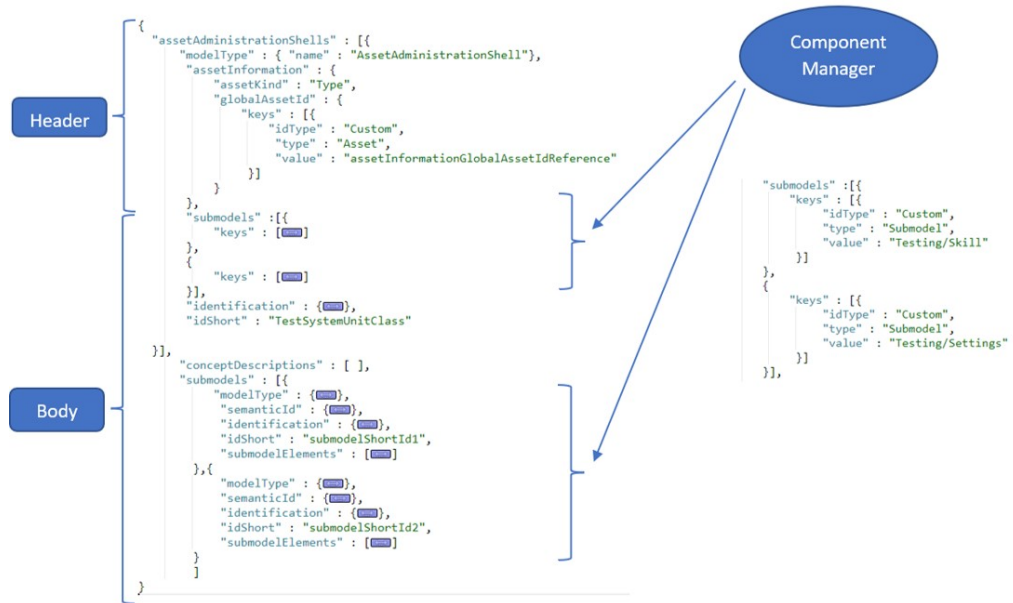


Figure 9.17: Generic asset administration shell representation (left). Skill submodel is developed, submodel elements contains the skill API calls along with pertaining data that can be used by the client services. Component manager is deployed when the submodels are listed (right).

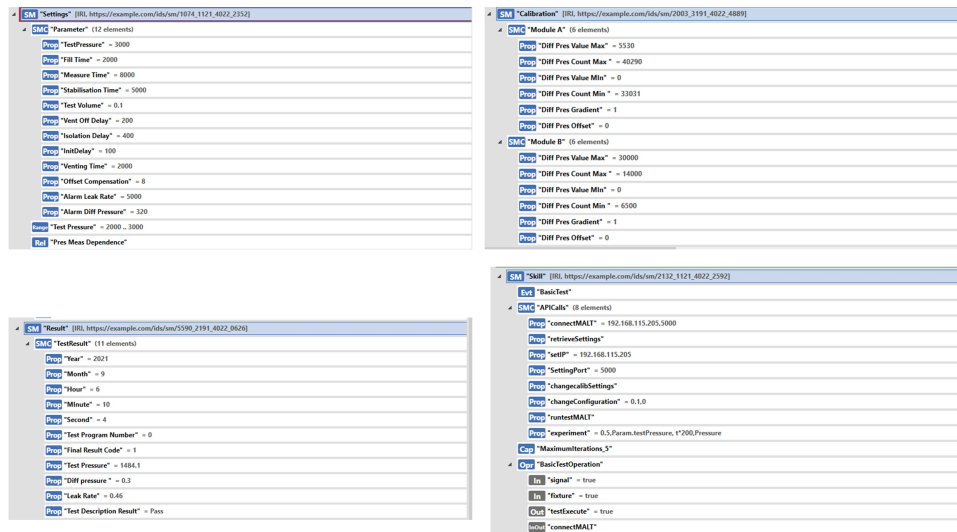


Figure 9.18: Detail of each submodel to execute “leak test” functionality.

ality as the event “Basic Test”. The API Calls that use the configuration to execute the functionality are stored as a submodel collection. The capability of the functionality and the detail of the functional operation are also stored in this submodel.

### **Tool Utility for Coordinating Leak Test Functionality:**

The Tool Utility generates a variable list based on functionality corresponding to its configuration, consisting of variables, relationships, and constraints. For MALT, the "Basic leak test" functionality is set through the tool. The values of these variables are identified through machine learning pipelines hosted at endpoints and assigned to the production system for execution.

The deployment of variable values involves linking to the system and executing the functionality with the updated configuration.

### **Experimental Methodology**

**Test Plan:** A comprehensive plan was designed. The details of the plan are as follows:

**Objectives:** The experiment aimed to determine the best suitable configuration setting for leak testing.

Leak testing is the process of determining the leakage flow rate from a part under test. The leak-testing production system consists of multiple processes. "Basic Leak Test" involves filling the part under the test process (i.e. over-pressure) with a testing medium or subjecting it to a complete vacuum and allowing it to stabilise under pressure. The pressure change is measured under leakage flow. This change in pressure or for simplicity proportional value of leakage flow rate is used as an indicator to pass or fail on part. There are several variations for leak testing such as differential pressure measurement pressure decay, dosing leak test, chamber leak test, blockage testing, and coarse or gross leak test (Mount (2015)).

Testing is carried out at normal room conditions on a clean, dry, room temperature stabilised product. These conditions are maintained devoid of excess moisture due to the temperature-sensitive nature of the dry-air leak testing consideration. This testing process plays an important role in ensuring product quality, product's integrity, safety, and reliability.

The test products consist of multiple test volumes 0.1L, 0.2L, 0.4L, 0.8L, 3L and 10L. The experiment aims to validate the adaptation strategy for volume-based configuration determination under KPIs.

**Test Case:** The product is tested to determine the most suitable leak test configuration setting. This test's main focus is ensuring the adaptation strategy correctly identifies volume and sets the corresponding leak test parameters. It should demonstrate that the system can maintain consistent pass/fail outcomes under constraints and do so efficiently in comparison to manual configuration settings. The test case can be briefly detailed as:

- **Basic Volume Identification:** The system determines the cylinder volume for which the test is to be performed.
- **Volume Range Adaptation:** The system identifies the suitable configurations for executing the test on the volume under test, i.e. candidate configuration.
- **KPI Constraint:** The system identifies the most suitable configuration setting based on KPI constraint. Here, in this experiment, it is taken as the accuracy of the configuration setting for giving the correct leakage rate.
- **Configuration Adaptation:** The system adapts the new configuration setting based on KPI constraint from the candidate configu-

ration.

**Performance Metrics:** The tests performed will measure:

- Accuracy of volume identification
- Accuracy of the resulting configuration (compared to expert-derived settings)
- Configuration time
- Pass/Fail outcomes against the configuration settings for the product volume

**Baseline Establishment** Before introducing self-configuration capabilities, a baseline assessment of the MALT system's performance was conducted. A series of leak tests were performed using manual configuration on a set of representative products with volumes ranging from 0.1L to 10L. These baseline results will serve as the benchmark against which the self-configuring system's improvements will be measured. Figure 9.19 summarises the experimentation carried out for developing baseline and data-set for MALT.

### Validation

- **Performance Comparison:** Leak test configuration settings were analysed against the baseline, to validate the accuracy of settings.
- **Expert Review:** Configuration settings were validated by a domain expert to ensure correctness and optimality.

## 9.4. EXPERIMENTS

Parameters	Value						Experimentation	
	0.1 L	0.2L	0.4L	0.8L	3L	10L	Variation Range	Description
Test Pressure	1550 mbar	1550 mbar	1550 mbar	1550 mbar	1550 mbar	1550 mbar	1000 mbar-3000 mbar	1000 tests
Initdelay	0.1 sec	0.1 sec	0.1 sec	0.1 sec	0.1 sec	0.1 sec	No change	No change
Vent Off Delay	0.2 sec	0.2 sec	0.2 sec	0.2 sec	0.2 sec	0.2 sec	0.1-0.3	1000 tests
Fill time	2 sec	10 sec	12 sec	18 sec	20 sec	40 sec	1 sec - 50 sec	1000 tests
Stabilisation Time	25 sec	60 sec	60 sec	90 sec	60 sec	70 sec	10 sec - 75 sec	1000 tests
Isolation Delay	0.4 sec	0.4 sec	0.4 sec	0.4 sec	0.4 sec	0.4 sec	0.01- 1.0	1000 tests
Measuring Time	2 sec	20 sec	20 sec	20 sec	30 sec	50 sec	0.05 sec - 50 sec	1000 tests
Venting time	2 sec	10 sec	12 sec	20 sec	20 sec	40 sec	0.05 sec - 50 sec	1000 tests
Offset Compensation	8 mbar/sec	5 mbar/sec	5 mbar/sec	5 mbar/sec	5 mbar/sec	5 mbar/sec	1 mbar/sec - 100 mbar/sec	1000 tests
Alarm Leak Rate	50 mbar/sec	15 mbar/sec	21 mbar/sec	40 mbar/sec	40 mbar/sec	40 mbar/sec	1 mbar/sec - 50 mbar/sec	1000 tests
Alarm Diff Pressure	32 mbar	50 mbar	50 mbar	50 mbar	50 mbar	50 mbar	1 mbar - 50 mbar	1000 tests
Others	Requirement specific						Combination	1076 tests
							<b>Total Tests Performed</b>	<b>11076 tests</b>

Figure 9.19: Summary of experimentation carried out for developing baseline and data-set for self-configuration on MALT. The parameters for each volume are varied over the range mentioned. The range is volume and MALT-specific (constraints).

### Experimental Setup

#### Hardware

- MALT leak test system
- Representative product samples (0.1L, 0.2L, 0.4L, 0.8L, 3L and 10L).

#### Software

- Agent system
- CAEX engine
- ML endpoints for configuration generation
- MALT interface API for data logging and analysis

### Experiments and Results

**Goal:** Demonstrate self-configuration in response to varying product volumes.

The goals for the MALT experimentation can be summarised as follows:

1. **Configuration Capture:** Capture the configuration settings for leak test functionality in terms of variables, relationships, and constraints through the information model, and CAEX engine. The captured information should be compliant with the AAS standard.
2. **Self-Configuration Capability:** Using the features identified during level-based classification, incorporate self-configuration capability by improving features.
3. **Optimisation and Adaptation:** Using adaptation strategy and ML/GA endpoint deployment achieve self-configuration.
4. **Agent System:** Implement an agent system, as enabler, for coordinating with data pipelines to achieve self-configuration goals.
5. **Cloud-Based Data Pipeline:** Host data pipeline components in a cloud environment (e.g., Google Cloud Platform) for better data processing and coordination.

**Procedure:** A sequence diagram that completely represents the interaction of product in the environment is presented in figure 9.20.

The sequence diagram (figure: 9.20) shows the flow of information between the agents during the operation. The agents are asset administration shell (AAS), system device (MALT as a Use-case), Transport Agent (TA), Product Agent(PA) , Resource Agent (RA), ML Endpoint on Cloud (Configuration Library, Database/ML Data Pipeline) and DF Service (for querying and registering). The step-wise approach to self-configuration in the developed strategy is presented as follows;

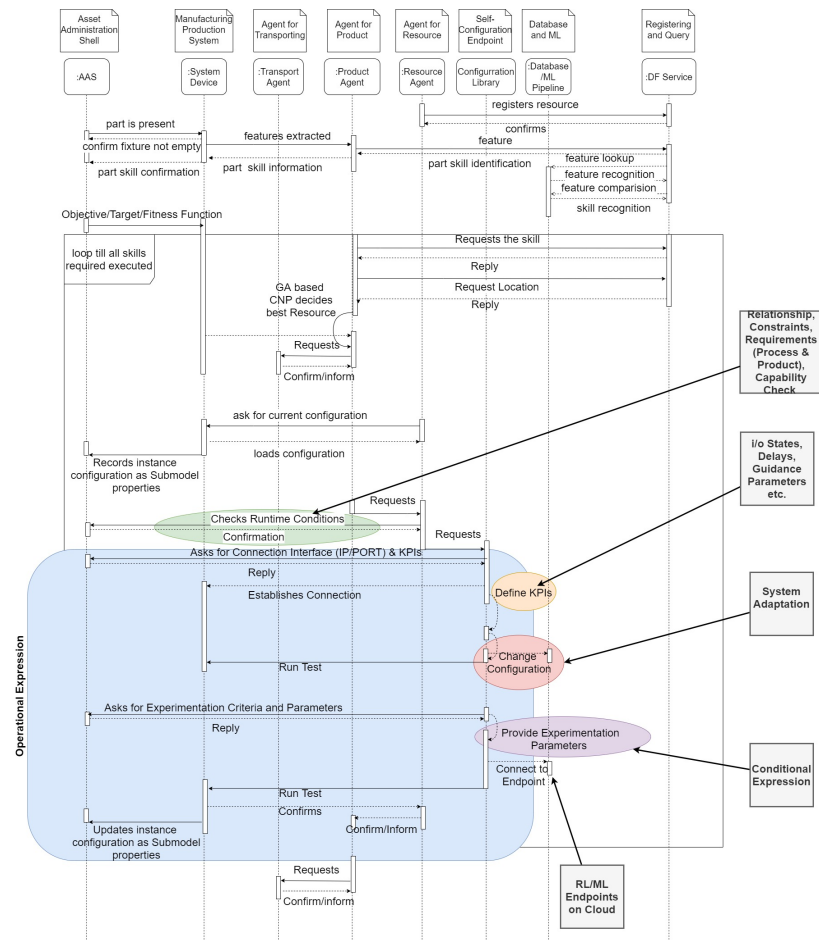


Figure 9.20: Sequence diagram for a distributed manufacturing environment with self-configuration and testing.

- At the start of the operation all the resources register with the Directory Facilitator (DF) Service.
- The AAS contains information about the product introduced in the system, its criterion, KPIs, objectives and all other representative information.
- These information features are extracted and relevant skills are identified.
- An objective/target/fitness function is fetched from the asset administration that corresponds to the target for the product being produced.



- The PA requests the DF Service for the skills and find the RA that is able to execute the skill. It then inquires the location of the skill. The PA selects the best resource to execute the skill based on Contract Net Protocol (CNP) negotiation with the RA by using the genetic algorithm (GA).
- The product is transported to the selected resource. and current configuration of the resource loaded and recorded in AAS as a submodel.
- The run-time conditions are checked for the resource (RA) with regard to the product (PA). The run-time conditions include relationship, capability, constraint, and operation requirements. If all the necessary conditions are satisfied, then the skill is able to be executed on that resource.
- The Operational Expressions are loaded that define the skill requirement as well as sequence for the self-configuration assisted test to happen.
- The Operational Expression is executed that establishes the connection with the system device. The KPIs (key performance indicators) are loaded for the system to be configured according to it. KPIs serve as guiding parameters.
- The system is adapted (configured) as per KPI provided. The test is executed. The configuration parameters are queried individually from the ML/GA endpoints deployed.
- An experimentation is planned by providing the experimentation criteria along with the parameters necessary. These criteria and parameters are loaded that define the manner of self-configuration. These are basically provided/loaded in the form of conditional expressions.

- The test is executed, and the configuration instance is updated in AAs.
- Once the experiment is performed, the RA confirms the PA of skill execution and the PA requests the TA to move to perform next skill.
- If no more skill is required then PA requests the TA to move the product out of the system.

The run-time condition checks give depth to operation execution (skill execution). This could be best understood as a check on the current state of the system device. Considering that a part needs to be picked up by the robot. Although the robot has the capability to pick up the part if it is already in the state of holding another component the check condition will return as a negative. Similarly, the approach checks the state condition against relationship, constraint, capability, and specific requirements (i.e. product and process). This matching of conditions is done by the respective AAS of the resource against the AAS of the product. This approach is taken as AAS of both resource and product are updated regularly during execution to account for the influence of changing state during operation.

The Self-Configuration library that is developed relies on the input from the User that is fed through AAS for the product and resource. The KPIs can be one entity or many, depending on the desired criteria. KPIs in the self-configuration approach can be I/O states that can drive PLC control, delay parameters for feedback control or guidance parameter such as a unique feature or business parameter. KPIs serve as an anchor for self-configuration, i.e. the self-configuration library configures the system device based on these guiding parameters. Figure 9.21 illustrates the self-configuration library formulation with the endpoints hosted in the cloud.

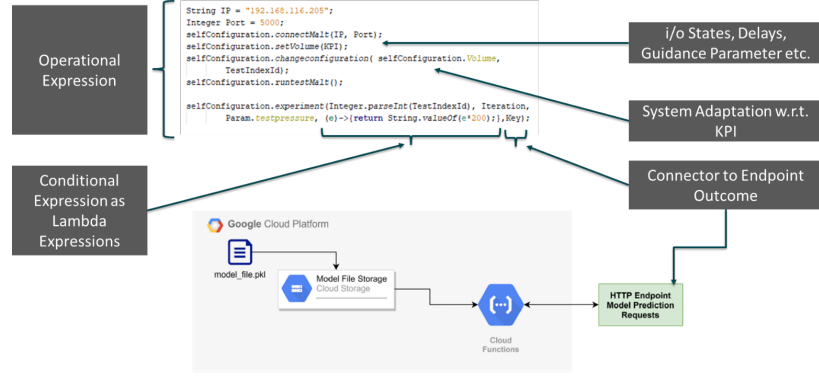


Figure 9.21: Self-Configuration library formulation with Endpoint hosted on Google Cloud Platform (GCP). Developing parametric optimisation code that is modular and can execute an experiment in 4-5 lines.

The ML Endpoint for self-configuration of the current system device involves logistic regression. The model trained is deployed on the Google Cloud platform endpoint, trained on a data set for 0.1L-3L test volumes and can be processed through the library directly. The connector in the library extracts the settings based on the “key” value (figure 9.21). This method extensively relies on the baseline data set to make accurate predictions for the configuration settings. For this experiment, the configuration settings for a test volume of 10L were predicted using the endpoint through the library.

**Expression for Leak Test Functionality Execution:** Product Agent (EA1) looks for the skills required for testing based from an “operational” expression. The “operational” expression lists down the skill required by product being tested as well as the sequence of execution for that part. This expression is stored in cloud based and loaded by EA1 based on part identification.

$$T_{EA1} = C_{\phi} \text{basic}.C_{\phi} \text{cond}.C_{ST1} \text{stab}.C_{FA1} \text{fill}. \tag{9.1}$$

$$C_{\phi} \text{param}.C_{\phi} \text{iter}.T_{EA1}$$

The EA1 searches for the agents responsible for these skills. These agents then look for the “conditional” expressions that define the behaviour on how these skills may be executed.  $C_\phi$  is a null condition that means no condition expression is present for executing the skill. So the expression may be simplified;

$$T_{EA1} = basic.cond.C_{ST1} stab.C_{FA1} fill.param.iter.T_{EA1} \quad (9.2)$$

TI1 agent is responsible for “basic” skill that starts the connection with the test system, CA1 executes “cond” skill that uses condition to change test parameters, ST1 changes the stabilisation time through “stab” skill, FA1 and PA1 induces change to fill time and any parameter (taken as argument) through “fill” and “param” skill respectively. IT1 is the agent responsible for “iter” skill that gives number of test iterations to be performed and executes the tests as per those iterations. Giving each agent responsibility of individual skill gives capability to define test for each part driven by “operational” expression. Here, TI1, CA1, ST1, PA1, FA1, and IT1 are a group of agents previously established as resource agents.

These agents once they have received request to execute a skill look for a conditional expression from cloud service. This expression is loaded and the skill behaviour modified as per expression. For the purpose of this test the fill time and stabilisation time was varied by each increment. The conditional expressions are;

$$C_{FA1} = i * fill \quad (9.3)$$

where  $i = 1..i$  increments

$$C_{ST1} = stabilisation\ time + i * t \quad (9.4)$$

where  $i = 1 \dots i$  increments and  $t = \text{constant}$

These expressions are loaded from cloud service and execution of the test carried out.

**Executing Leak Test Functionality:** Leak testing process involves the process of determination of leakage flow rate of product under test. The product being tested is subjected to complete vacuum or filled under pressure. The product is then stabilised over time. The differential pressure is measured against a reference. The differential pressure or proportional leakage flow rate value gives the part or fail for the product. A corresponding layout is designed for demonstrating the approach working.

The skills and the location of the manufacturing system are entered and assigned in the layout. Each manufacturing system is corresponding to one RA. The requirement of product skill is entered through the interface instantiated by a PA, here it is leak testing. The PA instructs TA, after negotiation through CNP, to transport the product to the manufacturing system location. As the product reaches the location, the system coordinates with the data pipeline components to adapt the manufacturing system, configuring as be defined objective and achieving self-configuration. The experimental setup and agent system driving execution can be visualised through figure 9.22.

A simple demonstration of the approach working can be witnessed through the figure 9.23 and the sequence diagram through the figure 9.24. The configuration change of the manufacturing system before and after can be seen in figure 9.25.

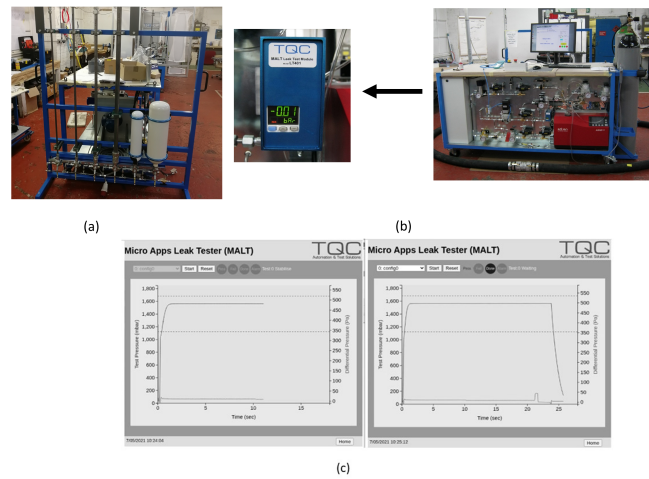


Figure 9.22: The leak testing setup: (a) the cylinder volumes under test (b) MALT test system being a part of test bench for general leak testing (c) Interface for leak testing; agent system drives the execution.

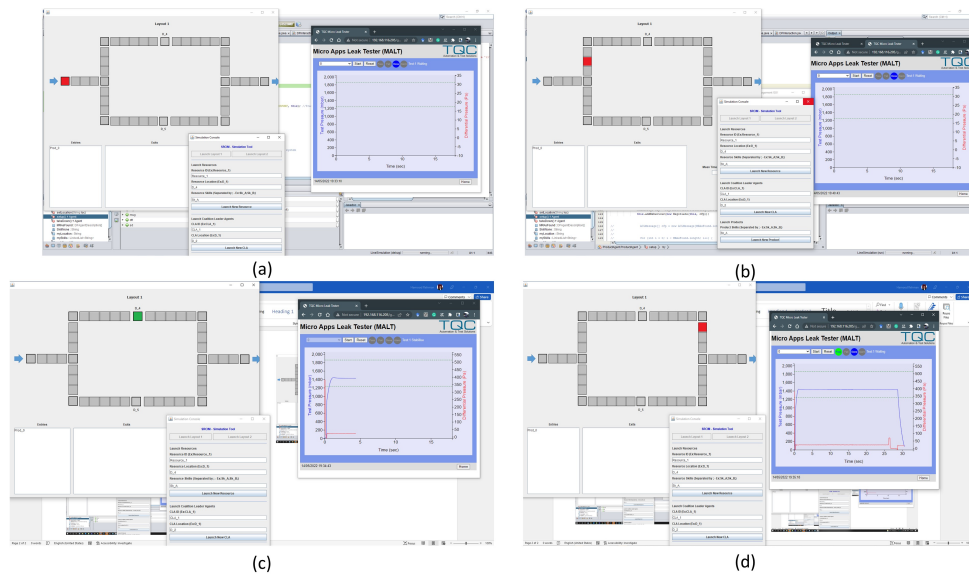


Figure 9.23: Execution of the self-configuration framework on MALT system. (a) PA (representing a product) is entered into the system. (b) TA transports the product to the manufacturing system (MALT) location. (c) Manufacturing system is self-configured and test is executed by agent coordination through components of data pipeline.(d) TA transports the product from the system once test is carried out.

**Results:**

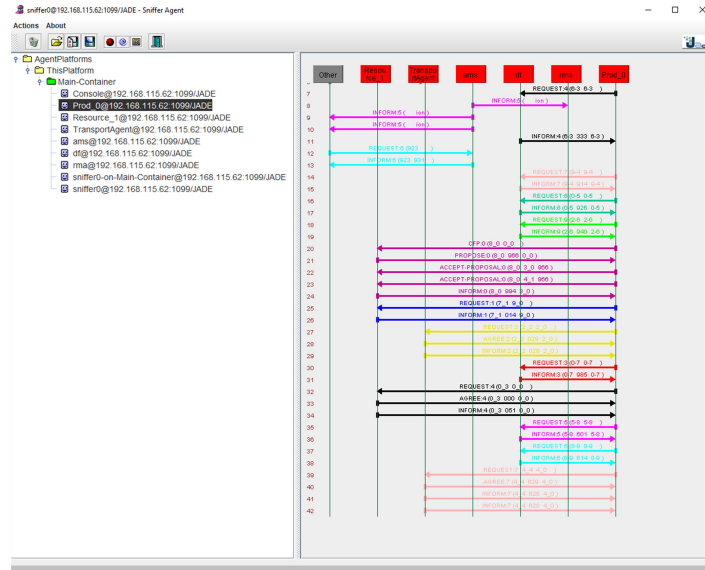


Figure 9.24: Sequence Diagram for self-configuration and Test Process (JADE Interface).

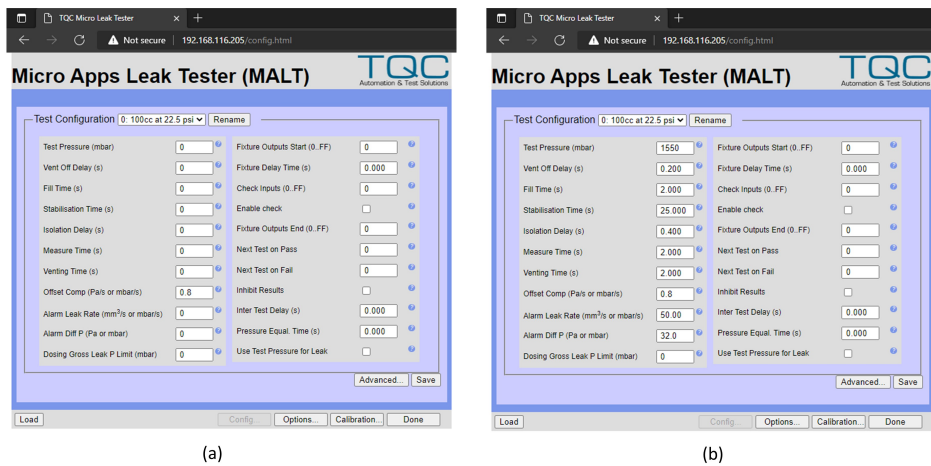


Figure 9.25: Change in configuration settings of the manufacturing system (MALT). (a) No configuration setting (b) Configuration change after execution before test.

**Validation of Adaptation Strategy on MALT:** This approach provides a means to adapt MALT to product, process, and KPI requirements. As an advantage, this approach assists in achieving self-configuration for leak test functionality on the MALT by updating configuration specific to requirements. Figure 9.26 provides a comparison on the configuration achieved through the deployed approach and the configuration derived

through an expert.

Parameters	Value						Time Taken	
	0.1L	0.2L	0.4L	0.8L	3L	10L	Expert	Adaptation Strategy
Test Pressure	1550 mbar	1550 mbar	1550 mbar	1550 mbar	1550 mbar	1550 mbar	120 sec	0.1 sec
Initdelay	0.1 sec	0.1 sec	0.1 sec	0.1 sec	0.1 sec	0.1 sec	60 sec	0.1 sec
Vent Off Delay	0.2 sec	0.2 sec	0.2 sec	0.2 sec	0.2 sec	0.2 sec	60 sec	0.1 sec
Fill Time	2 sec	10 sec	12 sec	18 sec	20 sec	40 sec	900 sec	0.2 sec
Stabilisation Time	25 sec	60 sec	60 sec	90 sec	60 sec	70 sec	3600 sec	0.2 sec
Isolation Delay	0.4 sec	0.4 sec	0.4 sec	0.4 sec	0.4 sec	0.4 sec	60 sec	0.1 sec
Measuring Time	2 sec	20 sec	20 sec	20 sec	30 sec	50 sec	3600 sec	0.15 sec
Venting Time	2sec	10 sec	12 sec	20 sec	20 sec	40 sec	30 sec	0.1 sec
Offset Compensation	8 mbar/sec	5 mbar/sec	5 mbar/sec	5 mbar/sec	5 mbar/sec	5 mbar/sec	140 sec	0.2 sec
Alarm Leak Rate	50 mbar/sec	15 mbar/sec	21 mbar/sec	40 mbar/sec	40 mbar/sec	40 mbar/sec	7200 sec	0.2 sec
Alarm Diff Pressure	32 mbar	50 mbar	50 mbar	50 mbar	50 mbar	50 mbar	3600 sec	0.2 sec
Total Time To Test Setting Determination							19370 sec	1.65 sec

Figure 9.26: Validation of Adaptation Strategy for leak test functionality on MALT. The time taken to determine a valid configuration is significantly less through the research approach in comparison to an Expert. Also, the approach allows updating this configuration automatically, while the expert has to enter the values manually.

The 10L volume was presented to the system as a test volume, and then configuration settings were determined by the system and the expert. The time taken by both to reach the optimal configuration under constraint was recorded (figure 9.26). The adaptation strategy was able to configure and execute functionality in 1.65 sec in comparison to an expert that took 19370 sec respectively.

Figure 9.26 demonstrates the application of self-configuration on the test volumes (from the experimentation section). The same configuration was determined as optimal by the expert in a significantly longer duration. This was carried out in an industrial setting, providing a clear improvement and significant cost-saving.

**Mapping to Research Questions:** The implementation is mapped to the research questions established of the research project as seen in figure 9.27.



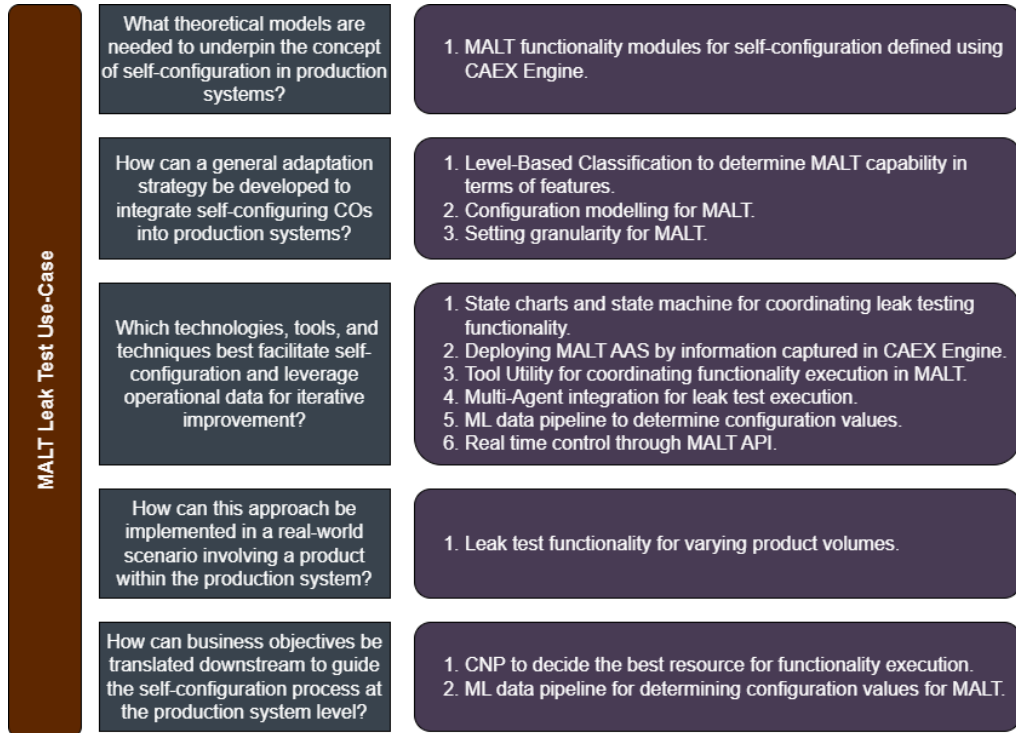


Figure 9.27: Mapping project research questions to MALT Use-Case Implementation.

### Overall Findings

The results of the MALT experiments validate the research by achieving its objectives in the following manner:

1. **Theoretical Foundations:** The adaptation of the leak test system to the varying volumes and utilisation of data to determine configuration settings demonstrates the usefulness of the developed data and conceptual models.
2. **Configuration Abstraction:** The adaptation and leak test functionality execution under changing volumes validates the abstraction developed for the configuration and the self-configuration logic.
3. **Enabling Technologies:** The reduction of expert dependency for determining leak test settings and the implementation of adaptation

strategy through the developed tools confirms usefulness of enabling technologies for self-configuration in MALT.

4. **System Architecture:** MALT modules and adaptability to suit the leak test requirements show the ability of the system to adapt for functionality execution based on the developed architecture.
5. **Implementation Strategy:** The self-reconfiguration of MALT for leak testing and reduced expert reliance validates the implementation strategy.

#### **Level-Based Classification on MultiAgent and ML Integrated MALT:**

Applying level-based classification on improved MALT clearly demonstrates that features that contribute to self-configuration have been enhanced and now lie at a higher level compared to the baseline. Figure 9.28 illustrates the application of level-based classification.

**Realisation of Adaptation Strategy on MALT:** The use case discusses the implementation of the adaptation strategy on MALT. This adaptation strategy realises the tools developed to capture information, configure the system and execute functionality. In terms of the Adaptation Strategy levels, the realised tools and their descriptions are as follows:

- **High Level - Production System Coordination**

- *State Charts:* Activates the self-configuration process. Initiates information capture, configuration change, and functionality execution through coordination of state machine actions.
- *State Machine:* Coordinates the functionality execution by interacting with other layers. Groups state charts into stages to

		Multi-Agent MALT System	
Data Management	Self-Capability	Self-adaptation	MALT is capable of determining its state, function and components (fixture) attached. It is able to adapt to changes and disturbances.
		S6   L4	
	Data Collection	Combine useful data from data sources without disruptions (partial)	MALT can combine data sources to give an outcome i.e. pass/fail criteria based on objective. Also, it can eliminate noise from collected data.
		S6   L4	
	Data Storage	Choice of local, network or cloud storage	MALT configuration and test data can be stored locally, network or cloud storage. It is also possible to store portions of data to specific storage locations.
		S6   L4	
	Data Conversion	Useful information along with filtration	The system can filter information to determine important aspects that influence outcome. Agent integration enables to drive the objective while suppressing aspects that do not contribute to goal.
		S4   L3	
	Data Transmission	Data transmission and linked to knowledge-base	Data is available in structured and transmissible format with other devices or even cloud-based services.
		S4   L2	
Operation Identification	Goal oriented operational awareness	Multi-Agent MALT integration assists in directing to achievement of a goal with operational awareness about the process.	
	S6   L3		
Optimisation	Optimisation by objective function manipulation	Optimisation functionality is present based on objective function, where the objective function guides the behaviour.	
	S6   L3		
Monitoring	Monitoring for disruption management	MALT is able to monitor accurately the configuration and test data. Agents based integration assists in overcoming disruptions.	
	S6   L4		
Control	Monitoring driven Automated Control	Based on monitored values multi-agent integration can drive automated control in testing application.	
	S6   L3		

Figure 9.28: Level Based Classification for Self-Configuration applied to Multi-Agent & ML integrated MALT System. The features have been significantly enhanced making MALT capable of self-configuration.

orient the state chart actions in a batch control behaviour.

- **Mid Level - Production System Runtime**

- *Asset Administration Shell*: Captures MALT information in real-time. It is the digital footprint of the physical asset. The AAS is used to represent the Configurable Object (MALT) in the form of a module. State, configuration, requirement, and func-

tionality information is represented in submodels of the AAS. The low-level layer uses this information to produce configuration changes and execute leak test functionality as per requirement through a single source of truth guided by the high-level coordination layer.

- *CAEX Engine*: Assists in querying information hosted in AAS by providing an interface in the AAS structure of MALT. The coordination level interacts with the AAS to find out the current state of the MALT, the configuration and the current requirements. The high-level layer then induces the change, updates requirements, and executes functionality through the low-level layer.
- *Tool Utility*: Provides a means for the high-level layer to interact with runtime and with the low-level control layer.

- **Low Level Production System Drivers & Control**

- *MALT API*: Hardware abstraction for the MALT. It provides a means to interact with physical MALT. The coordination layer uses this to change configuration or to execute functionality. The runtime layer also uses this to query the current values/settings and state.

The process of self-configuration for MALT leak test execution depends on matching requirements with variables. This chapter discusses this in detail and how these variables can be configured through the endpoint hosted in the cloud.

The possibility of using a Generic Algorithm for CNP resource selection is also discussed with each resource, representing MALT, configurable through

changing requirements in terms of functionality-driven variables. Multiple possibilities are discussed here, to achieve self-configuration, that can be explored as future research to achieve a smarter leak test functionality.

**Limitations and Future Research:** The experiments focused primarily on volume-based configuration. Further research incorporating additional part characteristics (e.g., complex shapes, and material variations) would enhance the strategy’s versatility. The adaptation strategy considers static parameters, and integrating real-time monitoring of ambient conditions would further improve its ability to maintain optimal settings in dynamic conditions. Exploring other machine-learning techniques could lead to better precision in the self-configuration process.

This research offers advancement in the field of leak testing. The self-configuration system streamlines leak testing processes, reduces errors, and minimises downtime in manufacturing industries by automating and optimising configuration.

## 9.4.2 PRIME

### Introduction

This experiment validated the self-configuration strategy’s ability to dynamically adapt the force testing settings for the PRIME test station. Specifically, they address the following:

1. **Product Variability:** This experiment tests the data and conceptual models developed in this research (Objective 1). By adapting force testing parameters to different hinge assemblies, the experi-

ment will assess the ability of the adaptation strategy to capture and utilise configuration information. It will also examine how well the system can dynamically identify and adjust to these hinge assemblies, validating the configuration abstraction of the system elements (Objective 2).

2. **Dynamic Production Requirements:** The experiment addresses the system architecture objective (Objective 4). By evaluating the system's ability to maintain optimal configuration in the face of changing assembly configurations, the experiment will assess the modularity and adaptability of the architecture.
3. **Reduction of Expert Dependency:** This experiment is designed to validate the implementation strategy (Objective 5). By minimising the need for specialised knowledge in force testing, the experiment will test the system's ability to seamlessly adapt the force test settings based on the defined configuration rules (Objective 2). This will demonstrate the practical application of the identified enabling technologies (Objective 3).

#### **Application of Level-Based Classification on PRIME test station:**

Level-based classification can be applied to force testing equipment, such as a PRIME test station, to identify areas that need improvement for achieving self-configuration. The identified features, as shown in Figure 9.29, can be enhanced through stage-wise transitions to enable the test station to achieve self-configuration.

		PRIME Test Station	
Data Management	Self-Capability	Self & context Awareness S4   L2	PRIME test station is capable of determining its state, function, and components (fixture) attached. Camera in place that identifies the fixture. Management system to move shuttle in front of the test station.
	Data Collection	Combine useful data from multiple data sources S5   L3	PRIME test station can combine data sources to give an outcome i.e. pass/fail criteria based on force test and KPI objectives.
	Data Storage	Stored locally and some also at network S3   L2	Configuration data is stored locally in PLC memory and also test data is populated on shared local storage directory on main PLC.
	Data Conversion	Useful information extraction from data S3   L2	PRIME test station extracts useful information from data to develop inference for outcome. This is carried out through PLC logic.
	Data Transmission	Data transmission and linked to knowledge-base S4   L2	Data is available in structured and transmissible format with other devices.
	Operation Identification	Operational awareness present S3   L1	PRIME test station has awareness of operation based on event triggers and configuration selected for execution.
	Optimisation	Utilisation mechanism not present S1   L1	Currently optimisation functionality does not take into account the objective. Optimisation depends on the operator.
	Monitoring	No monitoring capability -   L0	PRIME test station does not have monitoring capability, a boolean true or false is received as pass/fail result.
	Control	Partial-automated control S3   L1	The system can carry out the test but human input is required for selecting the configuration for execution and also starting the test. Also, the operator has to acknowledge the result.

Figure 9.29: Level Based Classification for Self-Configuration applied to PRIME test station

**Capturing Information on PRIME Test Station:**

The dynamic self-configuration is established for the PRIME test case as changes made to the system AAS, through the CAEX Engine, is updated using a configurable module interfaced with the physical system. The AAS for the PRIME test station contains submodels, such as the “settings”, “results”, “calibration” and “skill” submodels. The configuration settings (i.e. variables, relationships, and constraints for PRIME force-test configuration) of the PRIME test station for “force test” functionality are updated

through the CAEX Engine based on requirements captured through the state charts. Figure 9.30 shows captured PRIME test station information from CAEX Engine.

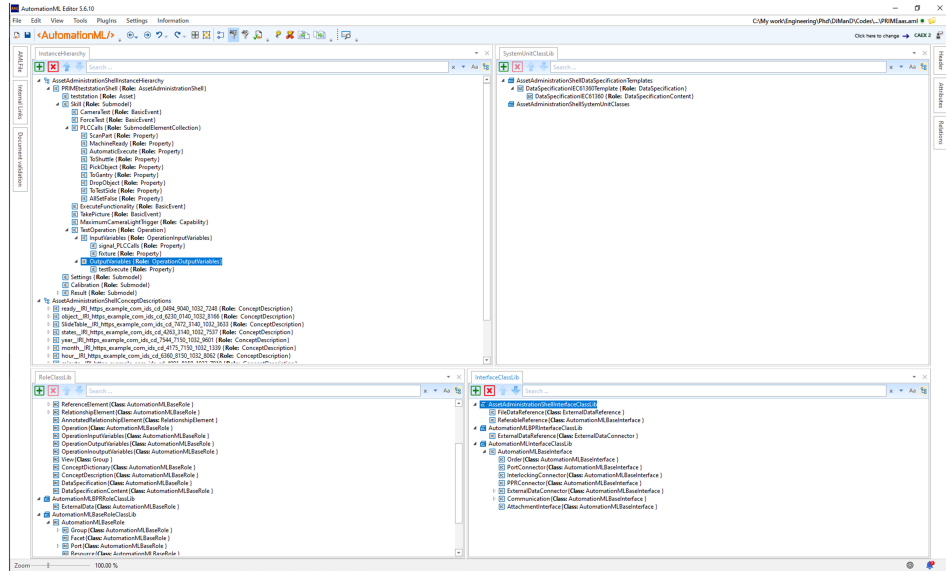


Figure 9.30: Capturing PRIME test station information from the CAEX Engine

### Representing Information in AAS:

The updating process for the AAS through the approach is driven by the respective submodels, that are updated in the PRIME test station through the interfaced configurable module. These submodels follow strict structural requirements for the interface to understand and update them in the relevant AAS. This means that in the case of the PRIME test station, the whole AAS may not be provided for updating the configuration. Figure 9.31 illustrates the AAS of the PRIME test station.

### Tool Utility for Coordinating Force Test Functionality:

The tool generates a variable list and captures customer, functionality, and part requirements, which are related to the variables of the configuration



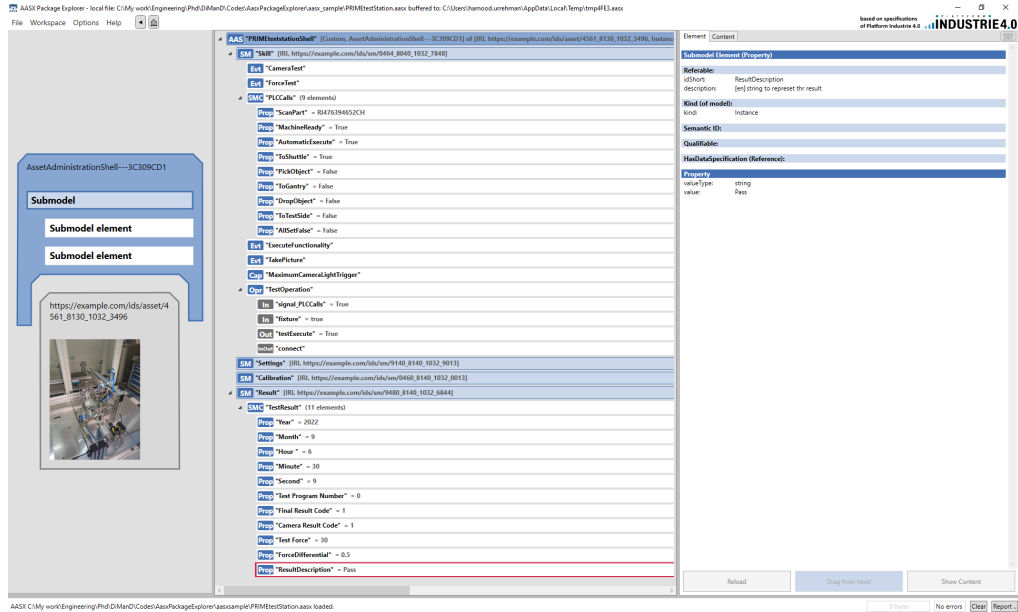


Figure 9.31: AAS for PRIME test station

in the production system that need to be changed. The values for variables are queried through ML pipelines hosted at an endpoint, which are then assigned to the production system for functionality execution. Finally, the deployment of the variables involves connection to the system and deployment of the changed configuration, followed by execution of the functionality.

For PRIME test station, as per state chart, “Activation ” leads to “Part Recognition”. The identification of the part is dependent on the fixture on which the part is clamped for the PRIME case. There exists a camera on the setup that takes an image of the data matrix on the fixture and relates it to the part, therefore acting as an identifier for the part being force-tested. The force test functionality requirement revolves around the type of part being tested, more specifically its assembly components. Each hinge assembly has a different force test requirement. Along with the part variation, there is a requirement of maximum force and the fore differential. Here, the variation is the customer requirement, maximum force is the functionality requirement and force differential can be considered as

both the customer and functionality requirement. These requirements are linked to the variable list, that needs to be configured. These variables are matched to the requirements for the part being tested. The variable values are queried through ML pipeline hosted in Google Cloud Environment and updated on the PRIME test station as a configuration setting.

### **Experimental Methodology**

**Test Plan:** A comprehensive plan was designed. The details of the plan are as follows:

**Objectives:** Demonstrate configuration setting adaptation based on fixture-product association (identified visually). Quantify improvements in predicting configuration wrt time and accuracy compared to the manual baseline.

Force testing can be considered a means of determining the failure, breakage, or actuation of a product. Manufacturers use this to determine the ease or difficulty with which a product or aspects of the product can be pushed, bent or compressed. This ensures the accurate functionality of the product as it is used. Traditionally, force testing is carried out to ensure product safety and performance along with verifying the likelihood of functionality as per intended design and usage. Force tests on a product, if not carried out, may affect the safety and acceptance of the product.

Force testing can be carried out through specialised equipment providing force measurements from a product. These measurements can be the maximum force the product can handle, the minimum force to actuate the product or the force range that is to be maintained on the product for operation.

A force test may be carried out in various modes like amplitude (i.e. to measure effect under a single force application), and periodic or increasing/decreasing modes to ascertain whether the product still maintains functionality and the presence of weakness in the product through continuous load application occurring during the lifecycle. In a traditional manufacturing case, the force magnitude is displayed on an HMI for the test carried out on a product. To get a more detailed understanding of the effect of force on the product during test, typically, force measurements can be plotted in a graph against other dimensions with an acceptance band. Failing to fall in the acceptance band means that the product does not pass the test for functionality.

The force test may be of static or dynamic nature. The static test involves loading the product or applying force until the product reaches the desired deformation level and ascertaining the behaviour of the product. This kind of test is carried out to predict how a product reacts under force over a long time period. Dynamic tests are carried out with force applied over time periods/intervals to determine the product behaviour over said intervals. The load can be kept the same or vary over the time periods. These kinds of tests are usually carried out to predict product reaction to short-term forces.

The test products consist of hinge assemblies of different configurations. These hinge assemblies are subjected to force test requirements. The experiment aims to validate the adaptation strategy by determining force configuration settings for hinge assemblies under KPIs.

**Test Case:** The fixture is used to determine the most suitable test configuration setting by associating it with the product. The tests main focus

is ensuring the adaptation strategy correctly identifies fixtures relating to the product and sets the corresponding force test parameters. It should demonstrate that the system can maintain consistent pass/fail outcomes under constraints and do so efficiently in comparison to manual configuration settings. The test case can be briefly detailed as:

- **Fixture Identification:** The system determines the fixture relating to the product for which the test is to be performed.
- **Force Range Adaptation:** The system identifies the suitable force configurations for executing the test on the product under test, i.e. candidate configuration.
- **KPI Constraint:** The system identifies the most suitable configuration setting based on KPI constraint. Here, in this experiment, it is taken as the accuracy of the configuration setting for giving the correct force.
- **Configuration Adaptation:** The system adapts the new configuration setting based on KPI constraint from the candidate configuration.

**Performance Metrics:** The test performed measured the time taken to identify the fixture and determined the optimal force test parameters. Accuracy of force parameters against expert-derived settings and the pass/fail outcomes based on force test results.

**Baseline Establishment** The PRIME test station is configured for a range of product variations and force tests performed. The configuration time for each test by an expert, exact force test parameters used, and

Pass/Fail outcome are recorded. The average configuration times and force test configuration by the expert are stored.

Cloud-based machine learning (ML) model is trained on fixture images and deployed for analysis on an end-point. The architecture of the pipeline is shown in figure 9.32 and the deployment detail with training is detailed in figure 9.33. As more and more classifications take place the model is improved. The classified images are sent to the pipeline for training with every iteration. The cloud-based deployment of for identification of fixtures relating to parts is used as the basis for agent-based decision-making.

Images ingested from the gateway device obtained by the camera module are stored in the storage housed by the cloud platform. This event of image storage acts as an event trigger that executes a script sending the stored image to the machine learning service endpoint. The endpoint houses the model that is determined by using NASNET, trained for the task and on the dataset in GCP. At this endpoint, the image is labelled as per the classification obtained by the trained, tested, and validated model. The

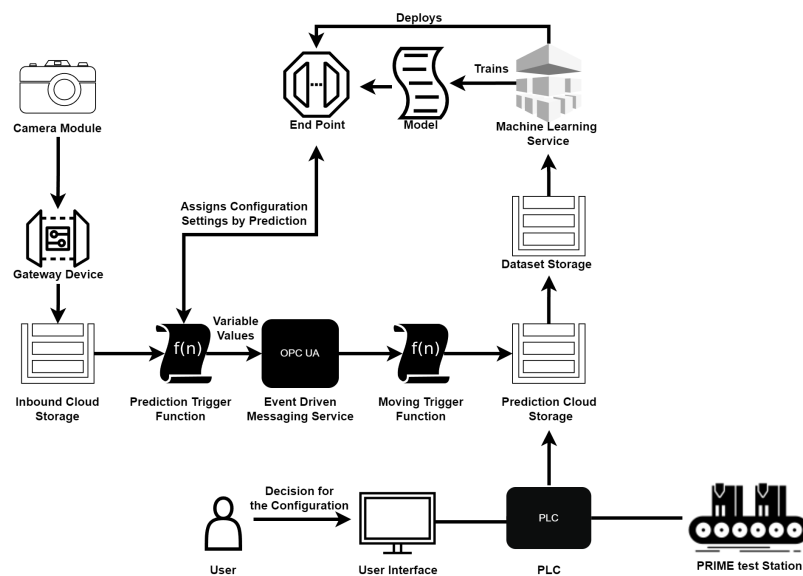


Figure 9.32: Architecture of the Cloud Based Machine Learning Pipeline for Configuration Setting Detection.

labelled image is written to the message topic (OPC UA). This message topic hosts the force configuration settings for the labelled image and triggers another event that moves the labelled image with the force setting to the predicted cloud storage, offering separate storage services for the force setting categories classification. The identified force settings are loaded into the production system through the PLC behaviours.

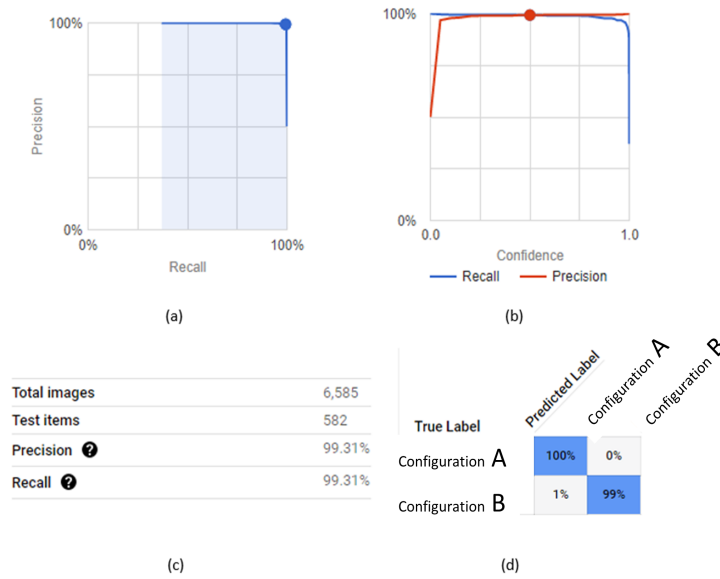


Figure 9.33: ML Pipeline Deployment. (a) Precision vs Recall against identified part labels, for fixture identification. Precision is a measurement of positive label assignment (ratio between the True Positives and all the Positives) and recall is the measure of the model correctly identifying True Positives. (b) Maximum confidence at Threshold = 0.5. (c) Dataset size (total and test images). (d) Confusion Matrix representing True and Predicted Labels.

### Validation

- **Performance Comparison:** Analyse predicted force configuration settings against the baseline.

- **Expert Review:** A domain expert reviews the accuracy and optimality of the strategy-generated force settings.

## Experimental Setup

### Hardware:

- PRIME test station
- Fixtures with variations in hinge assemblies
- Camera for fixture identification
- Beckhoff Control System

### Software:

- Agent system
- CAEX engine
- Image-based classification ML endpoint (cloud-based)
- Connection mechanisms for ML endpoint output to the test station (e.g., PLC control)

## Experiments and Results

**Goals:** The goals for the PRIME test station experimentation, kept the same as the MALT case study, are summarised as follows:

1. **Configuration Capture:** Capture the configuration settings for force test functionality in terms of variables, relationships, and constraints through the information model, and CAEX engine. The captured information should be compliant with the AAS standard.
2. **Self-Configuration Capability:** Using the features identified during level-based classification, incorporate self-configuration capability by improving features.
3. **Optimisation and Adaptation:** Using adaptation strategy and ML endpoint deployment, achieve self-configuration.
4. **Agent System:** Implement an agent system, as an enabler, for coordinating with data pipelines to achieve self-configuration goals.
5. **Cloud-Based Data Pipeline:** Host data pipeline components in a cloud environment (e.g., Google Cloud Platform) for better data processing and coordination.

**Procedure:** Once a product's order has been launched into the system, the component moves through the conveyor until the camera module is reached. Immediately, it takes a picture of the fixture and compares it with an ML cloud-based classifier to determine the force test configuration settings. The setting is updated on the system and the functionality is executed.

The experimental setup, as seen in figure 9.36, includes services employed in a cloud environment, agent resources, and deployed physical resources. The camera module is connected to the Cloud-Based Machine Learning Pipeline. The functionality employed by cloud-based services is of visual quality inspection for determining force test configuration for the force test station.



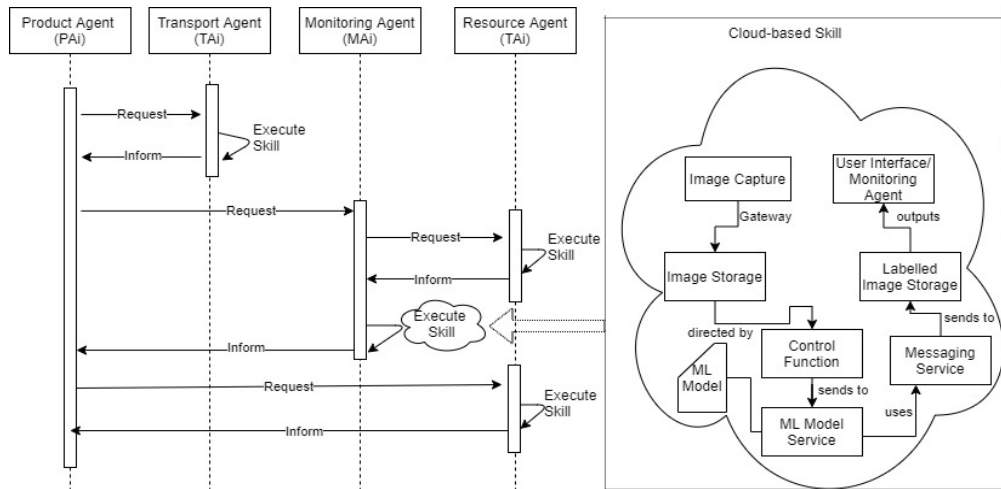


Figure 9.34: Sequence diagram for the force-test use-case incorporating image oriented configuration setting.

The sequence diagram as per figure 9.34 shows the interaction between products, transportation assets, monitoring elements, and resources. It assumes that the agents have been already launched.

**Product Agents (PA)** guides the sequence of the processes of creating a product by requesting the skills (functionalities) from other agents to be performed on the physical product instance. This request of skills could vary depending on the product requirements and the skills required to create the product (i.e. functionality requirements). In the PRIME test case, this is the “force test” functionality.

**Transport Agents (TA)** execute the skills requested by the PA to move the product to the required resources and to inform the current position of the product to PA. The TA also considers the buffers and potential bottlenecks that might arise before executing its skills.

**Resource Agents (RA)** represent shop-floor resources (such as machining centers, robots, sensors, cameras, test stations). RAs can provide their availability, task status info, and resource information as per request.

**Monitoring Agents (MA)** doesn’t have a physical entity associated with it but utilizes the skills provided by RAs if required (e.g. image capturing

skills from a camera RA). The MA offers the cloud computing functionality as skills if required for configuration setting prediction. For example, PA requests MA to perform a skill of configuration setting prediction of the product associated with it. MA if required, requests RA to perform its skill (e.g. take a photo of the product). MA then uses a cloud platform for decision-making and informs PA about the force configuration setting for the product.

Agent programming is implemented with the JADE (Java Agent Development Environment) platform like the previous use case (MALT). The agents deployed in JADE use cloud-based classification input generated by vision model trained on the dataset (for fixture identification), as input leading to further actions. All the actors in the production environment are controlled by means of agents.

ML endpoint triggered through PLC behaviours provides the force settings after the image of the fixture is captured by the camera on the PRIME test station.

**Expression for Force Test Functionality Execution:** The product Agent looks for skills required for force testing based on the “operational” expression. The “operational” expression lists down the skill required by the product being tested along with the sequence of execution of those skills. For the PRIME test case, the Operational expression is given as:

$$\begin{aligned}
 T = & C_{\phi} AllSetFalse.C_{pick} PickObject.C_{drop} DropObject.C_{scan} ScanPart. \\
 & C_{testside} ToTestSide.C_{exec} ExecuteFunctionality.ReturnPart.T
 \end{aligned}
 \tag{9.5}$$

The agents execute the skills in these manners through the PLC Call triggers. If conditions are accompanying the trigger calls, then these are satisfied first through the respective “conditional” expressions. In the PRIME case, these conditional expressions are stored locally at the PC connected to the PRIME test station. A few example of these conditions are as follows;

$$C_{pick} := input\ TransferCylinder\ A = True \quad (9.6)$$

$$C_{drop} := input\ TransferCylinder\ B = True \quad (9.7)$$

These condition present the states of the mechanical components that must be satisfied for the functionality to execute in the illustrated sequence. As this sequence is executed, the part is tested and returned to the shuttle on the same fixture.

**Executing Force Test Functionality:** Figure 9.37 shows the execution of the self-configuration strategy on the system and sequence diagram, as per figure 9.35, the agent system interaction. It demonstrates the product’s entry, transportation, self-configuration of the manufacturing system, test execution through agent coordination, and product transportation after testing. Additionally, a configuration change in the manufacturing system before and after execution can be observed in the corresponding figure 9.38.

**Results:**

**Validation of Adaptation Strategy on PRIME Test Station:** This approach provides a means to adapt PRIME to product, process, and KPI

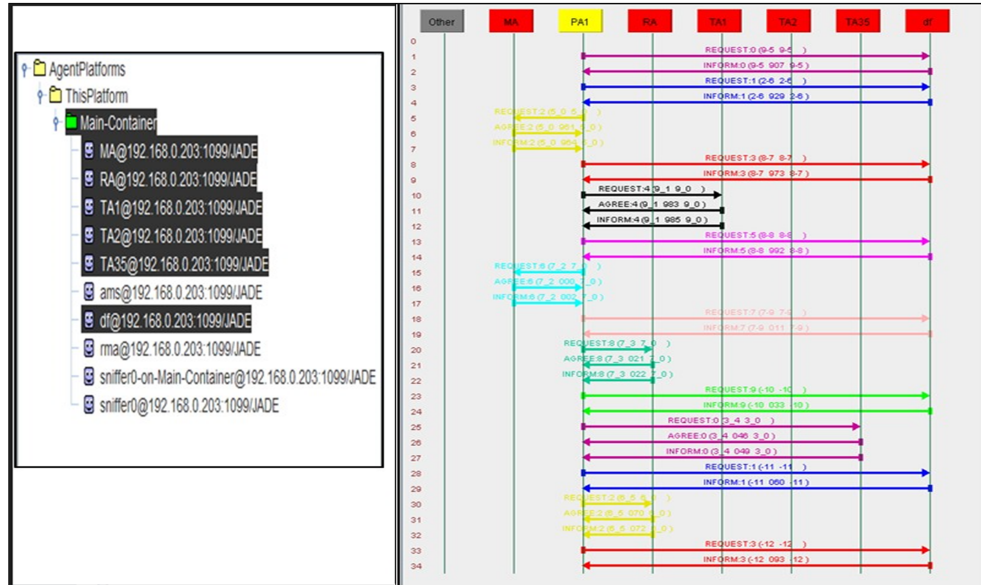


Figure 9.35: Sequential diagram for functionality execution on the PRIME test station.



Figure 9.36: PRIME experimental setup involving teststation, two robots and a shuttle on rail.

requirements. As an advantage, this approach assists in achieving self-configuration for force test functionality on the PRIME test station by updating configuration specific to requirements. Figure 9.39 provides a comparison of the configuration achieved through the deployed approach and the configuration derived without it.

A hinge assembly was presented to the system as a test part, and then force configuration settings were determined by the system and the expert. The time taken by both to read the optimal test configuration under constraint

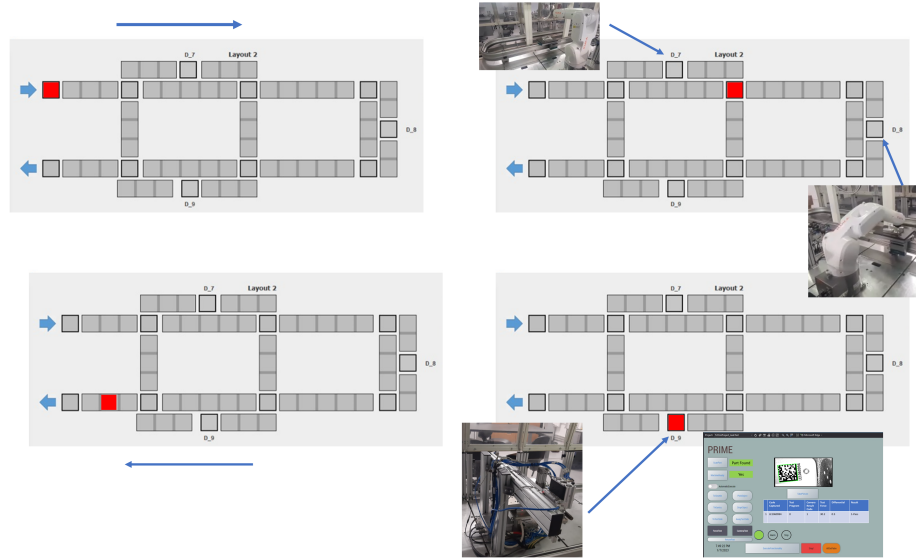


Figure 9.37: The force test functionality execution on PRIME. The robots and the test station are allocated respective locations. The agent system routes the part to the test station where depending on the fixture the part is identified and the force test configured.

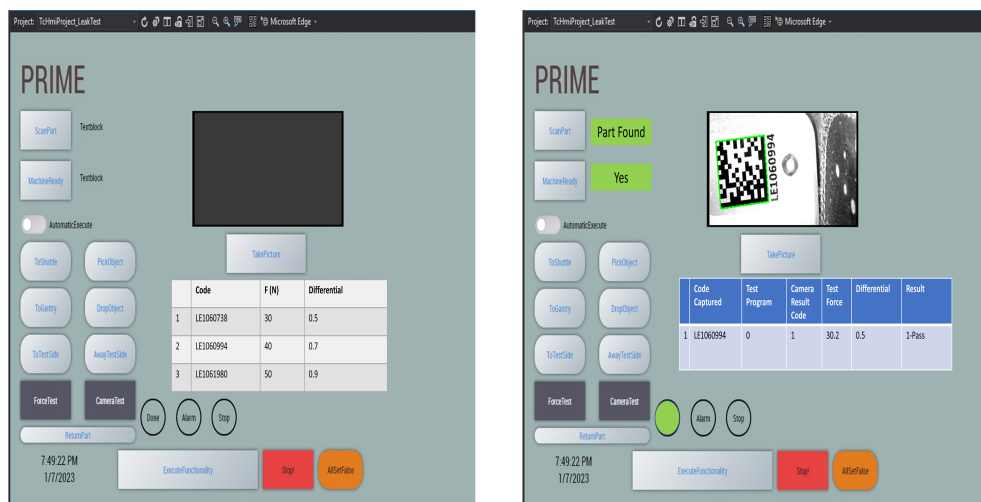


Figure 9.38: Configuration change in the PRIME test station (a) before (b) after detecting the part.

was recorded. The adaptation strategy was able to configure and execute the functionality in 0.2 sec in comparison to an expert that took 1500 sec respectively.

Figure 9.39 demonstrates the application of self-configuration on a test part.

Parameters	Value			Time Taken	
	LE1060994	LE1060738	LE1061980	Expert	Adaptation Strategy
Test Force	30.2 N	45 N	35	900 sec	0.1 sec
Differential	0.5	0.7	0.6	600 sec	0.1 sec
Total Time To Test Setting Determination				1500 sec	0.2 sec

Figure 9.39: Validation of Adaptation Strategy for force test functionality on PRIME. The time taken to determine a valid configuration is significantly less through the research approach in comparison to an Expert. Also, the approach allows updating this configuration automatically, while the expert has to enter the values manually.

The same configuration was figured out in a significantly longer duration and fed back to the station externally so that part could be evaluated. This was carried out in an academic setting on the PRIME assembly system, providing a clear improvement and significant cost-saving.

**Mapping to Research Questions:** The implementation is mapped to the research questions established of the research project as per figure 9.40.



Figure 9.40: Mapping project research questions to PRIME test station Use-Case Implementation

### Overall Findings

The result of the PRIME for test experiment validate the research by achieving its objectives in the following manner:

1. **Theoretical Foundations:** The adaptation of the force test system to the different hinge assembly and utilisation of data to determine configuration settings demonstrates the usefulness of the developed data and conceptual models.
2. **Configuration Abstraction:** The adaptation and force test functionality execution under changing hinge assemblies validate the abstraction developed for the configuration and the self-configuration logic.
3. **Enabling Technologies:** The reduction of expert dependency for determining force test settings and the implementation of adaptation strategy through the developed tools confirms the usefulness of enabling technologies for self-configuration in the PRIME test station.
4. **System Architecture:** PRIME test station module and adaptability to suit the force test requirements show the ability of the system to adapt for functionality execution based on the developed architecture.
5. **Implementation Strategy:** The self-reconfiguration of PRIME force test station for force testing and reduced expert reliance validates the implementation strategy.

### Level-Based Classification on MultiAgent and ML Integrated PRIME

**Test Station:** Applying level-based classification on improved PRIME

test station clearly demonstrates that features that contribute to self-configuration have been enhanced and now lie at a higher level compared to the baseline (see figure 9.41).

		Multi-Agent & ML PRIME Test Station			
Data Management	Self-Capability	Self-adaptation	S6	L4	PRIME test station is capable of determining its state, function, and components (fixture) attached. It is able to adapt to changes and disturbances. Image from fixture can be used to find optimal configuration.
	Data Collection	Combine useful data from data sources without disruptions (partial)	S6	L4	PRIME test station can combine data sources to give an outcome, i.e. pass/fail criteria based on KPIs.
	Data Storage	Choice of local, network or cloud storage	S6	L4	PRIME configuration and test data can be stored locally, or cloud storage through OPC UA.
	Data Conversion	Useful information along with filtration	S4	L3	Agent integration enables to drive the objective while suppressing aspects that do not contribute to the goal.
	Data Transmission	Data transmission and linked to knowledge-base	S4	L2	Data is available in structured and transmissible format with other devices or even cloud-based services.
	Operation Identification	Goal oriented operational awareness	S6	L3	Multi-Agent integration assists in directing functionality execution with operational awareness about the functionality.
	Optimisation	Optimisation by objective function manipulation	S6	L3	Optimisation of the functionality is guided by the objective through cloud-based data pipeline.
	Monitoring	Monitoring for disruption management	S6	L4	PRIME test station is able to monitor accurately the configuration and test data. Agents based integration assists in overcoming disruptions.
	Control	Monitoring driven Automated Control	S6	L3	Based on monitored values, multi-agent integration can drive automated control in testing application.

Figure 9.41: Level Based Classification for Self-Configuration applied to Multi-Agent & ML integrated PRIME test station. The features have been significantly enhanced making MALT capable of self-configuration.

**Realisation of Adaptation Strategy on PRIME Test Station:** The implementation of an adaptation strategy for the PRIME test station is discussed in this use case. The strategy incorporates a range of tools to



capture information, configure the system, and execute functionality. These tools are classified into different levels within the Adaptation Strategy, each serving a specific purpose. Their descriptions are as follows:

- **High Level - Production System Coordination**

- *State Charts:* State charts are utilized to initiate the self-configuration process and facilitate information capture, configuration changes, and functionality execution.
- *State Machine:* The state machine, operating within this level, groups the state charts into stages to enable effective coordination with other layers and facilitate batch control behaviour.

- **Mid Level - Production System Runtime**

- *Asset Administration Shell:* The Asset Administration Shell (AAS) plays a crucial role in capturing real-time information about the test system. Serving as a digital representation of the physical asset, the AAS represents the Configurable Object (i.e. PRIME Station) through submodels that encompass state, configuration, requirement, and functionality information. By utilising the AAS, the high-level coordination layer can ensure a single source of truth, enabling smooth configuration changes and execution of force test functionality through the low-level layer.
- *CAEX Engine:* The CAEX Engine provides an interface to query information hosted in the AAS, enabling the coordination layer to access the current state, configuration, and requirements of the test system.
- *Tool Utility:* Provides a means for the high-level layer to interact with runtime and with the low-level control layer.

- **Low Level Production System Drivers & Control**

- *PLC Control*: The PLC control serves as a hardware abstraction layer that facilitates interaction with the physical test station. Through the PLC control, the coordination and runtime layers modifies configurations, execute functionality, and retrieve real-time values and state information. These interactions at the low level contribute to the overall self-configuration process for force-test execution.

By configuring resources, which represent the test system, based on changing requirements related to functionality-driven variables, the self-configuration process can be further improved.

By implementing this adaptation strategy, the PRIME test system benefits from a hierarchical arrangement of tools across different levels of production system coordination, runtime, and drivers/control. This approach ensures effective information capture, configuration changes, and functionality execution. Through the use of state charts, a state machine, the AAS, the CAEX Engine, and the PLC control, the test station achieves enhanced self-configuration capabilities.

**Limitations and Future Research:** The adaptation strategy identified fixtures using images captured, relating it to the product and adjusted force test parameters accordingly. This demonstrates the ability to automate configuration based on the specific product variations being tested. The experiments primarily focused on fixture-based configuration. Further investigation of additional part features (e.g., shape, holes etc.) would expand the prediction capabilities. The strategy can be expanded to include the possibility of real-time monitoring of equipment states for even

more robust configuration in dynamic environments. Exploring more image processing and machine learning techniques could enhance fixture identification accuracy.

This research signifies advancements in force testing within manufacturing. By automating the configuration process based on fixture identification, the self-configuration system aims to streamline force testing operations, improve accuracy, and reduce reliance on specialised expertise.

## 9.5 Conclusion

In this chapter, the implementation of the adaptation strategy developed in the previous chapter is discussed. This implementation is a critical step in realising self-configuration capabilities in two industrial use cases. The primary goal of this chapter is to validate the research conducted within the scope of this PhD thesis, and the work aligns closely with the research objectives established in Chapter 3. Respective sections in this chapter relate to the contribution each use case makes to these objectives.

The validation process focuses on two specific industrial use cases: the Multi-Application Leak Test (MALT) and the PRIME test station. MALT is a leak test module developed by the SME (TQC Ltd.), designed to test parts for leaks under pressure. Changes in part features, functional conditions, or any constraints can necessitate adjustments to the test configuration. The self-configuration application for MALT aims to assist in adapting the test configurations to accommodate these changes. MALT can be applied in various settings, whether as part of a larger production system or as a stand-alone tool.

The PRIME test station, on the other hand, is used to apply force to a part to assess its deformation characteristics. Similar to the MALT use case, variations in part features, functionality conditions, or constraints can lead to changing test configuration requirements. The PRIME test station is just one component within the broader PRIME assembly system, and the self-configuration application here aims to automatically determine the force test configuration settings for each part.

The research conducted in this chapter represents a contribution towards enhancing the adaptability and flexibility of industrial testing processes. Self-configuration, as applied to these specific use cases, has the potential to streamline testing operations, improve the ability to adapt to changing conditions and product variations, and reduce the dependency on expert knowledge for configuring these complex testing systems.

Currently, this research work that integrates self-configuration capability into production systems is at the forefront of research. This realisation is carried out in an industrial setting to test applications in manufacturing, to the knowledge of the author, has not been done in either research or industry. This enhances the value of this research towards the field of self-configuring production systems. The application of this research in the industrial was quite challenging, as each feature of self-configuration needed to be developed and integrated into production systems. As both use cases were industrial equipment, care had to be maintained on not affecting functionality over the objective of achieving self-configuration. Enhanced interoperability between manufacturing assets, granularity defined for functionality, and smart controllers in production systems by manufacturers can make it easier for more systems to achieve self-configuration by following the proposed adaptation strategy. Simplicity in adaptation can be achieved as more systems are made capable of self-configuration and

tools become more standardised/adopted.

### 9.5.1 Potential Cost Saving by Adaptation Strategy:

To further validate the argument, a small study at the industrial partner (TQC Ltd.) was carried out. The associated configuration cost for the production systems that they deploy for their customers was found. These costs form a significant percentage of total project costs. The time associated with the configuration operation also contributed to the project time. By adopting a self-configuration strategy in their existing business model, this time and costs can be saved for significant profit. A simple plot demonstrating the value utilisation potential can be seen in figure 9.42 and figure 9.43.

Projects	Project Description	Project Cost	Time (hrs)	Setup Time(hrs)	Configuration Cost
RAL6	GEN2 Truck Sensor Assembly Automation	£144,322.57	2,164	232	£ 15,469.80
BOS1	Helium Leak Test - Fuel Tanks	£306,954.11	1,576	72	£ 14,023.28
PIOL1	P33B Glovebox Latch Assembly and Test Machine	£52,887.24	1,792	56	£ 1,652.73
RAC21	Robot Cell for Clutch Housing	£118,782.80	2,488	320	£ 15,277.53
TAP1	Leak Test Mach for Caterpillar Transfer Case	£76,485.83	2,096	104	£ 3,795.10
TFS3	Robot Packaging Machine For DNA Tray Packer	£74,954.12	2,920	144	£ 3,696.37
BINX1	PoP Leak Tester	£18,757.70	848	40	£ 884.80
NNL2	Nat Nuclear Can Welder & Helium Leak Test	£133,818.37	2,656	352	£ 17,734.96
ALG4	Leak Test and Vision Machine For 'Y' Connector's	£46,333.71	1,160	40	£ 1,597.71
EVT1	Leak Test machine for MLA I6 Sump	£68,098.32	984	64	£ 4,429.16
Total		£1,041,394.77	18684	1424	£ 79,368.14

Figure 9.42: A practical example from TQC (Industrial Partner) on configuration costs incurred in respective projects. This presents a potential for value utilisation by self-configuration strategy, reducing these costs during deployments.

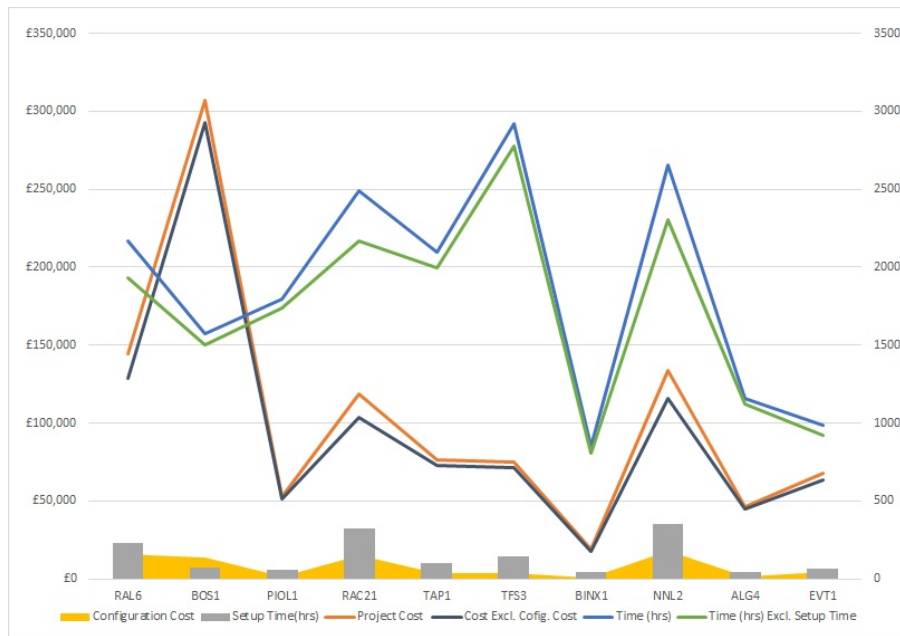


Figure 9.43: A plot demonstrating the value utilisation potential on industrial partner data by adaptation strategy.

---

# Chapter 10

# Conclusion

## Contents

---

10.1 Research Questions and Contributions . . . . .	322
10.2 Summary of the Findings . . . . .	328
10.3 Outputs, Implications, and Novel Findings . . . . .	334
10.4 Limitations . . . . .	336

---

## 10.1 Research Questions and Contributions

The current trend in manufacturing research revolves around incorporating entire systems with intelligence while assuming the capabilities of constituent manufacturing assets based on input and output needs. This study contributes by focusing on self-configuring production systems, incorporating intelligent components. These intelligent components address limitations of previous assumptions, enabling them to adjust to meet product, customer, and system requirements.

In this research, the focus is on addressing the configuration change in production systems associated with a change in system settings. This change involves updating the physical parameters along with variations in calibration and connection settings. Time constraint conditions, performance criteria or other KPIs may motivate this configuration change in production systems. The product can be operated on by the production system with different configurations depending on these circumstances.

In Chapter 1 of this thesis, the Research Question (RQ) is framed as:

*How to achieve self-configuration in production systems at the machine level?*



In Chapter 1, this RQ was further elaborated and divided in terms of the specific research questions:

- **Theoretical Foundations:** What theoretical models are needed to underpin the concept of self-configuration in production systems?
- **Adaptation Strategy:** How can a general adaptation strategy be developed to integrate self-configuring COs into production systems?
- **Technology, Tools, and Techniques:** Which technologies, tools, and techniques best facilitate self-configuration and leverage operational data for iterative improvement?
- **Implementation:** How can this approach be implemented in a real-world scenario involving a product within the production system?
- **Business Objectives:** How can business objectives be translated downstream to guide the self-configuration process at the production system level?

Figure 10.1 provides details of research contribution towards research questions. These are;

**Theoretical Foundations:** To address the “Theoretical Foundations” of self configuration in production systems, features of self-configuration were identified. The concept of self-configuration in manufacturing is further strengthened by presenting an idea of components in production systems that can be adapted (i.e. configured) to meet requirements, referred to as “Configurable Objects”.



Figure 10.1: Contributions of the research towards the research questions

**Adaptation Strategy:** Development of the “Adaptation Strategy” for self-configuration was achieved iteratively. Through the literature, features of self-configuration were identified. These features were mapped to the system through a classification model. This model can then be used to transition to a complete, self-configuring production system by evaluating the features for self-configuration readiness.

To frame an “Adaptation Strategy” a fundamental direction was to establish a case for granularity and modularity in production systems. In this research to develop an approach for self-configuration, it was necessary to define some separation of concern. It was essential to determine how to split the elements of production systems so that these splits could be treated as

one entity to be configured.

A case for defining this separation of concern was presented. The granularity is defined in the production system in terms of the functional decomposition style. This research bases that modularity is linked with granularity, each module is dependent on a careful assessment of the physical and cyber components. This module is a structured representation of the configurable object. These configurable objects execute functionality as per their configuration settings. The self-configuration adaptation loop, in the Adaptation Strategy, processes the configuration change through Module Driven Configuration.

In the Module Driven Configuration chapter, the concept of the module is elaborated, with a focus on configuration driven by functionality subjected to constraints in production systems. In this realisation of self-configuration, a module consists of variables, and the associated functionality rules and variable rules. These configuration settings are updated on the system through allocation and link. A model to capture the configuration information is provided. The captured information is related to Asset Administration Shells representing functionality operation as a submodel in AAS. The modules can be combined to form a production system where each module is responsible for the functionality, requiring variable configuration change and governed by rules. This kind of production system is presented as a Module-Based System.

The proposed Adaptation Strategy, a three-level architecture, operates through integration with CAEX Engine. The above-mentioned contribution on Module Driven Configuration forms the Adaptation Strategy's Mid-Level (i.e. Production System Runtime) layer. The high-level (i.e. Production System Coordination) layer deals with the coordination of the

state chart functionality in terms of state machine behaviours. This layer uses the information stored in the Production System Runtime model for coordination. The requirement captured in this layer can be used to adapt functionality configuration settings, achieving self-configuration. The low-level (i.e. Production System & Control) layer provides an interface to the production system. It interacts with the runtime layer, accessing information stored to update system configuration and executing functionality. This layer can also provide an interface to the service observers and actors, accommodating the need for monitoring and interaction with external services (e.g. cloud pipelines).

**Tools and Techniques:** In this research, various tools have been adopted and applied to address each layer of the Adaptation Strategy. Realised tools and their descriptions are as follows:

- *State Chart:* Responsible for self-configuration process activation. The coordination for the execution of functionality is carried out through state-machine actions.
- *State Machine:* The state machine groups state charts into stages, orienting the state actions into behaviour.
- *Asset Administration Shell:* Captures production system information in real-time. Acts as a digital footprint of the asset and the runtime to assist the configuration change by interacting with other layers. State, configuration, requirement, and functionality information is represented in submodels of the AAS.
- *CAEX Engine:* Assists in querying and updating information hosted in AAS by providing an interface in the AAS structure.

- *Tool Utility*: A tool for interaction between layers of adaptation strategy.
- *Control*: Hardware abstraction for the production systems. Provides a means of interaction with the physical system.

**Implementation:** The developed adaptation strategy is applied to MALT and PRIME test station use cases. This application demonstrates validation of the developed strategy to achieve self-configuration for testing functionalities. Based on KPIs, constraints, and requirements, a configuration setting is determined by the deployed self-configuration strategy in these production systems for test functionalities.

**Business Objectives:** The industrial surveys were carried out to gather knowledge/insight on self-configuration. Detail on the industrial surveys is presented in Chapter 4. This was done to establish an understanding towards the level of self-configuration (if any) capability present in the current production system on the shop floor.

Configuration change was formulated and discussed by updating the variables through endpoints hosted in the cloud. The possibility of using a Genetic Algorithm for CNP resource selection is also discussed, with each resource configurable through changing requirements in terms of functionality-driven variables. This research provides integration with business objectives through this approach.

An illustration of contributions related to chapters in the thesis can be seen in table 10.1.

Table 10.1: Contributions to the Research Related to Chapters in the Thesis

<b>Contributions</b>	<b>Chapters</b>
Module and Configurable Object Concept	Chapter 4 & 6
Level-Based Classification	Chapter 5
Case of Granularity and Modularity	Chapter 8
Module Driven Configuration	Chapter 6
Standard Configuration Model	Chapter 7
Adaptation Strategy Architecture	Chapter 8
State Charts for Self-Configuration	Chapter 8
Expressions, Agent Systems & ML Endpoints for Self-Configuration	Chapter 8
Tool Utility	Chapter 8 & 9
MALT Leak Test Use Case Implementation	Chapter 9
PRIME test case Implementation	Chapter 9
Industrial Insight Surveys	Chapter 4
KPI Integration and Resource Selection for Self-Configuration	Chapter 8

## 10.2 Summary of the Findings

This thesis presents the extensive research efforts and their practical application in the industrial domain in regard to self-configuring production systems. In this section of the thesis, the theoretical contributions along with practical implementation are listed, providing a comprehensive perspective on the research carried out. This section demonstrates the actionable insight and solutions applicable to industrial practices. The research carried out in this thesis can be summarised by the following details:

- **The Need for Self-Configuration in SMEs: A Survey & Business Model:** A survey to capture industrial insight on self-configuration was carried out. A potential business model taking advantage of the introduction of self-configuration capability in existing systems was presented. **Thesis Chapter:** 4
- **Level-Based Classification for Self-configuration in Produc-**

**tion Systems:** A level-based classification system was developed, as seen in figure 10.2 to classify the self-configuration capability in the existing production systems in terms of its features. These features are categorised into System Readiness and System Execution. A stage-wise transition is presented using which the mapped features in the system can go from system readiness to system execution category, achieving self-configuration. Technical enablers are also evaluated. **Thesis Chapter:** 5

- **Module Driven Configuration: A general framework for Self-Configuration:** Representation of a production system in terms of its modules was carried out, as seen in figure 10.3. Modules are structured representations of Configurable objects (COs). These modules consist of variables driven by functionality, subject to constraints (functionality and variable rules). These COs execute functionality as per the configuration settings. Module Driven Configuration processes the change in configuration settings. These settings are updated on the production system through Allocation and Links, which form part of the Module. A way of capturing information in a module is presented. A module is an embodiment of the separation of concern in production systems. Each module encapsulates a functionality, linking all physical and cyber components into one. The approach in this contribution can be applied to a variety of applications in manufacturing. **Thesis Chapter:** 6
- **Adaptation Strategy for Self-Configuration in Production Systems:** An Adaptation Strategy, as seen in figure 10.4, was developed that presented a layer-based approach to achieve self-configuration. The interaction between layers utilised runtime to capture production system information in real time (i.e. in the form of AAS). The top-

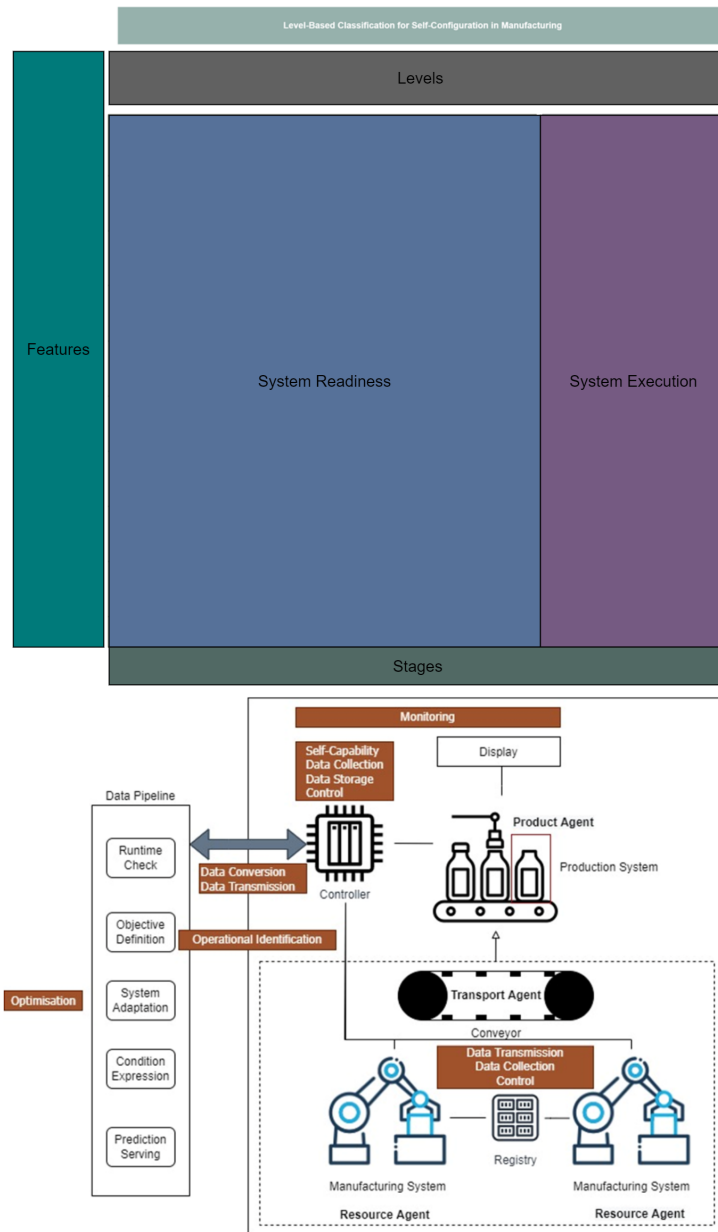


Figure 10.2: Level Based Classification for Self-Configuration and Mapping to Production System.

level coordination layer used the information captured in the runtime to coordinate the execution of functionality through state machine behaviour. The middle-level production system runtime layer captures information about production systems in terms of configurable objects. The low-level layer updates the configuration and executes functionality in the production system. The low-level layer provides



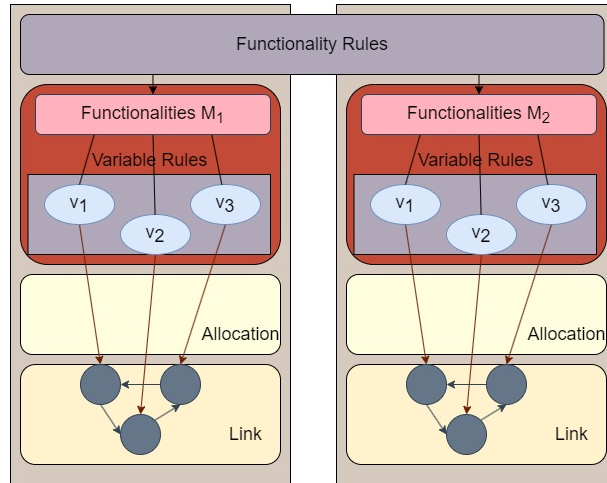


Figure 10.3: An illustration of a Module-based system having two modules hosting functionalities  $M_1$  and  $M_2$ .

an interface to the system by interacting with the runtime and coordination layers. **Thesis Chapter: 8**

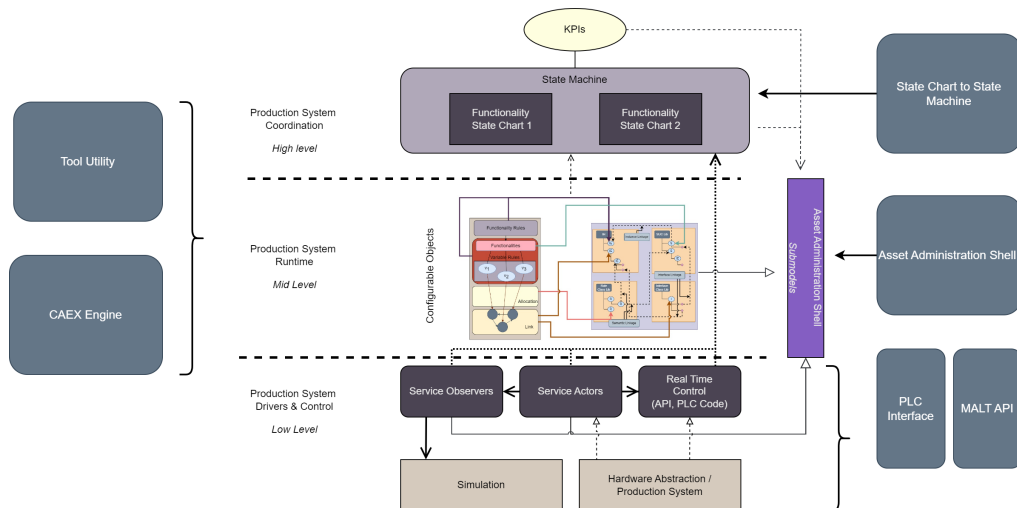


Figure 10.4: The developed Adaptation Strategy with supporting tools at a Glance.

- Deploying State Charts to State Machines for Production Systems to achieve Real-Time Control:** This contribution involves self-configuration process activation, information capture, configuration change and functionality execution control for coordination. The state chart actions are grouped into state machine stages. These stages are operated in a batch control behaviour. In this con-

tribution, a means of event-driven coordination of real-time control in production systems is provided. **Thesis Chapter:** 8

- **A Generalised Workflow For Self-applications Based On Cloud-computing For Production Systems:** Each functionality in the production system presents a requirement of variables that need to be configured. A generalised framework for providing machine learning capability for predicting the values of these variables in a configurable object is presented. In this research, a generalised workflow is provided for applying machine learning to production system self-applications. The mentioned workflows provide a basis for possible integration with machine learning services. These ML services can be hosted on the cloud, making it possible for low-powered industrial devices to use ML-assisted decision-making capability. This is carried out for a self-configuration use case, but it can be expanded to other “self-\*” applications. **Thesis Chapter:** 8
- **Application of Multi-Agent Systems for Testing Applications:** An approach to control functionalities through agent systems was developed. The concepts of “Operational Expression” and “Conditional Expression” are introduced. These expressions provided a basis for configuration updates to meet the functionality requirements. The operational expressions deal with “what” to do to execute functionality, and the conditional expression elaborates on the “how” to execute functionality. The integration of the expressions allows agents to execute event-driven behaviours (e.g. PLCs) and work with the State Machine for the self-configuration process. **Thesis Chapter:** 8
- **Cloud Based Decision-Making for Multi-Agent Production Systems:** An cloud-based integration of ML pipelines for decision-

making in a multi-agent production system environment. A method was developed where the images taken of the product can be used to query the configuration settings. These configuration settings can then be updated before executing the functionality of the production system. Initially done for quality control applications, this contribution can be expanded to load configuration to meet other KPI (e.g. time, and cost) requirements. **Thesis Chapter:** 8

- **Service Based Approach to Asset Administration Shell for Controlling Testing Processes in Manufacturing:** An integrated approach was developed that used services to control the functionalities of the production system by deploying Asset Administration Shells. The AAS approach was used to incorporate intelligence, and context awareness in the production systems. The contribution was applied to an industrial test case but can be expanded to be applied to other cases. This contribution provided validation of using agents (as services) with AAS towards configuring the system to meet requirements on functionalities. **Thesis Chapter:** 7 & 8

In the industrial use-case validation of the research, the following contributions are observed:

- **Simulated Environment:** A simulated environment was developed that can work with agent systems and AAS. This simulated environment works with ML services hosted in the cloud for decision-making. Such an environment assisted in simulating multiple scenarios like resource allocation, time, and cost along with business priority objectives. Using the developed simulation environment, the impact of decisions can be illustrated for CNP and GA algorithms. **Thesis Chapter:** 9

- **MALT Leak Test System:** Leak test application was studied with a focus on self-configuration. The product volume and test pressure were taken as input, and all other configuration settings were determined through the developed Adaptation Strategy. It is measured that such an approach saves time and effort in setting up configuration settings to meet product and leak test requirements. A potential configuration was achieved in a relatively short time period and then can be tweaked or agreed upon by the experts. With the increasing demand for customisation, such an approach that configures systems itself is beneficial in saving time and cost. **Thesis Chapter: 9**
- **PRIME Test Station:** In the PRIME case, force testing of the product was linked with the image recognition of the fixture. The image captured was related to product identification and determining a configuration for the force test. This approach can be used to integrate ML services on the cloud to find out the best optimum configuration settings through image recognition of the fixture in place. This makes the approach applicable to a variety of industrial applications. **Thesis Chapter: 9**

## 10.3 Outputs, Implications, and Novel Findings

This research into self-configuring production systems has yielded several key outputs with significant implications for the field of manufacturing:

**Adaptation Strategy for Self-Configuration:** A core output is a multi-layered strategy facilitating self-configuration in production systems.

The strategy incorporates a high-level coordination layer, runtime information captured in AAS, and a low-level control layer with hardware abstraction. This framework provides a structured approach to making manufacturing systems more adaptable.

**Module-Driven Configuration:** The research introduced the concept of module-driven configuration as a means of organising production systems. This modular approach groups system components by functionality while maintaining flexibility and granularity. It enables easier reconfiguration based on evolving requirements.

**Tools for System Assessment and Configuration:** To assist in adapting this framework within industry settings, practical tools were developed. These include a classification tool to assess production systems' readiness for self-configuration and a CAEX Engine for interfacing with and updating Asset Administration Shells of individual components or systems.

**Interoperability in Production System:** This work underscores the critical importance of interoperability in production systems. The research demonstrated how Asset Administration Shells (AAS) can enable standardised information exchange between components from different manufacturers.

**Security Considerations:** The adoption of AAS raises valid concerns about intellectual property protection. The research highlights the need to balance information accessibility with security protocols such as selective submodel sharing or rights-based access control. This will be a target for future work.

**Integration of Machine Learning:** This research introduces a workflow for deploying machine learning endpoints in self-configuration applications. This workflow allows for automated configuration updates, reducing manual intervention, especially for edge devices.

**Novel Findings** This research presents several novel insights for the field of self-configuration:

1. **The Link Between Real-Time Control and Self-Configuration:**

The ability to dynamically update process variables and execute functionality through hardware abstraction is essential. Robust hardware abstraction, including context awareness capabilities, emerged as a key factor in enabling true self-configuration behaviour.

2. **The Importance of Functional Decomposition:**

For effective module-driven configuration, production assets need to be functionally decomposed. Representing these functions using AAS submodels is critical for successful integration.

3. **Insights from Industry Surveys:**

Surveys conducted in this research confirmed an existing infrastructure in many manufacturing environments, along with the need for self-configuring systems. Yet, the lack of knowledge, training, and expertise hinders their ability to achieve self-configuration.

## 10.4 Limitations

In this research, it is observed that there exist certain limitations and challenges that must be addressed, as shown in figure 10.5, to understand and

leverage the potential of research fully. These limitations can arise from the complexity of the functionality and the system, standardisation in the systems and lack of practical implementation cases in domains. These limitations establish the scope of future work and the potential impact such research will generate. The limitations of this work are as follows:

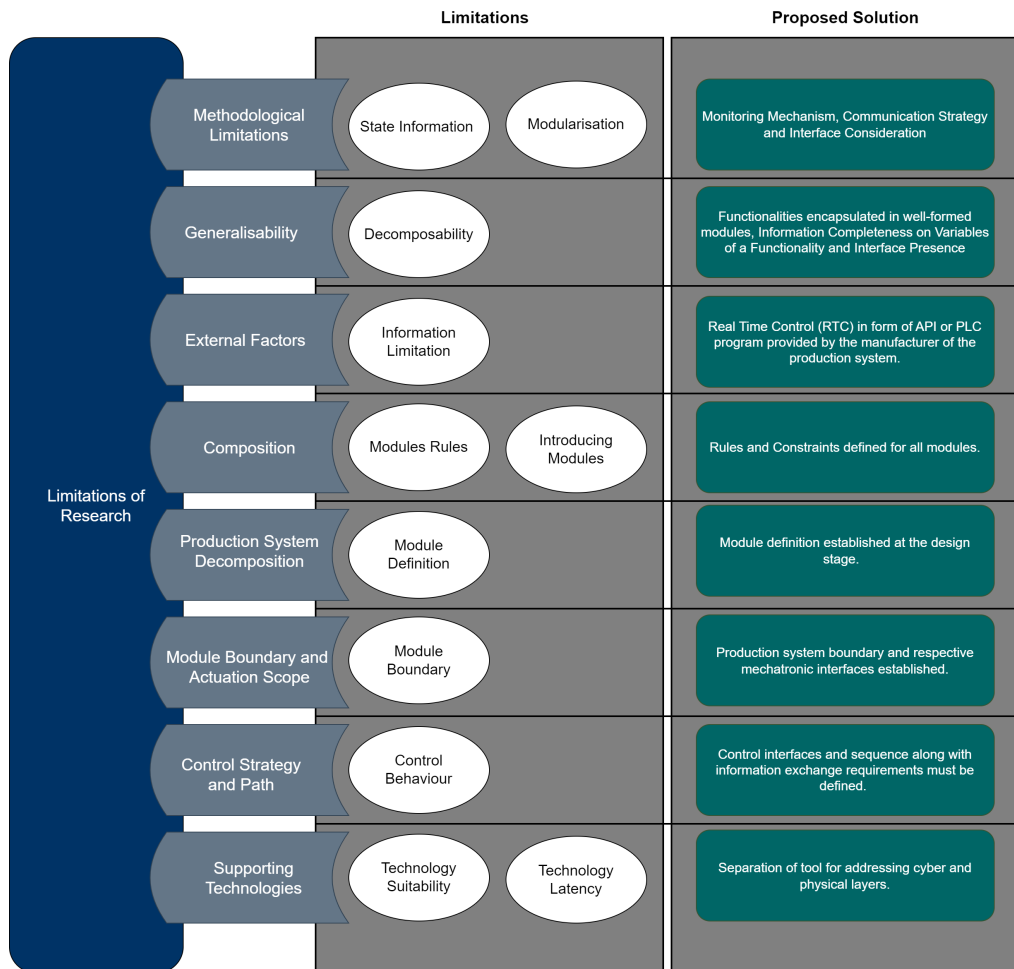


Figure 10.5: Limitations of the research. More detail on this is expanded on in this section.

1. **Methodological Limitations:** The self-configuration adaptation strategy relies on capturing state information. The module functionality is the transition from one state to another in the self-configuring process.

### State Information

**Affected Contribution:** Deploying State Charts to State Machines for Production Systems to achieve Real-Time Control & Adaptation Strategy for Self-Configuration in Production Systems.

**Limitation:** The information on states must be captured along with complete detail on the module, representing functionality. The developed approach has a requirement of contextual awareness.

**Proposed Solution:** A monitoring mechanism must be present that infers meaning from state transitions.

### Modularisation

**Affected Contribution:** Module Driven Configuration: A general framework for Self-Configuration & Adaptation Strategy for Self-Configuration in Production Systems.

**Limitation:** The developed approach promotes operations in the form of functional modules. These modules require interfaces to each other and between layers of adaptation strategy. Depending on the module, the information exchange between interfaces can be quite varying and infrequent.

**Proposed Solution:** A careful design of communication strategy and interface consideration is required to operate a distributed production system setting.

2. **Generalisability:** The approach is applied as use-cases to leak test system configuration and vision system guided force test configuration. The approach has been envisioned to apply to generalised manufacturing systems, but this does cause some potential issues.



**Decomposability**

**Affected Contribution:** Module Driven Configuration: A general framework for Self-Configuration

**Limitation:** A problem in this approach is that for achieving self-configuration, the configurable object needs to be decomposed to granular levels. The granularity can be broken down to the very basic physical entity of the system. This becomes a problem, as the approach warrants that variables that represent configuration be given a definition. In this approach, it is proposed that every CO is a module that brings functionality having requirements of variables that need to be configured. For this solution to be generalised, the functionalities should be encapsulated in well-formed modules. There also must be interfaces between modules so that variables may interact under constraints (e.g. functionality rules). To have such kind of module structure, it is necessary to have complete knowledge of the production operation that the system needs to perform (i.e. giving variables) and the CO in the system that needs to work together to execute the production system operation. One main limitation to this is the lack of information for encapsulating the variables to a functionality module and the presence of the interfaces that ensure communication and connection. This also requires distributed control through state transitions by accessing functionality modules. A centralised control mechanism that follows a sequence program can not be self-configured using this approach.

**Proposed Solution:** Functionalities encapsulated in well-formed

modules, all variables of the functionality defined and interface present.

3. **External Factors:** This approach relies on information represented for functionality in the form of modules. There can be factors that influence information capture to represent the functionality completely.

### **Information Limitation**

**Affected Contribution:** Adaptation Strategy for Self-Configuration in Production Systems

**Limitation:** There can be cases where the manufacturer of the system/product limits the information capture. This hinders the self-configuration approach as the variable information becomes restricted and state actions cannot map the requirements to variables.

In the case of the Adaptation Strategy, the approach requires an interface between functionality modules to the respective production system. This will require that each functionality in terms of variables can have its configuration updated on the system in real-time. So there is a need for having real-time control of the functionality modules on the production system.

**Proposed Solution:** RTC comes in the form of a production system-specific API or PLC program. The feature of RTC is dependent on the manufacturer supplying the production system.

4. **Composition:** This research involves the use of modules as building blocks for the production systems. The execution of the functionality of the production system will involve multimodule interactions based

on rules. The introduction of new modules is also a possible limitation factor.

### Module Rules

**Affected Contribution:** Module Driven Configuration: A general framework for Self-Configuration

**Limitation:** The modules consist of variables based on the functionality (i.e. of the module). To execute system functionality, the interaction between variables may be permitted subject to constraint rules (e.g. functionality rules). The composition requirement, in this case, is that the rule must be enabled, and an interface must exist between the variables. It can be said that in a production system composed of modules, the interaction is enabled between two or more variables present in different modules through rule constraint and the value taken by the variable is related to the constraint binding the variable.

**Proposed Solution:** Rules and constraints for all modules must be defined at the design stage.

### Introducing Modules

**Affected Contribution:** Module Driven Configuration: A general framework for Self-Configuration

**Limitation:** A limitation can be in the introduction of new modules, bringing additional functionalities, to the production system. The composition can execute the behaviour of the production system under underlying constraints. Each new functionality will be represented with a module, introducing a set of variables under constraint. This composition

of the production system relies on the variable and functionality rules.

**Proposed Solution:** Variable rules that confront the variable logic in respective modules and functionality rules representing rule logic and constraint on variables must be defined for the production system aggregate to operate.

5. **Production System Decomposition:** This approach assumes careful consideration of the modularity, defining the granularity level in the production system to meet all the functional requirements. This is necessary as this approach defines a module that performs a functionality and is a complete-structured representation of the Configurable Object.

#### **Module Definition**

**Affected Contribution:** Adaptation Strategy for Self-Configuration in Production Systems

**Limitation:** If consideration is not given to careful module definition, then unbounded structural decomposition aspects may result in a decreased performance, as there may be a physical obstruction or information being processed at different granular levels. This will result in inconsistency in the behaviour of the production system.

**Proposed Solution:** Careful module definition is necessary at design stage.

6. **Module Boundary and Actuation Scope:** This approach consists of modules, each responsible for certain functionality, connected through simple interfaces forming a production system.

#### **Module Boundary**

**Affected Contribution:** Module Driven Configuration: A general framework for Self-Configuration

**Limitation:** In current production systems, it is observed that the automation solutions' integral design generally maximises the production system's performance but lacks modularisation, enlarging the scope of action of the automation solution and misaligning it with physical components. This causes configuration settings to affect a large section of the production system, along with any failure. This may be coupled with other limitations of the presence of unstructured granularity, causing interaction between components at distinct abstraction levels, and creating inconsistencies in the behaviour of the system.

**Proposed Solution:**To define a module, it is necessary to consider the overall boundary of the production system, the manufacturing asset (that form the system) boundaries, and the respective mechatronic interfaces.

7. **Control Strategy and Path:** This approach provides a means of self-configuration at the functionality level. This means that there exists a distributed control in this approach to self-configuration.

#### **Control Behaviour**

**Affected Contribution:** Adaptation Strategy for Self-Configuration in Production Systems, Level-Based Classification for Self-configuration in Production Systems.

**Limitation:** One limitation is that the less hierarchical the control strategy is, the more critical the production system's boundary definition and action scope becomes. The emergent control behaviour, as all these functional modules

work together, becomes increasingly important. This emergent control behaviour because of the distributed nature becomes less controllable. The nature of information exchange through interfaces will vary in nature, size and time criticality. This will warrant the establishment of multiple communication channels and interfaces for different functional modules.

**Proposed Solution:** The control strategy and the logic that needs to be followed for the production system will define the number and operation of the control interfaces.

8. **Supporting Technologies:** Tools have been developed and used for the Adaptation Strategy. These supporting technologies have been developed for the application and objective of self-configuration. The tools developed facilitate the functional decomposition of the production system components. The tools support the optimisation objectives while combining local and network control while supporting cloud abstractions.

### **Technology Suitability**

**Affected Contribution:** A Generalised Workflow For Self-applications Based On Cloud-computing For Production Systems, Application of Multi-Agent Systems for Testing Applications, Cloud Based Decision-Making for Multi-Agent Production Systems & Service Based Approach to Asset Administration Shell for Controlling Testing Processes in Manufacturing

**Limitation:** A limitation in this approach is to use the technologies by understanding and deciding on the suitability in regard to the application. The factor of distributed control

is taken as the main motivator behind these technologies. In current industrial settings, most of the control technologies are positioned between networked and distributed systems. They rely on local system controls along with network servers to carry out operations. However, the central underlying control is still hierarchical, with optimisation usually focused on infrastructure rather than optimising the functionality as per objective.

### **Technology Latency**

**Affected Contribution:** A Generalised Workflow For Self-applications Based On Cloud-computing For Production Systems, Application of Multi-Agent Systems for Testing Applications, Cloud Based Decision-Making for Multi-Agent Production Systems & Service Based Approach to Asset Administration Shell for Controlling Testing Processes in Manufacturing

**Limitation:** A limitation observed in this approach is related to the actuating time frames of the functionality modules and the need to information, capture, optimise and update in these. Although mainstream automation solutions provide reasonable mechanisms for real-time control, there still exist some issues of proper control execution. The tools that support self-applications, need to be designed to value time-based contracts between the control constructs and the interfaces of the production system.

**Proposed Solution:** To overcome such limitations so that this approach may be applied to other self-applications, tools must be developed that address separately the cyber and physical com-

ponent representations for time-based controls. The rationale can be to:

- This will promote more intelligent automation setups without changing the local control of the system.
- Host the cyber representation in a computationally powerful environment for carrying out tasks of optimisation.
- Still maintain a simplistic RTC with an additional capability of connecting to other industrial middleware.

This is necessary as complex simulation will be required with functionality-based control. Simulations can assist in observing the emergent behaviours and test coverage of the change in the system.



# Bibliography

- Abbasi, M. and Houshmand, M. (2011). Production planning and performance optimization of reconfigurable manufacturing systems using genetic algorithm. *The International Journal of Advanced Manufacturing Technology*, 54:373–392.
- Aceto, L., Larsen, K. G., and Ingólfssdóttir, A. (2005). An introduction to milners ccs. *Course Notes for Semantics and Verification. Constantly under revision. The most recent version is available at the URL <http://www.cs.auc.dk/~luca/SV/Intro21ccs.pdf>, BRICS, Department of Computer Science, Aalborg, Denmark.*
- Aggour, K. S., Gupta, V. K., Ruscitto, D., Ajdelsztajn, L., Bian, X., Brosnan, K. H., Kumar, N. C., Dheeradhada, V., Hanlon, T., Iyer, N., et al. (2019). Artificial intelligence/machine learning in manufacturing and inspection: A ge perspective. *MRS Bulletin*, 44(7):545–558.
- Aguilar, J., Garces-Jimenez, A., R-moreno, M., and García, R. (2021). A systematic literature review on the use of artificial intelligence in energy self-management in smart buildings. *Renewable and Sustainable Energy Reviews*, 151:111530.
- Ahuja, S. P. and Myers, J. R. (2006). A survey on wireless grid computing. *The Journal of Supercomputing*, 37(1):3–21.

- Aldanondo, M., Veron, M., and Fargier, H. (1999). Configuration in manufacturing industry requirements, problems and definitions. In *IEEE SMC'99 Conference Proceedings. 1999 IEEE International Conference on Systems, Man, and Cybernetics (Cat. No.99CH37028)*, volume 6, pages 1009–1014 vol.6.
- Altıntaş, Y. (1994). Direct adaptive control of end milling process. *International Journal of Machine Tools and Manufacture*, 34(4):461–472.
- Ammar, M., Haleem, A., Javaid, M., Bahl, S., and Verma, A. S. (2022). Implementing industry 4.0 technologies in self-healing materials and digitally managing the quality of manufacturing. *Materials Today: Proceedings*, 52:2285–2294.
- Anderson, P. (1994). Towards a high-level machine configuration system. In *Proceedings of the 8th Conference on Systems Administration, LISA 1994*, pages 19–26.
- Andronie, M., Lăzăroiu, G., Iatagan, M., Uă, C., tefănescu, R., and Co-coatu, M. (2021). Artificial intelligence-based decision-making algorithms, internet of things sensing networks, and deep learning-assisted smart process management in cyber-physical production systems. *Electronics*, 10(20):2497.
- Anthony, R., Rettberg, A., Chen, D., Jahnich, I., de Boer, G., and Ekelin, C. (2007). Towards a dynamically reconfigurable automotive control system architecture. In *Embedded System Design: Topics, Techniques and Trends*, pages 71–84. Springer.
- Antsaklis, P. J., Passino, K. M., and Wang, S. J. (1989). Towards intelligent autonomous control systems: Architecture and fundamental issues. *Journal of Intelligent and Robotic Systems*.

- Antzoulatos, N. (2017). *Towards self-adaptable intelligent assembly systems*. PhD thesis, University of Nottingham.
- Antzoulatos, N., Castro, E., de Silva, L., Rocha, A. D., Ratchev, S., and Barata, J. (2017). A multi-agent framework for capability-based re-configuration of industrial assembly systems. *International Journal of Production Research*, 55(10):2950–2960.
- Appavoo, J., Hui, K., Soules, C. A., Wisniewski, R. W., Da Silva, D. M., Krieger, O., Auslander, M. A., Edelsohn, D. J., Gamsa, B., Ganger, G. R., McKenney, P., Ostrowski, M., Rosenburg, B., Stumm, M., and Xenidis, J. (2003). Enabling autonomic behavior in systems software with hot swapping. *IBM Systems Journal*.
- Arden, N. S., Fisher, A. C., Tyner, K., Lawrence, X. Y., Lee, S. L., and Kopcha, M. (2021). Industry 4.0 for pharmaceutical manufacturing: Preparing for the smart factories of the future. *International Journal of Pharmaceutics*, 602:120554.
- Bachula, K. and Zajac, J. (2013). The study of distributed manufacturing control system self-configuration. *Solid State Phenomena*, 196:148–155.
- Baetge, J. (1974). Betriebswirtschaftliche systemtheorie: Regelungstheoretische planungs-überwachungsmodelle für produktion. *Lagerung und Absatz, Opladen*, 24.
- Banerjee, D., Sen, S., and Chatterjee, A. (2016). Self learning analog/mixed-signal/RF systems: Dynamic adaptation to workload and environmental uncertainties. In *2015 IEEE/ACM International Conference on Computer-Aided Design, ICCAD 2015*, pages 59–64. Institute of Electrical and Electronics Engineers Inc.

- Bannat, A., Bautze, T., Beetz, M., Blume, J., Diepold, K., Ertelt, C., Geiger, F., Gmeiner, T., Gyger, T., Knoll, A., Lau, C., Lenz, C., Ostgathe, M., Reinhart, G., Roesel, W., Ruehr, T., Schuboe, A., Shea, K., Stork genannt Wersborg, I., Stork, S., Tekouo, W., Wallhoff, F., Wiesbeck, M., and Zaeh, M. F. (2011). Artificial cognition in production systems. *IEEE Transactions on Automation Science and Engineering*, 8(1):148–174.
- Barring, M. (2019). Increasing the Value of Data in Production Systems.
- Bauer, D., Schumacher, S., Gust, A., Seidelmann, J., and Bauernhansl, T. (2019). Characterization of autonomous production by a stage model. *Procedia CIRP*, 81:192–197.
- Beden, S., Cao, Q., and Beckmann, A. (2021). Semantic asset administration shells in industry 4.0: A survey. In *2021 4th IEEE International Conference on Industrial Cyber-Physical Systems (ICPS)*, pages 31–38. IEEE.
- Ben Ida, I., Balti, M., Chabaane, S., and Jemai, A. (2020). Self-adaptive early warning scoring system for smart hospital. In *The Impact of Digital Technologies on Public Health in Developed and Developing Countries: 18th International Conference, ICOST 2020, Hammamet, Tunisia, June 24–26, 2020, Proceedings 18*, pages 16–27. Springer.
- Berardinelli, L., Drath, R., Maetzler, E., and Wimmer, M. (2016). On the evolution of caex: A language engineering perspective. In *2016 IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA)*, pages 1–8. IEEE.
- Berns, A. and Ghosh, S. (2009). Dissecting self-\* properties. In *SASO 2009 - 3rd IEEE International Conference on Self-Adaptive and Self-Organizing Systems*.

- Best, A., Narang, S., Pasqualin, L., Barber, D., and Manocha, D. (2018). Autonomi-sim: Autonomous vehicle simulation platform with weather, sensing, and traffic control. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 1048–1056.
- Binotto, A. P. D., Wehrmeister, M. A., Kuijper, A., and Pereira, C. E. (2013). Sm@rtConfig: A context-aware runtime and tuning system using an aspect-oriented approach for data intensive engineering applications. *Control Engineering Practice*.
- Black, J. T. (2000). *Production systems flexible in manufacturing systems-MANUFACTURING SYSTEMS*, pages 423–431. Springer US, Boston, MA.
- Blum, M. and Schuh, G. (2017). Towards a data-oriented optimization of manufacturing processes. In *Proceedings of the 19th International Conference on Enterprise Information Systems*, volume 8, page 257.
- Bond, A. H. and Gasser, L. (2014). *Readings in distributed artificial intelligence*. Morgan Kaufmann.
- Borangi, T., Morariu, O., Răileanu, S., Trentesaux, D., Leitão, P., and Barata, J. (2020). Digital transformation of manufacturing. industry of the future with cyber-physical production systems. *Romanian Journal of Information Science and Technology*, 23(1):3–37.
- Bordel, B., Alcarria, R., Martín, D., Robles, T., and de Rivera, D. S. (2017). Self-configuration in humanized cyber-physical systems. *Journal of Ambient Intelligence and Humanized Computing*, 8:485–496.
- Bordel, B., Alcarria, R., Sanchez de Rivera, D., Martín, D., and Robles, T. (2018). Fast self-configuration in service-oriented smart environments

- for real-time applications. *Journal of Ambient Intelligence and Smart Environments*, 10(2):143–167.
- Botygin, I. A. and Tartakovsky, V. A. (2014). The development and simulation research of load balancing algorithm in network infrastructures. In *Proceedings of 2014 International Conference on Mechanical Engineering, Automation and Control Systems, MEACS 2014*.
- Braun, A.-T., Colangelo, E., and Steckel, T. (2018). Farming in the era of industrie 4.0. *Procedia Cirp*, 72:979–984.
- Brennan, R. W., Fletcher, M., and Norrie, D. H. (2002). An agent-based approach to reconfiguration of real-time distributed control systems. *IEEE Transactions on Robotics and Automation*.
- Burrell, J., Brooke, T., and Beckwith, R. (2004). Vineyard computing: Sensor networks in agricultural production. *IEEE Pervasive computing*, 3(1):38–45.
- Campos Sabioni, R., Daaboul, J., and Le Duigou, J. (2022). Concurrent optimisation of modular product and reconfigurable manufacturing system configuration: a customer-oriented offer for mass customisation. *International journal of production research*, 60(7):2275–2291.
- Cannon, R. H. (2003). *Dynamics of physical systems*. Courier Corporation.
- Casalicchio, E. and Gualandi, G. (2021). Asimov: A self-protecting control application for the smart factory. *Future Generation Computer Systems*, 115:213–235.
- Cavalieri, S., Mulé, S., and Salafia, M. G. (2019). Opc ua-based asset administration shell. In *IECON 2019-45th Annual Conference of the IEEE Industrial Electronics Society*, volume 1, pages 2982–2989. IEEE.

- Cavaliere, S. and Salafia, M. G. (2020). Asset administration shell for plc representation based on iec 61131-3. *IEEE Access*, 8:142606–142621.
- Cavaliere, S. and Salafia, M. G. (2021). Predictive maintenance model based on asset administration shell. In *ICEIS (2)*, pages 681–688.
- Cengiz, K., Sharma, R., Kottursamy, K., Singh, K. K., Topac, T., and Ozyurt, B. (2021). Recent emerging technologies for intelligent learning and analytics in big data. *Multimedia technologies in the internet of things environment*, pages 69–81.
- Cerpa, A. and Estrin, D. (2004). Ascent: Adaptive self-configuring sensor networks topologies. *IEEE transactions on mobile computing*, 3(3):272–285.
- Chatzigiannakis, I., Hasemann, H., Karnstedt, M., Kleine, O., Kröller, A., Leggieri, M., Pfisterer, D., Römer, K., and Truong, C. (2012). True self-configuration for the IoT. In *Proceedings of 2012 International Conference on the Internet of Things, IOT 2012*, pages 9–15.
- Chen, H., Xu, J., Zhang, B., and Fuhlbrigge, T. (2017). Improved parameter optimization method for complex assembly process in robotic manufacturing. *Industrial Robot: An International Journal*, 44(1):21–27.
- Cheng, S. W., Huang, A. C., Garlan, D., Schmerl, B., and Steenkiste, P. (2004a). An architecture for coordinating multiple self-management systems. In *Proceedings - Fourth Working IEEE/IFIP Conference on Software Architecture (WICSA 2004)*.
- Cheng, S.-W., Huang, A.-C., Garlan, D., Schmerl, B., and Steenkiste, P. (2004b). An architecture for coordinating multiple self-management

- systems. In *Proceedings. Fourth Working IEEE/IFIP Conference on Software Architecture (WICSA 2004)*, pages 243–252. IEEE.
- Chiriac, N., Hölttä-Otto, K., Lysy, D., and Suk Suh, E. (2011). Level of modularity and different levels of system granularity.
- Chirn, J.-L. and McFarlane, D. C. (2000). A holonic component-based approach to reconfigurable manufacturing control architecture. In *Proceedings 11th International Workshop on Database and Expert Systems Applications*, pages 219–223. IEEE.
- Cohen, J., Dasgupta, A., Ghosh, S., and Tixeuil, S. (2008). An exercise in selfish stabilization. *ACM Transactions on Autonomous and Adaptive Systems*.
- Coito, T., Martins, M. S., Firme, B., Figueiredo, J., Vieira, S. M., and Sousa, J. M. (2022). Assessing the impact of automation in pharmaceutical quality control labs using a digital twin. *Journal of Manufacturing Systems*, 62:270–285.
- Collados, K., Gorricho, J.-L., Serrat, J., Zheng, H., and Xu, K. (2015). An intelligent two-agent self-configuration approach for radio resource management. In *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, pages 706–712. IEEE.
- Council, N. R. and others (1998). *Visionary manufacturing challenges for 2020*. National Academies Press.
- da Silveira, F., Neto, I. R., Machado, F. M., da Silva, M. P., and Amaral, F. G. (2019). Analysis of industry 4.0 technologies applied to the health sector: systematic literature review. *Occupational and environmental safety and health*, pages 701–709.



- Dafflon, B., Moalla, N., and Ouzrout, Y. (2021). The challenges, approaches, and used techniques of cps for manufacturing in industry 4.0: a literature review. *The International Journal of Advanced Manufacturing Technology*, 113:2395–2412.
- De la Prieta, F. and Corchado, J. M. (2016). Cloud Computing and Multiagent Systems, a Promising Relationship.
- De Souza, L. M. S., Spiess, P., Guinard, D., Köhler, M., Karnouskos, S., and Savio, D. (2008). SOCRADES: A Web service based shop floor integration infrastructure. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*.
- Del Giudice, M., Scuotto, V., Papa, A., Tarba, S. Y., Bresciani, S., and Warkentin, M. (2021). A self-tuning model for smart manufacturing smes: Effects on digital innovation. *Journal of Product Innovation Management*, 38(1):68–89.
- Deuter, A. and Imort, S. (2021). Product lifecycle management with the asset administration shell. *Computers*, 10(7):84.
- Dey, A. (2016). Machine Learning Algorithms: A Review. *International Journal of Computer Science and Information Technologies*, 7(3):1174–1179.
- Dittrich, M. A. and Fohlmeister, S. (2020). Cooperative multi-agent system for production control using reinforcement learning. *CIRP Annals*, 69(1):389–392.
- Drath, R. (2012). Let’s talk automationml what is the effort of automationml programming? In *Proceedings of 2012 IEEE 17th International*

*Conference on Emerging Technologies & Factory Automation (ETFA 2012)*, pages 1–8. IEEE.

Drath, R. (2021). *Automationml: A practical guide*. Walter de Gruyter GmbH & Co KG.

DAniello, G., De Falco, M., and Mastrandrea, N. (2021). Designing a multi-agent system architecture for managing distributed operations within cloud manufacturing. *Evolutionary Intelligence*, 14:2051–2058.

Ebrahimi, M., Baboli, A., and Rother, E. (2018). A roadmap for evolution of existing production system toward the factory of the future: A case study in automotive industry. In *2018 IEEE International Conference on Technology Management, Operations and Decisions (ICTMOD)*, pages 274–281. IEEE.

ElMaraghy, W. and Urbanic, R. (2004). Assessment of manufacturing operational complexity. *CIRP annals*, 53(1):401–406.

Elements, C. (2018). IEC 61131-3: a standard programming resource. page 2.

ElMaraghy, H., Azab, A., Schuh, G., and Pulz, C. (2009). Managing variations in products, processes and manufacturing systems. *CIRP annals*, 58(1):441–446.

ElMaraghy, H., Monostori, L., Schuh, G., and ElMaraghy, W. (2021). Evolution and future of manufacturing systems. *CIRP Annals*, 70(2):635–658.

Elsken, T., Metzen, J. H., Hutter, F., et al. (2019). Neural architecture search: A survey. *J. Mach. Learn. Res.*, 20(55):1–21.

- Estrada-Jimenez, L. A., Pulikottil, T., Hien, N. N., Torayev, A., Rehman, H. U., Mo, F., Hojjati, S. N., and Barata, J. (2021). Integration of cutting-edge interoperability approaches in cyber-physical production systems and industry 4.0. In *Design, Applications, and Maintenance of Cyber-Physical Systems*, pages 144–172. IGI Global.
- Feng, L. (2009). Robustness evaluation of flexible manufacturing system considering the static & dynamic manufacturing environment. In *2009 International Conference on Information Management, Innovation Management and Industrial Engineering*, volume 4, pages 392–395. IEEE.
- Fernández-Sáez, A. M., Caivano, D., Genero, M., and Chaudron, M. R. (2015). On the use of uml documentation in software maintenance: Results from a survey in industry. In *2015 ACM/IEEE 18th International Conference on Model Driven Engineering Languages and Systems (MODELS)*, pages 292–301. IEEE.
- Ferreira, P. and Lohse, N. (2012). Configuration model for evolvable assembly systems. *CIRP Conference on Assembly Technologies and Systems (CATS) 2012*, (May):75–79.
- Fragapane, G., Ivanov, D., Peron, M., Sgarbossa, F., and Strandhagen, J. O. (2022). Increasing flexibility and productivity in industry 4.0 production networks with autonomous mobile robots and smart intralogistics. *Annals of operations research*, 308(1):125–143.
- Fritze, A., Mönks, U., and Lohweg, V. (2016). A concept for self-configuration of adaptive sensor and information fusion systems. In *2016 IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA)*, pages 1–4. IEEE.

- Gao, Q., Xu, H., and Li, A. (2022). The analysis of commodity demand predication in supply chain network based on particle swarm optimization algorithm. *Journal of Computational and Applied Mathematics*, 400:113760.
- Garcia, M. A. R., Rojas, R., Gualtieri, L., Rauch, E., and Matt, D. (2019). A human-in-the-loop cyber-physical system for collaborative assembly in smart manufacturing. *Procedia CIRP*, 81:600–605.
- Ghadimi, P., Toosi, F. G., and Heavey, C. (2018). A multi-agent systems approach for sustainable supplier selection and order allocation in a partnership supply chain. *European Journal of Operational Research*, 269(1):286–301.
- Gheibi, O., Weyns, D., and Quin, F. (2021). Applying machine learning in self-adaptive systems: A systematic literature review. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, 15(3):1–37.
- Givehchi, O., Landsdorf, K., Simoens, P., and Colombo, A. W. (2017). Interoperability for industrial cyber-physical systems: An approach for legacy systems. *IEEE Transactions on Industrial Informatics*, 13(6):3370–3378.
- Graessler, I., Hentze, J., and Poehler, A. (2019). Self-organizing production systems: Implications for product design. *Procedia CIRP*, 79:546–550.
- Günther, O., Ivantysynova, L., Rode, J., and Ziekow, H. (2009). It infrastructures in manufacturing: insights from seven case studies. *AMCIS 2009 Proceedings*, page 664.
- Guo, L., Wang, B., and Wang, W. (2009). Research of energy-efficiency algorithm based on on-demand load balancing for wireless sensor networks.

- Guo, Z., Zhang, Y., Liu, S., Wang, X. V., and Wang, L. (2023). Exploring self-organization and self-adaption for smart manufacturing complex networks. *Frontiers of Engineering Management*, 10(2):206–222.
- Guo, Z., Zhang, Y., Zhao, X., and Song, X. (2020). Cps-based self-adaptive collaborative control for smart production-logistics systems. *IEEE transactions on cybernetics*, 51(1):188–198.
- Har, L. L., Rashid, U. K., Te Chuan, L., Sen, S. C., and Xia, L. Y. (2022). Revolution of retail industry: from perspective of retail 1.0 to 4.0. *Procedia Computer Science*, 200:1615–1625.
- Hawkins, W., Brandl, D., and Boyes, W. (2010). *Applying ISA-88 in discrete and continuous manufacturing*, volume 2. Momentum Press.
- Hayes, T., Rustagi, N., Saia, J., and Trehan, A. (2008). The forgiving tree: A self-healing distributed data structure. In *Proceedings of the Annual ACM Symposium on Principles of Distributed Computing*.
- Hees, A. F. (2017). *System zur Produktionsplanung für rekonfigurierbare Produktionssysteme*, volume 331. Herbert Utz Verlag.
- Hofmeyr, S. A. and Forrest, S. (2000). Architecture for an artificial immune system. *Evolutionary computation*.
- Hülsmann, M. and Windt, K. (2007). *Understanding autonomous cooperation and control in logistics: the impact of autonomy on management, information, communication and material flow*. Springer Science & Business Media.
- Iarovyi, S., Lastra, J. L. M., Haber, R., and del Toro, R. (2015). From artificial cognitive systems and open architectures to cognitive manufacturing systems. In *2015 IEEE 13th International Conference on Industrial Informatics (INDIN)*, pages 1225–1232.

- Jann, J., Browning, L. M., and Burugula, R. S. (2003). Dynamic reconfigurations: Basic building blocks for autonomic computing on IBM pSeries servers.
- Järvenpää, E., Siltala, N., Hylli, O., Nylund, H., and Lanz, M. (2023). Semantic rules for capability matchmaking in the context of manufacturing system design and reconfiguration. *International Journal of Computer Integrated Manufacturing*, 36(1):128–154.
- Jia, F., Tzintzun, J., and Ahmad, R. (2020). An improved robot path planning algorithm for a novel self-adapting intelligent machine tending robotic system. In *Industrial and Robotic Systems: LASIRS 2019*, pages 53–64. Springer.
- Jones, G. M. and Romig, S. M. (1991). Cloning customized hosts (or customizing cloned hosts). In *Proceedings of the Fifth Large Installation Systems Administration Conference (LISA V)(USENIX Association: Berkeley, CA)*, page 233.
- Joseph, S. and Ignatious Monterio, J. (2019). Self configuring wireless sensor network for precision irrigation. In *Government College of Engineering Kannur, International Conference on Systems, Energy & Environment (ICSEE)*.
- Junker, U. (2006). Configuration. In *Foundations of Artificial Intelligence*, volume 2, pages 837–873. Elsevier.
- Kannisto, P., Hästbacka, D., and Kuikka, S. (2017). System architecture for mastering machine parameter optimisation. *Computers in Industry*, 85:39–47.
- Kao, H. A., Jin, W., Siegel, D., and Lee, J. (2015). A cyber physical inter-

face for automation systems-Methodology and examples. *Machines*, 3(2):93–106.

Karabegović, I., Karabegović, E., Mahmić, M., and Husak, E. (2020). Implementation of industry 4.0 and industrial robots in the manufacturing processes. In *New Technologies, Development and Application II 5*, pages 3–14. Springer.

Kephart, J. O. and Chess, D. M. (2003). The vision of autonomic computing. *Computer*.

Khalgui, M. and Mosbahi, O. (2010). Intelligent distributed control systems. *Information and Software Technology*, 52(12):1259–1271.

Khan, S., Phillips, P., Jennions, I., and Hockley, C. (2014). No fault found events in maintenance engineering part 1: Current trends, implications and organizational practices. *Reliability Engineering & System Safety*, 123:183–195.

Koronios, A., Nastasie, D., Chanana, V., and Haider, A. (2006). Integration through standards an overview of international standards for engineering asset management. *2nd World Congress on Engineering Asset Management and the Fourth International Conference on Condition Monitoring (WCEAM 2007)*, Harrogate, United Kingdom.

Kostal, P., Mudrikova, A., and Michal, D. (2019). Possibilities of intelligent flexible manufacturing systems. In *IOP Conference Series: Materials Science and Engineering*, volume 659, page 012035. IOP Publishing.

Kramer, J. and Magee, J. (1985). Dynamic Configuration for Distributed Systems. *IEEE Transactions on Software Engineering*.

Kruger, G. H., Shih, A. J., Hattingh, D. G., and Van Niekerk, T. I. (2011). Intelligent machine agent architecture for adaptive control optimiza-

tion of manufacturing processes. *Advanced Engineering Informatics*, 25(4):783–796.

Kuan, A. L., Rauschecker, U., Meier, M., Muckenhirn, R., Yip, A., Jagadeesan, A., and Corney, J. (2011). Cloud-based manufacturing-as-a-service environment for customized products. *eChallenges e-2011 Conference Proceedings*, pages 1–8.

Kubler, S., Derigent, W., Rondeau, ., Thomas, A., and Främling, K. (2013). Embedded Data on Intelligent Products - Impact on Real-Time Applications. *Communications in Computer and Information Science*, 183:25–34.

Lal, S. and Onwubolu, G. (2007). Three tiered web-based manufacturing system part 1: System development. *Robotics and Computer-Integrated Manufacturing*, 23(1):138–151.

Lee, C. K., Lin, B., Ng, K., Lv, Y., and Tai, W. (2019). Smart robotic mobile fulfillment system with dynamic conflict-free strategies considering cyber-physical integration. *Advanced Engineering Informatics*, 42:100998.

Lee, S. K., Kuo, H. C., Balkir, N. H., and Ozsoyoglu, G. (1997). Database server architecture for agile manufacturing. In *Proceedings - IEEE International Conference on Robotics and Automation*.

Leitão, P., Mařík, V., and Vrba, P. (2013). Past, present, and future of industrial agent applications. *IEEE Transactions on Industrial Informatics*.

Leitão, P. and Restivo, F. (2003). Towards autonomy, self-organisation and learning in holonic manufacturing. In *Lecture Notes in Artificial*



*Intelligence (Subseries of Lecture Notes in Computer Science)*, volume 2691, pages 544–553. Springer Verlag.

Leitão, P., Ribeiro, L., and Strasser, T. (2016). Smart Agents in Industrial. *Proceedings of the IEEE*, 104(5):1–16.

Lettner, D., Petruzelka, M., Rabiser, R., Angerer, F., Prähofer, H., and Grünbacher, P. (2013). Custom-developed vs. model-based configuration tools: Experiences from an industrial automation ecosystem. In *Proceedings of the 17th International Software Product Line Conference co-located workshops*, pages 52–58.

Li, C., Chen, Y., and Shang, Y. (2022). A review of industrial big data for decision making in intelligent manufacturing. *Engineering Science and Technology, an International Journal*, 29:101021.

Li, P. and Jiang, P. (2021). Enhanced agents in shared factory: enabling high-efficiency self-organization and sustainability of the shared manufacturing resources. *Journal of Cleaner Production*, 292:126020.

Li, S., Wang, H., Hu, S. J., Lin, Y.-T., and Abell, J. A. (2011). Automatic generation of assembly system configuration with equipment selection for automotive battery manufacturing. *Journal of Manufacturing Systems*, 30(4):188–195.

Liang, Q., Zhao, S., Zhang, J., Deng, H., Damm, W., Hess, D., Schweda, M., Sztipanovits, J., Bengler, K., Biebl, B., et al. (2024). Cyber-physical systems. *ACM Transactions on*, 8(1).

Liao, L. and Pavel, R. (2012). Machine tool feed axis health monitoring using plug-and-prognose technology. In *Technical Program for MFPT 2012, The Prognostics and Health Management Solutions Conference - PHM: Driving Efficient Operations and Maintenance*.

- Lieberoth-Leden, C., Fischer, J., Vogel-Heuser, B., and Fottner, J. (2019). Implementation, self-configuration and coordination of logistical functions for autonomous logistics modules in flexible automated material flow systems. *International Journal of Mechanical Engineering and Robotics Research*, 8(4):498–505.
- Liu, K., Liu, H., Li, T., Liu, Y., and Wang, Y. (2019). Intelligentization of machine tools: comprehensive thermal error compensation of machine-workpiece system. *The International Journal of Advanced Manufacturing Technology*, 102:3865–3877.
- Löcklin, A., Vietz, H., White, D., Ruppert, T., Jazdi, N., and Weyrich, M. (2021). Data administration shell for data-science-driven development. *Procedia CIRP*, 100:115–120.
- Löppenber, M. and Schwung, A. (2023). Self optimisation and automatic code generation by evolutionary algorithms in plc based controlling processes. In *2023 IEEE 21st International Conference on Industrial Informatics (INDIN)*, pages 1–6. IEEE.
- Lupton, D. (2013). The digitally engaged patient: Self-monitoring and self-care in the digital health era. *Social Theory & Health*, 11:256–270.
- Marrella, A., Mecella, M., and Sardina, S. (2017). Intelligent process adaptation in the SmartPM system. *ACM Transactions on Intelligent Systems and Technology*, 8(2).
- May, M. C., Overbeck, L., Wurster, M., Kuhnle, A., and Lanza, G. (2021). Foresighted digital twin for situational agent selection in production control. *Procedia CIRP*, 99:27–32.
- Mcfarlane, D. and Giannikas, V. (2013). Product intelligence in industrial control : Theory and practice. (November 2017).

- McFarlane, D., Giannikas, V., Wong, A. C., and Harrison, M. (2013). Product intelligence in industrial control: Theory and practice. *Annual Reviews in Control*, 37(1):69–88.
- Merz, M., Frank, T., and Vogel-Heuser, B. (2012). Dynamic redeployment of control software in distributed industrial automation systems during runtime. In *IEEE International Conference on Automation Science and Engineering*.
- Messig, M. and Goscinski, A. (2005). Self healing and self configuration in a wsrf grid environment. In *International Conference on Algorithms and Architectures for Parallel Processing*, pages 149–158. Springer.
- Monostori, L., Kádár, B., Bauernhansl, T., Kondoh, S., Kumara, S., Reinhart, G., Sauer, O., Schuh, G., Sihn, W., and Ueda, K. (2016). Cyber-physical systems in manufacturing. *Cirp Annals*, 65(2):621–641.
- Morgan, J., Halton, M., Qiao, Y., and Breslin, J. G. (2021). Industry 4.0 smart reconfigurable manufacturing machines. *Journal of Manufacturing Systems*, 59:481–506.
- Mount, D. J. (2015). Trends in leak testing: advancing the state-of-the-art. *Quality*, 54(2):S8–S8.
- Mourtzis, D., Vlachou, E., and Milas, N. (2016). Industrial Big Data as a Result of IoT Adoption in Manufacturing. *Procedia CIRP*, 55:290–295.
- Nayyar, A. and Puri, V. (2016). Smart farming: Iot based smart sensors agriculture stick for live temperature and moisture monitoring using arduino, cloud computing & solar technology. In *Proc. of The International Conference on Communication and Computing Systems (ICCCS-2016)*, pages 9781315364094–121.

- Neogi, N., Mohanta, D. K., and Dutta, P. K. (2014). Review of vision-based steel surface inspection systems. *EURASIP Journal on Image and Video Processing*, 2014(1):1–19.
- Neuhausen, J. (2001). *Methodik zur Gestaltung modularer Produktionssysteme für Unternehmen der Serienproduktion*. PhD thesis, Bibliothek der RWTH Aachen.
- Nie, Q., Tang, D., Zhu, H., and Sun, H. (2022). A multi-agent and internet of things framework of digital twin for optimized manufacturing control. *International Journal of Computer Integrated Manufacturing*, 35(10-11):1205–1226.
- Olsen, S., Wang, J., Ramirez-Serrano, A., and Brennan, R. W. (2005). Contingencies-based reconfiguration of distributed factory automation. In *Robotics and Computer-Integrated Manufacturing*.
- Opara-Martins, J., Sahandi, R., and Tian, F. (2016). Critical analysis of vendor lock-in and its impact on cloud computing migration: a business perspective. *Journal of Cloud Computing*, 5:1–18.
- Orehounig, K., Evins, R., and Dorer, V. (2015). Integration of decentralized energy systems in neighbourhoods using the energy hub approach. *Applied Energy*, 154:277–289.
- Padayachee, J. and Bright, G. (2012). Modular machine tools: Design and barriers to industrial implementation. *Journal of Manufacturing Systems*, 31(2):92–102.
- Park, K. T., Lee, S. H., and Noh, S. D. (2021). Information fusion and systematic logic library-generation methods for self-configuration of autonomous digital twin. *Journal of Intelligent Manufacturing*, pages 1–31.

- Perez-Leguizamo, C. (2016). Autonomous decentralized database system self configuration technology for high response. *IEICE Transactions on Communications*, 99(4):794–802.
- Pernkopf, F. and O’Leary, P. (2002). Visual inspection of machined metallic high-precision surfaces. *EURASIP Journal on Advances in Signal Processing*, 2002(7):1–12.
- Pethig, F., Niggemann, O., and Walter, A. (2017). Towards industrie 4.0 compliant configuration of condition monitoring services. In *2017 IEEE 15th international conference on industrial informatics (indin)*, pages 271–276. IEEE.
- Pierreval, H. and Tautou, L. (1997). Using evolutionary algorithms and simulation for the optimization of manufacturing systems. *IIE transactions*, 29(3):181–189.
- Priego, R., Armentia, A., Orive, D., Estévez, E., and Marcos, M. (2014). A model-based approach for achieving available automation systems. *IFAC Proceedings Volumes (IFAC-PapersOnline)*, 19:3438–3443.
- Puliafito, A., Celesti, A., Villari, M., and Fazio, M. (2015). Towards the integration between IoT and cloud computing: An approach for the secure self-configuration of embedded devices. *International Journal of Distributed Sensor Networks*, 2015.
- Pushkov, R., Ljubimov, A., and Evstafieva, S. (2021). Approach to build a universal communication protocol for equipment of different manufacturers. In *Advances in Automation II: Proceedings of the International Russian Automation Conference, RusAutoConf2020, September 6-12, 2020, Sochi, Russia*, pages 832–841. Springer.
- Qin, Z. and Lu, Y. (2021). Self-organizing manufacturing network: A

- paradigm towards smart manufacturing in mass personalization. *Journal of Manufacturing Systems*, 60:35–47.
- Radetzky, M., Rosebrock, C., and Bracke, S. (2019). Approach to adapt manufacturing process parameters systematically based on machine learning algorithms. *IFAC-PapersOnLine*, 52(13):1773–1778.
- Raptis, T. P., Passarella, A., and Conti, M. (2019). Data management in industry 4.0: State of the art and open challenges. *IEEE Access*, 7:97052–97093.
- Rehman, H. U., Chaplin, J. C., Zarzycki, L., Jones, M., and Ratchev, S. (2021a). Application of multi agent systems for leak testing. In *2021 9th International conference on systems and control (ICSC)*, pages 560–565. IEEE.
- Rehman, H. U., Chaplin, J. C., Zarzycki, L., Mo, F., Jones, M., and Ratchev, S. (2022). Service based approach to asset administration shell for controlling testing processes in manufacturing. *IFAC-PapersOnLine*, 55(10):1852–1857.
- Rehman, H. U., Chaplin, J. C., Zarzycki, L., and Ratchev, S. (2021b). A framework for self-configuration in manufacturing production systems. In *Doctoral Conference on Computing, Electrical and Industrial Systems*, pages 71–79. Springer.
- Rehman, H. U., Mo, F., Chaplin, J. C., Zarzycki, L., Jones, M., and Ratchev, S. (2024). A modular artificial intelligence and asset administration shell approach to streamline testing processes in manufacturing services. *Journal of Manufacturing Systems*, 72:424–436.
- Rehman, H. U., Pulikottil, T., Estrada-Jimenez, L. A., Mo, F., Chaplin, J. C., Barata, J., and Ratchev, S. (2021c). Cloud based decision

- making for multi-agent production systems. In *EPIA Conference on Artificial Intelligence*, pages 673–686. Springer.
- Reinhardt, I. C., Oliveira, J. C., and Ring, D. T. (2020). Current perspectives on the development of industry 4.0 in the pharmaceutical sector. *Journal of Industrial Information Integration*, 18:100131.
- Renna, P. (2011). Multi-agent based scheduling in manufacturing cells in a dynamic environment. *International Journal of Production Research*, 49(5):1285–1301.
- Ribeiro, L., Barata, J., and Mendes, P. (2008). MAS and SOA: Complementary automation paradigms. In *IFIP International Federation for Information Processing*.
- Ribeiro, L. and Björkman, M. (2017). Transitioning from standard automation solutions to cyber-physical production systems: an assessment of critical conceptual and technical challenges. *IEEE systems journal*, 12(4):3816–3827.
- Rodič, B. (2021). Self-organizing manufacturing systems in industry 4.0: Aspect of simulation modelling. In *Handbook of Research on Autopoiesis and Self-Sustaining Processes for Organizational Success*, pages 346–363. IGI Global.
- Rodrigues, N., Leitão, P., and Oliveira, E. (2015). Adaptive services reconfiguration in manufacturing environments using a multi-agent system approach. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 9433, pages 280–284. Springer Verlag.
- Rožanec, J. M., Lu, J., Rupnik, J., Škrjanc, M., Mladenčić, D., Fortuna, B., Zheng, X., and Kiritsis, D. (2022). Actionable cognitive twins for

- decision making in manufacturing. *International Journal of Production Research*, 60(2):452–478.
- Sakly, H. (2020). Self-organization and autonomous network survey. *International Journal of Network Security & Its Applications (IJNSA)* Vol, 12.
- Sakurada, L., Leitao, P., and De la Prieta, F. (2021). Towards the digitization using asset administration shells. In *IECON 2021–47th Annual Conference of the IEEE Industrial Electronics Society*, pages 1–6. IEEE.
- Salehie, M. and Tahvildari, L. (2005). Autonomic computing: emerging trends and open problems. *ACM SIGSOFT Software Engineering Notes*, 30(4):1–7.
- Sanchez, M., Exposito, E., and Aguilar, J. (2020). Implementing self-\* autonomic properties in self-coordinated manufacturing processes for the industry 4.0 context. *Computers in industry*, 121:103247.
- Scanzio, S., Cena, G., Zunino, C., and Valenzano, A. (2022). Machine learning to support self-configuration of industrial systems interconnected over wi-fi. In *2022 IEEE 27th International Conference on Emerging Technologies and Factory Automation (ETF A)*, pages 1–8. IEEE.
- Scheifele, S., Friedrich, J., Lechler, A., and Verl, A. (2014). Flexible, Self-configuring Control System for a Modular Production System. *Procedia Technology*, 15:398–405.
- Schleipen, M. (2010). Automated production monitoring and control system engineering by combining a standardized data format (caex) with standardized communication (opc ua). In *Factory Automation*. IntechOpen.



- Schleipen, M., Drath, R., and Sauer, O. (2008). The system-independent data exchange format caex for supporting an automatic configuration of a production monitoring and control system. In *2008 IEEE International Symposium on Industrial Electronics*, pages 1786–1791. IEEE.
- Sharp, M., Ak, R., and Hedberg Jr, T. (2018). A survey of the advancing use and development of machine learning in smart manufacturing. *Journal of manufacturing systems*, 48:170–179.
- Shaw, M. J., Subramaniam, C., Tan, G. W., and Welge, M. E. (2001). Knowledge management and data mining for marketing. *Decision Support Systems*.
- Shen, H., Zhang, H., Xu, Y., Chen, H., Zhu, Y., Zhang, Z., and Li, W. (2022). Multi-objective capacity configuration optimization of an integrated energy system considering economy and environment with harvest heat. *Energy Conversion and Management*, 269:116116.
- Shi, W. and Dustdar, S. (2016). The Promise of Edge Computing. *Computer*.
- Sibaliija, T. (2018). Application of simulated annealing in process optimization: a review. *Simulated Annealing: Introduction, Applications and Theory*, pages 1–14.
- Sibaliija, T. V. (2019). Particle swarm optimisation in designing parameters of manufacturing processes: A review (2008–2018). *Applied Soft Computing*, 84:105743.
- Silva, M. A. L., de Souza, S. R., Souza, M. J. F., and Bazzan, A. L. C. (2019). A reinforcement learning-based multi-agent framework applied for solving routing and scheduling problems. *Expert Systems with Applications*, 131:148–171.

- Sittón, I. and Rodríguez, S. (2017). Pattern extraction for the design of predictive models in industry 4.0. In *Advances in Intelligent Systems and Computing*.
- Somers, R. J., Douthwaite, J. A., Wagg, D. J., Walkinshaw, N., and Hierons, R. M. (2023). Digital-twin-based testing for cyber-physical systems: A systematic literature review. *Information and Software Technology*, 156:107145.
- Steiner, J. G. and Geer, D. E. (1988). Network services in the athena environment. *Project Athena, Massachusetts Institute of Technology, Cambridge, MA*, 2139.
- Sterritt, R. and Bustard, D. (2003). Towards an autonomic computing environment. In *Proceedings - International Workshop on Database and Expert Systems Applications, DEXA*.
- Strasser, S., Tripathi, S., and Kerschbaumer, R. (2018). An approach for adaptive parameter setting in manufacturing processes. In *DATA*, pages 24–32.
- Strasser, T. and Froschauer, R. (2012). Autonomous application recovery in distributed intelligent automation and control systems. *IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews*.
- Streit, A., Rosch, S., and Vogel-Heuser, B. (2014). Redeployment of control software during runtime for modular automation systems taking real-time and distributed I/O into consideration. In *19th IEEE International Conference on Emerging Technologies and Factory Automation, ETFA 2014*.
- Surucu, O., Gadsden, S. A., and Yawney, J. (2023). Condition monitoring

- using machine learning: A review of theory, applications, and recent advances. *Expert Systems with Applications*, 221:119738.
- Tamizi, M. G., Yaghoubi, M., and Najjaran, H. (2023). A review of recent trend in motion planning of industrial robots. *International Journal of Intelligent Robotics and Applications*, pages 1–22.
- Tang, H., Li, D., Wang, S., and Dong, Z. (2017). Casoa: an architecture for agent-based manufacturing system in the context of industry 4.0. *Ieee Access*, 6:12746–12754.
- Tang, T., Hu, T., Chen, M., Lin, R., and Chen, G. (2020). A deep convolutional neural network approach with information fusion for bearing fault diagnosis under different working conditions. *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, page 0954406220902181.
- Tantik, E. and Anderl, R. (2017). Potentials of the asset administration shell of industrie 4.0 for service-oriented business models. *Procedia CIRP*, 64:363–368.
- Taylor, E. et al. (2021). Autonomous vehicle decision-making algorithms and data-driven mobilities in networked transport systems. *Contemporary Readings in Law and Social Justice*, 13(1):9–19.
- Tirkolaei, E. B., Mahmoodkhani, J., Bourani, M. R., and Tavakkoli-Moghaddam, R. (2019). A self-learning particle swarm optimization for robust multi-echelon capacitated location–allocation–inventory problem. *Journal of Advanced Manufacturing Systems*, 18(04):677–694.
- Tobon-Mejia, D. A., Medjaher, K., and Zerhouni, N. (2012). CNC machine tools wear diagnostic and prognostic by using dynamic Bayesian networks. *Mechanical Systems and Signal Processing*.

- Tran, N.-H., Park, H.-S., Nguyen, Q.-V., and Hoang, T.-D. (2019). Development of a smart cyber-physical manufacturing system in the industry 4.0 context. *Applied Sciences*, 9(16):3325.
- Trendafilova, I. and Van Brussel, H. (2001). Non-linear dynamics tools for the motion analysis and condition monitoring of robot joints. *Mechanical Systems and Signal Processing*.
- Tuck, C. and Hague, R. (2006). The pivotal role of rapid manufacturing in the production of cost-effective customised products. *International Journal of Mass Customisation*.
- UK, G. O. V. (2020). National Data Strategy. pages 1–73.
- Vaisi, B. (2022). A review of optimization models and applications in robotic manufacturing systems: Industry 4.0 and beyond. *Decision analytics journal*, 2:100031.
- Valilai, O. F. and Houshmand, M. (2013). A collaborative and integrated platform to support distributed manufacturing system using a service-oriented approach based on cloud computing paradigm. *Robotics and Computer-Integrated Manufacturing*.
- Van Brussel, H., Wyns, J., Valckenaers, P., Bongaerts, L., and Peeters, P. (1998). Reference architecture for holonic manufacturing systems: Prosa. *Computers in industry*, 37(3):255–274.
- Vishnu, V., Varghese, K. G., and Gurumoorthy, B. (2023). A data-driven digital twin framework for key performance indicators in cnc machining processes. *International Journal of Computer Integrated Manufacturing*, 36(12):1823–1841.
- Wan, J., Chen, M., Xia, F., Di, L., and Zhou, K. (2013). From machine-to-

- machine communications towards cyber-physical systems. *Computer Science and Information Systems*, 10(3):1105–1128.
- Wang, J., Ma, Y., Zhang, L., Gao, R. X., and Wu, D. (2018). Deep learning for smart manufacturing: Methods and applications. *Journal of Manufacturing Systems*, 48:144–156.
- Wang, J., Xu, C., Zhang, J., and Zhong, R. (2022). Big data analytics for intelligent manufacturing systems: A review. *Journal of Manufacturing Systems*, 62:738–752.
- Wang, L., Du, Z., Dong, W., Shen, Y., and Zhao, G. (2019). Hierarchical human machine interaction learning for a lower extremity augmentation device. *International Journal of Social Robotics*, 11(1):123–139.
- Wang, P., Gong, Y., Xie, H., Liu, Y., and Nee, A. Y. (2017). Applying cbr to machine tool product configuration design oriented to customer requirements. *Chinese Journal of Mechanical Engineering*, 30:60–76.
- Wang, W. and Liu, F. (2012). The research of cloud manufacturing resource discovery mechanism. In *ICCSE 2012 - Proceedings of 2012 7th International Conference on Computer Science and Education*, pages 188–191.
- Wenger, M., Zoitl, A., and Müller, T. (2018). Connecting plcs with their asset administration shell for automatic device configuration. In *2018 IEEE 16th International Conference on Industrial Informatics (INDIN)*, pages 74–79. IEEE.
- Weyer, S., Schmitt, M., Ohmer, M., and Gorecky, D. (2015). Towards industry 4.0-standardization as the crucial challenge for highly modular, multi-vendor production systems. *Ifac-Papersonline*, 48(3):579–584.

- Wooldridge, M. (2009). *An introduction to multiagent systems*. John Wiley & sons.
- Wu, D., Rosen, D. W., and Schaefer, D. (2014). Cloud-based design and manufacturing: status and promise. In *Cloud-based design and manufacturing (CBDM)*, pages 1–24. Springer.
- Wu, D., Zhang, Y., Ourak, M., Niu, K., Dankelman, J., and Vander Poorten, E. B. (2021). Hysteresis modeling of robotic catheters based on long short-term memory network for improved environment reconstruction. *IEEE Robotics and Automation Letters*, pages 1–1.
- Xia, K., Fan, H., Huang, J., Wang, H., Ren, J., Jian, Q., and Wei, D. (2021). An intelligent self-service vending system for smart retail. *Sensors*, 21(10):3560.
- Xia, W. and Wu, Z. (2005). An effective hybrid optimization approach for multi-objective flexible job-shop scheduling problems. *Computers & industrial engineering*, 48(2):409–425.
- Xie, X. (2008). A review of recent advances in surface defect detection using texture analysis techniques. *ELCVIA: electronic letters on computer vision and image analysis*, pages 1–22.
- Xu, W., Shao, L., Yao, B., Zhou, Z., and Pham, D. T. (2016). Perception data-driven optimization of manufacturing equipment service scheduling in sustainable manufacturing. *Journal of Manufacturing Systems*, 41:86–101.
- Yan, J. and Vyatkin, V. (2013). Extension of reconfigurability provisions in IEC 61499. In *IEEE International Conference on Emerging Technologies and Factory Automation, ETFA*.

- Yan, K.-Q., Wang, S.-C., Chang, C.-P., and Lin, J. (2007). A hybrid load balancing policy underlying grid computing environment. *Computer Standards & Interfaces*, 29(2):161–173.
- Yang, L., Zou, H., Shang, C., Ye, X., and Rani, P. (2023). Adoption of information and digital technologies for sustainable smart manufacturing systems for industry 4.0 in small, medium, and micro enterprises (smmes). *Technological Forecasting and Social Change*, 188:122308.
- Yelles-Chaouche, A. R., Gurevsky, E., Brahimi, N., and Dolgui, A. (2021). Reconfigurable manufacturing systems from an optimisation perspective: a focused review of literature. *International Journal of Production Research*, 59(21):6400–6418.
- Yeung, W. L. (2018). Efficiency of task allocation based on contract net protocol with audience restriction in a manufacturing control application. *International Journal of Computer Integrated Manufacturing*, 31(10):1005–1017.
- Zaidi, A. A. and Kupzog, F. (2008). Microgrid automation-a self-configuring approach. In *2008 IEEE International Multitopic Conference*, pages 565–570. IEEE.
- Zhang, D., Yu, Z., and Chin, C.-Y. (2005). Context-aware infrastructure for personalized healthcare. *Studies in health technology and informatics*, 117:154–163.
- Zhang, H., Li, Z., Shu, W., and Chou, J. (2019). Ant colony optimization algorithm based on mobile sink data collection in industrial wireless sensor networks. *EURASIP Journal on Wireless Communications and Networking*, 2019(1):1–10.

- Zhang, L., Guo, H., Tao, F., Luo, Y. L., and Si, N. (2010). Flexible management of resource service composition in cloud manufacturing. In *IEEM2010 - IEEE International Conference on Industrial Engineering and Engineering Management*.
- Zheng, X., Lu, J., and Kiritsis, D. (2022). The emergence of cognitive digital twin: vision, challenges and opportunities. *International Journal of Production Research*, 60(24):7610–7632.
- Zhong, R. Y., Xu, X., Klotz, E., and Newman, S. T. (2017). Intelligent Manufacturing in the Context of Industry 4.0: A Review. *Engineering*, 3(5):616–630.
- Zhou, T., Tang, D., Zhu, H., and Zhang, Z. (2021). Multi-agent reinforcement learning for online scheduling in smart factories. *Robotics and Computer-Integrated Manufacturing*, 72:102202.
- Zhou, Y., Li, W., Wang, X., Qiu, Y., and Shen, W. (2022). Adaptive gradient descent enabled ant colony optimization for routing problems. *Swarm and Evolutionary Computation*, 70:101046.
- Zoitl, A. (2009). *Real-time Execution for IEC 61499*. Instrumentation, Systems, and Automation Society Research Triangle Park, NC.
- Zoph, B., Vasudevan, V., Shlens, J., and Le, Q. (2017). Automl for large scale image classification and object detection. *Google AI Blog*, 2:2017.



# Appendices

# Appendix A

## Survey Questionnaires

### A.1 Survey Questions

#### A.1.1 Industrial Challenges in Adopting Intelligent Production System

- Q 1.1: Which of the following characteristics are required to make a production system intelligent in your opinion?
- Q 1.2: What criteria of intelligent production systems have you seen at your manufacturing facility?
- Q 1.3: What criteria of production system intelligence do you think would give the best opportunities for productivity increase without being too complex or expensive to implement?
- Q 1.4: Where should intelligence be located on a production system?
- Q 1.5: In your opinion, is your current manufacturing infrastructure capable of supporting such intelligent production systems?

- Q 1.6: Please scale the challenges in your manufacturing setup for adopting intelligent production systems.
- Q 1.7: If you were to adopt intelligent production systems, please provide an estimated time frame for overcoming the challenges in your manufacturing environment.

### **A.1.2 Data in Manufacturing**

- Q 2.1: In your perspective, which of the following options is the most relevant ‘useful’ activity in manufacturing?
- Q 2.2: Which of the following is the single most important method you use to categorise data as ‘useful’?
- Q 2.3: Which type of data has the greatest impact at different stages of the product life-cycle?
- Q 2.4: In your experience, how often is productivity on the shop floor impacted by decisions based on data gathered during operation?
- Q 2.5: Multiple processing levels are required for turning raw data into actionable insight. Please rate these levels by how challenging they are?
- Q 2.6: How much time and effort is spent on each of these data analytics stages in your manufacturing environment?
- Q 2.7(i): Does your company use descriptive data analysis techniques?
- Q 2.7(i)a: Please select any known descriptive data analysis technique that is used at your organisation.

- Q 2.7(ii): Does your company use diagnostic data analysis techniques?
- Q 2.7(ii)a: Please select any known diagnostic data analysis technique that is used at your organisation.
- Q 2.7(iii): Does your company use predictive data analysis techniques?
- Q 2.7(iii)a: Please select any known predictive data analysis technique that is used at your organisation.
- Q 2.7(iv): Does your company use prescriptive data analysis techniques?
- Q 2.7(iv)a: Please select any known prescriptive data analysis technique that is used at your organisation.
- Q 2.8: What are the industrial network/automation protocols employed in your organisation?

### **A.1.3 Self-Configuring Production Systems in Manufacturing**

- Q 3.1: How do you typically enter the settings in a machine before its operation?
- Q 3.2: How do you determine the settings of a machine before its operation?
- Q 3.3: Do you think that machines that automatically adjust their settings, in response to a change, can be beneficial to your production environment for time and cost savings?

- Q 3.4: What kind of addition to your existing technological infrastructure will make it capable of realising machines that automatically adjust their settings?
- Q 3.5: Where in your setup do you see the most beneficial application of machines that can automatically self-adapt their settings?
- Q 3.6: In which domain do you think self-adapting machines can be beneficial?
- Q 3.7: What do you think is the best strategy for the adoption of such machines in your setup?
- Q 3.8: Do you think a tool that assists in evaluating the self-adapting capability of your machine (i.e., adjusting their settings themselves) would be helpful?
- Q 3.9: In your opinion, what are the challenges in using machines that automatically adjust their settings on your shop floor?
- Q 4.0: How do you think machines that adjust their settings can address common manufacturing issues?
- Q 4.1: In your opinion, how do you perceive the significance of machines, that adjust their settings, in solving operational challenges?

## A.2 Detailed Survey Questionnaire



This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant No. 814078

### Industrial Challenges in Adopting Intelligent Production Systems

This survey targets the barriers and challenges the manufacturing industry faces in adopting intelligent production systems in its environment.

The ever-increasing complexity in manufacturing means a higher distribution of intelligence and automation tasks among system components could lead to improved productivity. Intelligent production systems (equipment) addresses the domain and facilitates reliability, reusability and modularity in the manufacturing environment. A lot of research is being carried out in design, implementation and operation of intelligent production systems to make it simplified, more comprehensible and acceptable in industries.

6

Industry 4.0 and manufacturing automation vendors discuss "intelligence" in production systems frequently. However, there is some debate over what would actually make a production system intelligent. \*

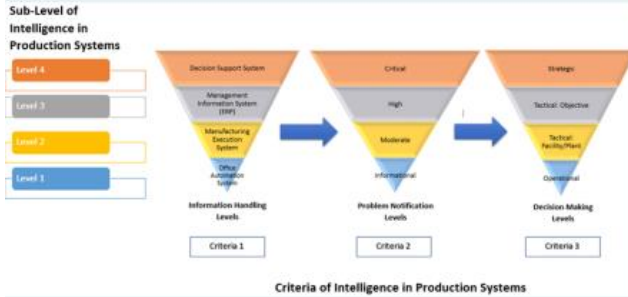
Which of the following characteristics are required to make a production system 'intelligent' in your opinion? Tick up to five.

- Has ability to uniquely identify itself
- Has ability to communicate with other devices
- Has ability to retain data
- Has ability to display its features and production requirements
- Has ability to make decisions
- Has ability to analyze data
- Has ability to predict and inform of future scenarios
- Has ability to manage its life cycle
- Has ability to ensure secure interactions
- Has ability to adapt to network changes
- Has ability to operate as a 'plug and play' product
- Has ability to provision and exchange data with multiple sources
- Has ability to cooperate with other machines
- Other

7

What criteria of intelligent production systems have you seen at your manufacturing facility? \*

Take a combination from the picture below to develop a mix that represents your intelligent production systems. For e.g. an intelligent production system in an industry provides 'information' to 'MES' which displays 'high' level error that may impact 'tactical objectives' of the company. Note: Problem identification at 'informational' level includes logs of all activities, at 'moderate' level the key activities, errors and results, at 'high' level the key parameters, errors and results and at 'critical' level includes only error and results.



	Level 1	Level 2	Level 3	Level 4
C1: Information Handling	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
C2: Problem Identification	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
C3: Decision Making	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

8

What criteria of production system intelligence do you think would give the best opportunities for productivity increase without being too complex or expensive to implement? \*

Please provide your answer in form of desirable mix from the picture.



	Level 1	Level 2	Level 3	Level 4
C1: Information Handling	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
C2: Problem Identification	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
C3: Decision Making	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

9

Where should the intelligence be located on a production system? Please share your opinion on the best way of gathering, storing and processing intelligence data. \*

- On the production system (Intelligence on product)
- Elsewhere on the network (Intelligence on Shared Network)
- Other

10

In your opinion, is your current manufacturing infrastructure capable of supporting such intelligent production systems? \*

Please provide best answer while considering your manufacturing setup, systems, processes, network and data processing capability.

- Yes
- No
- Maybe
- I lack awareness of the complete infrastructure
- Up to a certain extent

11

Please scale the challenges in your manufacturing setup for adopting intelligent production systems. \*

Please scale from least significant (1) to most significant (5) for the following challenges to adoption.

	1	2	3	4	5
Challenges in infrastructure	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Challenges in legislation	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Challenges in data security	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Challenges in data transport	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Challenges in filtration of data	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Challenges in training	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Bias/Resistance to change	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Challenges in business process re-engineering (BPR)	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Incompatible legacy equipment	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Lack of knowledge	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Challenge in cost/investment	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>



If you were to adopt intelligent production systems, please provide an estimated time-frame for overcoming the challenges in your manufacturing environment \*

Please select the most feasible time in your opinion that is required to overcome the challenge.

	0-3 months	4-8 months	9-12 months	2-3 years	4-5 years	> 5 years	Never
Challenges in infrastructure	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Challenges in legislation	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Challenges in data security	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Challenges in data transport	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Challenges in filtration of data	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Challenges in training	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Bias/Resistance to change	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Challenges in business process re-engineering (BPR)	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Incompatible legacy equipment	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Lack of knowledge	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Challenge in cost/investment	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Challenge of infeasible available solutions	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>



This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant No. 814078

## Self-Configuring Production Systems in Manufacturing

In the competitive age of globalised markets, we experience abrupt shifts in demand and customer requirements. To manage this, production systems must be flexible and easy to combine. Self-configuring production systems are at the forefront of this technology.

The main idea is to have decision-making capability for setting up parameters, optimization, and compatibility on the system itself rather than coordinated externally. An unknown, however, is the readiness of current manufacturing environments to accept self-configuring production systems.

This survey seeks the insight of industrial professionals from their own perspectives on the requirements of self-configuring production systems.

Note: This survey is the third in a series developed for this project. If you have not already done so, please contribute your opinions by completing the previous surveys once you have finished this survey. The link to the previous surveys are:

### 1. Industrial Challenges in Adopting Intelligent Production Systems :

<https://forms.office.com/Pages/ResponsePage.aspx?id=7qe9Z4D970GskTWEGckKHrO8XUjwhVtGsdRwEHoMR3xUN0RHtVdJT1ZCSkiWS0tSWIURENETTJYWC4u>

### 2. Data in Manufacturing :

<https://forms.office.com/Pages/ResponsePage.aspx?id=7qe9Z4D970GskTWEGckKHrO8XUjwhVtGsdRwEHoMR3xUM1VZQVlyNkpYWhLNVm4QVhFSTU5UDVVS54u>

In the competitive age of globalised markets, we experience abrupt shifts in demand and customer requirements. To manage this, production systems must be flexible and easy to combine. Self-configuring production systems are at the forefront of this technology.

The main idea is to have decision-making capability for setting up parameters, optimization, and compatibility on the system itself rather than coordinated externally. An unknown, however, is the readiness of current manufacturing environments to accept self-configuring production systems.

This survey seeks the insight of industrial professionals from their own perspectives on the requirements of self-configuring production systems.

6

How do you typically enter the settings in a machine before its operation?

- It is a Manual Process, settings are entered by the operator.
- Through a digital interface.
- Some settings are entered manually, others through a digital interface.

7

How do you determine the settings of a machine before its operation?

- Derived from Customer Requirements.
- Based on Product and Process Requirements.
- Depending on Quality Standard Criteria.
- Best-fit settings that are observed in-house.

8

Do you think that machines that automatically adjust their settings, in response to a change, can be beneficial to your production environment for time and cost savings?

- They will reduce cost significantly.
- They will decrease the process time.
- They will save both time and cost.
- They will not have significant impact on production.
- They will not have any impact.

9

What kind of addition to your existing technological infrastructure will make it capable of realising machines that automatically adjust their settings?

- No change needed, our technological infrastructure is capable.
- A standard data format is needed.
- Data connections between systems are required.
- We would require both data connections, and a data format.

10

Where in your setup do you see the **most beneficial** application of machines that can automatically self-adapt their settings?

- Setup and Changeover
- Testing and Development
- Production Operation
- Maintenance and Problem Identification
- Production Optimisation

11

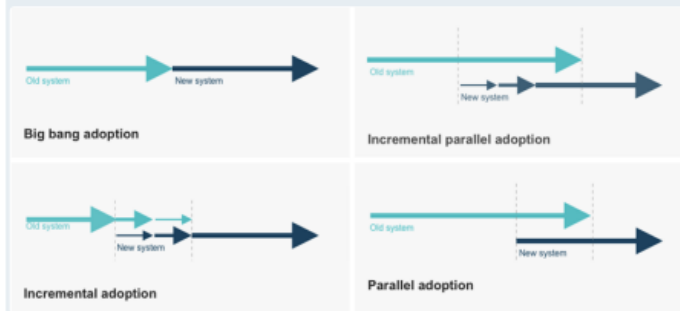
In which domain do you think self-adapting machines can be beneficial?

Please scale from least significant (1) to most significant (5) for the following domains in manufacturing.

	1	2	3	4	5
Factory Automation	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Warehousing	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Machine Efficiency	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Tracking and tracing	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Testing	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Minimizing downtime	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Process Integration	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Logistics	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Ergonomics	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Machining	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Assembly	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

12

What do you think is the best strategy for adoption of such machines in your setup?



- Incremental
- Big Bang
- Parallel
- Incremental parallel

13

Do you think a tool that assists in evaluating the self-adapting capability of your machine (i.e., adjusting their settings themselves) would be helpful?

- Yes
- No
- Maybe

In your opinion, what are the challenges in using machines that automatically adjust their settings on your shop-floor?

Please scale from least significant (1) to most significant (5) for each of the following challenges.

	1	2	3	4	5
Infrastructure Challenge	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Challenge in lack of benchmarks for process	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Infeasibility with deployed solutions	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Bias/Resistance to Change	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Lack of knowledge	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Lack of Control	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Challenge in business process re-engineering	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
No clear direction on adapting to changing conditions	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Fear of uncertain data or data loss	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Incompatibility with existing equipment	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

15

How do you think machines that adjust their settings can address common manufacturing issues?

Please scale from least effective (1) to most effective (5) for the following manufacturing issues.

	1	2	3	4	5
Employment disruptions	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
High Implementation Costs	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Lack of Skill	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Lack of IoT Knowledge	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Lack of Software	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Low Organisation and Process Adaptation Capability	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
No Standard Reference Architecture	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Privacy and Security concerns	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Compatibility Issues	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Regulatory and Legal Requirements	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Contractual Requirements	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

16

In your opinion, how do you perceive the significance of machines, that adjust their settings, in solving operational challenges?

Please scale from least significant (1) to most significant (5) impact for the following operational challenges.

	1	2	3	4	5
Difficulty in predicting settings	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Issues in managing product quality through the line	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Difficulty in Meeting objective	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Overcoming experience requirements	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Reacting to market demand	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>



This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant No. 814078

## Data in Manufacturing

With increasingly large amounts of data collected in manufacturing settings identifying, filtering and using the specific data that can create value is a challenge. The value of data can be defined not only in terms of economic benefits it may bring but also as an increase in quality and consistency of existing data.

These objectives are diverse and depend on the application. In regards to manufacturing it is important to understand the 'useful' aspect of gathered data, 'useful' filtering and analysis, 'useful' transformation and 'good' decisions. These questions aim to identify such components.

Note: This survey is second in list of surveys developed for this project. Please help by filling that too once you are done with this survey (if you have not already done so). The link to the previous survey is as follows:

<https://forms.office.com/Pages/ResponsePage.aspx?id=7qe9Z4D970GskTWEGCkKHrO8XlJwhVtGsdRwEHoMR3xUN0RHTVdJT1ZCSkIW50tSWIJBENETJYWC4u>

With increasingly large amounts of data collected in manufacturing settings identifying, filtering and using the specific data that can create value is a challenge. The value of data can be defined not only in terms of economic benefits it may bring but also as an increase in quality and consistency of existing data.

These objectives are diverse and depend on the application. In regards to manufacturing it is important to understand the 'useful' aspect of gathered data, 'useful' filtering and analysis, 'useful' transformation and 'good' decisions. These questions aim to identify such components.

6. In your perspective which of the following options is the most relevant 'useful' activity in manufacturing? \*

- A 'useful' manufacturing activity is the one that goes on to fulfil customer requirements.
- A 'useful' manufacturing activity is the one that goes on to fulfil manufacturer requirements.
- A 'useful' manufacturing activity is the one that goes on to improve manufacturing efficiency.
- A 'useful' manufacturing activity is the one that goes on to improve manufacturer budget utilisation.
- A 'useful' manufacturing activity is the one that goes on to be compliant with quality standards.
- A 'useful' manufacturing activity is the one that offers consistency in production.
- A 'useful' manufacturing activity is the one that reduces risk, defects and improves quality.
- I do not know

7. Which of the following is the single most important method you use to categorize data as 'useful'? \*

- Mathematical thresholding (Data value helps in getting insight)
- An Expert's opinion on accumulated data
- Historical data comparison for predicting future trends
- Identifying relevant/influencing data sources
- I do not know
- Other

8. Which type of data has highest impact at different stages of product life-cycle? \*

	Heuristic Data	Guidelines	Process Data	Material Data	Customer Data	Life Cycle Data	No Data
Requirement Generation	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Conceptual Design	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Embodiment Design	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Detail Design	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Production Planning	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Manufacturing	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Testing	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Dispatching	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

9. In your experience, how often is productivity on the shop floor impacted by decisions based on data gathered during operation? \*

- Never
- Rarely
- Sometimes
- Always
- Often

10. Multiple processing levels are required for turning raw data into actionable insight. Please rate these levels by how challenging they are. \*

Scale from least significant (1) to most significant (5).

	1	2	3	4	5
Challenges in COLLECTING data from data source	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Challenges in CLEANING the data	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Challenges in COMBINING data streams into models for analysis	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Challenges in CONDUCTING analysis	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Challenges in INTERPRETING the results	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Challenges in UTILIZING the insight (manually or automatically)	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>



11. Data Analytics is the science of analysing raw data gathered from the production system to find insights and to make better decisions.

How much time and effort is spent on each of these data analytics stages in your manufacturing environment? \*

Please select from none(0) to significant effort(5).

	0	1	2	3	4	5
Descriptive (what happened?)	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Diagnostic (why it happened?)	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Predictive (what might happen?)	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Prescription (what should be done?)	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

12. Does your company use descriptive data analysis techniques? \*

- Yes, and I am familiar with them.
- Yes, but I don't know specifics.
- No.
- I don't know.

13. Please select any known descriptive data analysis technique that is used at your organisation. \*

- Classification
- Thematic Analysis
- Typification
- Network Analysis
- Grounded Theory
- Close Reading
- Statistical Reporting
- Association Rule Mining
- Clustering
- Data visualization
- Querying
- Sequential pattern mining
- Time-series Analysis
- Casual Analysis
- Correlation Analysis
- Delphi Method

14. Does your company use diagnostic data analysis techniques? \*

- Yes, and I am familiar with them.
- Yes, but I don't know specifics.
- No.
- I don't know.

15. Please select any known diagnostic data analysis technique that is used at your organisation. \*

- Cluster Analysis
- Factor Analysis
- Multiregression
- Bayesian Clustering
- K-Nearest Neighbours
- Self-Organizing maps
- Principal Component Analysis
- Graph and Affinity Analysis
- On Board Diagnostic
- Root Cause Analysis
- FMEA
- Risk/Fault/Safety Modelling
- Reliability Engineering
- Quality Engineering
- Simulation
- Correlation
- Regression
- Social Network Analysis
- Sentiment Analysis

16. Does your company use predictive data analysis techniques? \*

- Yes, and I am familiar with them.
- Yes, but I don't know specifics.
- No.
- I don't know.

17. Please select any known predictive data analysis technique that is used at your organisation. \*

- Trend Analysis
- Reliability Growth
- Actuarial Growth
- Residual Life
- Simulation
- Data Mining
- Correlation
- Regression
- Social Network Analysis
- Sentiment Analysis
- What-IF Analysis
- Neural Networks
- Data Visualization
- Machine Learning

18. Does your company use prescriptive data analysis techniques? \*

- Yes, and I am familiar with them.
- Yes, but I don't know specifics.
- No.
- I don't know.

19. Please select any known prescriptive data analysis technique that is used at your organisation. \*

- Discrete Choice Modelling
- Linear Programming
- Non-linear programming
- Value Analysis
- Mash-Up Analytics
- Extended Data Model

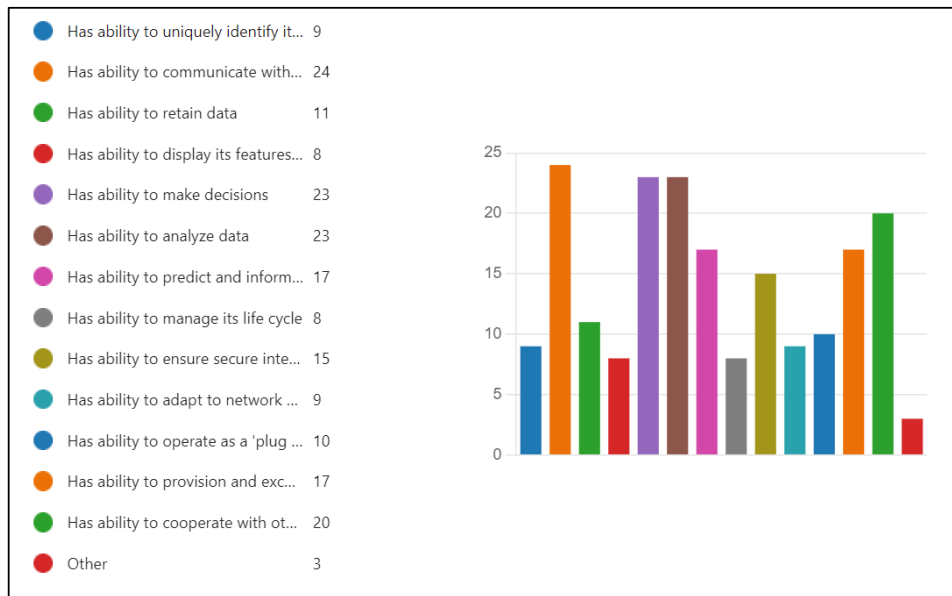
20. Gathering data from a manufacturing line often requires use of a manufacturing fieldbus, network, or protocol.

What are the industrial network/automation protocols employed in your organisation? \*

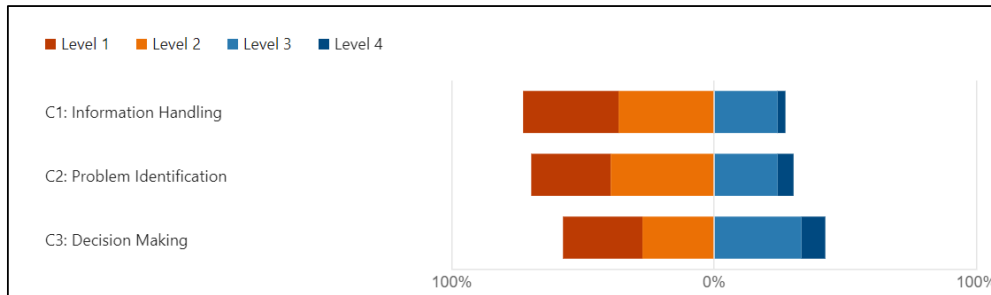
- EtherNet/IP
- PROFINET
- EtherCAT
- POWERLINK
- Modbus-TCP
- WLAN
- PROFIBUS DP
- CC-Link
- Modbus-RTU
- CANopen
- DeviceNet
- MTConnect
- OPC
- OPC UA
- OMG DDS
- Manual data gathering
- None
- Other

## A.3 Survey Response

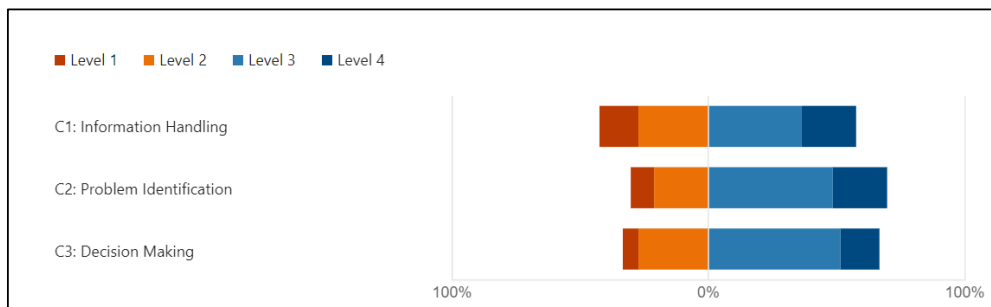
### Survey 1: Industrial Challenges in Adopting Intelligent Production Systems



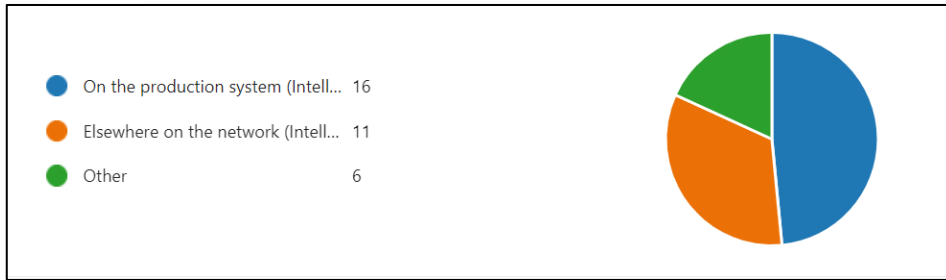
**Characteristics required to make a production system “Intelligent”**



**Response to Current Criteria of intelligence in the production system**



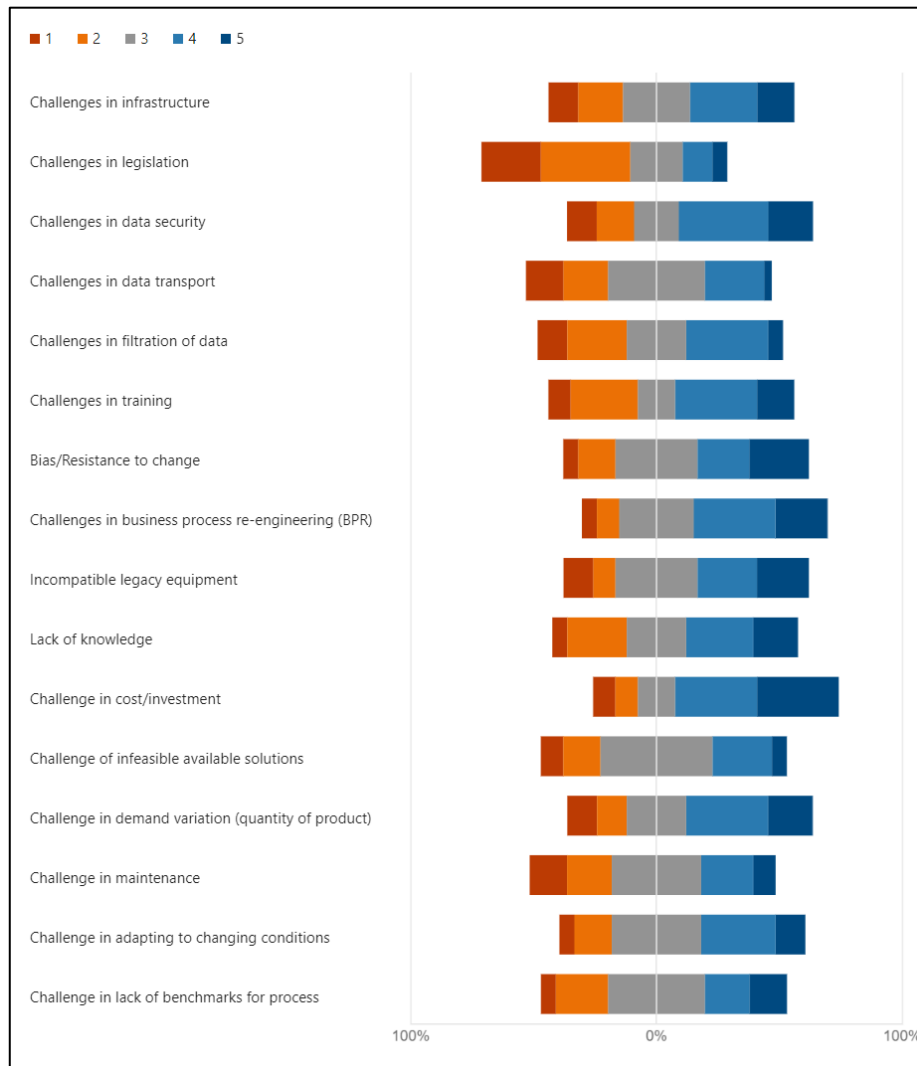
**Response to Desired Criteria of intelligence in production system for productivity increase**



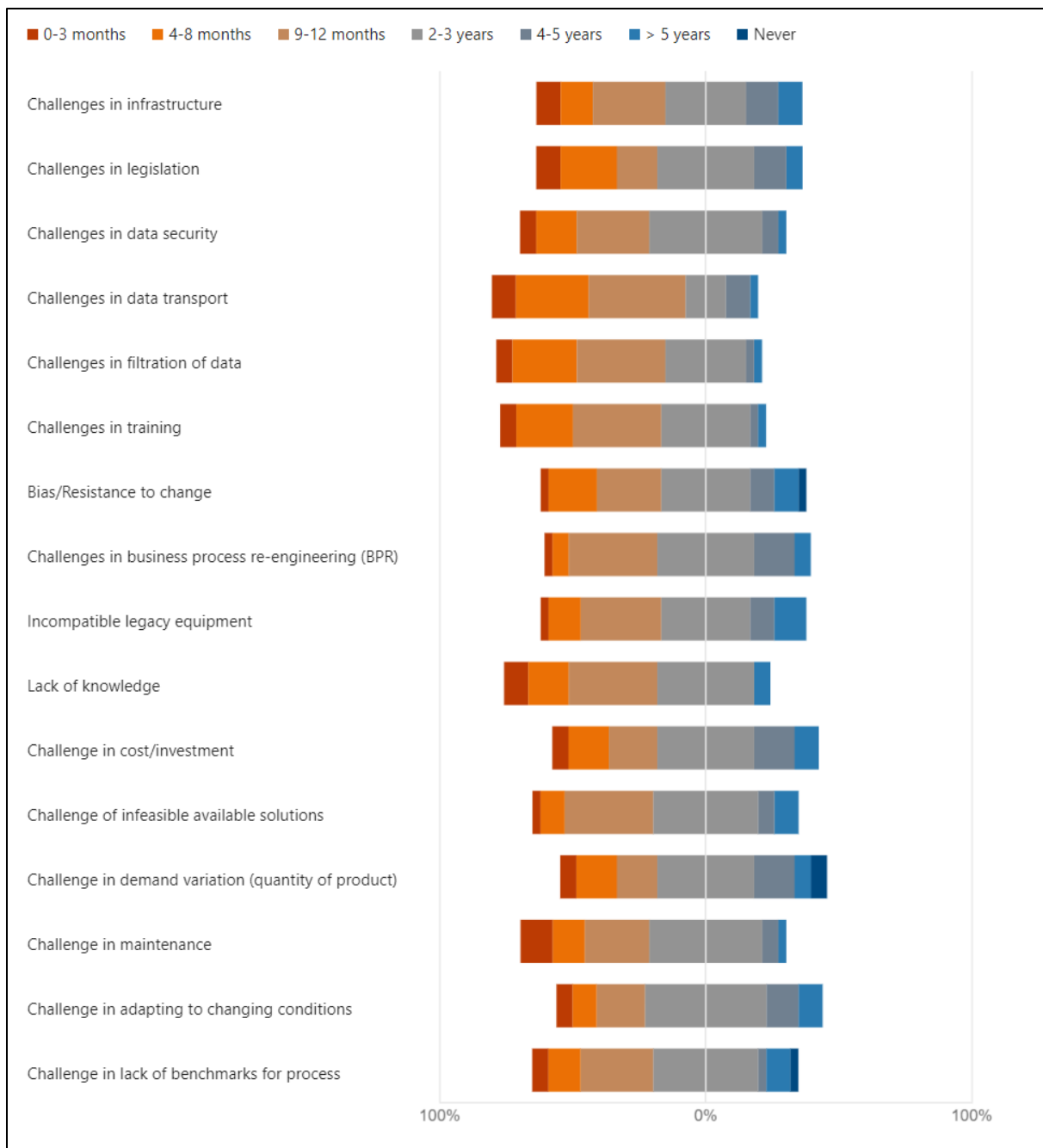
**Location of intelligence on the production system**



**Manufacturing Infrastructure for supporting intelligent production systems**

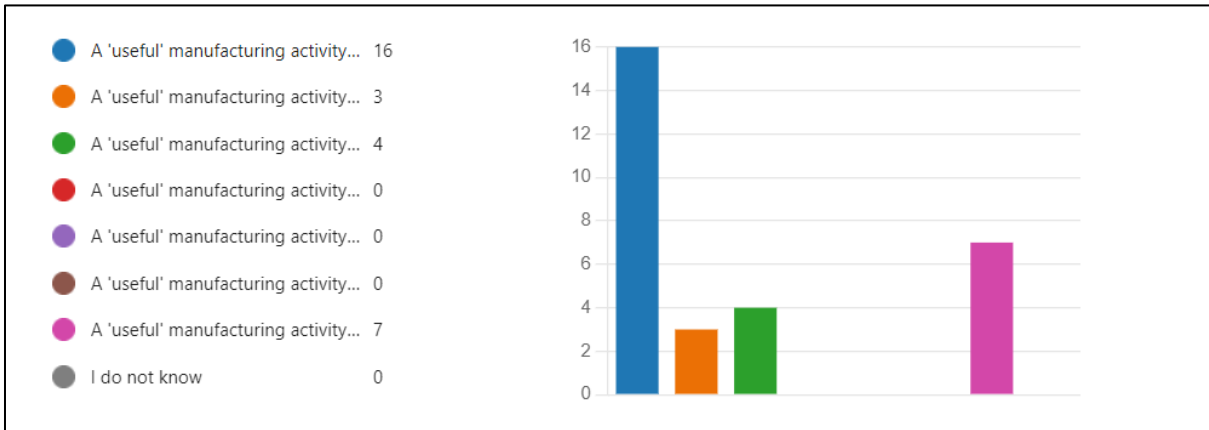


**Challenges in the adoption of intelligent production systems**

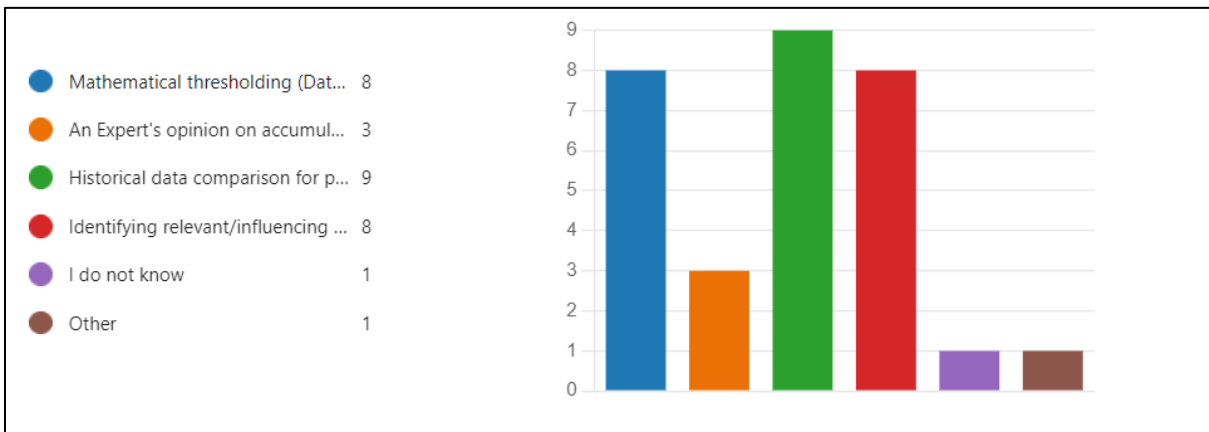


**Timeframe for overcoming the challenges in the manufacturing environment**

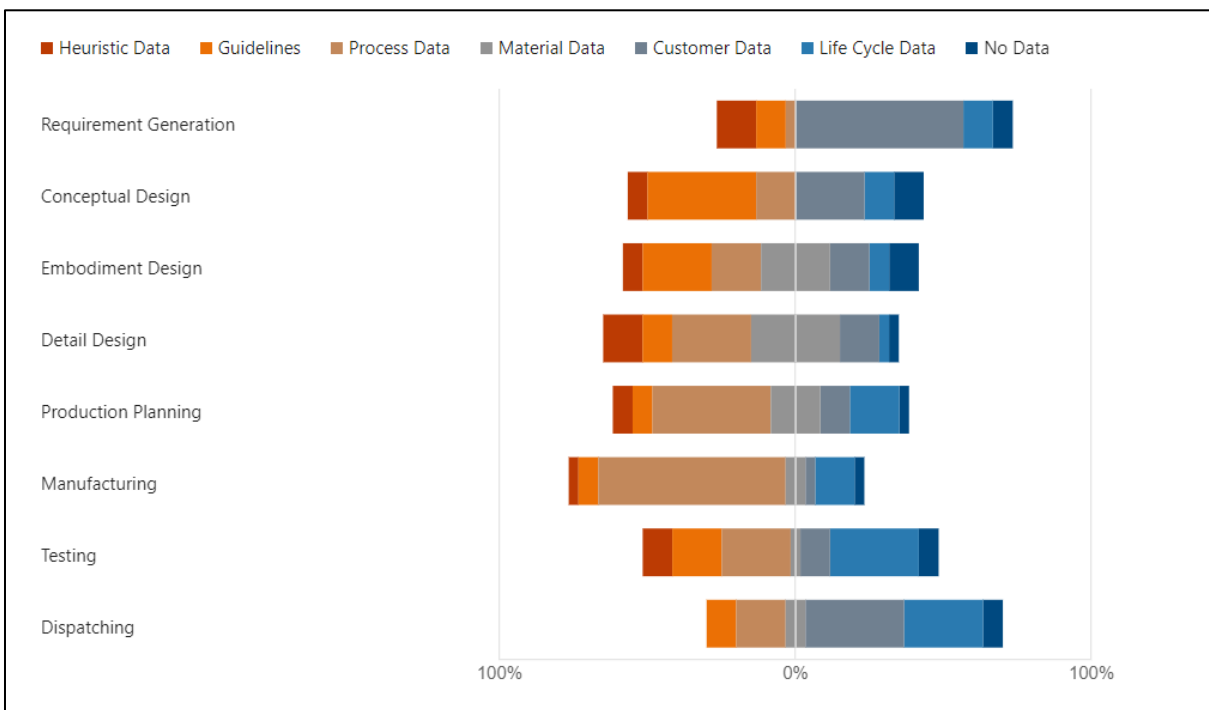
### Survey 2: Data in Manufacturing



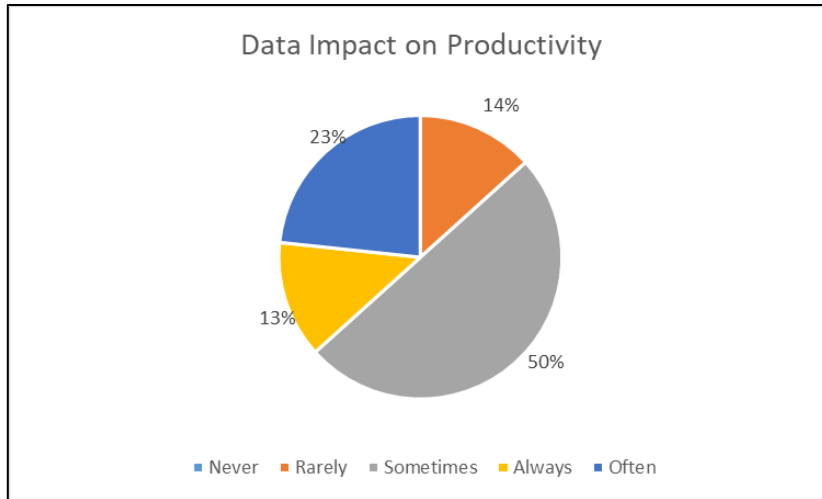
### Most relevant 'useful' activity in manufacturing



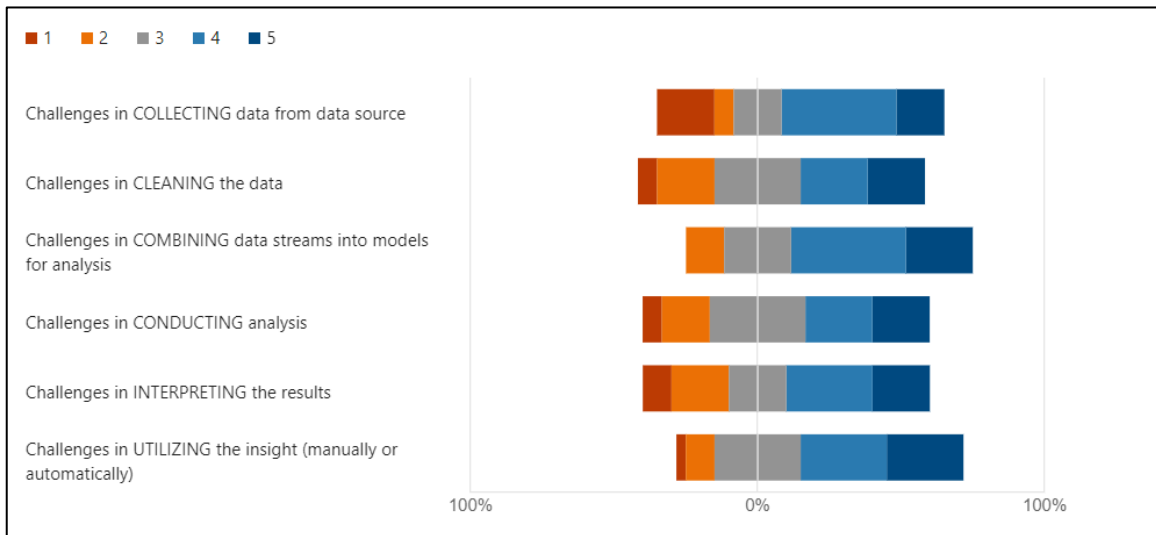
### Most important method to categorise 'useful' data



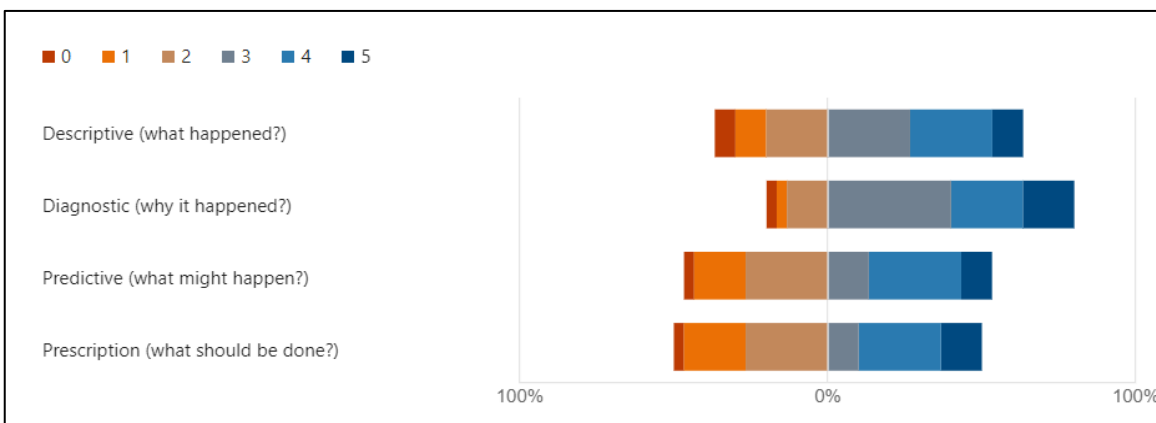
### Impact of data at different stages of product life-cycle



**Impact of data on shop-floor decisions**

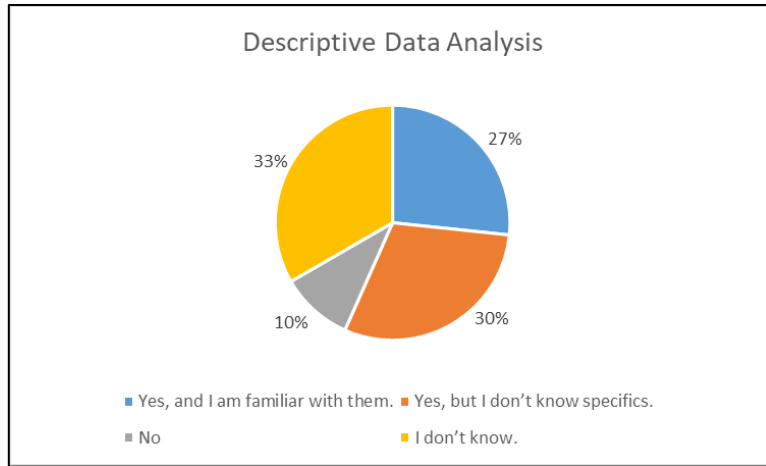


**Challenges in each processing level for converting data into insight.**

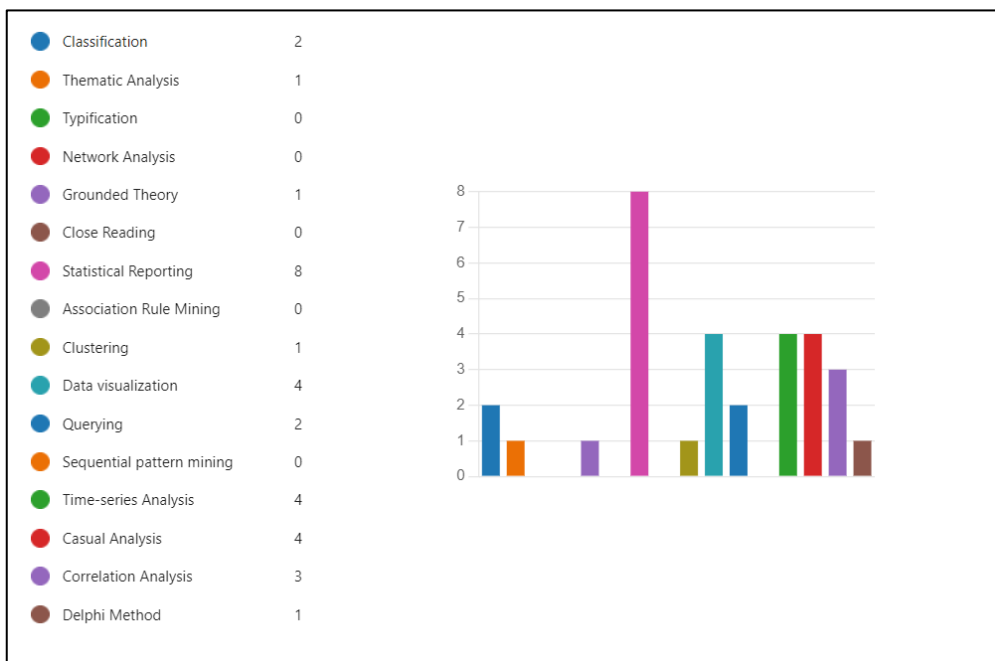


**Time and effort spent on each data analytics stage in manufacturing environment**

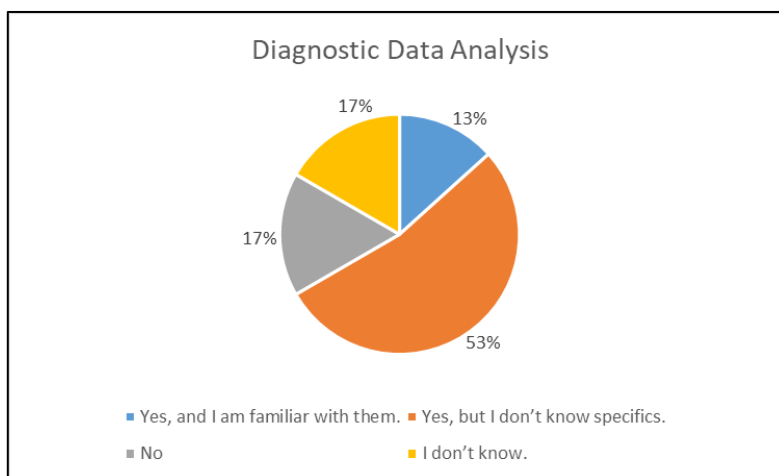




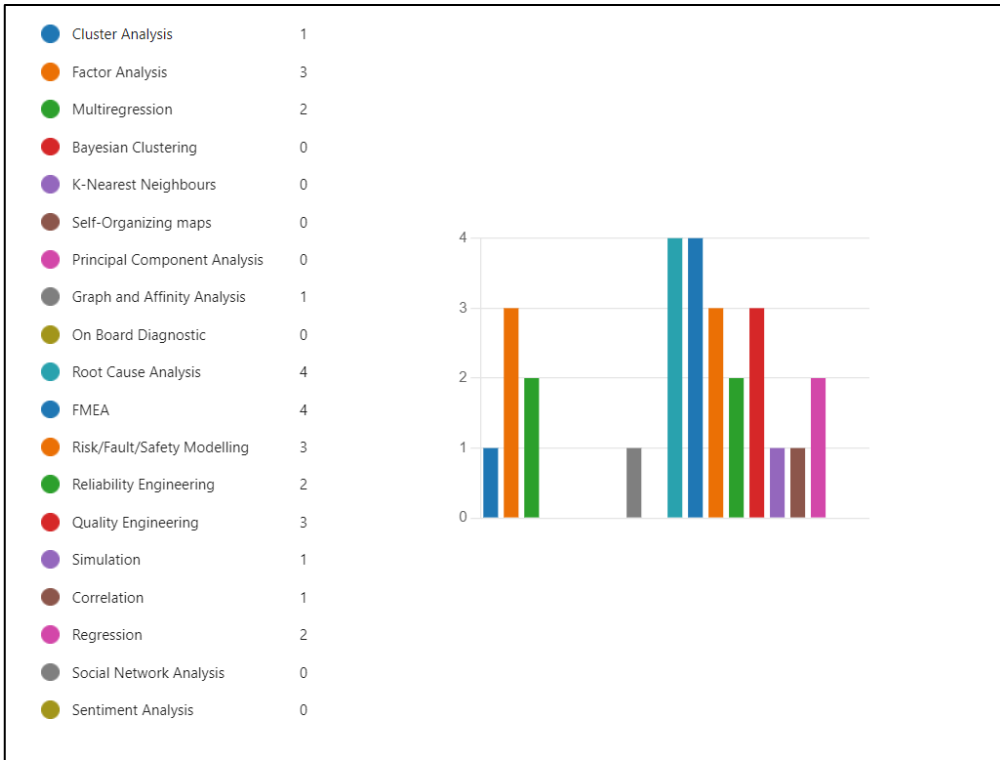
### Use of descriptive data analysis techniques



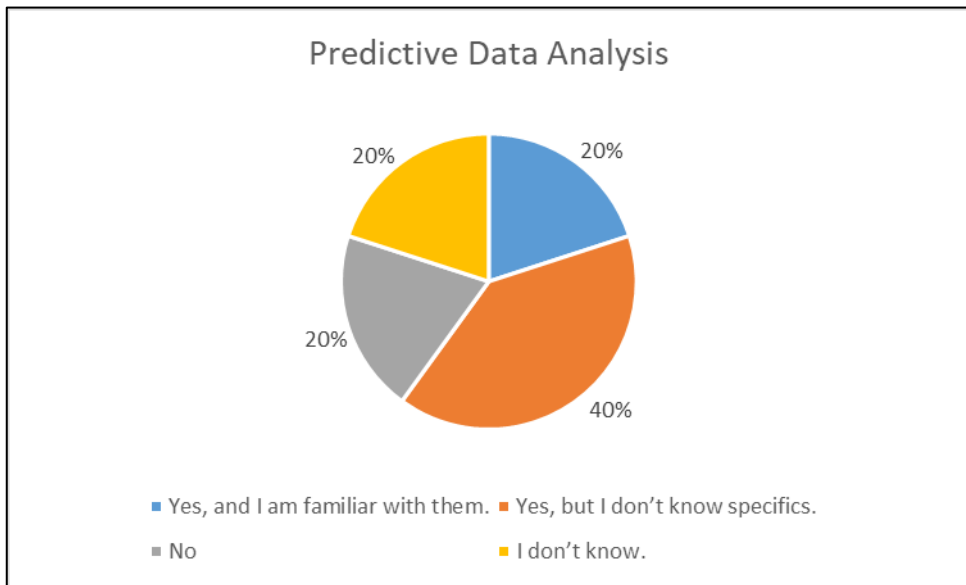
### Descriptive data analysis techniques used at the organisations



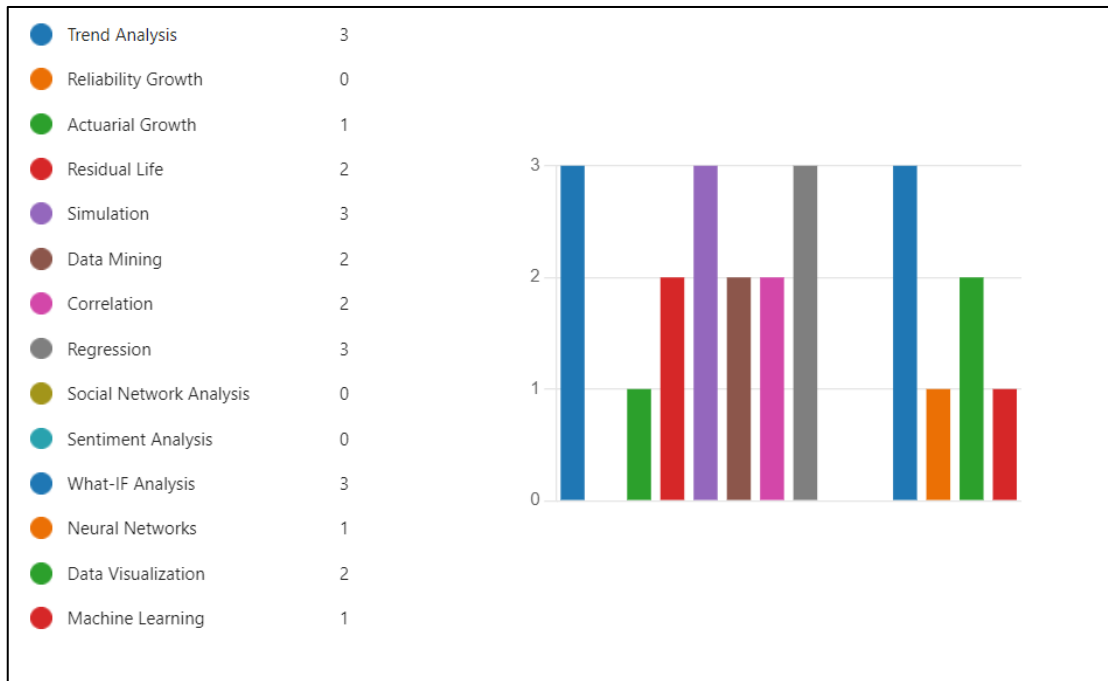
### Use of diagnostic data analysis techniques



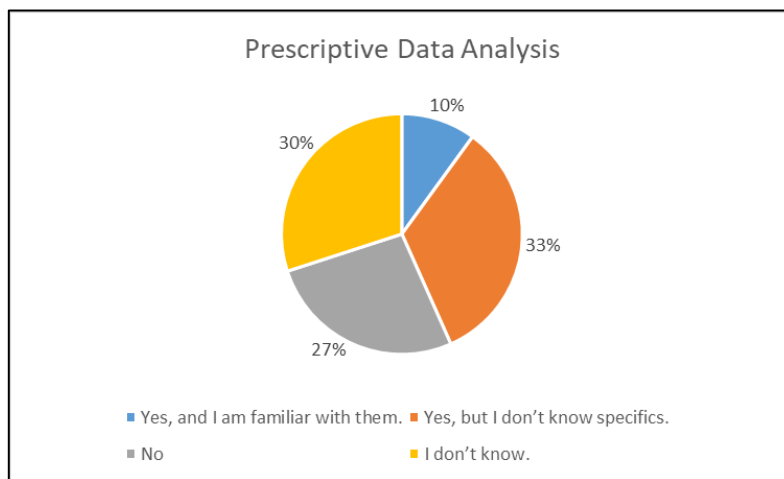
**Descriptive data analysis techniques used at the organisations**



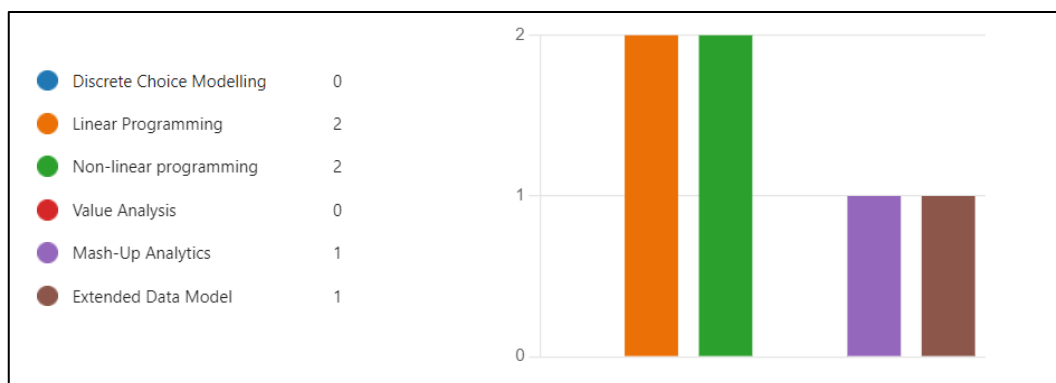
**Use of predictive analysis techniques**



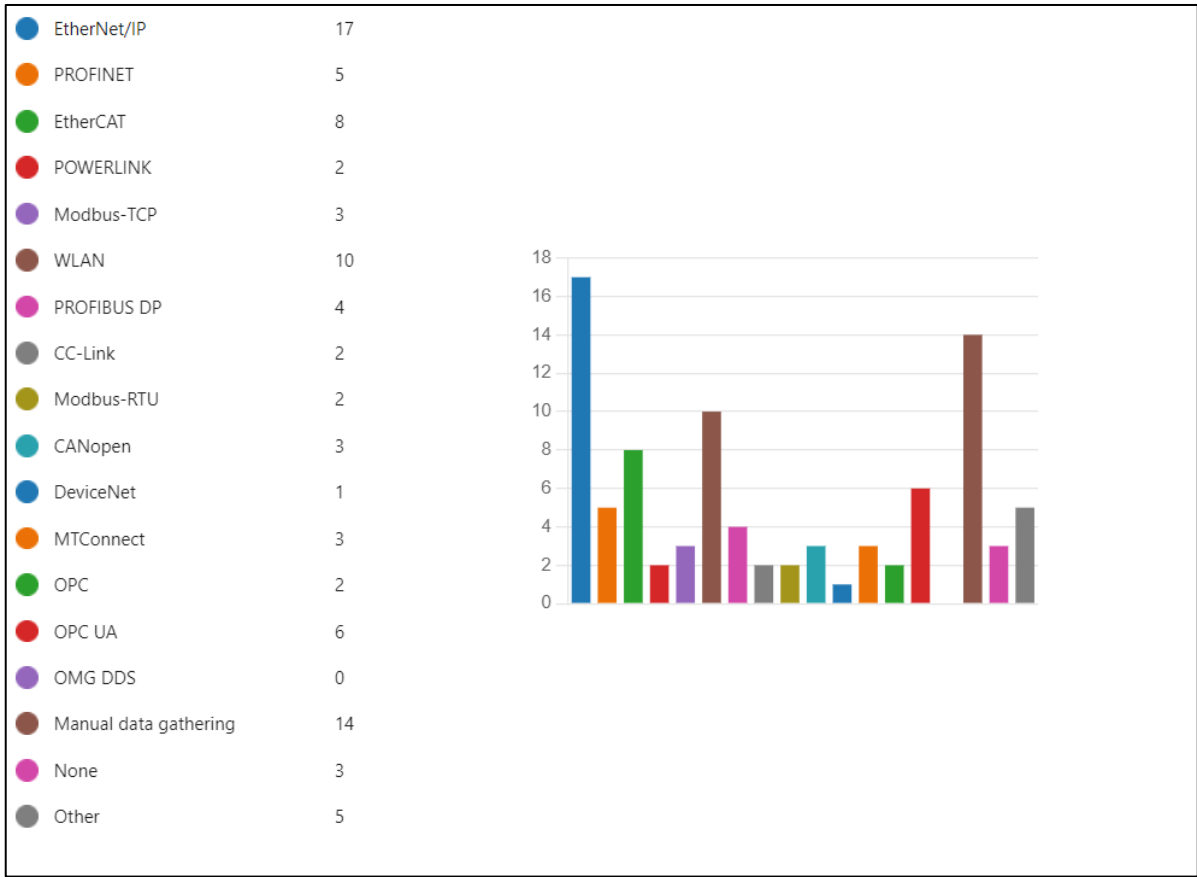
**Predictive data analysis techniques used at the organisations**



**Use of prescriptive analysis techniques**

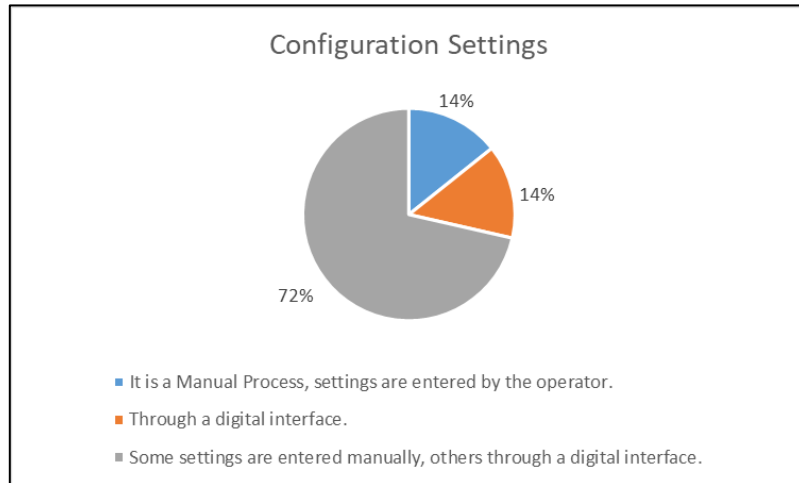


**Prescriptive data analysis techniques used at the organisations**

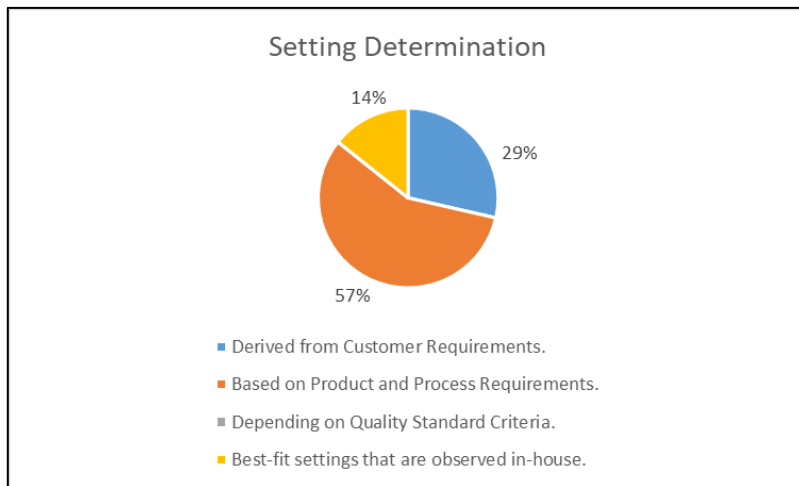


**Industrial network/automation protocols employed in the organisation**

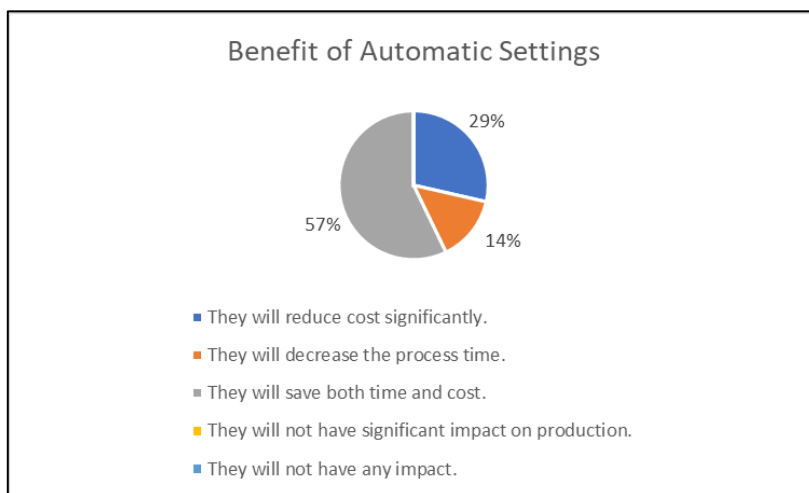
### Survey 3: Self-Configuring Production Systems in Manufacturing



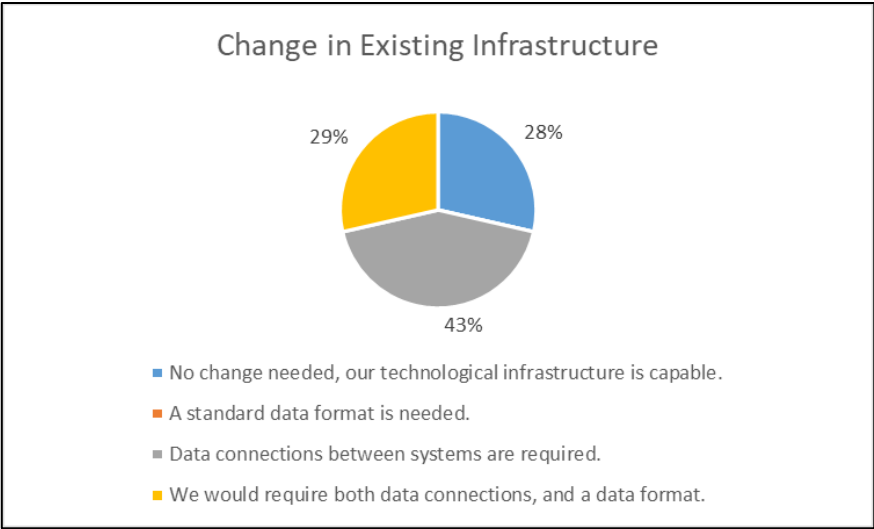
#### **Entering settings in a machine before Operation**



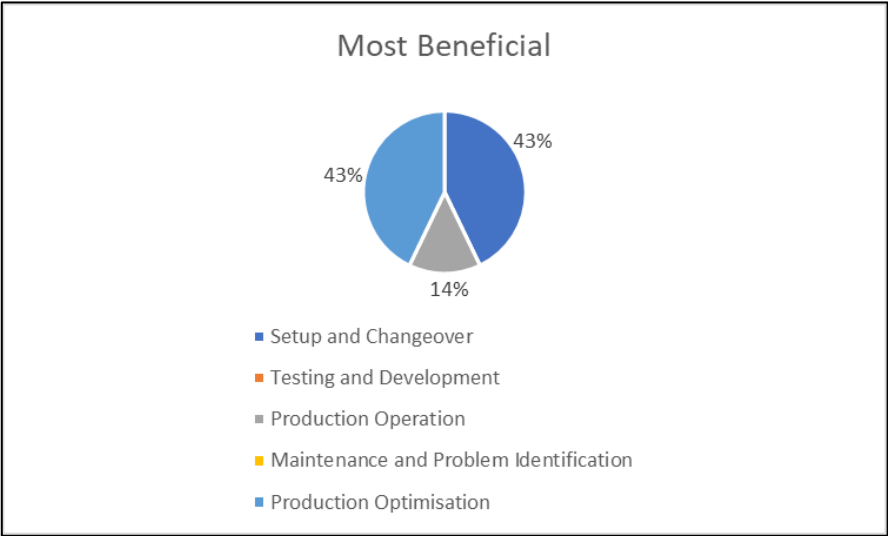
#### **Determining settings in a machine before Operation**



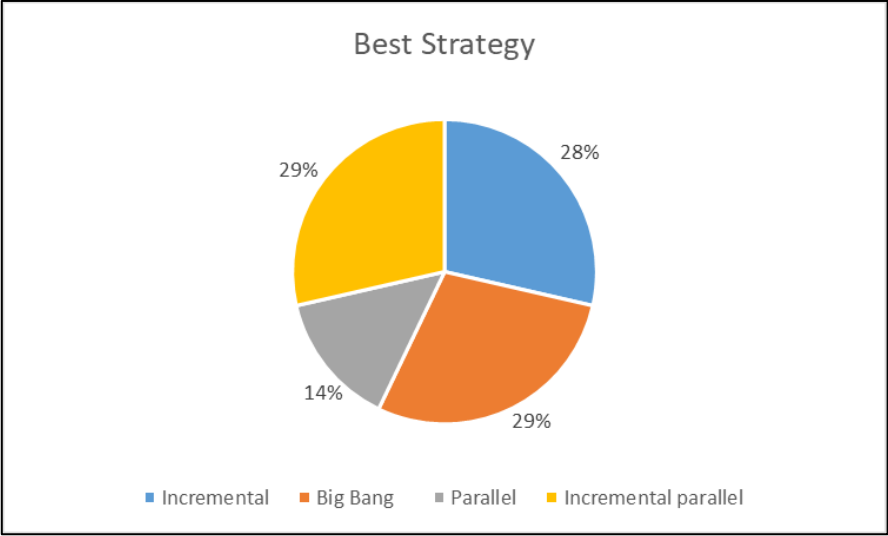
#### **The benefit of systems that automatically adjust their settings in a machine**



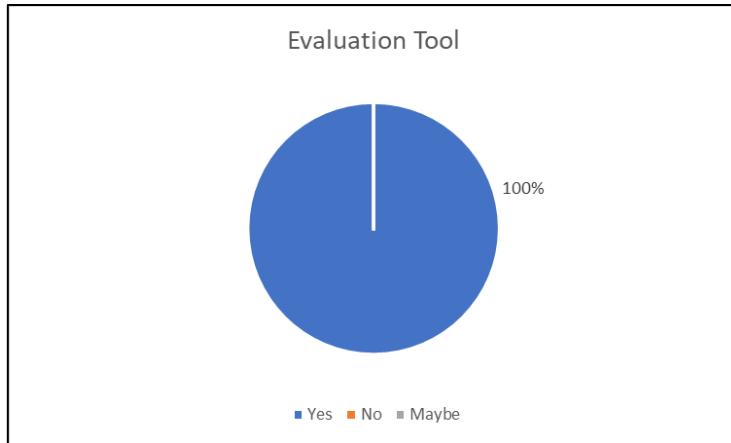
**Change in infrastructure required for self-configuring production systems**



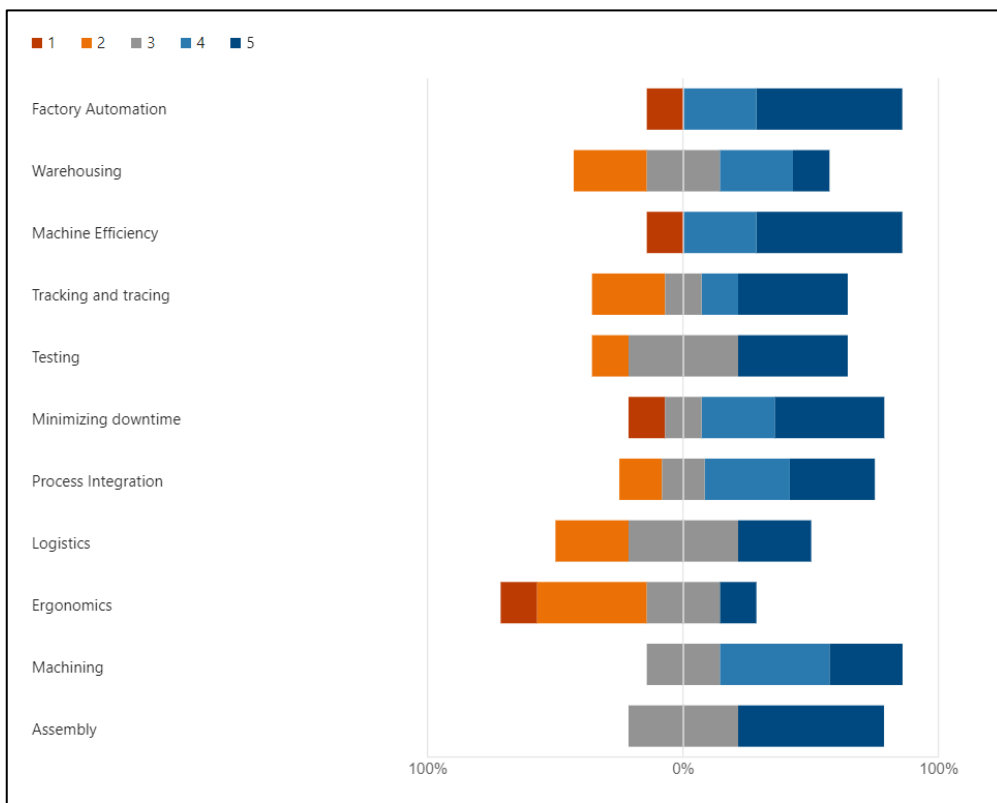
**The most beneficial application of self-configuring in production systems**



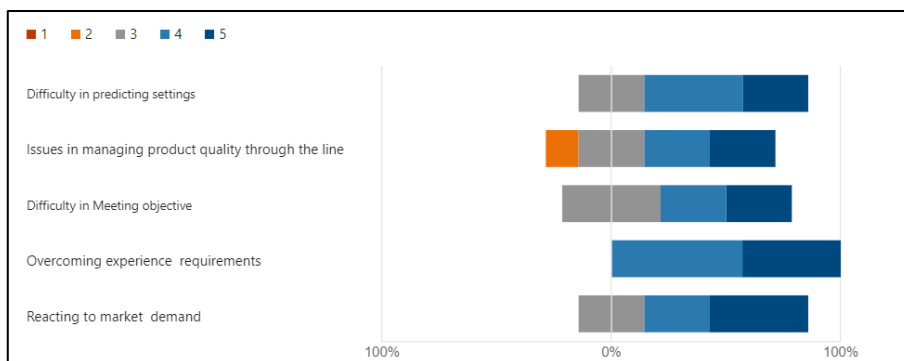
**Best strategy for adoption of self-configuring production system**



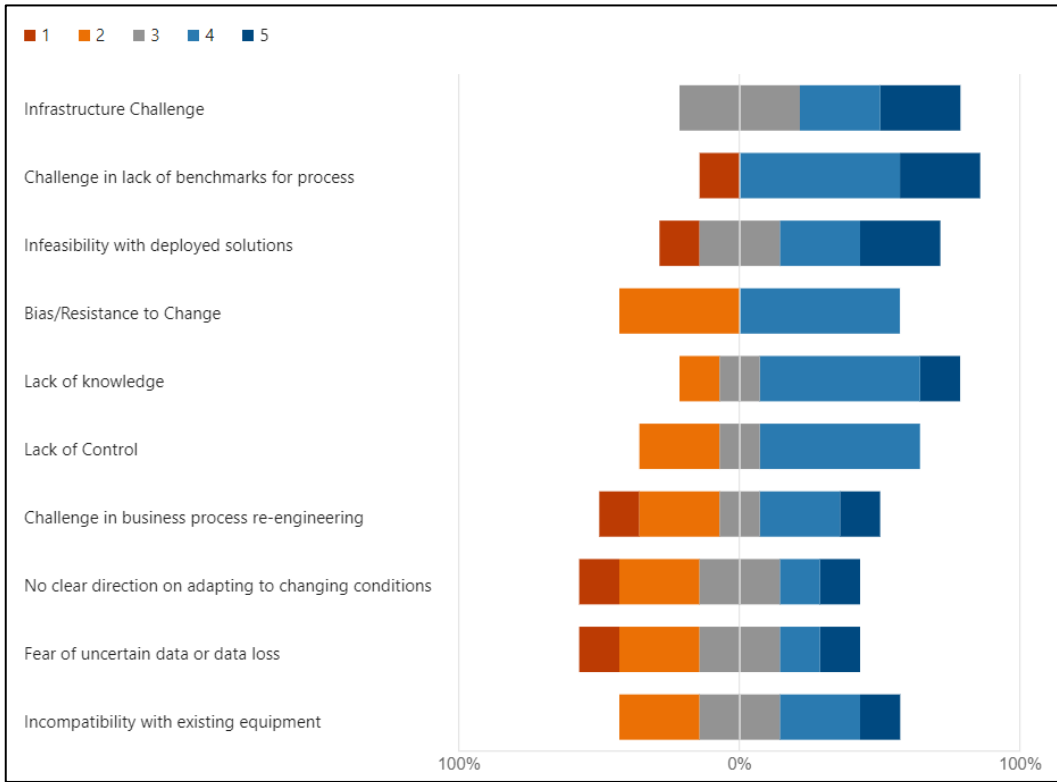
**Tool for evaluating self-adapting capability**



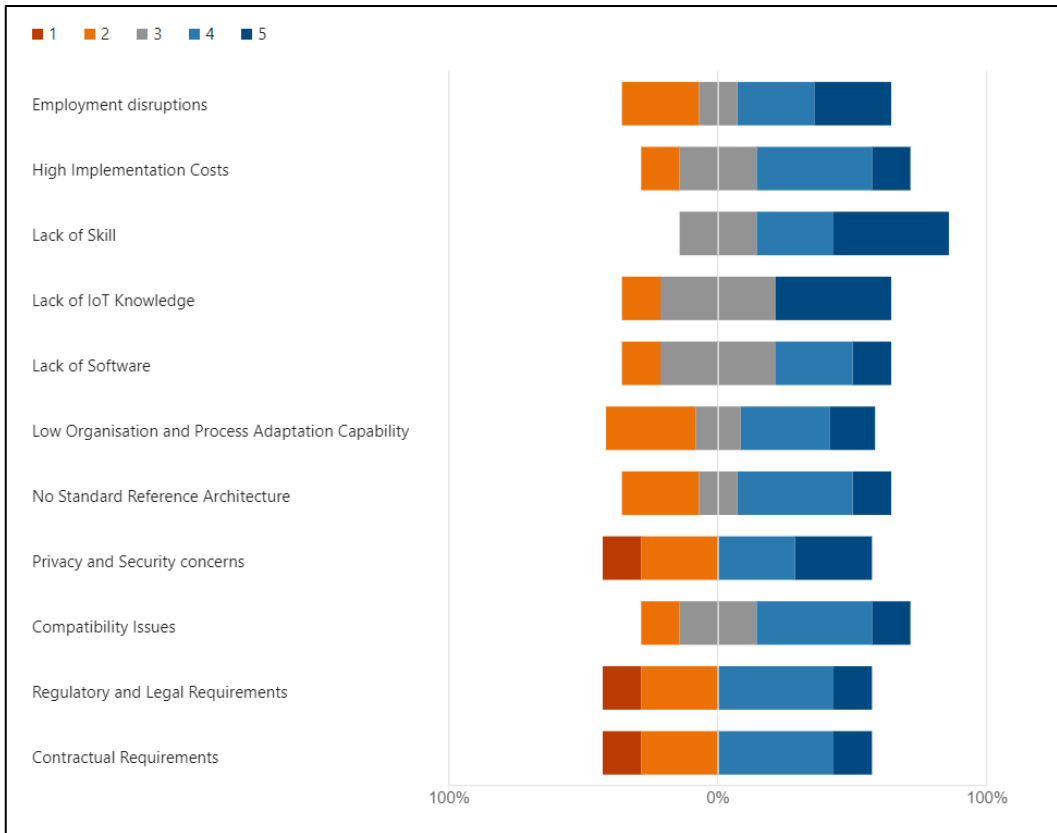
**Most beneficial domain of self-adapting machines**



**Significance of self-configuring machines in solving operational challenges**



**Challenges in using machines that automatically adjust their settings**



**Self-configuring machines addressing manufacturing problems**



# Appendix B

## Code Walkthrough

## B.1 Code Walkthrough

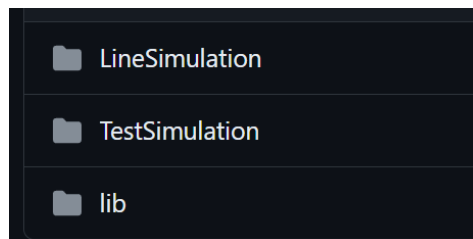
This section details a short tutorial on the implementation of self-configuration approach. The following steps can be undertaken to set up the environment. The code is hosted in GitHub repositories in respective folders.

Github Repo: <https://github.com/Hamood564/CodeWalkthrough.git>

### Agent Deployment

**Folders:** Line Simulation, Test Simulation & Lib.

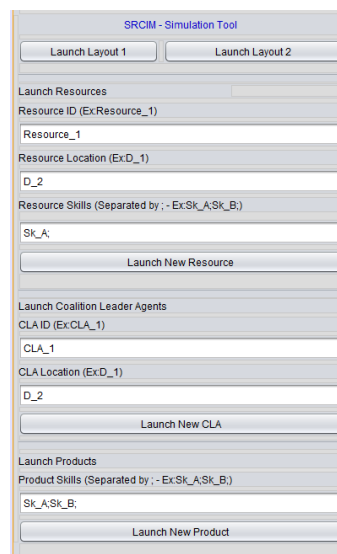
Download these folders. The Line Simulation and Test Simulation are separate NetBeans projects compatible with NetBeans IDE 8.2 and JAVA JDK 8. Lib folder contains additional supporting libraries for these projects.



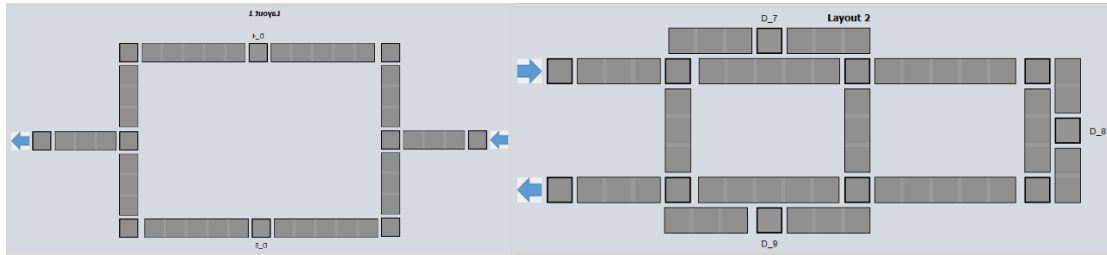
Line Simulation project contains the digital twin model for the functionality execution and the product routing. Test Simulation project contains the agent system for functionality execution. The operation of agent system is detailed in Chapter 8 and Chapter 9 respectively. The inner working of the agent system (architecture, behaviors and protocol) can be understood from the book *Developing Multi-Agent Systems with JADE*.

A brief description of the packages in each project are as follows:

1. **Console Agent:** Contains the console frame and the console agent code that interacts with the frame in the simulation tool. Frame to agent system communication is also present in this package.



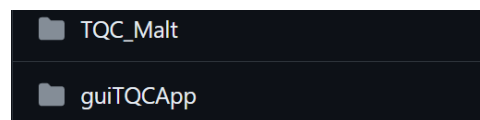
2. **Simulation:** This package contains the two layouts developed for the simulation. The resources can be allocated to resource locations and depending on negotiation criteria product be routed to these resources.



3. **Transport:** This package contains transport agent, the transport rules for the product and the relevant hardware interface abstraction. This package also contains the rules for the interface and the classes to map the state of product across the layouts.
4. **Transport Libs:** This package contains the hardware simulation and layout rules.
5. **Utilities:** Contains the constants and directory facilitator interaction mechanism that can be used by the Line Simulation project.
6. **Coalition Leader Agent:** This package contains information on coalition agent. Not implemented in the current strategy.
7. **Product Agent:** This package contains the classes to implement product agent, its behaviours and relevant communication with the simulation & physical environment.
8. **Resource Agent:** This package contains the classes to implement resource agent, its behaviours and relevant communication with the simulation & physical environment.
9. **Mqtt:** This package contains the implementation of mqtt protocol for publish and subscribe. These classes are used to communicate with the physical hardware controller.
10. **WebSockets:** This package contains the implementation of websockets between agents and hardware.
11. **Util:** Contains the constants and directory facilitator interaction mechanism that can be used by the Test Simulation project.

## State Chart and State Machine Functionality Representation

**Folder:** State Chart to State Machine



The folder contains two projects “TQC\_Malt” and “guiTQCApp”. These projects are compatible with Eclipse IDE and JAVA JDK 11. The description of these projects is as follows:

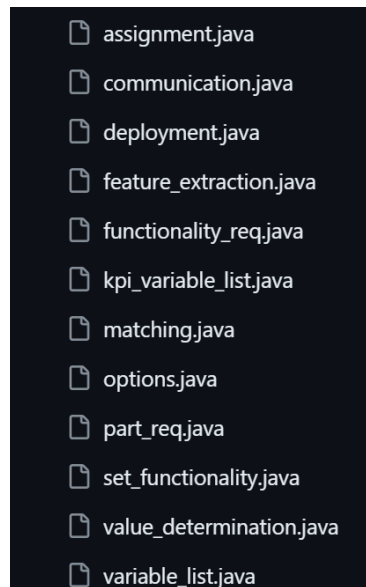
1. **TQC\_Malt:** This project contains the state chart implementation derived from Yakindu state chart modelling environment described in self-configuration approach strategy. The bin folder contains the compiled class files. The src-gen/malt folder contains the implementation of Yakindu state machine. The src folder contains MaltState package that has maltStateMain where the state chart can be executed. In guiTQCApp contains a State Machine package with some example classes on how the state chart can be encapsulated in state machine behaviors.
2. **guiTQCApp :** Contains a basic implementation of tool utility realization through JAVA FX framework. The behaviors can be defined in the State Machine package and leveraged through the app.

More on Yakindu State Chart and its implementation for self-configuration is detailed in relevant thesis chapter.

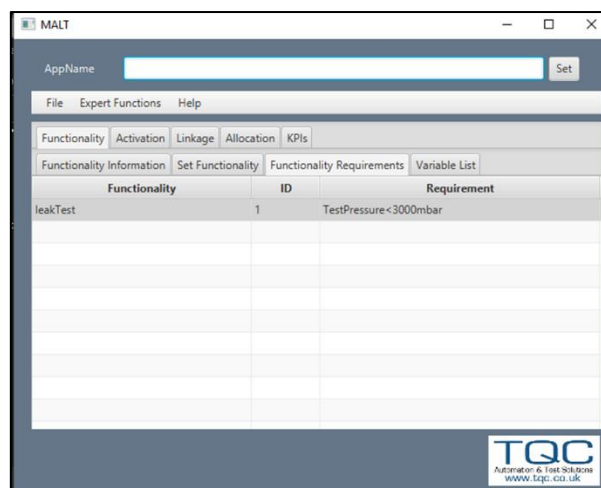
## Tool Utility for Functionality

The tool utility is developed and can be accessed through guiTQCApp project. The TQCMainScene.fxml contains the user interface definition for the app. The TQCMainSceneController class contains the controller classes for the app functions. These functions can be tied to state machines from previous section. The tool utility operates as a coordinator for the functionality execution. An example is present in the code.

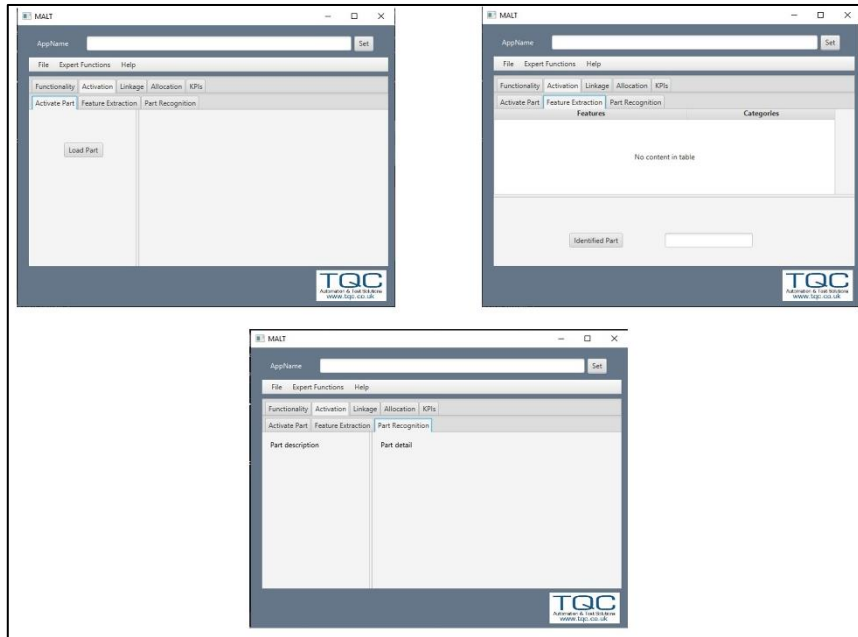
The information capture can be integrated into the guiTQCApp through relevant classes. The information capture is carried out through CAEX engine implementation.



As the information captured is integrated the app can be used to reflect that in real-time, along with coordinating functionality as per state machine behaviors.



The detail on each screen responds to the states in the state machine developed through yakindu environment.



The coordination of functionality execution is detailed in the respective PhD Thesis Chapters.

### Information Capture through CAEX Engine and AAS

**Folders:** CAEX\_Malt, CAEX\_test, CAEX\_testingwithoutinterface and CAEX3

The information capture is carried out through CAEX engine and provides computability to the AAS standard. The detail on mapping between CAEX engine and AAS is present in the respective thesis chapter.

Each folder is a separate JAVA project, with generated classes that can be used to interface with CAEX files. The CAEX files are generated through AAS Explorer, can be referenced through AML editor. The following steps can be followed:

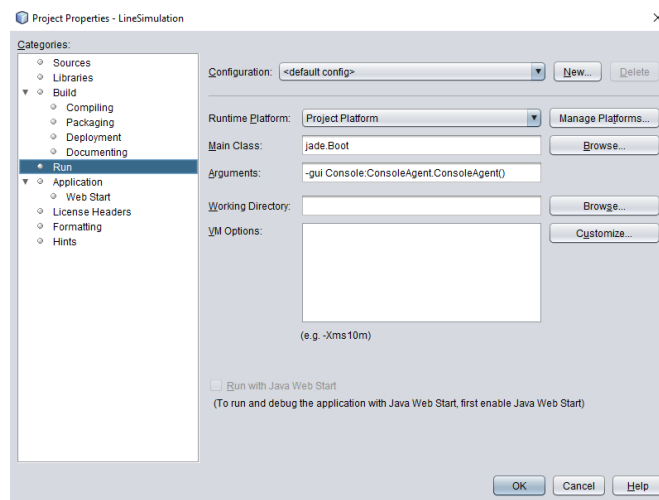
1. Information on modules in a production system can be mapped to the AAS submodels. Then whole AAS of the system can be exported to AML (CAEX) file.
2. The AML editor can be used to access the CAEX file for the system.
3. The folders contain code that can interact with the CAEX files through generated classes. The information can be extracted and updated.
4. The classes can be integrated within state machine behaviors for functionality coordination. Tool utility detailed before can be used together with these state machine behaviors.

A basic implementation example is present in each code project.

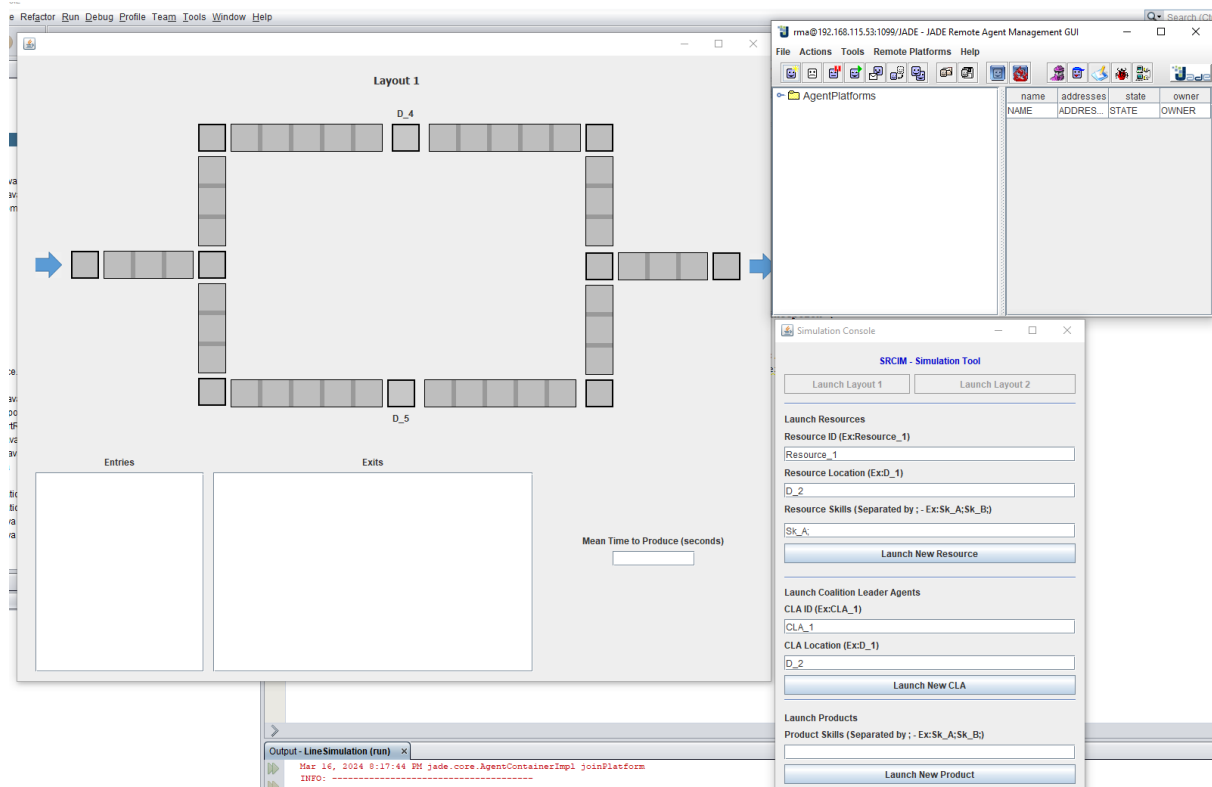
### Agent System Execution

**Folders:** Line Simulation, Test Simulation & Lib.

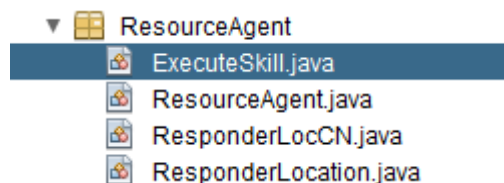
The Agent simulation can be started by run the Line Simulation Project with following configuration:



Once the simulation is started the location of resources can be defined through the console screen:



The functionality execution behavior can be triggered through Resource Agent package in Test Simulation. The functionality is executed through Execute Skill class. The Response from location is handled in Responder Location class.



Integrating business objectives (KPIs) and negotiating functionality execution is carried out through Contract Net Protocol (CNP) in ResponderLocCN class.



## ML Pipelines

**Folders:** pass\_fail\_endpoint, collab\_notebooks, endpointStrategy-main and cloud\_functions.

These folders contain the machine learning code for the thesis project. These code implementations rely (depending on application) on JAVA, Python and Google Cloud Platform. The main folders are as follows:

1. **Pass\_Fail\_Endpoint:** This folder contains the pretrained models for leak test and force test prediction. Should be in GCP Storage bucket of same name.
2. **Collab\_Notebooks:** This folder contains code for machine learning applied to datasets. The pretrained models are generated through these.
3. **Cloud\_Functions:** This folder contains the trigger scripts responsible that query values from pretrained models as they receive a http request. Should be configured in GCP Cloud Functions.
4. **Endpoint\_Strategy:** Contains modular ML endpoint implementation through Google's Vertex AI. The detail is presented in the code and more insight available through Google official documentation.

GCP is used to deploy the ML models in cloud storage as well as the trigger functions. The http calls can be integrated in the state machine behaviours and in agent systems to query predictions on settings. GCP official documentation to be followed for deployment. Detail on each implementation is present in respective folders.

## Real Time Control

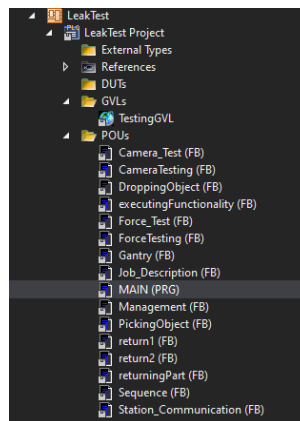
**Folders:** Self-ConfigurationMALT.zip and Testing1.rar

The archived files are the implementation of real time control (RTC) for physical industrial systems. The two industrial systems are Multi-Application Leak Tester (MALT), a bespoke leak tester, developed at TQC and PRIME test station respectively. The RTC is implemented as per following detail:

1. **MALT:** The testingMALT is the main class for the MALT self-configuration RTC. The KPIs can be provided and the configuration be changed through a single call function. Currently it is set to varying volumes. The configuration settings are determined through machine learning model hosted in cloud environment (GCP). This RTC can be integrated with the tool utility for functionality coordination under constraints.

```
String IP = "192.168.115.205";  
Integer Port = 5000;  
selfConfiguration.connectMalt(IP, Port);  
selfConfiguration.setVolume("2");  
selfConfiguration.changeconfiguration( selfConfiguration.Volume, "0");
```

2. **PRIME:** The PRIME test station code is a Beckhoff TwinCAT 3 project that can be loaded into TwinCAT 3 environment. The variables responsible for settings and functionality execution can be accessed through TestingGVL by OPCUA. The main execution happens through Main FB in the POUs. The configuration settings are determined through machine learning model hosted in cloud environment (GCP) actuated through trigger scripts on storage buckets. This RTC can be integrated with the tool utility for functionality coordination under constraints.



*NOTE: For any assistance contact the author of the Thesis at [hamood564@outlook.com](mailto:hamood564@outlook.com).*



# Appendix C

## State Chart

### C.1 Understanding States and Actions in State Charts

Statecharts have vertices and (directed) edges because they are depicted as directed graphs. An object's states are represented by its vertices. Edges are state transitions or changes in the state. A rectangle with rounded corners is used to symbolise states, and they can have names and actions. There are three different types of state actions:

- When the state is triggered, the entry action is carried out.
- After the entry-action is complete, the do-action is carried out.
- When the status is deactivated, the exit action is carried out.

Pseudo-states are a subset of states. They exist solely for modelling purposes and are not actual states of the item. The state chart must not remain or perform actions in these pseudostates. The start-state, which is

a pseudostate with only outbound transitions, is the principal entry triggered when the statechart begins event handling. Instead, the final state is a true state with just incoming transitions.

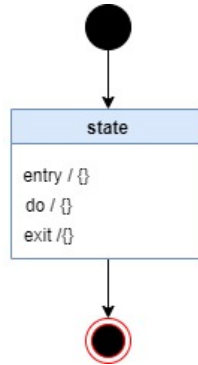


Figure C.1: A basic state representation using statecharts.

### C.1.1 Transitions in a State Chart

#### Events

Normally, changing a state is prompted by events. An event is the outcome of an arbitrary action by the system or its surroundings. There is no particular definition for events. The most essential event kinds are given in the UML specification:

- The surrounding environment causes the signal event to occur.
- The state chart itself initiates the time-event.

A completion-transition, which is a transition without an event, can occur in any state. If no such transition exists in the state (all transitions are triggered by events), the object remains in this state until an appropriate event can be handled by an outgoing transition. The present active state is the only one that can handle all events. If the state is unable to handle

the event, it is discarded. The completion of the state's exit-action is understood as the trigger for the end-transition.

### **Guards and Actions**

A guard and an action can be added to a transition in addition to the event. The first is labelled [guard], while the second is labelled /action. Guards are employed to ensure that the transition occurs only if the guard's evaluation is true. As a result, transitions using the same event and various guards are feasible. Depending on the guard, different actions or goal states can be achieved.

It must be ensured that the model is deterministic when modelling. As a result, adding outgoing transitions to the state that can trigger both on an event is not permitted.

### **Time-triggered transitions**

The transition marks the time event with the keyword after (x). This is a time-triggered transition. "x" represents the amount of time that the source state must be active before the transition may occur (if the optional guard evaluation is true). When a state is triggered, the clock begins to tick.

### **Segmented transitions**

Segmented transitions are another feature of state charts. They are used to concatenate numerous transitions, for example, to execute an action with each sub transition. The modelling of branching is another valuable

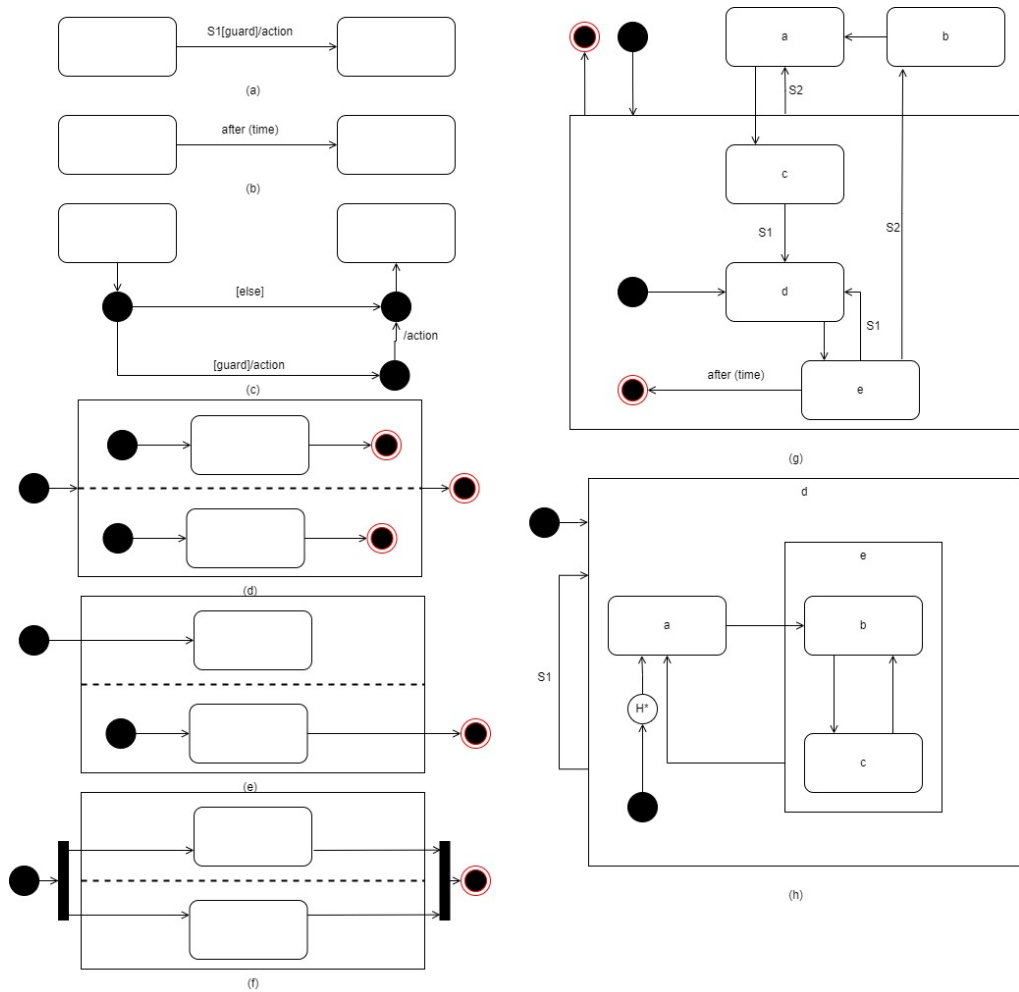


Figure C.2: An collaborative illustration on state charts for a production system. (a) Guards and actions during the transition, (b) time-triggered transition, (c) segmented transition, (d) regions, (e) concurrent transitions, (f) complex transitions, (g) state transition, (h) history states.

application of segmented transitions. All sub transitions are concatenated with the junction point pseudostate, which is depicted in the figure as a little black circle.

The fact that segmented transitions are always atomic is an important characteristic. As a result, the transition triggers either fully or not at all. That is, before a state change happens, the segmented transition must determine if an authorised path to the next actual state exists. However, there is another important distinction to make: only the first part of the transition can contain an event as a trigger. Otherwise, guards and actions

can be found in any part.

### **C.1.2 Hierarchical states**

Complex systems frequently call for the abstraction of a real-world scenario and the later specification of concrete behaviour. Hierarchical states, commonly referred to as or-states, can be utilised in statecharts to break down complicated states into substates that more fully describe the behaviour. One substate is automatically triggered as the object achieves this composite or-state, and only one substate is active at any one moment (which explains the name or-state).

#### **Start- and final-states**

Knowing which substate to activate when the parent state is active and when the or-state is complete, respectively, is crucial within an or-state. These data are simulated using the start- and end-states that are previously known. It is not always necessary for these pseudostates to exist. Why will be explained in the paragraph that follows.

#### **Entering and leaving the hierarchical state**

There are two ways that a complicated state can be activated:

- At the or-state's border an incoming transition comes to a halt. In this situation, the start-state presence is required.
- An inbound transition terminates at a substate and crosses the or-state's boundary. Since the aim of the transition informs the hier-

archical state of the first active substate, the start-state is thus not necessary.

An analogy for leaving a hierarchical state is:

- An incoming (end-)transition can deactivate the state. If there is no trigger for this transition, the completion of the composite state serves as the signal for the end-transition. A final-state is necessary in this situation.
- A transition from one state to another outside of the or-state, with the source being a substate, might deactivate the state. A final-state is not necessary in this situation.

### **State transitions**

Transitions inside hierarchies raise two issues for which a semantics is required. What occurs with incoming event-triggered transitions from the or-state at first? When the or-state is in effect, the object is semantically in precisely one substate. Therefore, it is essential that each substate be able to manage this occurrence as well. Every event-triggered outgoing transition of the hierarchical state is inherited by all substates in order to implement this semantics. To put it another way, every substate manages any egress-triggered transitions of its parent states that are on the route from the state to the hierarchy-root tree's node.

What happens if a substate "overrides" an outgoing transition with the same event, as a result of this semantics? In this instance, many transitions may activate. Because t1 is an outgoing transition of the state s1 and s1 is a transitively accessible substate of s2, a mechanism to prioritise transitions

is required.  $t_2$  manages the same event as  $t_1$  and is an outgoing transition of  $s_2$ . In this instance, inner transition  $t_1$  is given priority over outer transition  $t_2$ . This rule guarantees that the hierarchical tree's lowest transition will always trigger.

### **History states**

Many systems need to know how a complicated state was configured before it was disabled. With this knowledge, reactivating the complicated state may be done while maintaining the same configuration. History-pseudostates in hierarchical states are used to model this data. In the figure, history is denoted by the letters  $H$  or  $H^*$  and is classed as follows:

- The shallow history  $H$  stores the last active substate.
- The deep history  $H^*$  stores all active substates on the path from the node to the leaf in the hierarchy tree.

### **C.1.3 Concurrent states**

Concurrency modelling is possible with statecharts. At least two concurrently active substates are required for a concurrent state. The and-state is divided into these regions. These regions are divided by a dashed line in the figure. Concurrency entails fresh conceptual ramifications, which are now discussed.

## **Regions**

The semantics mentioned above also applies here since regions are specified by hierarchical states. As soon as the and-state is triggered, practically all areas are active. They are a particular kind of processes that are active in the concurrent state.

It is not feasible for some regions to be "deactivated" and not others. All regions always handle incoming events, hence many transitions may be able to fire at once. specifically up to changes in region count. Like in or-states, all regions inherit event-triggered outgoing transitions (and thereby all substates). Here too, the transition firing priority rule is in effect.

## **Entering and leaving the concurrent state**

All regions must be active to enter a concurrent state, and vice versa. Two scenarios can be given, similar to hierarchical states.

- The and-state marks the conclusion of the entering transition. Each area in this scenario has to have a start-state.
- The incoming transition comes to an end in a single region's substate. The and-state is thereby implicitly triggered at this substate. The start-states of all other areas are enabled. Thus, n-1 start-states are required.

## **Complex transitions**

Above, the implicit and explicit examples of activating a concurrent state were discussed. Whereas later n-1 start-states are required, the initial all



n regions require start-states. However, what happens if n-m regions are enabled.

Complex transitions were included to the model to represent this behaviour. The control flow was divided. Maximum one transition from an incoming transition is divided into each area with a target substate. It is possible to give an analogous example for exiting the and-state. These two semantics can appear in a complex transition:

- Divide a single transition at least twice. All areas that are blocked by incoming transitions are automatically enabled at their start states..
- Synchronising the departure of certain regions from the and-state upon activation of particular substates. All regions that are prohibited from outbound transitions are immediately disabled.

# Appendix D

## CAEX Standard

### D.1 Utilising the CAEX Standard

The CAEX (Computer Aided Engineering Exchange) standard has been used for modelling information about production systems in manufacturing (Drath (2012)), allowing illustration of semantics in terms of defined roles and then collecting them in groups known as role class libraries. Interfaces between production system objects can be specified in interface class libraries. System Unit Classes can contain information about system objects collected in system unit libraries. Finally, instance hierarchy can contain information about internal elements that reference instances of system unit objects and derive their semantics from role class objects. Along with this, internal elements reference interface objects providing a mechanism to interlink internal element objects and a means to reference externally stored information.

The model for representing production system-related information utilises the CAEX standard for modelling the system topology and its elements

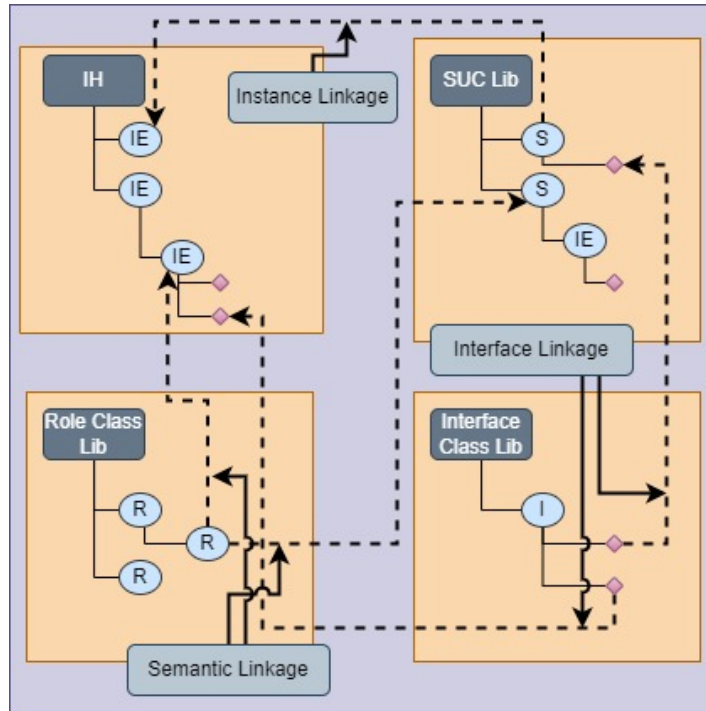


Figure D.1: CAEX architecture for capturing information on the module and operational semantics (Drath (2021)).

(Drath (2021)) (see figure D.1).

1. **Role Classes:** Role (R) classes, which are collected in form of role class libraries, describe the abstract functionality of the system unit object without describing the fundamental implementation, therefore acting as a representation of semantics. Role Classes are dependent on the need for capturing data and the respective data interchange required. Role classes follow a unique name identification within the role class library hierarchy, making it easy to be referenced within a hierarchy path. These role classes may possess attributes and interfaces, and assist in the interpretation and processing of information.
2. **Interface classes:** Interface classes implement abstract relationships between elements and to other information not contained within CAEX based model. An interface class will have a unique identification along with attributes. These attributes can take up values in

each occurrence of an instance (I) of the interface class.

3. **System Unit Classes (SUC):** System unit classes are representative reusable system components or templates. These reflect a library of components or a set of templates representing aspects of the production system that be instantiated as an instance. Each of the system unit class objects (S) may contain sub-objects known as internal elements (IE), along with the respective attributes and interfaces. These system unit class objects may also be derived from other system unit classes through reference path attributes.
4. **Instance hierarchy(IH):** Instance hierarchy consists of an integrated hierarchy of internal elements that themselves consist of instances of system unit class objects. Internal elements (IE) are representations of objects in a production system that can be physical or digital elements. Internal elements contain instances of interfaces derived from the interface class library and possibly refer to one or more than one role class library. The attribute and referenced role class define the semantics of the internal element.

Within the CAEX standard, modules, which contain critical functionality information, can be effectively represented. To capture the operational semantics embedded within a module, following approach is followed:

1. **Instantiating a System Unit Class Template:** A system unit class template is created that contains essential information about the functionality of configurable objects. This template acts as a blueprint for defining the module's functionality.
2. **Creating an Instance in the Instance Hierarchy:** Within the instance hierarchy, an instance of the module is created. This instance

holds detailed information about the functionality and can include specific values instantiated for that instance.

3. **Instantiating Operations as Internal Elements:** The individual operations within a module are instantiated as internal elements within the instance hierarchy. These internal elements represent the specific operations that the module can perform.
4. **Defining the Functionality as a Role Class:** The functionality information can be based on semantics based on the role class definition for functionality. The functionality information can be defined for a process as per the following description;

- **Identifier:** This distinguishes the functionality from other functionality capabilities. It carries a unique identification.
- **Kind:** This represents an instance of the functionality with specific input and output variables. It can also be a representation of a template present within the system unit class.
- **Semantic ID:** This presents a link to semantic reference. This is necessary to automatically identify and understand the meaning of the functionality if it is linked to a Concept Description.
- **Qualifiable:** It presents the constraints the functionality may be subjected to.
- **Data Specification:** They reference from a template that can be used to define attributes for a functionality. They refer to prescribed standard data model semantics.

The functionality is actuated through the Event interface represented by identifier, kind, semantic ID, qualifiable and data specification.

The best representation of functionality information can be understood by figure D.2.

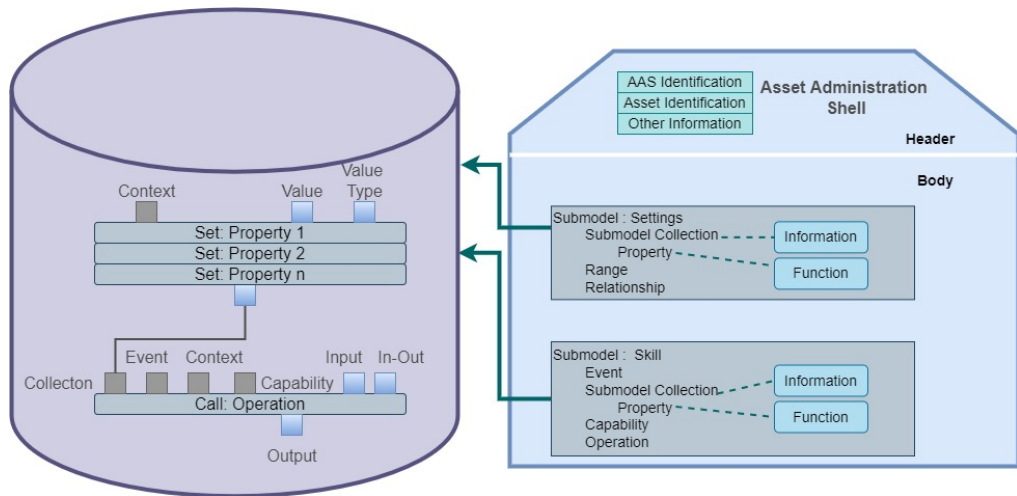


Figure D.2: Representation of functionality operation as a Submodel in Asset Administration Shell deployed through CAEX engine. *Context* is a combination of identifier, kind, semantic ID, qualifiable, and data specification for a production operation. *Event* defines the change happening as a result of the operation, *Submodel Collection* contains information and functions about the operations relevant for that production system, *Capability* is the ability or extent of the operation, and *Operation* contains information about the input, output and in-out condition for the operation. *Value* and *Value Type* are used to assign value as per a unit to the property.