

This item was submitted to Loughborough's Institutional Repository (<https://dspace.lboro.ac.uk/>) by the author and is made available under the following Creative Commons Licence conditions.



CC creative commons
COMMONS DEED


Attribution-NonCommercial-NoDerivs 2.5


You are free:

- to copy, distribute, display, and perform the work

Under the following conditions:

 **Attribution.** You must attribute the work in the manner specified by the author or licensor.


 **Noncommercial.** You may not use this work for commercial purposes.

 **No Derivative Works.** You may not alter, transform, or build upon this work.

- For any reuse or distribution, you must make clear to others the license terms of this work.
- Any of these conditions can be waived if you get permission from the copyright holder.

Your fair use and other rights are in no way affected by the above.

This is a human-readable summary of the [Legal Code \(the full license\)](#).

[Disclaimer](#) 

For the full text of this licence, please go to:
<http://creativecommons.org/licenses/by-nc-nd/2.5/>

OPERATOR INTERFACES FOR THE LIFECYCLE SUPPORT OF COMPONENT BASED AUTOMATION SYSTEMS

A Doctoral Thesis Submitted in Partial Fulfilment of the Requirements for the
Award of Doctor of Philosophy of Loughborough University

By

Vishal .A. Barot

1st June, 2012

Wolfson School of Mechanical and Manufacturing Engineering

Loughborough University

United Kingdom

© Vishal Barot 2012

Dedication

*To my encouraging parents Mr and Mrs A. Barot, my wife Tanvi and my son
Kian*

Acknowledgements

During the course of my research at the Wolfson School of Mechanical and Manufacturing Engineering, I have learnt many lessons and acquired many skills. The most invaluable asset is the knowledge and experience I have obtained working within the MSI group under direct research supervision of Professor Robert Harrison. I would like to thank my supervisor for his kind support and advice throughout my research.

When carrying out a research study, one is bound to face many challenges and this requires support from various sources to achieve the end result. I would like to sincerely thank my colleagues at the MSI group for their technical expertise, guidance and professionalism.

For my financial support and case studies, I would like to thank EU FP7 SOCRADES, EPSRC, IMCRC and BDA projects, and their collaborators, especially the Ford Motor Company.

I feel indebted to my father Mr. Arun Barot and my mother Mrs. Rekha Barot for their unconditional love and patience.

Finally, I deeply wish to thank Tanvi, my wife and my best friend, for the psychological support, tenderness and care she steadily provided me with throughout the years. It is for her that I try everyday to become not only a better engineer / scientist but a better person.

Abstract

Current manufacturing automation systems (specifically the powertrain sector) have been facing challenges with constant pressures of globalisation, environmental concerns and ICT (Information and Communication Technology) innovations. These challenges instigate new demands for shorter product lifecycles and require customised products to be manufactured as efficiently as possible. Manufacturing systems must therefore be agile to remain competitive by supporting frequent reconfigurations involving distributed engineering activities.

The most agile components within any industrial system are the human personnel involved in controlling and monitoring production machines using operator interface systems. Current operator interface systems are proprietary and machine-dependent i.e. they offer poor connectivity to other manufacturing systems, and locks end users to costly and difficult to modify support solutions. Furthermore, they do not provide support at various key phases of the machine lifecycle to closely monitor and control its activities. These factors unnecessarily increase machine lifecycle costs and complicate its support process.

The emphasis of this research is a novel operator interface system implemented within a system components architecture that can better support lifecycle usage requirements of powertrain manufacturing machines engineered using the Component Based automation approach, and enable different classes of users throughout the supply-chain to more efficiently interact with these systems. Specifically, the research focuses on automatic generation of vendor-independent web-based operator interface systems with integrated diagnostics and remote support functionalities developed using open standard technologies.

The research of this innovative approach to operator interface system design, development and implementation has been prototyped and evaluated using case studies based on the Ford-Festo test rig (programmed using the web services-based FTB control devices), Oil Pan Rundown engine assembly machine simulation and theoretical plant layout to system architecture mapping, identifying strengths and weaknesses for its industrial application.

Keywords: Operator Interface System, Component Based Automation, Machine Lifecycle, System Architecture, Agile Manufacturing, Manufacturing Automation Systems.

Table of Contents

Dedication.....	i
Acknowledgements.....	ii
Abstract.....	iii
Table of Contents.....	v
List of Figures.....	xi
List of Tables.....	xv
Chapter 1 : Introduction	1
1.1 Research Inspiration	1
1.2 Problem Description.....	3
1.3 Research Questions.....	5
1.4 Research Formation.....	6
Chapter 2 : Manufacturing Systems Review	8
2.1 General Overview	8
2.1.1 Preliminary Explanation.....	8
2.1.2 Paradigm Shift.....	9
2.1.3 Factors Driving the Change.....	11
2.2 Part A: Manufacturing Trends	16
2.2.1 Agile Manufacturing.....	16
2.2.2 RMS (Reconfigurable Manufacturing System)	19
2.3 Part B: Existing Manufacturing State.....	23
2.3.1 Hierarchical Levels of Operations in Manufacturing Facility	23
2.3.2 Operator Interface – PLC System Architecture	25

2.3.3	Operator Interface System Scope within Machine Life Cycle	29
2.3.4	Operator Interface Supporting Machine Maintenance	37
2.3.5	Operator Interface Roles in Control and Monitoring Machines	42
2.4	Manufacturing Review Analysis	45
Chapter 3 : Research Context and Focus		49
3.1	External and Internal Automation Research.....	49
3.1.1	Research Centre at the University of Michigan	49
3.1.2	Rockwell Automation	51
3.1.3	ITEA SIRENA and SOCRADES	51
3.1.4	Other Miscellaneous Research in the Academia	53
3.1.5	MSI Research Institute	54
3.2	Component Based (CB) Automation.....	55
3.2.1	General Description.....	55
3.2.2	CB Application.....	56
3.3	Focused Attributes	60
3.4	Research Aim and Novel Contributions	63
3.4.1	Aim	63
3.4.2	Novel Contributions	64
Chapter 4 : Enabling Technologies and Methods.....		65
4.1	General Overview	65
4.2	User Interface Modelling and Engineering	65
4.3	User Interface Implementation	71
4.4	Real-Time Machine Data Transmission Options and Issues.....	75
4.5	Communication Mechanisms	80
4.5.1	Communication Queues	80
4.5.2	System Interaction Styles	84

4.6	Architectural Patterns.....	85
4.6.1	Model View Controller (MVC)	86
4.6.2	Layered Architecture	87
4.6.3	Repository	88
4.6.4	Client – server Model.....	89
4.7	Technological Review Analysis.....	90
Chapter 5 : Architectural Design		92
5.1	General Overview	92
5.2	Requirements Design.....	92
5.2.1	Design Guidelines	92
5.2.2	Design Capture.....	96
5.3	System Components Architecture.....	104
5.3.1	Architectural Evolution.....	104
5.3.2	Architecture Justification.....	109
Chapter 6 : System Components Detailed Design		112
6.1	Blackboard-based Methodology.....	112
6.1.1	General Description.....	112
6.1.2	Blackboard	116
6.1.3	Knowledge Source	118
6.1.4	Controller.....	119
6.2	System Components Design.....	120
6.2.1	Overall Design Description	120
6.2.2	Broadcaster Blackboard Model	123
6.2.3	Marshaller Blackboard Model	128
6.2.4	Web-HMI Blackboard Model.....	133
6.2.5	BB Component – Knowledge Source Structure.....	138

6.2.6	BB Component – Controller Operation.....	140
Chapter 7 : System Components Implementation.....		145
7.1	General Overview	145
7.2	Overall System Runtime Operation.....	145
7.3	Broadcaster Implementation	147
7.3.1	I / O Outline	147
7.3.2	Process Runtime Implementation.....	148
7.3.3	Interface Description.....	150
7.3.4	Reconfigurable Memory Buffer.....	154
7.3.5	Graphical User Interface View.....	155
7.4	Marshaller Implementation.....	157
7.4.1	I / O Outline	157
7.4.2	Process Runtime Implementation.....	158
7.4.3	Top-level Database Schema	159
7.4.4	Machine Control Sharing Mechanism.....	161
7.4.5	Graphical User Interface View.....	166
7.5	Web-HMI implementation.....	167
7.5.1	I / O Outline	167
7.5.2	Process Runtime Implementation.....	168
7.5.3	Operator Interface Template to Configuration Mapping.....	171
Chapter 8 : Industrial Case Studies.....		175
8.1	General Overview	175
8.1.1	Operator Interface Context within CB Machine Lifecycle.....	175
8.2	Stage 1 Case Study: Ford-Festo Test Rig	177
8.2.1	Test Rig Description	178
8.2.2	Case Study Setup.....	180

8.2.3	Research Attributes Assessment.....	182
8.2.4	Reconfigurability and Reuse Support	182
8.2.5	Information Transparency and Mobility.....	186
8.2.6	Loose Mapping of HMI to Actual Machine or its Control Logic ...	188
8.2.7	Real-time Remote Control, Monitoring and Maintenance	192
8.3	Stage 2 Case Study: Oil Pan Rundown Machine Simulation	196
8.3.1	Oil Pan Rundown Description.....	197
8.3.2	Case Study Setup.....	201
8.3.3	Research Attributes Assessment.....	202
8.3.4	Virtual Machine Validation	202
8.3.5	Early HMI Verification	204
8.3.6	Early HMI Training.....	207
8.4	Stage 3 Theoretical Case Study: Fox Assembly Plant Layout	208
8.4.1	Production Plant Architecture Description	208
8.4.2	System Components to Plant Architecture Mapping	211
8.4.3	Considerable Issues	213
Chapter 9 : System Components Evaluation		215
9.1	General Overview	215
9.2	Safety.....	215
9.2.1	Configuration Whitelisting and Token Sharing.....	216
9.2.2	Command Execution Confirmation and Handshaking	218
9.3	Security	219
9.3.1	Authentication and Authorisation.....	220
9.3.2	Data Encryption	223
9.4	Reliability.....	226
9.5	Robustness	227
9.5.1	Application Heartbeat	228

9.5.2	Data Filtering	230
9.6	Performance	231
9.6.1	System Throughput	231
9.6.2	Response Times.....	233
9.7	Scalability.....	235
9.7.1	Proposed Plant Architecture Mapping Strategy Evaluation	237
Chapter 10 : Discussion, Conclusion and Future Work.....		238
10.1	Research Discussion and Conclusion	238
10.1.1	Contribution to Knowledge	238
10.1.2	Fulfilling the Industrial Requirements	239
10.1.3	Benefits to the Powertrain Manufacturing Lifecycle.....	241
10.2	Future Work Recommendations	242
10.2.1	Remote Data Transmission.....	242
10.2.2	Operator Interface Functionality.....	244
10.2.3	Application to a PLC-based Control System	245
10.2.4	General Work Areas.....	245
10.2.5	Application to Other Industries	246
References.....		248
Appendices.....		268

List of Figures

Figure 1-1: Research Plan Stages	7
Figure 2-1: Globalisation Demanding Faster Design, Build and Ramp Up.....	13
Figure 2-2: Key Agile Enablers.....	17
Figure 2-3: Hierarchical Levels of Operations in Manufacturing Facility.....	25
Figure 2-4: Generalisation of Existing HMI - Control System Architecture	26
Figure 2-5: Machine Lifecycle Process with HMI System Usage Requirements	32
Figure 2-6: Comparison of Existing and Required Approach to Control Logic - HMI Integration.....	36
Figure 2-7: Existing Machine Maintenance Procedure.....	38
Figure 2-8: Required Machine Maintenance Procedure.....	41
Figure 2-9: User Roles Providing Control and Monitoring Support in Machine Lifecycle	44
Figure 3-1: RMS Implementation at University of Michigan	50
Figure 3-2: SOCRADES Approach	52
Figure 3-3: Component Based Automation Approach.....	58
Figure 3-4: Author's Involvement in CB Automation Implementation	59
Figure 3-5: Focused Attributes Addressed within this Research.....	63
Figure 4-1: Iterative User Interface Design Approach	71
Figure 4-2: User Interface Implementation Tools	72
Figure 4-3: RemoteIMS Tool Implementation at Ford Motor Company, U.K....	78
Figure 4-4: Point-to-Point Communication Queue System.....	81
Figure 4-5: Publish / Subscribe Communication Queue System.....	82
Figure 4-6: Message Bus Communication Queue System.....	83
Figure 4-7: MVC Pattern	87
Figure 4-8: OSI Layer Architecture.....	88
Figure 4-9: Client – server Model.....	90
Figure 5-1: Implemented Transline HMI Screen Example.....	94
Figure 5-2: HMI Screen Structure Division - Transline Standard.....	94
Figure 5-3: Machine Component Screen Convention - Transline Standard.....	96
Figure 5-4: Use Cases Supporting HMI User Roles.....	100

Figure 5-5: Example Storyboard Capturing Fault Details Navigational Structure	103
Figure 5-6: Evolution of Control and Monitoring System Architecture in this Research.....	105
Figure 5-7: Proposed Control and Monitoring System Architecture	107
Figure 6-1: Overall Blackboard Systems Model	114
Figure 6-2: Indirect Communication and Ease of Reconfigurability in Blackboard Methodology.....	116
Figure 6-3: Knowledge Source Basic Configuration.....	119
Figure 6-4: Overall System Components Design Model	122
Figure 6-5: Broadcaster Model.....	126
Figure 6-6: Marshaller Model	132
Figure 6-7: Web-HMI Model.....	136
Figure 6-8: Knowledge Source Structure	139
Figure 6-9: Controller Division.....	141
Figure 6-10: Runtime Catalogue Logic Element's Record.....	142
Figure 6-11: Runtime Scanner Logic Element's Record	143
Figure 6-12: Blackboard Processing Example	144
Figure 7-1: System Components Runtime Operation.....	146
Figure 7-2: Broadcaster I / O Overview.....	148
Figure 7-3: Broadcaster Process Runtime Implementation.....	149
Figure 7-4: Broadcaster KS1 Interface Representation.....	151
Figure 7-5: Broadcaster KS1 "SystemLoader" Class	151
Figure 7-6: Broadcaster Control Interfaces	153
Figure 7-7: Reconfigurable Circular Queue.....	155
Figure 7-8: Broadcaster System Component Graphical User Interface	156
Figure 7-9: Marshaller I / O Overview	157
Figure 7-10: Marshaller Process Runtime Implementation	159
Figure 7-11: Top-level Database Schema.....	160
Figure 7-12: Dual-level Control Safety Mechanism	162
Figure 7-13: Machine Control Sharing Mechanism	165
Figure 7-14: Marshaller System Component Graphical User Interface	166
Figure 7-15: Web-HMI I / O Overview	168
Figure 7-16: Web-HMI Process Runtime Implementation	170

Figure 7-17: Web-HMI Template Representation and Configuration Population Process	172
Figure 7-18: Operator Interface Template to Configuration Mapping at Runtime	174
Figure 8-1: Ford-Festo Test Rig	178
Figure 8-2: Ford-Festo Test Rig Illustrating Major Components.....	179
Figure 8-3: Ford-Festo Test Rig Case Study Setup	181
Figure 8-4: Reconfigurability and Reuse Scenario Case Study	185
Figure 8-5: Schematic Setup for SAP xMII Demonstration	187
Figure 8-6: SAP xMII Tool Integration in SOCRADES	188
Figure 8-7: Machine Independence Scenario Case Study	191
Figure 8-8: Real-time Control and Monitoring Scenario Case Study.....	193
Figure 8-9: Remote Maintenance Scenario Case Study	195
Figure 8-10: Example of Engine Assembly Plant	197
Figure 8-11: Oil Pan Rundown Machine	198
Figure 8-12: Op 1900 Process Time Chart	200
Figure 8-13: Op 1900 Machine Case Study Setup.....	201
Figure 8-14: Virtual Machine Validation Case Study Demonstration.....	204
Figure 8-15: Machine Ramp up Activities Within this Research.....	206
Figure 8-16: Early HMI Verification Case Study	207
Figure 8-17: Fox Production Program Plant Architecture.....	209
Figure 8-18: System Components to Auto Station Mapping	212
Figure 8-19: System Components to Manual Station Mapping	213
Figure 9-1: Web-HMI IP Whitelisting Strategy Implementation	217
Figure 9-2: Command Execution Confirmation and Handshaking Strategy Evaluation	219
Figure 9-3: ASP.NET Web Configuration Tool for Web-HMI System Component	221
Figure 9-4: Web-HMI Authentication Strategy Evaluation.....	223
Figure 9-5: HTTPS Data Encryption Session Setup.....	224
Figure 9-6: Web-HMI Data Encryption Evaluation Using Wireshark Protocol Analyser	225
Figure 9-7: System Components Heartbeat Strategy Evaluation	229
Figure 9-8: Broadcaster's Data Filtering Strategy Evaluation.....	230

Figure 9-9: Broadcaster's System Throughput Strategy Evaluation.....	232
Figure 9-10: Response Time Strategy Evaluation.....	233
Figure 9-11: Broadcaster's Scalability Evaluation Process	235
Figure 10-1: Potential Research Directions.....	243
Figure 10-2: RemoteComm Conceptual Illustration	244
Figure 10-3: Research Application to Other Industries.....	247

List of Tables

Table 2.1: Key RMS Characteristics	22
Table 2.2: Manufacturing - HMI Requirements Summary	48
Table 4.1: User Interface Modelling Components	70
Table 4.2: Attributes and Shortcomings of Model Based User Interface Development	73
Table 4.3: Examples of Common Interaction Styles.....	85
Table 5.1: Operational Requirements	98
Table 5.2: Scenario-based Design Capture	102
Table 5.3: Summary Justifying the Proposed System Architecture Benefits..	111
Table 8.1: Reconfiguration and Reuse Scenario.....	184
Table 8.2: Virtual Machine Validation Demonstration.....	203
Table 8.3: Considerable Issues when Implementing System Architecture at Ford's Plant.....	214
Table 9.1: Whitelisting and Token Sharing Strategy Evaluation.....	217
Table 9.2: Web-HMI Authentication Strategy Evaluation Results.....	222
Table 9.3: Reliability Evaluation Results	227
Table 9.4: Broadcaster's System Throughput Evaluation Results.....	232
Table 9.5: Response Times Evaluation Results.....	234
Table 9.6: Scalability Evaluation Results for Ford-Festo Test Rig.....	236
Table 9.7: Plant Architecture Mapping Strategy Evaluation Results	237

Chapter 1 : Introduction

Chapter Contribution to this Thesis:

Contributions of this chapter to this thesis are to identify the main inspiration for carrying out this research and to highlight the scope of work involved within this thesis. A description of research is introduced, research questions are raised and the overall formation of the research is mapped out.

1.1 Research Inspiration

Automotive industry (specifically powertrain manufacturing process) has been facing challenges with constant pressures of globalisation [1], environmental concerns [2] and ICT (Information and Communication Technology) innovations [3]. Globalisation has resulted into manufacturers to geographically distribute their activities and deliver wide variety of high quality customised products at lower prices [1, 4]. Environment regulations are proceeding towards strict regulations regarding waste discharge and fugitive carbon dioxide (CO₂) engine emissions and energy consumptions [5], and ICT development is pushing the boundaries of accessing machine critical information regardless of the location and distribution of the implemented system [6]. These factors impact product lifecycles and its manufacturing lead times, all of which need to be shorter.

Enterprises are changing their manufacturing paradigms and systems towards mass customisation to improve the efficiency and the support required by current agile trends, which have evolved from mass production, beyond lean manufacturing, into agile manufacturing [7]. There is a strong need for manufacturing industries to change their attitudes towards machine design and building process, and their maintenance structure, to accommodate agile changes. Designing and building manufacturing machines for producing vehicle engines, and their maintenance, are key areas in the lifecycle of manufacturing machines [8, 9]. A project initiated by the Manufacturing System Integration (MSI) Research Institute at Loughborough University has successfully examined a Component Based (CB) Approach to design and implementation of vehicle engine production machines to support reconfigurability and reuse requirements of the agile manufacturing paradigm. This research is based

within automotive domain with Ford Motor Company as a primary industrial collaborator in the United Kingdom.

The most agile components within any industrial system are the human personnel involved in controlling and monitoring production machines throughout their lifecycle. Control and monitoring in automation is a high investment industrial sector worth around 188 billion Euros, with an annual estimated growth of 7.8% (prior to worldwide recession). Control and monitoring of factory automation systems (manufacturing and process industries combined) represents €62 billion of the European market [10], consisting of segments like application design, simulation, manufacturing, installation and maintenance [11]. This demonstrates the importance of this sector in contributing to the overall world economic stability.

To adopt and implement the CB approach in industries, it would be essential and beneficial to design a web interface between the human and the machine, and to have system architecture in place, that can support control and monitoring of production machines throughout their lifecycles, in order to:

- provide more flexibility and promote sustainability in terms of reconfiguration, remote operation and maintenance,
- support more natural human machine interaction and recover from failures at a lower cost regardless of machine types or locations,
- save costs and encourage innovation opportunities by providing an integration framework where third-party solutions can easily be accommodated (i.e. vendor-independent (“open”) resource integration), and
- increase confidence in the employment of manufacturing practices by enabling virtual machine validation prior to its actual implementation, leading to a faster ramp up.

1.2 Problem Description

Competitive markets of today require industries to respond quickly to changes in order to maintain their competitive edge. These changes are usually driven by global, environmental and ICT factors which impact manufacturing product lifecycles such that their development and lead times need to be shorter [4, 12]. Industries are tremendously pressurised to market their products faster and achieve a quicker machine build, requiring a shift of existing manufacturing paradigms towards agility. Agile manufacturing requires machines to be flexible to changes and easier to manage through their lifecycles to provide more intuitive human machine interaction and recovery from failures at a lower investment.

A critical component of any manufacturing automation system is the control system and how operators interface to it. It is essential that the interface between the operator and the machine is designed and implemented to meet the demands of agility. Machine critical information needs to be retrieved and utilised at real-time through the lifecycle phases to support faster machine ramp up, maximise its operational effectiveness and decrease its downtime.

The existing approaches to operator interface system implementation and utilisation are not suitable to the current trends and business needs as they do not support key lifecycle phases of manufacturing machines. Specifically, a number of limitations can be identified as follows:

- They are vendor-specific and implemented on vendor-provided panels. This prohibits machine information to be collected freely from variety of control devices and locks end users to expensive but closed support solutions, creating support islands on the shop-floor and introducing unnecessary additional burdens such as license management issues and separate maintenance contracts. Furthermore, it restricts implementation of innovative approaches to machine monitoring and support.
- The interface screens are not automatically generated from the machine control description. Any change to a machine component is programmed

(and not configured) on the screens using proprietary software tools. This causes unnecessary expense and can lead to introduction of new errors to the system. Moreover, this activity requires specialised skills and knowledge to modify operator interface system codes.

- Machine builders cannot evaluate the machine logic using virtual simulation prior to the actual build event. This adds to the costs and time associated with any product engineering changes that may be required later as the design matures.
- The operator interface system is specified and implemented in machine-specific form such that the screens are uniquely tied up with the machine components. This creates lack of portability in terms of reusing or reapplying the same operator interface system to other machines within an engine programme.
- There is no provision of remotely controlling, monitoring and maintaining machines to save unnecessary costs, time and efforts incurred in travel and troubleshooting machine issues.
- Operators can only be trained after the machine has been physically installed at the end user's site. This delays production and can lead to improper training due to time constraints.

There is a need for more flexible and scalable next-generation operator interface system that can handle variability at moderate cost and within acceptable time. The research addressed within this thesis looks at the powertrain manufacturing lifecycle, specifically use cases in the context of automotive domain particularly engine assembly.

1.3 Research Questions

To articulate the research problems described in the section 1.2, a number high-level research questions have been raised, each addressed in its respective chapters as follows:

- What is the existing practice of utilising operator interface systems for the lifecycle support of the powertrain manufacturing machines? What key phases of the lifecycle are not currently supported by the operator interface systems and why does the powertrain automation need to address this issue? - (refer to the chapter 2)
- What other researches in academia and industry are working upon to address some of the lifecycle usage requirements obtained from the existing state-of-the-art? What automation design approach can be used within the operator interface system solution to satisfy these requirements and what are the desired attributes of this system that are needed to be supported to provide lifecycle usage of the selected automation design approach? – (refer to the chapter 3)
- What are the enabling technological opportunities and methods that can provide a materialising platform for designing and developing operator interface systems that can satisfy the research requirements? – (refer to the chapter 4)
- Are there any design guidelines used within the industry which govern the operator interface system screen structure and its navigation? How can these design requirements be captured and represented to enable operator interface system implementation to comply with the required industrial standard? What system architecture is needed for the operator interface system to be integrated with the automation approach (i.e. CB paradigm) to support these design and lifecycle requirements in industry? – (refer to the chapter 5)

- How can various components of the system architecture be designed and implemented such that operator interface system supports the machine lifecycle requirements? (refer to the chapter 6 and chapter 7)
- Are the research requirements qualitatively and quantitatively evaluated to realise the industrial adoption of the proposed solution? (refer to the chapter 8 and chapter 9)
- To what extent this research fulfils its requirements and what is the visionary impact of this solution on the powertrain manufacturing lifecycle? How can this operator interface system research be applicable to other industrial sectors? (refer to the chapter 10)

1.4 Research Formation

The general research methodology conducted and described within this thesis is illustrated in the figure 1.1. The plan pursued during the course of this research has been split up into six research design stages. The first stage deals with exploration which involves reviewing the current state-of-the-art in powertrain manufacturing lifecycle to establish the need for this research (covered in the chapter 2). The second stage deals with formalisation where a conceptual solution to the industrial requirements and opportunities identified in the previous stage is addressed using set of implementation features (covered in the chapter 3). This stage also involves reviewing technological trends in operator interface systems design and engineering (covered in the chapter 4).

The third stage (covered in the chapter 5 and 6) deals with interpretation where a hypothetical solution to the formalised features identified in the previous stage is designed. This solution entails developing a theoretical HMI operational requirements model and specification of the overall control and monitoring system architecture. Forth stage deals with instantiation where the conceptual theories and associated system architecture is implemented in the form of a proof-of-concept demonstration system (covered in the chapter 7). The

penultimate stage is evaluation where the research systems approach is qualitatively and quantitatively examined and validated (against research needs) to understand its effectiveness using a set of scenarios (covered in the chapter 8 and 9). The last stage is conclusion which formulates the knowledge accumulated, identifies the research strengths and proposes future opportunities within this research (covered in the chapter 10).

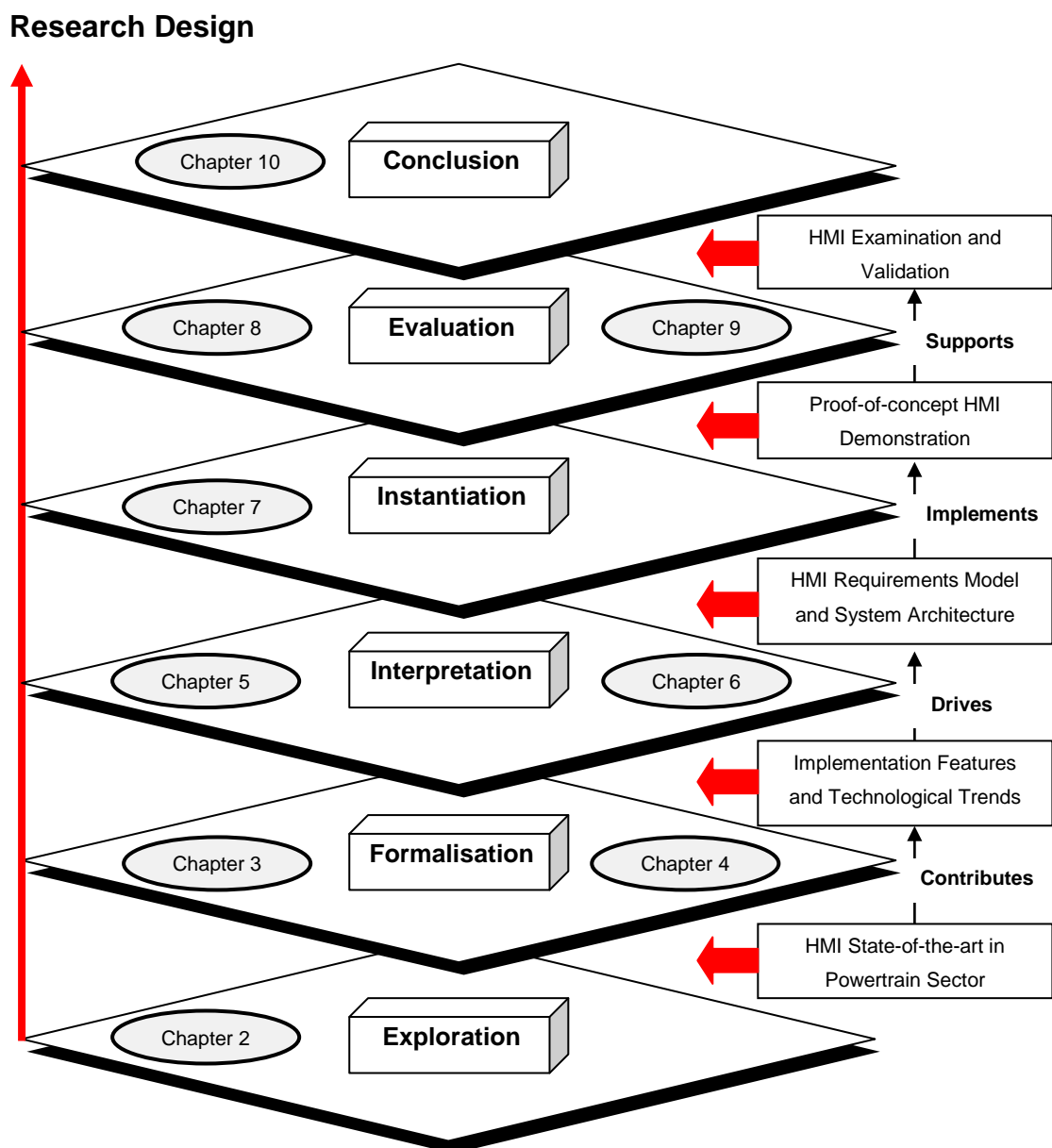


Figure 1-1: Research Plan Stages

Chapter 2 : Manufacturing Systems Review

Chapter Contribution to this Thesis:

From manufacturing systems' aspect of automation, the main contribution of this chapter is to demonstrate an engineering need to address the implementation process for operator interface systems, effectively providing the required level of control and monitoring support to production machines through their lifecycle phases.

2.1 General Overview

To appreciate demands and various challenges in the engineering process of production machines, it is fundamental to initially provide an explanation of frequently used concepts within this chapter and identify various factors that drive the change. Subsequently, a background literature of traditional and existing manufacturing practices is carried out to determine research requirements of next-generation operator interface systems for the lifecycle support of machines. In conclusion, a review analysis is carried out to summarise the need for this research.

2.1.1 Preliminary Explanation

Throughout this chapter, a number of concepts have been referred. Author preliminarily explains some of the essential terminology in this section of the chapter.

Manufacturing Automation: It is any system which automates the process of transforming resources (such as raw materials) into a finished product to gain profits and / or to capture a market [13], by satisfying customer demands. The final product is usually an assembly of basic parts which are designed and manufactured prior to putting them together [14]. To operate any manufacturing system, two management actions are required such as production planning and production control [15].

Operator interface: Operator interface is defined as a user interface which enables interaction between a human (i.e. typically a machine operator, a diagnostic engineer or a maintenance engineer in manufacturing automation environment) and a manufacturing machine [16]. In industries, this interface is usually termed as a HMI (Human Machine Interface), thus it is also interchangeably used to describe operator interface within this thesis. HMI acts as a front-end to gather information, monitor and diagnose industrial systems [17]. From machine operator's perspective, this interface monitors current machine status, accepts input (such as button pushes or keystrokes) and enables controlling of a machine using a set of actions.

PLC (Programmable Logic Controller): It is a widely used industrial microprocessor-based control system in manufacturing automation. It is able to store machine instructions to implement its functions such as sequencing, timing, counting, arithmetic, data manipulation and communication, to control industrial machines and processes [18]. It usually incorporates a number of input/output terminals for interfacing to machines [19].

Supply-chain partners: In this thesis, they correspond to end user (vehicle manufacturer), machine builder and technology vendor. An end user designs and produces vehicle engines using a production machine and subsequently assembles a complete vehicle. A machine builder is responsible for the design, manufacturing and installation of a production machine for the end user. A technology vendor is responsible to providing required components and technology to implement a machine for an end user. The participating entities (i.e. end user, machine builder and technology vendor) in a machine lifecycle are also termed as the stakeholders. In this thesis, technology vendor is also termed as controls vendor.

2.1.2 Paradigm Shift

During the industrial revolution in the late 19th century, mass production was presented as an approach to support high volumes of customer demands with minimum variety and cost [20]. Manufacturers were producing more per worker-

hour, eventually lowering the cost of the end product. This was a successful paradigm due to the stability provided by the markets in terms of established customer demands, limited varieties and fewer competitions. Profits were made by manufacturing large product batches in order to minimise costs associated with the production process [21].

Today's market requires accommodating customers in terms of offering different variety of products (or its variants), for example, Mazda 323 vehicle model came in 180 different colour palettes, including four shades of black [22]. The cost of offering product variety includes the actual cost of customising or configuring products, all the setup costs, excessive inventory costs, operational costs, and procedures and process costs. Under mass production paradigm, this cost increases exponentially with an increase in product variety in the market [23]. Owing to these variation requirements and factors, Dean [24] reports that the mass production is not an effective approach to mass customisation production process. Mass customisation is a paradigm where individually designed products and services are provided for every type of customer through process agility, flexibility and integration [25].

Modern production systems must be able to handle changes and manufacture highly customised, design-to-order products, where additional services and value added benefits such as product upgrades and any future changes are as important as the product itself [26]. A paradigm shifting process is thus required to be adopted by manufacturers in order to fabricate their business models to be compatible with today's agility requirements such as reconfigurability, flexibility and efficient use of resources [13, 27-29]. Reconfigurability can be achieved through increasing interoperability between shop-floor devices and reducing design, build, installation, and programming efforts. Flexibility can be enabled by both compatibility and cross-company communication (regardless of geographical locations), and legacy system support. In order to efficiently use resources, human errors are required to be reduced and machine maintenance needs to be optimised [27].

In response, a number of organisations and research institutions have proposed agile manufacturing solutions to these rapid, continuous and unpredicted

changes [30, 31]. The concept and enablers of agile manufacturing are covered in the section 2.2.1 of this thesis; however the author has identified number of major factors that are driving this change resulting into a paradigm shift from mass production to mass customisation, eventually affecting the overall machine lifecycle process. These factors are detailed in the next section of this chapter.

2.1.3 Factors Driving the Change

Changes to manufacturing automation practices are occurring more frequently in this era compared to the past. In this research, the term “change” means those that occur at machine production and engineering level. The MASCADA report as described in [28], distinctly defines two types of changes namely, production change and production disturbance. A production change is usually planned and refers to intentional adjustments to production machines where as a production disturbance is an unexpected adjustment experienced by production machines, and it is usually reacted upon.

A number of factors are driving the requirements for a change in current manufacturing machine design and build processes, eventually impacting upon implementation of operator interface systems to support next-generation trends (described in the chapter 3.3). These are as follows:

Globalisation

In the 21st century, globalisation is not a new concept as the process of integrating activities in an international scale has been underway for decades; however, this era has seen an unprecedented growth on the grounds of domestic and international competitions, higher market share needs and low volume product customisations. A number of firms are selling vehicles to mature economies such as USA, Germany and Japan [32], and a wave of assembly plants in low wage economies have emerged due to the rising costs of resources and better opportunities.

Companies are off- shoring (and sometimes outsourcing) their manufacturing activities (such as design, production, marketing, distribution) and needs to (and sometimes from) overseas emerging global markets (such as Brazil, China, India and Russia) to survive these competitions. Motivation for this action is not only to lower the upfront capital investment but to lower the labour costs too, enabling industries to produce low-cost high quality innovative products [18]. Since these markets have emerged as high quality automotive producers as well as low-cost centres, many supply-chain partners have begun moving their production activities into these areas to search for more customers. As a consequence, a number of new suppliers, including Mexico, South Korea and Spain, have been very successful in penetrating world automotive market [33]. This has created a business world where manufacturing activities are globally distributed for the need of acquiring better skills, market coverage and reducing costs [18, 34, 35]. In the light of these issues, it is essential that vehicle manufacturers offer products to customers in a shortest possible time, thus a prevailing trend is to shorten a product lifecycle [36].

Figure 2.1 illustrates how shorter product lifecycles are demanding faster design, build and ramp up of manufacturing machines [37]. Figure 2.1 (a) shows a traditional manufacturing sequential approach where the initial development of a product (i.e. vehicle engine) is followed by the design and build of a production machine. This machine is then ramped up to its full production capacity (where the product is then produced). However, the current global market trends (such as shorter product lead-times, more variants, low and fluctuating volumes, and low price, in addition to quality and durability [13]) demand rapid introduction of new, upgradeable, customised products to remain competitive. Machines must be quickly ramped up to full production capacity and readily reconfigured to meet the manufacturing requirements of new products of the same family (as shown in the figure 2.1 (b)). This requires providing careful attention to the existing machine lifecycle phases and the process through which operator interface systems are implemented within a production programme.

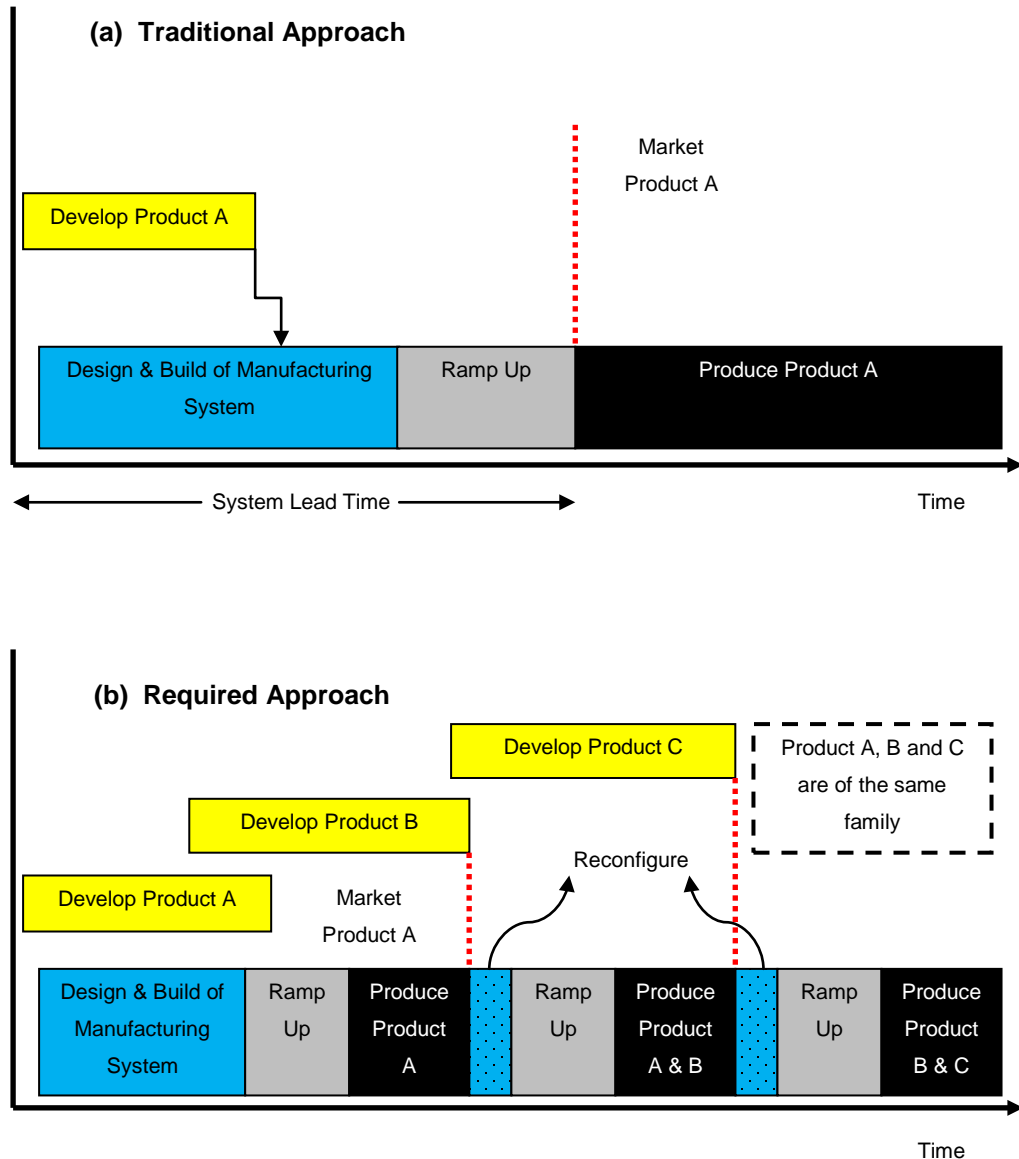


Figure 2-1: Globalisation Demanding Faster Design, Build and Ramp Up

(Adopted from [37])

Environmental Concerns

The environment is deteriorating at an enormous rate as resources like raw materials are diminishing, waste discharge sites are overflowing and pollution is increasing [38]. For adopting an environmental thinking approach to production of vehicles, major revisions are usually required throughout the entire supply-chain management, including material sourcing and selection, product design, manufacturing processes, final product delivery and end-of-life of the product

[38]. Sullivan [39] has found that material production and manufacturing stages contribute up to 65% of particulate emissions and consumes around 14% energy. This may sound like energy consumed by manufacturing process is a tiny portion of the overall utilised energy; however, it has to be noted that the rest of the energy used during vehicle usage (around 86%) is distributed over 10-15 years period, while the manufacturing energy demand (and accompanying CO₂ emissions) occur over a much smaller period. This requires one to have a closer look at the powertrain subsystem of a vehicle manufacturing lifecycle.

Presently, vehicles are expensive for most people in developing economies like China and India; however, with an increase in standard of living of these countries, this situation may no longer remain stagnant. As the demand for vehicle increases, more vehicles are going to be manufactured and used, eventually deteriorating the environment. All these facts indicate that drastic amount of changes are required in the design and build of manufacturing machines to produce more efficient vehicle engines [2]. Efficient engines require less fuel, saving the customer money and reducing negative effect on the environment. To produce environmental friendly vehicles, it is extremely fundamental to review and rectify existing manufacturing practices to manage resources like costs, raw materials, energy and ultimately the environment. A number of global vehicle manufacturers, such as Ford Motor Company, General Motors and Toyota are aware of these issues and have started addressing them for improving environmental performance [40].

ICT (Information and Communication Technologies)

ICT has developed at an exponential pace in the last decade. Likewise, opportunities and pressures of implementing innovative ideas at industrial levels have increased. With improved speeds of communication networks such as standard Ethernet (e.g. 100Mbps, 1Gbps) making its way to the shop-floors [41], and improvements in communication paradigms such as SOA (Service Oriented Architecture) - based web services and the WWW (World Wide Web

[42, 43], firms are now having opportunities to deploy state-of-the-art in ICT for their day-to-day manufacturing design, build and support process using the Internet [18, 44]. In the field of software engineering, modern programming frameworks (for example, Microsoft dot Net [45]) are able to provide smart and secure support tool functionalities which were unrealistic before. Coupled with the performance capabilities offered by today's personal computers (PC), ICT is surely finding its way at all the phases of a manufacturing machine lifecycle. Within manufacturing automation industry, ICT can be used at any number of different levels and stages such as simulation, data acquisition and display, control system, robotic design, etc [46].

Large and complex manufacturing systems cannot be efficiently and safely managed without flexible control and monitoring support applications. ICT provides flexibility to manufacturing industries and enables them to closely monitor and carefully control their activities [44], and thus drastically alter their production practices by increasing production rates, improving production quality, reconfiguring production lines, reducing energy consumption and shortening production machine lifecycle, to save resources and obtain a competitive edge [47]. In order to achieve such targets, human resources and technological opportunities should be used effectively for providing a timely response to increasing customer demands. While currently, most of these applications are proprietary, huge savings and innovative approaches to designing operator interface systems can be obtained if ICT opportunities are properly deployed through the machine lifecycle with vendor-independence in mind.

To cope up with these above described factors, agile manufacturing concept is a proposed solution in the academia. Next section identifies manufacturing trends and existing manufacturing state to justify a need for this research.

2.2 Part A: Manufacturing Trends

2.2.1 Agile Manufacturing

Though the concept of agile manufacturing was first introduced by Yusuf [48] in 1991, it has been widely disseminated as representing essential requirements for next-generation manufacturing automation systems. It can be defined as an adaptive capability to survive and succeed in a competitive environment of continuous and unpredictable changes, by quickly reacting to changing markets which are driven by customer-designed products and services [49]. Agile manufacturing facilitates re-allocation of production line capacity to higher than expected demanded products and quickly launches new products, yet retaining the production ability for other lower than expected demanded products. Automotive industry is attracted to agile manufacturing as it potentially offers equipment reuse and investment cost-reductions over time [50].

With reference to multi-facet description of agile manufacturing as discussed by Gunasekaran [30], focus of this thesis surrounds the production and system integration facet. The production facet is characterised by flexibility and reconfigurability of production systems to shorten machine lifecycle [51] whereas system integration facet is characterised by collaborative methodologies in integrating various supply-chain applications for supporting efficient productivity and performance of manufacturing systems [52].

The main aim of this research is to provide lifecycle support using operator interface systems that function in flexible and reconfigurable production lines to encourage creation of varying product types / volume. The conceptual model illustrating enablers of agile manufacturing has been studied and modified to fit the previously mentioned research aim as shown in the figure 2.2. From the production system facet, the key agile enablers are modularisation and change capability. From the system integration facet, the key agile enablers are business-production system integration and information retrieval and utilisation.

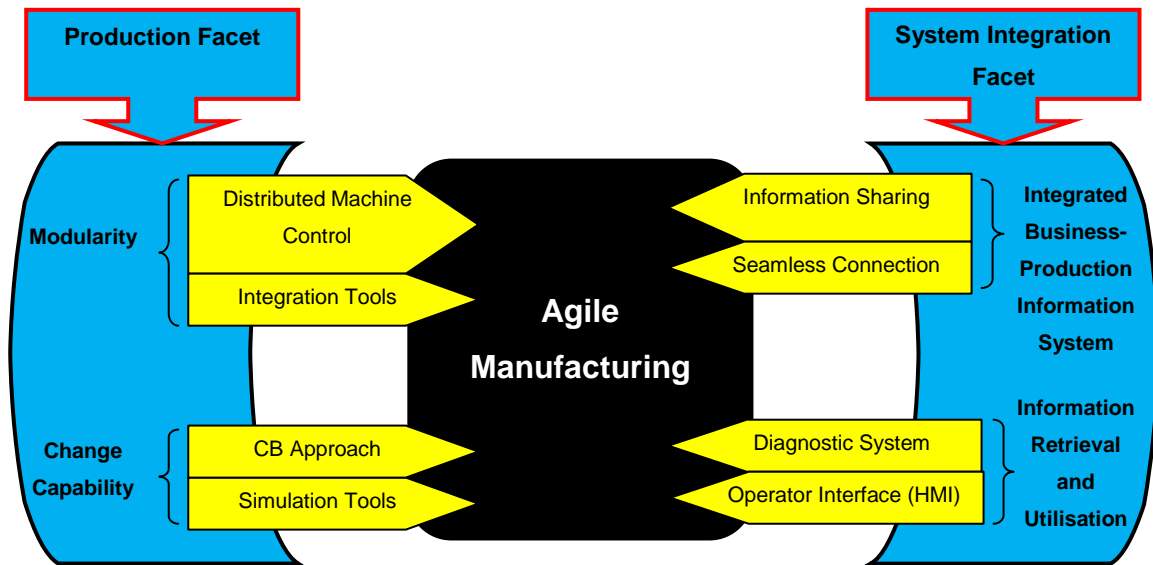


Figure 2-2: Key Agile Enablers

Modularisation and Change Capability

Introducing modularity increases system functionality in terms of flexibility and reconfiguration, when conditions demand changes. Modularisation allows changes to be made to a few isolated functional elements of production without necessarily affecting the design of other elements [53]. Modular production system design research has proposed solutions for both, reconfigurable mechanical structures and control software applications, enabling quick changeovers in decentralised automation systems to improve flexibility and adaptability of machine systems as presented in [54].

Key modularisation concepts are distributed machine control and integration tool for flexible and reconfiguration support. In distributed machine control, industrial control devices (such as a PLC) are connected through communication networks, distributed throughout shop-floor. The control functionality is decomposed and distributed to individual controllers to match the required physical modularity of the machine [54]. Any modification of device control functionality is independent of the functionality of the others since they are interlocked using configuration data and internal state variables. Consequently, any change to the implemented system is carried out through reconfiguration rather than reprogramming. In order to support adoption of

modular control systems, integrated engineering tools (as described in the chapter 3.2.2) are required for building, changing and managing machine applications, for example, synchronisation and close monitoring of control devices.

For rapidly changing and commissioning automation system (i.e. change capability), concurrent design of manufacturing activities are needed to be supported using new engineering automation design approaches. One such approach (adopted within this thesis) is a Component Based (CB) design (described in the chapter 3.2). This integrates and validates individual machine components (using, for example, simulation tools) in early machine lifecycle phases to detect failures and deviations from requirement specification as early (and hence economically) as possible.

Business-Production Integration System, and Information Retrieval and Utilisation

Integrated business-production information systems are needed to enhance flexibility and reconfigurability of shop-floor devices by increasing information transparency and data mobility across heterogeneous platform systems [55]. Critical information distributed across manufacturing enterprise (for example, status of machine components, production capability) has to be shared between various automation levels (described in the section 2.3.1) for added benefit, for example, documentation sharing can shorten design life cycle of products and manufacturing machines [56].

Applying information sharing approach to other manufacturing activities, such as machine maintenance, can support learning from outcomes of previous projects, consequently, reducing and resolving new problems [57]. Maximum benefit from information sharing can be obtained when the content is characterised in a standard way that can be accessed and understood by others (seamless connection) [30, 56].

Current manufacturing trends recognise the strategic importance of collecting, disseminating and analysing production operation information. With the high-speed data communication of the Ethernet and field bus technologies,

distributed shop-floor devices can be monitored, analysed and configured / upgraded in real-time. This provides opportunities for having manufacturing support systems that are more open, flexible, distributed and extensible. Information from the shop-floors can be utilised in diagnostic systems and operator interfaces (HMI). Utilisation of collected data (such as temperatures, pressures, flow rates, RFID tags, etc) from shop-floor devices can be monitored and visually analysed to identify any issues (such as process downtime, parts degradation, faults, throughput, etc).

Furthermore, plant managers need information systems support for product planning and scheduling. HMI's and Supervisory Control And Data Acquisition (SCADA) systems are the traditional proprietary tools which provide control and performance visualisation at the shop-floor level. Currently, these systems are the main interfaces typically transferring significant amount of data to be converted into information that is utilised by the production management systems. Emerging trend is towards more open (vendor-independent) systems which allow machine information to be collected freely from variety of vendor control equipments and preventing manufacturers from being locked into proprietary solutions (as identified in the section 2.3), ultimately increasing flexibility significantly.

The focus of agile manufacturing is on the manufacturing enterprise and the business practices needed to adapt to ever changing global uncertain market. Any operational techniques or any engineering solutions are not provided by agile manufacturing. As a consequence, agile manufacturing compliments reconfigurable manufacturing (described next) owing to the share of focus on the objective of manufacturing responsiveness (i.e. agility) [58].

2.2.2 RMS (Reconfigurable Manufacturing System)

In manufacturing, especially discrete manufacturing, changes occur frequently in the production lines with every change made to a product [27]. In manufacturing automation, a new product is introduced every 6 to 9 months, which in turn requires modifications to existing production lines. When

compared to the past practices where production lines were usually sold after being used for a specific product, there is a need for these lines to stay operational (through reconfiguration) to manufacture any new product [27]. Machines from production lines have to be reconfigured and reused more efficiently in order to maximise the return on investment [59]. This has motivated new paradigms within the research community, such as RMS.

In contrast to agile manufacturing, reconfigurable manufacturing does not deal with the entire enterprise, but only with the responsiveness of the production machine system to new product demands in a globally competitive environment with niche market production. RMS methodology of rapid system design and ramp up, as well as the capability to dynamically increment the capacity and the functionality in response to market demands, is one aspect of agility (i.e. agile manufacturing) [58]. Instead of providing general flexibility through the use of equipment (with built-in high functionality), RMS provides customised flexibility for a particular part-family through scalability and reconfiguration [60] to improve or upgrade a system rather than completely replacing it.

RMS is designed at the outset for quick change in structure, as well as, hardware and software components [13], where changes do not affect rest of the system. A manufacturing machine system can be created by integrating basic process components, both hardware and software, that can be rearranged or replaced rapidly and reliably [37]. Any adjustment to production capacity or functionality can be carried out through reconfiguration (and not reprogramming), thus enabling addition, removal or modification of specific process capability, control, software, or machine structure. Open-architecture control (i.e. reconfigurable software), modular machines (i.e. reconfigurable hardware) and early operator training are RMS key enabling technologies [61].

Overall, RMS aims at [60]:

- reducing lead time for launching new systems and reconfiguring existing systems, and

- rapidly modifying and quickly integrating new technology and new functionality into existing systems through rearrangement of basic components (hardware and software).

For a manufacturing machine system to be readily reconfigurable (and achieve desired reduction in lifecycle cost and time), it must possess certain RMS key characteristics as prerequisites, summarised in the table 2.1 below [37, 58, 60-63]:

Key Characteristic	Summarised Description
<i>Modularity</i>	All the major components (for example; mechanical structural elements, control systems, software and tooling) have to be designed in a modular fashion. Manufacturing machines (and its peripherals) can be quickly built and ramped up, if its software and hardware is designed with modularity in mind.
<i>Integrability</i>	The machine system has to be engineered from a pre-designed set of components for ready integration, and any future introduction of new technologies has to be accommodated.
<i>Customisation</i>	By utilising the concepts of customised flexibility and customised control, RMS can be configured to meet the requirements of a whole part family. Customised flexibility means that the dominant features of the part family being manufactured, has to determine the overall machine configuration. Customised control can be achieved by integrating control components with the help of open-architecture technology which can provide the exact control functionality needed.

<p><i>Convertibility</i></p>	<p>System has to permit easy conversion between existing products, which enables rapid calibration of the machines after reconfiguration. Furthermore, machine system should be adaptable for future products.</p>
<p><i>Scalability</i></p>	<p>RMS is highly scalable which enables rapid capacity change, incrementally and economically. Machines may require, for example, addition of spindles to increase their productivity. This requires machine systems to be as scalable as possible, especially in product volume.</p>
<p><i>Diagnosability</i></p>	<p>RMS enables quick identification of reliability and quality issues that occur in machine systems throughout their lifecycles. Machine information (i.e. machine status and manufacturing process) has to be distributed using web-based technologies (in the form of web-based HMI's) for organisations to closely (and remotely) monitor and control their manufacturing activities (using flexible system architecture). These features are very critical in reducing machine ramp up time for RMS (hence the main focus of the research described in this thesis).</p>

Table 2.1: Key RMS Characteristics

A project initiated by the Manufacturing System Integration (MSI) Research Institute at Loughborough University has successfully examined a Component Based (CB) Approach (described in the chapter 3.2) to engineering machine control. This approach enables design and implementation of production machines to support next-generation reconfigurability requirements discussed earlier. Since the primary focus of this research is the operator interface system (i.e. HMI) aspect of these machines that provide control and monitoring

throughout their lifecycles (as described in the chapter 3.4.1), a review of existing manufacturing practice is covered next to identify the state-of-the-art in manufacturing automation practices.

2.3 Part B: Existing Manufacturing State

To appreciate the state-of-the-art in discrete manufacturing, this section initially depicts the hierarchical levels of operations generally found in a manufacturing facility followed by the existing operator interface to PLC architecture. The scope of operator interface system within manufacturing machine lifecycle and existing maintenance procedure is then described. An overview of operator interface users (i.e. roles) is provided to integrate requirements of various users within next-generation operator interface system design and implementation for control and monitoring machines. Lastly, major limitations (which eventually instigate the research need) are summarised.

2.3.1 Hierarchical Levels of Operations in Manufacturing Facility

Figure 2.3 shows a typical manufacturing hierarchy based on the ANSI/ISA standard-1995 [64], ISA-1999 [65], Purdue reference control model [66] and Ford Motor Company's Fox programme assembly layout (chapter 8.4). It depicts the integration levels and zones describing a link between a typical shop-floor manufacturing system and business systems. This depiction is useful for exchanging machine-critical information consistently and securely between manufacturing systems and business systems, and forms basis for their integrations. It comprises of a number of hierarchical levels described as follows:

Enterprise Zone: This zone's level is concerned with systems supporting enterprise resource planning and distribution, supply-chain management, factory production scheduling, order management, business planning and operational management. In terms of network, this forms part of corporate

network which is usually isolated from the other levels. This level typically interfaces downward to the level 3.

Manufacturing Zone: This zone's level is where all the key applications needed for collecting data for analysing and logging purposes are found. Typically, Ford UK implements production monitoring system called POSMON [67] and other quality assurance systems at this level. Collected data (from the level 1) can be displayed in various formats (for example, Microsoft Excel) and can be accessed from various locations throughout the plant. In terms of network, this level is typically found in an intranet network zone.

Cell / Area Zone: This zone has three levels. Level 0 is termed as a field level which comprises of sensors and actuators that perform various operations (based on commands from the level 1) and propagates its states to higher level. Devices operating at this level are usually connected using field bus networks [68].

Level 1 is responsible for processing data and directly influencing machine operations [69]. At this level, PLCs and microcontrollers are implemented. Applications at this level require cyclic transport functions which enables source information to be transmitted at regular intervals due to strict timing constraints [70]. The size of machine status data is usually very small where as the data representation is as compact as possible to reduce message transfer time [71]. This level can sometimes be further decomposed into a group level and a unit level. A group level controller coordinates activities of several unit level controllers. This level contains the true heart of the automation system [72].

Level 2 is where operator interface system is usually found within the hierarchy. It is this level where machine data is actually displayed to a human operator, which in turn may aid in decision making and carrying out of necessary operational and maintenance tasks on manufacturing machines. Next section illustrates the existing operator interface to PLC control system architecture currently implemented in powertrain manufacturing automation.

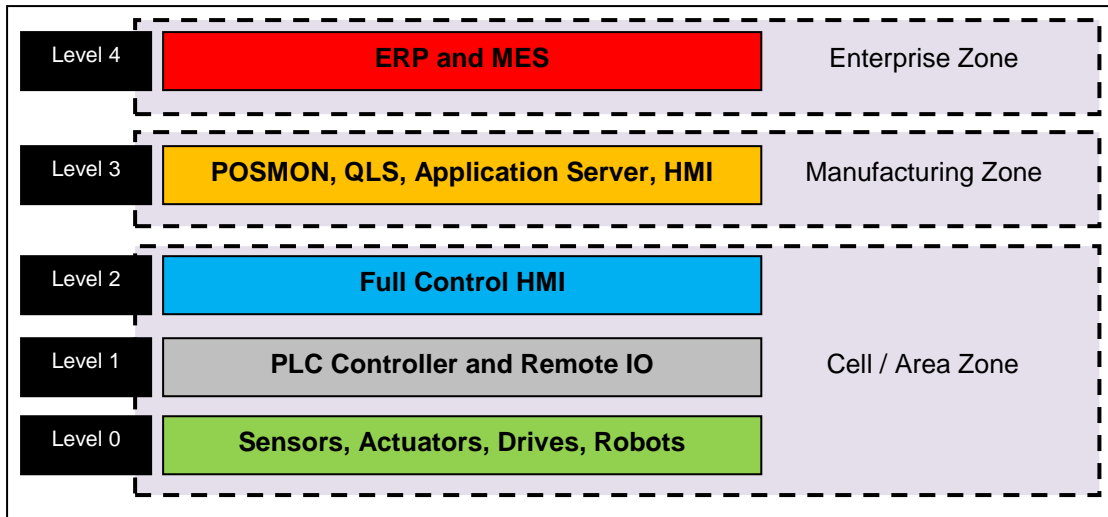


Figure 2-3: Hierarchical Levels of Operations in Manufacturing Facility

2.3.2 Operator Interface – PLC System Architecture

Operator interface system has evolved, from being just an entity supporting simple requirement for monitoring an activity to a sophisticated tool, for controlling and monitoring complex processes in manufacturing systems [73]. The future of HMI is certainly deviating from its existing implementation trend where anytime anywhere access to shop-floor information is no longer considered impractical.

The discrete manufacturing hierarchy described in the section 2.3.1 showed an overview of manufacturing system’s architectural level integration to business systems. Drawing from the existing literature and Ford Motor Company’s existing setup (chapter 8.4), typical system architecture in terms of data control and monitoring using operator interface system is conceptually shown in the figure 2.4 [17, 19, 47, 72, 74-77]. This common architecture is independent of the technology and can be mapped to any implemented networks, servers and applications in a physical world.

In the setup below, control device 1 and control device 2 correspond to PLC devices. PLC is the most widely used industrial control device; its logic can be programmed using languages like Instruction List, Structured Text, Function Block Diagram, Ladder Diagram and Sequential Function Charts, confirming to the IEC 61131 / 61499 international standards [78]. The implemented control

program is usually referred to as the machine control logic. This logic contains the production machine's sequencing and interlocking reasoning that supports real-time cyclic communication with the operator interfaces and I/O (Input / Output) signals at the field level for handling sensors and actuators.

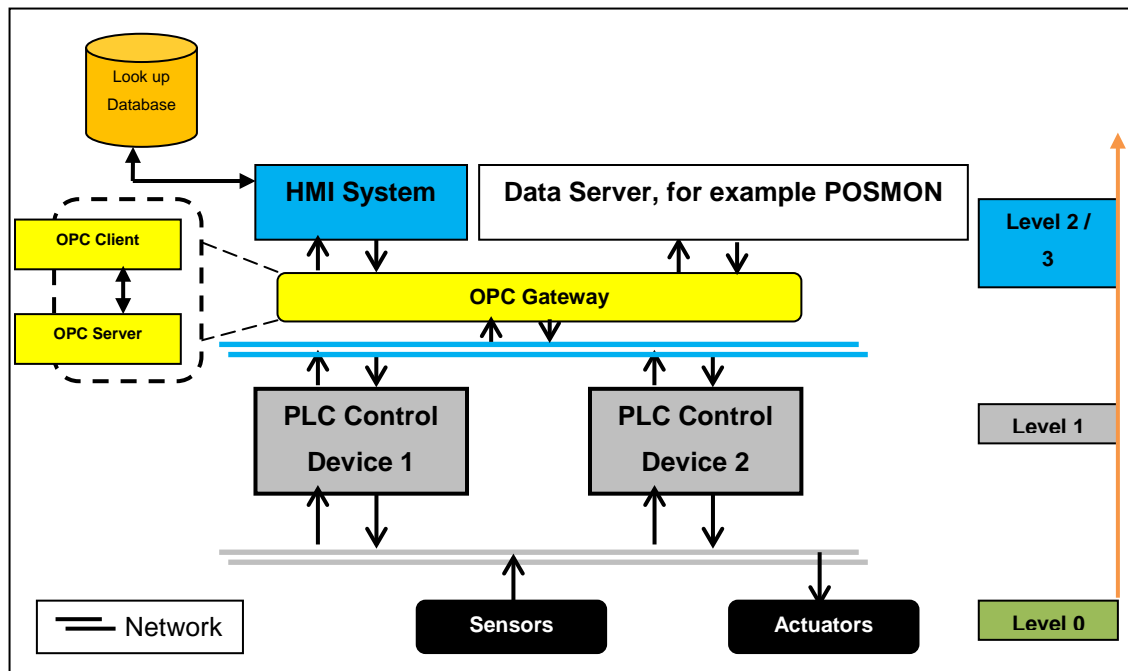


Figure 2-4: Generalisation of Existing HMI - Control System Architecture

The common language for data transfer within this architecture is through the OPC (OLE for Process Control) gateway. OPC gateway is the communication link which has become a defacto standard for retrieving data from PLC applications to the operator interface system [17, 76].

Operator Interface system is a software system usually programmed using vendor's proprietary software tools, and installed within an industrial vendor-supplied panel that interfaces to the machine's control logic. Two types of data can usually be visualised within the HMI system, i.e. soft real-time and historical machine status. Multiple HMI panels can usually be implemented to simultaneously monitor machine's records from various locations around the machine. To accommodate this functionality, PLC controls the access of various HMI clients to operate and monitor the machine. Furthermore, HMI system

interfaces with the control network to transfer low-level control device's register information which is used to describe the machines' real-time status. The HMI system contains a look up database that converts machine information into human readable machine descriptions and propagates it to the HMI screens.

Limitations of the Current Approach

The current approach of PLC-HMI architecture implementation has a number of problems as discussed next. All these factors ultimately increase machine lifecycle costs and complicate its support process.

- Supply-chain vendors (such as machine builders) tend to use proprietary technologies to retain their customers for obtaining maximum benefits when any changes are required. A change is not only costly but also requires a specialist knowledge base (or skill set) to develop and edit HMI programs due to difficulties in understanding the implemented solution [27]. End users and machine builders have found that when changes are required to such systems, the diagnostic engineers would spend large amount of time learning the programmer's individual coding style before diagnosing the actual machine faults. Difficulties occur in maintaining consistency between machine control system and HMI. The HMI and diagnostics are usually difficult to follow as they are not linked to the main control program and may not follow the same program sequence. The result is that when the PLC program itself is altered the HMI program may not be updated and hence the displayed messages soon fail to be correct or useful for an operator [79-81]. Thus, the HMI software must be updated with consistent naming and messaging data in order to display the correct message to an operator. A large effort is therefore required to accurately maintain the accuracy of any updates and control software releases.
- Data interoperability issues exist because machine support systems (or solutions) adopted and implemented by end users are provided by

various vendors [11]. A single supplier (i.e. controls vendor) may not provide all the control equipments and software required in an engine production programme. Technically, the goal is to connect automation components from different vendors in order to obtain the most efficient and cost-effective solution. Since each vendor adopts a different approach to their data model and communication protocol implementation, interoperability issues between the control device and HMI application still may exist. Owing to these interoperability issues, end user's are usually locked to a specific supplier to keep the overall implementation complexity lower.

- While existing HMI system is traditionally used for control and monitoring industrial machines, the greatest disadvantage of these packages are their high costs to companies and the restrictions imposed by the limited types of controllers and monitoring devices they support [47]. Often end users are tied up with a specific vendor for a production machine owing to the closed support solutions they offer. Since control systems and their associated operator interface systems are usually generated by the same vendors, this increases isolated islands at the shop-floors. Presence of these isolated islands requires drawing out separate machine maintenance contracts for various vendors.

Emerging Trends and Issues

While traditionally, operator interface systems have been functioning on a panel around immediate vicinity of a production machine, current advancement in the field of ICT has created an opportunity of operating a web-based HMI which can enable anytime, anywhere access to the shop-floors. This has resulted into possibilities of driving operator interfaces beyond just being user interfaces by accommodating advanced capabilities easily. To address security aspects of implementing remote connectivity, a firewall can be usually setup. The firewall isolates the internal network from the external network (i.e. internet) by allowing specific connections to pass while blocking others, protecting the internal

network from any unauthorised access. Additional mechanisms such as user credential logins and data transfer encryptions can also be implemented by vendors for added peace-of-mind.

In addition, applying web services for defining control devices is attractive both to manufacturers and system vendors as it simplifies device installation through reconfiguration and enables high-level control and monitoring applications to easily integrate with the control system [82]. A typical example of this is the design of FTB (Field Terminal Block) by Schneider Electric (more information is regarding this device is provided in the chapter 8.2). The traditional PLC implementation uses OPC gateway for communication, which in turn is made up of an OPC server (proprietary designed and provided by control vendors such as Siemens, Schneider Electric, etc) and an OPC client (can be programmed by anyone to link the PLC's OPC server to third-party engineering tools). When comparing this approach to the web services-based approach, the later promotes open standards through the use of XML over the web. Since XML presents data in a uniform format, it encourages open system implementation and makes it a good fit for distributed web architecture, where as PLC-based systems are characterized by their lack of agility and are often difficult to modify and extend due to their proprietary nature, and therefore do not provide the high degree of flexibility that is required for today's production machines.

However, web services use SOAP (Simple Object Access Protocol) as a message envelope for data transmission which potentially contributes to higher network bandwidth usage owing to its large size of 1 kilobyte per message. This may lead to system integrity issues especially when web services are employed at both levels of the system architecture (i.e. control device level and web-based operator interface system level).

2.3.3 Operator Interface System Scope within Machine Life Cycle

Since global production systems have become more important, machine lifecycle support (and its maintenance) has become an integral part of manufacturing systems [11]. With demands triggering frequent product

changes, their lifecycles are often measured in months rather than years. This trend of shorter product life is forcing production machine development lifecycle to be ever-shorter. Concurrent engineering of the product and the production machine through compressing timescales may be the only possible solution to bringing products rapidly in the market [54]. Though product development time has been significantly reduced by usage of Computer Aided Design (CAD) tools, this reduction is not paralleled in the design and development of production machines [60].

Presently, a development and implementation project for a new product typically takes 42 months (i.e. twice as long as the targeted time) in car manufacturing [83]. It begins with conceptualisation of a new vehicle (i.e. car) model and the engine associated with it, which in turn leads to the product design process. Midway through the product conceptualisation process, an end user contacts a machine builder to begin the process of the machine design and build for the vehicle engine system. The process of machine design and build normally takes 53 weeks to complete as shown in the figure 2.5 and it is deemed to be the riskiest process since it is extremely hard to modify the software and the hardware in existing systems, especially later within the lifecycle. Furthermore, since the engineering activities are resourced by distributed partners with different foci of concern, expertise and goals, ad hoc integration methods and mechanisms are currently utilised. It is essential to provide support at various key phases of the machine lifecycle to closely monitor and control its activities.

To appreciate the extent of problems associated with the life cycle of a production machine, specifically from the operator interface systems' perspective as it is the focus of this research, an insight to the design and build process has been described in this section of the thesis (see accompanying figure 2.5). This discussion has been determined from the available literature, interviews with a number of industrial collaborators (i.e. Schneider Electric, Siemens, Ford Motor Company, Krause and SAP) and research projects (i.e. SOCRADES and BDA) [11, 80, 83-90]. The main aim is to institute application engineering requirements for next-generation operator interfaces, implemented

within suitable system architecture, for control and monitoring of production machines throughout their lifecycles.

HMI Requirements for Machine Design and Build Process Support

The engineering process of designing and building a production machine involves an end user, a machine builder and a controls vendor. It is almost entirely a sequential process. Initially, end users have a requirement for a new production machine which addresses a set of business needs, for example, production capacity needs to increase or a new product has to be manufactured. HMI aspect of the production machine is not given any attention at this phase (i.e. concept phase in the figure 2.5). Machine builder is responsible for looking after the machine related aspects of the project where as the controls vendor is responsible for providing a suitable proprietary control technology (i.e. software) and hardware, and proposing network architectures which can meet the project requirements. Subsequent phase is the specification formalisation phase of the lifecycle where technology for the machine (including the required number and locations of the HMI stations within a production line) is specified. This phase (i.e. specification phase in the figure 2.5) is a result of a combined effort from the end user, the machine builder and the controls vendor.

HMI specification is defined by drawing a schematic (either through manual sketches or stand-alone graphical screen layout tools) of each individual screen detailed with its associated view and buttons for navigational, monitoring and control purposes. This process cannot begin until the machine has been roughly designed as HMI screens relate specifically to the particular machine's configurations. These configurations (with the associated process cycle charts) describe timings for machine movements. The requirements specification is then agreed upon and approved by supply-chain partners. It has to be noted that the HMI specification (and thus the final HMI system) is machine-dependent, i.e. its design is uniquely tied up with the machine components and cannot be reused or reapplied to other production machines within an engine programme.

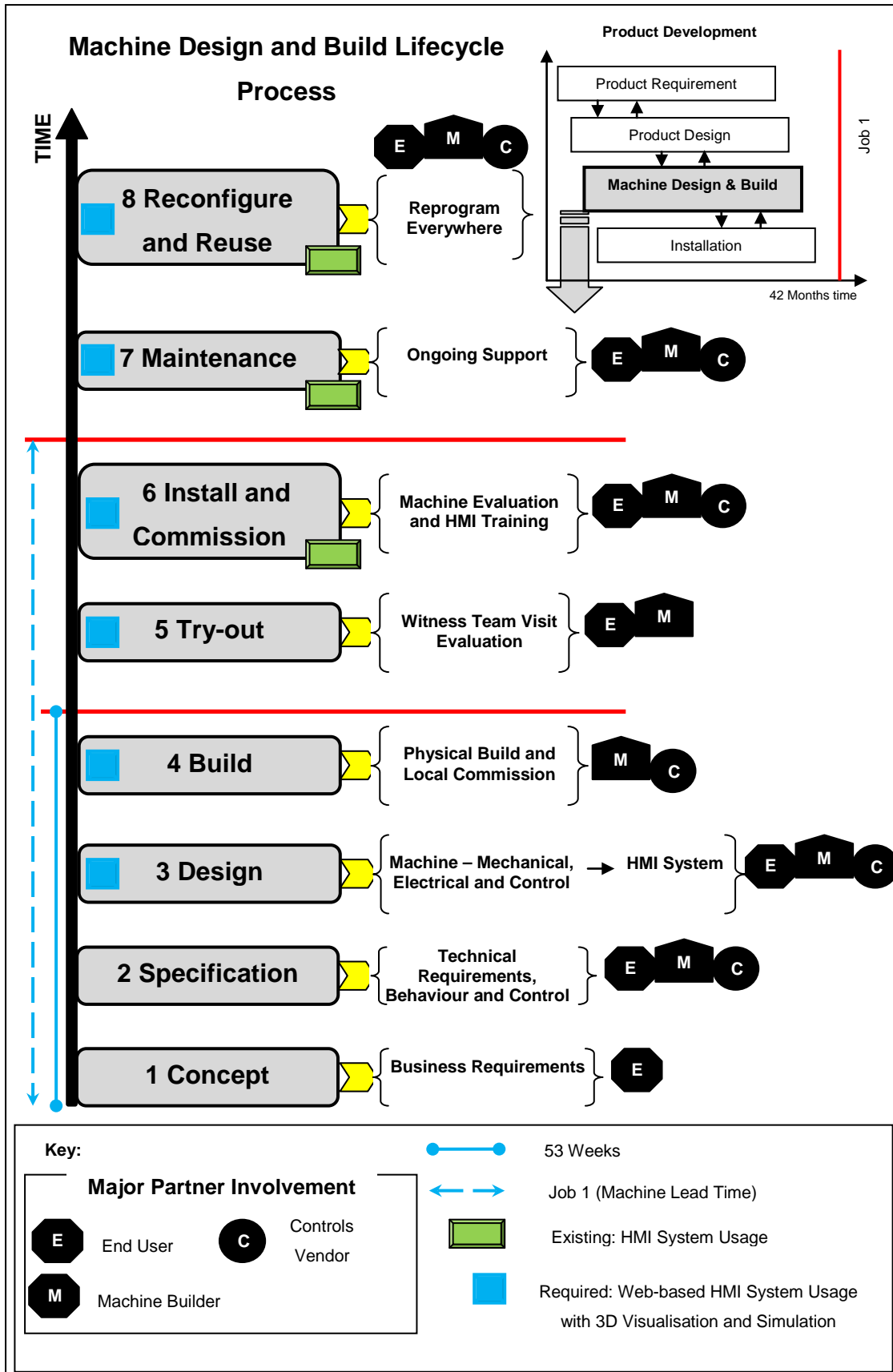


Figure 2-5: Machine Lifecycle Process with HMI System Usage Requirements

The design phase of the machine lifecycle (i.e. machine design, and HMI design and development) is a sequential process, beginning with mechanical engineering followed by electrical and hydraulic system design. Due to inherent characteristics of conventional approaches to HMI system implementation, its development cannot be completed until the control system's logic mapping information is available, the mechanism which allows the HMI to interface to the machine's control system (i.e. PLC system). The control software is engineered by specialist programmers and implemented to the end user required standard as per the specification. Associated HMI system is then added later on with the required monitoring and control functionalities by experts having participated in previous similar projects. Usually this process involves copy-paste coding techniques where PLC control logic codes are duplicated in the HMI side for it to reflect the machine configurations. This sequential approach of performing activities contributes significantly to the machine development efforts and time – critical metrics in today's manufacturing environment. It is necessary to compress the time for these activities where possible through concurrency and use of previous designs with the aid of engineering tools.

The next phase is the build phase of the lifecycle as shown in the figure 2.5 where the machine is physically built and commissioned at the machine builder's site prior to delivery to the end user. At this phase, sections of the machine are tested in order to reduce costs and to reduce time needed later to commission the complete machine system at the end user's site. The physical machine design and the control system design remain isolated from one another prior to this phase. In practical powertrain commissioning process, machine builder cannot currently evaluate commission activity in a virtually simulated environment prior to the actual machine build. Simulation can enable physical machine manufacturing to be delayed as long as possible which is beneficial as product engineering changes impact on the specification and build of the machines. Cost and time savings can be made if the machine can be built later when the product design has matured.

The next phase is the try-out where end users' witness team travels to the machine builder's site for evaluating the to-be commissioned system. Presently,

the whole system cannot be validated as there is large number of components and no tool practically exist supporting runtime evaluation of every component. Furthermore, the cost incurred for the end users in travelling and subsistence is significantly huge. At this phase, there is no provision of remotely monitoring and controlling production machines within Ford production programme to reduce or to avoid unnecessary costs incurred in visiting machine builder's site.

After this visit, the machine builder ships the machine to the end user's production facility. The machine builder's engineers come to the end users site to install and test the machine (i.e. install and commission phase in the figure 2.5). Simultaneously, the HMI system is implemented for the first time and any required support becomes the responsibility of the controls vendor. It is not until the machine has been installed and commissioned at the end user's site that the machine operators can be trained on the HMI system usage. This significantly delays and sometimes compromises the machine operator training process. It is often in the initial ramp up periods of a machine's production and operation when this training occurs, generally reducing the efficiency of the production machine. Once the machine has been installed and fully tested, the end user is ready to produce the first car engine (i.e. Job1 also known as the machine lead time). At this phase, production rates are monitored to ensure that the machine is ramping up satisfactorily and system reliability targets are being met. Although end users are trained to maintain these machines, it is a very common practice of consulting machine builders when machine problems are not readily resolved. Existing machine maintenance process is problematic and better opportunities are available to provide a cost-effective maintenance solution (described in the next section of this chapter).

Due to the translation required between the process engineers, who define the requirements, and the system engineers, who implement the systems, quite a few problems can often occur, for example;

- Process engineers cannot change or modify the control logic in installed systems easily on incremental basis at a later date without involving a highly specialist skilled system engineer who is familiar with the control logic and

HMI software. This is because the HMI system is tightly coupled with the machine control logic.

- The usability and suitability of the operation of the new HMI system remains largely unknown to end user until the physical machine has been built, controls have been wired up, the HMI software implemented and the system is finally tested. This is because the lifecycle is sequential and thus the HMI software is immediately developed after the control logic has been designed and programmed.
- HMI system is not integrated with production machine's process and control logic; therefore often the same information is entered many times into different systems through copy-paste coding techniques. In some cases, end users have reported that the assigned machine component names in the control logic and the HMI are not consistent which consumes more time and leads to confusions, when operating and troubleshooting a machine system.

The manner in which production machine is currently engineered is a traditional top down approach that does not support simultaneous engineering, is not reversible, uses proprietary techniques and often errors cannot be found until the system is completely installed. Moreover, commissioning and maintenance requires experts owing to the complex hard-coded programming and vendor specific technology implementation. These factors raise data interoperability issues, contributes to the implementation complexity and performance degradation, and increases process ramp up time [82]. According to Haq [11], 20 million Euros can be saved in a typical European production line if the ramp up is done twice as fast. Any machine reconfiguration activity needs additional programming efforts at the HMI side. Figure 2.6 (a) describes the existing sequential process of integrating control logic with the HMI system in a machine lifecycle, and any subsequent modification requirement accommodation. Figure 2.6 (b) shows the required process of integrating control logic with the HMI system within the machine lifecycle. It can be clearly seen that the required process is much more efficient than the existing approach to integration and reconfiguration.

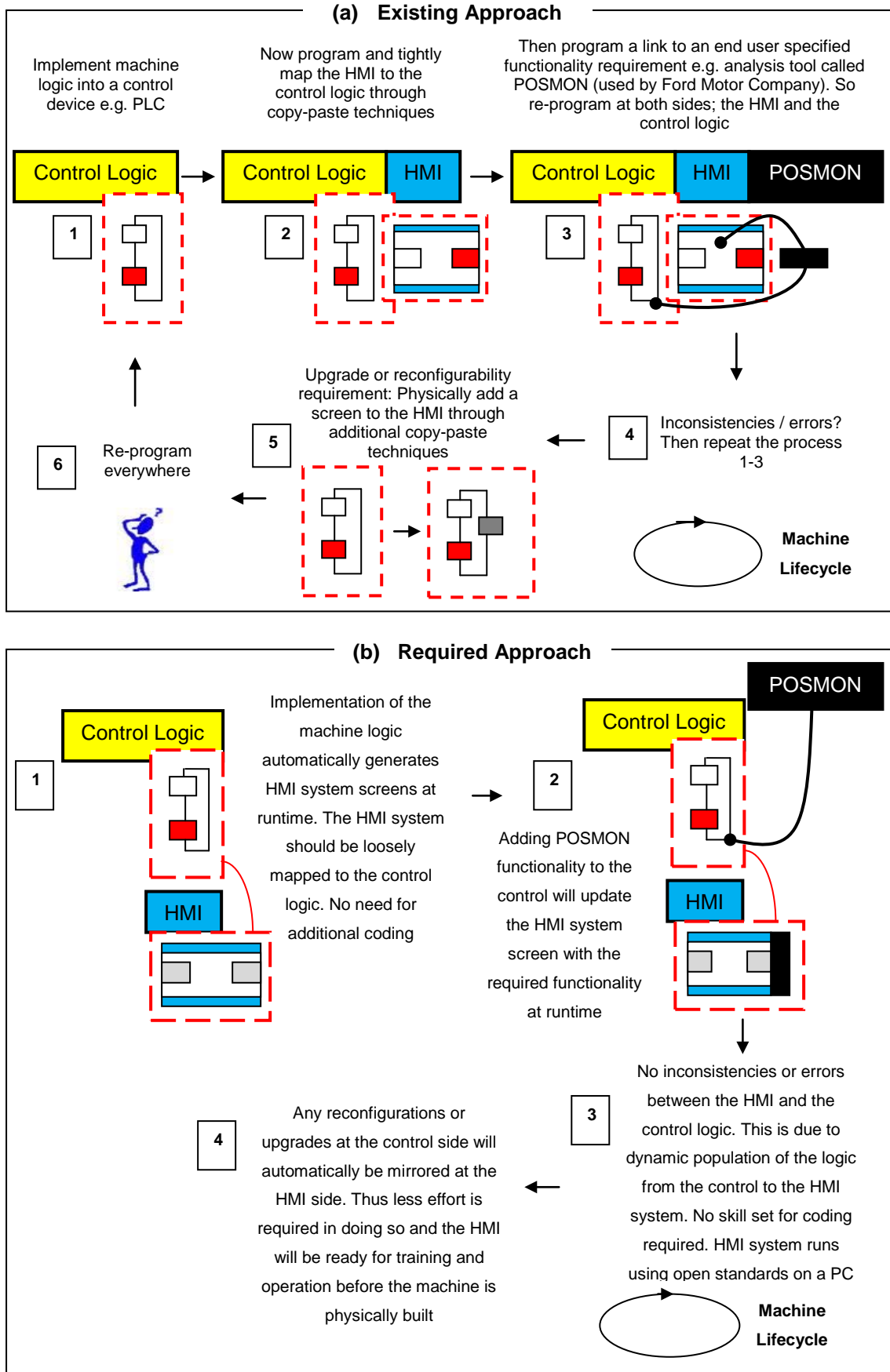


Figure 2-6: Comparison of Existing and Required Approach to Control Logic - HMI Integration

Summarising the illustration shown in the figure 2.5, HMI is only utilised from the commission phase onwards but there is a need and potential for extending the scope of its usage to support additional key machine lifecycle phases such as design, build, try-out and reconfiguration.

2.3.4 Operator Interface Supporting Machine Maintenance

A maintenance process is usually regarded as an expensive activity, which never create profits [91]. Machine maintenance is the most efficient way of keeping its functional and physical level as per the business requirements and environmental regulations. A well-maintained production machine is more likely to produce quality products; where as poorly maintained machine may lead to regular equipment failures, delayed production schedules and lower availability [92]. While maintenance can be planned, most of the machines are usually repaired when unexpected changes occur, owing to the limitations in trained manpower and production running costs. Any unexpected changes at the shop-floor level such as introduction of a new process or product, detection of missing parts or breaking down of the machine causes a ripple effect to the entire production and maintenance budget [28]. Any downtimes caused by such changes need to be reduced to keep the overall manufacturing cost as lower as possible.

As production has become more complex and globalised, it is becoming more difficult to maintain machines cost-effectively. Due to geographical distribution of supply-chain partners and manufacturing activities, and competition among vendors to market their products quickly, Mean Time To Repair (MTTR) the machines need to dramatically decrease. In automotive sector, maintenance of production machine is usually carried out by vehicle manufacturers (i.e. end users) with the help from the machine builders. An existing machine maintenance procedure between an end user and a machine builder is depicted as a typical scenario in the figure 2.7 [79, 80]. Drawing from this typical maintenance process, when an end user (for example, Ford Motor Company) identifies any problem with a machine (for example, an Oil pan rundown machine as described in the chapter 8.3) operating in a production line (for

example, Fox program's Craiova line as described in the chapter 8.4), their maintenance team will attempt to establish root cause of the problem and solutions for it using a vendor-specific operator interface. If the problem cannot be solved or it is completely new, then the end user seeks the machine builder's help (for example, ThyssenKrupp Krause). This consultation process is carried out using telephone and email conversations since the end user's machine operator interface (designed and implemented by the machine builder) cannot be remotely accessed by the machine builder's maintenance team. In most of the cases, machine builder's maintenance engineers rely on the archived machine records (usually paper-based representation) of the machine's control logic (e.g. written in ladder logic language). In doing so, machine builder's engineer tries to understand the remote situation based on verbal descriptions provided by the end user's engineer, and generate a possible solution.

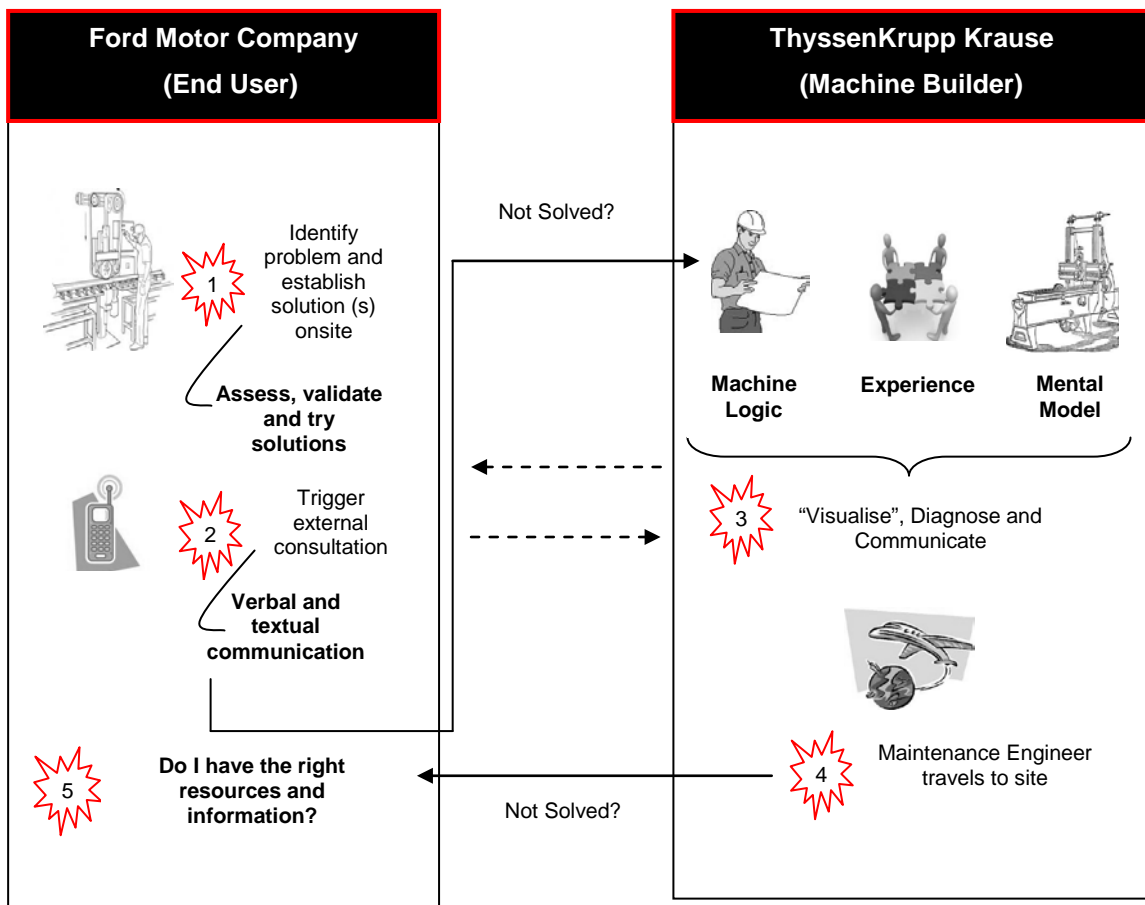


Figure 2-7: Existing Machine Maintenance Procedure

The engineers, relying on their personal experiences in the area of problem diagnostics coupled with the available information (i.e. archived machine logic and email/telephone messages); try to form mental models of the machine by “visualising” the problems. A great deal of time is required in establishing the current state of the machine. The problem solving process involves suggestion of various probable solutions to on-site maintenance engineers to attempt on the faulty machine. This process of recovering from a problem by means of verbal instructions is highly problematic. If the problem cannot be solved, then the machine builder’s maintenance engineer(s) have to travel to the end user’s production plant, which is costly in terms of resources. Sometimes, maintenance engineer(s) do not have accurate problem description beforehand (due to lack of visualisation) when they arrive onsite, leading to having inappropriate tools to solve the actual problem. In this case, further valuable time is wasted waiting for the required resources to reach the site. In some cases, machine builders negotiate contracting a control’s vendor engineer at end user’s site for certain time for troubleshooting control issues. This negotiation costs lots of money to the machine builders.

HMI Requirements for Remote Control and Monitoring Support

There is a need for maintenance teams to visualise remote machines at real-time by accessing machine’s operator interface to focus on analysing the cause of the problem accurately rather than attempting to establish the current state of a machine, in order to reduce its downtime. As identified from these research materials [51, 93], every minute of delay or malfunctioning in a machine production line, costs up to 6000 Euros for the end users. With the increasing complexity of production lines and strong market competitions, maintenance teams (and machine builders) are under a lot of pressure to ensure high machine availability and productivity throughout its lifecycle [94].

In response to these issues, remote maintenance is a solution which can provide high-quality, cost-effective and quick response service [95]. This refers to maintenance, repair and diagnosis of the machine over a spatial distance,

and through ICT, to enable real-time assessment of its performance and reliability. In order to provide such a service, engineering tools are needed that can closely monitor its performance and effectively control it remotely, while documenting its behaviour for best practices [79, 96]. Some attempts have been made by machine builders to remotely logon to the end user's machine; however, they are only able to access the status of the machines' control logic, under tight supervision of the onsite maintenance team. They do not have access to end user's operator interface with integrated three-dimensional simulation of the machine's control logic and physical representation of the machine in real-time at their home site [80].

Feedback from machine builders and end users demonstrates the importance of remotely accessing operator interface with integrated virtual representation and emulation of the machine at real-time through its lifecycle [29, 79, 80, 83]. Machine simulation would provide a facility to evaluate the physical machine behaviour against current operation of the machine's control logic, consequently reducing the effort involved in machine diagnosis. Furthermore, since machines are required to be designed and built with reconfigurability in mind, stakeholders are looking for remote engineering solutions which support reconfigurability, can provide cost-effective maintenance capabilities, and meet the challenges of globalisation, security, safety and environmental regulations [1, 96, 97]. Current engineering tools and solutions used by supply-chain partners do not facilitate these functionalities [12]. There is a strong need to utilise and implement opportunities provided by the ICT (especially web-based technologies) in suitable shop-floor system architecture to support the requirements identified in this research. Figure 2.8 presents the required procedure to maintaining production machines and compares it with the existing process as described in the figure 2.7.

In summary, having an integrated remote support solution within a machine's operator interface will be beneficial to all the partners involved in the lifecycle of a machine, as it would support its design, build, try-out, commissioning, maintenance and reconfiguration processes as described in the section 2.3.3. Firstly, the machine builder will be able to assign man power accordingly when

delivering a machine and training maintenance engineers on their home site using operator interface. This will improve both the cost and time efficiency for the machine builder. Secondly, this facility will save maintenance engineer's travel time when troubleshooting machine problems. Even if they travel to the end user's site, these engineers will be able to go with a detailed knowledge of the problems. Thirdly, all the involved parties will be able to remotely monitor the machine status (throughout its lifecycle) to be able to evaluate its performance (for example, when an actuator wears out), plan periodical repairs and predict machine degradation, ensuring that machine maintenance is scheduled accordingly. Such a service will enable end users to reduce their inventory of components and acquire them if and when required.

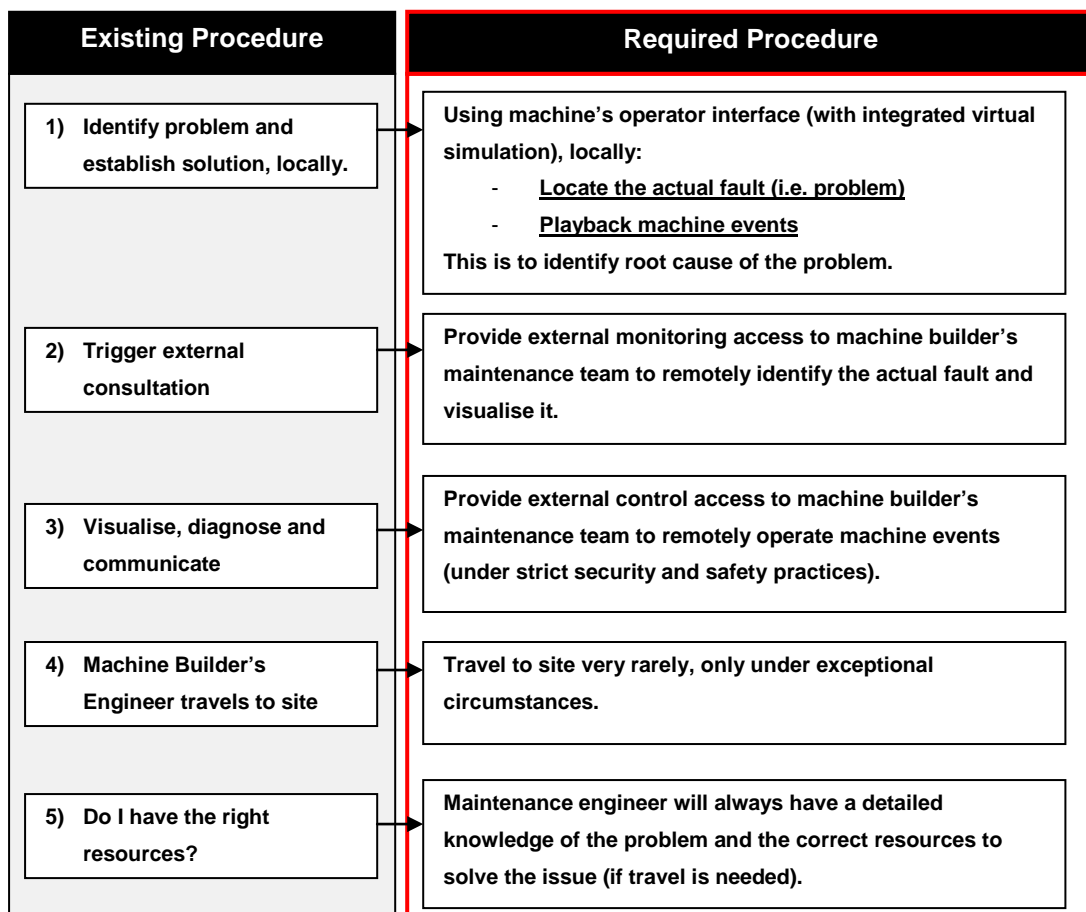


Figure 2-8: Required Machine Maintenance Procedure

2.3.5 Operator Interface Roles in Control and Monitoring Machines

With manufacturing trends focusing on the global markets, application of remote control and monitoring support functionality throughout the machine lifecycle (as described in the section 2.3.3 and 2.3.4) involves a number of user roles. Figure 2.9 illustrates various user roles participating in providing local / remote control and monitoring support in a typical production machine lifecycle [74, 79, 80, 82, 83, 87, 96]. Figure 2.9 has been divided into three parts namely; A, B and C. Part A shows the actual setup for a typical production machine line implemented at either an end user's shop-floor site or a machine builder's commissioning site. Part B corresponds to the local machine monitoring and control functionality implementation where as part C presents the remote aspect of monitoring and control support.

In this setup, the role of "Value-Added Service" is to mediate the interaction between the local site and the remote site using any communication mechanism (for example, Ethernet). Furthermore, it may perform complex data analysis and manages access security. In implementation practice, these functions may typically be carried out in a server or a network of servers. Users connect to the servers through the Internet or a company Intranet; however, the data content and the communication character may be different based on the user location / profile. That is the reason to show two "clouds of communication" in the figure 2.9. The roles of various users can be briefly described as follows:

- (a) **Local Machine Operator:** A local machine operator currently monitors and controls a machine using a proprietary HMI panel usually located in a centralised control room or close to the actual machine. His / Her access proximity is local to the machine system after its implementation at the end user's site. The required approach is to support similar tasks but using vendor-independent, cost-effective and portable HMI system.
- (b) **Remote Machine Operator:** A remote machine operator has tasks similar to a local operator but with remote access proximity enabling him / her to access system without any geographical constraints. In existing implementation, there is no provision of this role owing to the limitations

explained in the lifecycle description (section 2.3.3 and 2.3.4) of this thesis. The required approach is to enable a remote machine operator role to monitor and control a production machine, firstly during its build and commissioning at the machine builders site to speed up the operator training process, and secondly to remotely view a machine's operational status at runtime (i.e. after its implementation at an end user's site).

- (c) End User Diagnostic Engineer:** An end user diagnostic engineer is the first point of contact for local machine diagnostics, should a machine fails. His / Her access is usually local to the machine when it is fully implemented at the end user's shop-floor. He / She may use the same type of HMI screens as the local / remote machine operator. It is not practical to remotely support this role since there is a need for end user diagnostic engineer to always be physically available at a local site.
- (d) Machine Builder Diagnostic Engineer:** Since a machine builder supplies the production machine, they are responsible to provide an after-sales service to an end user by supporting the maintenance activities, if the end user's maintenance team are unable to troubleshoot any faults in the machine system. It is the responsibility of the machine builder diagnostic engineer to step into this role (in some situations this role may be sub-contracted to controls vendor engineer). In the current practice, his / her access proximity is local to the machine which forces unnecessary travel and associated costs (if the engineer is not locally available). Furthermore, if machine builder sub-contract this work to a controls vendor engineer (who is locally based on end user's premises), he / she may not be familiar with programming issues which may arise when troubleshooting problems during maintenance. The requirement is to use generic auto-generated HMI screens throughout the lifecycle so that the HMI system can be remotely shared for overall efficiency in the maintenance process.

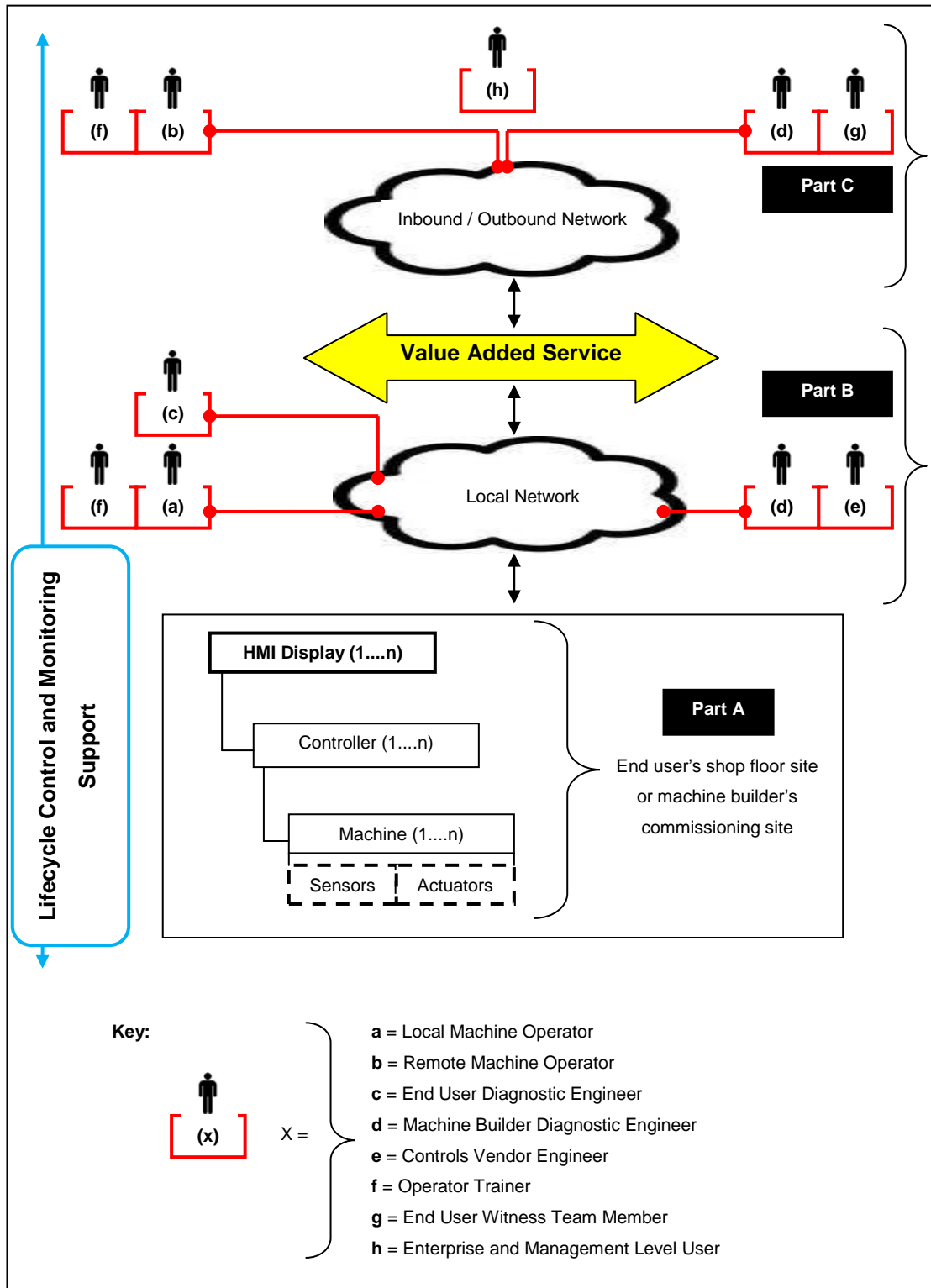


Figure 2-9: User Roles Providing Control and Monitoring Support in Machine Lifecycle

- (e) **Controls Vendor Engineer:** Currently, a controls vendor engineer usually supports the HMI system (which is tightly mapped to the corresponding machine control logic and installed in control vendor's HMI

panels) best suited for the application and plant location. The required approach is to have a loose coupling between the control and the HMI system such that HMI can be installed on PC's using standard open technology. Access proximity for this role is local to the machine.

- (f) **Operator Trainer:** An operator trainer is responsible for training machine operators using HMI system to perform daily control and monitoring activities. Currently, their access proximity is local to the machine such that they train operators after the machine has been physically implemented at an end user's shop-floor. The required approach is to provide them remote access proximity to enable them to train operators on the machine prior to the machine build and commissioning phase.
- (g) **End User Witness Team Member:** End user witness team member evaluates a machine by travelling to a machine builder's site during its build and commissioning phase. Their access proximity is local as they test machines by physically travelling to the machine location. It is a requirement to provide this role remote access proximity enabling one to evaluate machine's operational status using the HMI without incurring additional time and costs in travelling and subsistence.
- (h) **Enterprise and Management Level User:** The primary interest of this user role is targeted towards asset management and production planning; hence they do not directly access remote monitoring and control functionality using HMI screens throughout the machine lifecycle; however, they do indirectly analyse the collected operational information for strategic business actions. This is why they are not linked to any of the access levels in the figure 2.9.

2.4 Manufacturing Review Analysis

In this chapter, the major limitations and requirements from the manufacturing aspect of automation research have been derived from examining the current

literature. As identified and discussed, the demands for a change in manufacturing practices have been driven by factors such as globalisation, environmental concerns and ICT opportunities. To address global pressures web-based technological opportunities are further pushing the operator interface system's access boundary to easily accommodate remote control and monitoring functionality within current shop-floor system architecture. Existing HMI-PLC control system architecture has a number of issues which clearly demonstrate that they contribute to the overall unnecessary costs and complexities experienced throughout machine lifecycle process; owing to driving factors like rigidity and closeness of the implemented solutions. A web services-based control description is practical, attractive and the way forward; however, owing to the SOAP packet overheads, system integrity issues may arise especially if web services define the entire system architecture.

This chapter also discussed various types of HMI user roles involved within machine lifecycle to identify their needs. Their immediate needs are to have a solution that supports generic, vendor-independent, portable and cost-effective HMI screens which are loosely coupled to the implemented control logic. The existing limitations and the need to address next-generation operator interface system requirements (with its associated benefits) have been summarised in the table 2.2 below. This summary justifies a research need to having an operator interface system, implemented within suitable system architecture, which can provide effective control and monitoring support to production machines throughout their lifecycles.

No	Research Requirement (s)	Major Benefit (s)
1	Faster machine design, build and ramp up or support for rapid machine reconfiguration.	<ul style="list-style-type: none"> • Rapid introduction of a new product hence shortens its lifecycle. • Saves time and costs associated with duplication of efforts.
2	Increase information transparency and data mobility across heterogeneous platform systems.	<ul style="list-style-type: none"> • Information sharing shortens the product lifecycle and encourages innovation. • Eases the machine maintenance process

		<p>through learning from past documented information (i.e. lessons learned), consequently, reducing the MTTR of machine systems.</p> <ul style="list-style-type: none"> Establishes relationship between supply-chain partners by integrating various applications of suppliers, machine tool builders and end users. Linking production with the business enterprise will enable utilisation and sharing of machine critical information which aims at reducing cost and improving the throughput of a production system. This will improve business collaboration and support, through effective data sharing, visualisation, control and monitoring system integration.
3	<p>Locally or remotely collect, disseminate and analyse production operational information at real-time, regardless of the machine control type or its geographical location.</p>	<ul style="list-style-type: none"> Prevents manufacturers from being locked into proprietary solutions that are costly and functionally limited. Improves system performance owing to lack of heavy vendor-dependent protocol implementations. Close, but flexible monitoring improves overall system productivity. Any development or modification doesn't heavily rely on the availability of experienced engineers.
4	<p>Implement a machine independent HMI solution which doesn't relate to the actual machine or its control logic.</p>	<ul style="list-style-type: none"> Usage of common HMI system screens throughout various machines (or even an entire engine production program). Reduces system implementation complexity as devices from different vendors can easily be integrated into the control and monitoring architecture. Saves efforts, time and costs associated with duplication of laborious activities.

5	Evaluating machine commissioning processes (at machine builder's site) in a virtual environment prior to the machine build.	<ul style="list-style-type: none"> • Verifies the control logic and the machine design prior to the physical machine manufacturing. • Enables the product design to mature thus delaying the actual hardware build process. This saves costs and time associated with frequent reconfigurations to the machine, to match the updated product design.
6	Early verification of the HMI system.	<ul style="list-style-type: none"> • Rapid machine design and quick ramp up to full production capacity.
7	Early training of the machine operators (before the actual commission) or transfer an operator directly from other machine with the common HMI.	<ul style="list-style-type: none"> • Efficient use of human resources. • Production begins immediately after the machine has been installed and commissioned at the end user's production plant.

Table 2.2: Manufacturing - HMI Requirements Summary

Chapter 3 : Research Context and Focus

Chapter Contribution to this Thesis:

This chapter concentrates on reviewing a number of research approaches to obtain some background knowledge in order to provide design and focus direction to this research. The major contribution is to highlight the aim and novelty of this research.

3.1 External and Internal Automation Research

A large amount of research has been carried out in the academia and in industry to address the requirements summarised in the chapter 2.4. The domains of these researches surround distributed machine system, the component based design approach, business-process application integration and the development of engineering tools. In this section, some of these research approaches are reviewed to identify their achievements and obtain some background knowledge in developing the next-generation operator interface system solution that supports previously identified research requirements.

3.1.1 Research Centre at the University of Michigan

Engineering Research Centre at the University of Michigan has implemented the concept of reconfigurable manufacturing system. Their implementation consists of both real and virtual machines that are controlled over a communication network and coordinated through the systems unified software architecture [98]. Figure 3.1 illustrates the implemented primary components as described below [99]:

- **Hardware Testbed:** It consists of two manufacturing machines linked through a conveyor system. These machines operate in parallel and the conveyor carries pallets which transport the parts to be processed from one machine to the next.

- Virtual Factory: Factory simulation software simulates both the real machine and virtual parts that do not exist in the actual hardware. The virtual factory is controlled in the same manner as the actual hardware.
- Database and Middleware: Data-centric software infrastructure connects all the aspects of the machine system to provide rapid prototyping, integration, and transfer of newly-developed software systems.
- Remote Viewing and Collaboration Tools: These tools are implemented in the form of vendor-specific web-based HMI's that provide an operator with detailed information about the machine's status. Internet-based implementation enables remote access of the HMI screens. External partners user interface (i.e. HMI) tools are adopted and integrated within the RMS approach. While machine components are implemented with reconfigurability in mind, this approach is not paralleled when implementing operator interface systems as they are adopted from external partners having vendor-specific solutions.

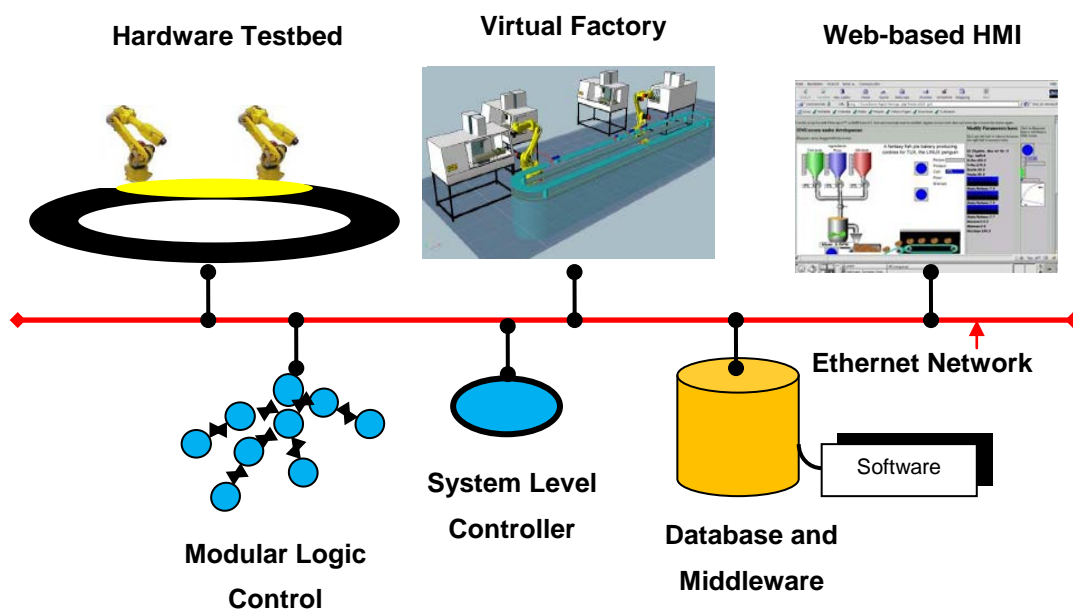


Figure 3-1: RMS Implementation at University of Michigan

3.1.2 Rockwell Automation

Rockwell Automation has focused on developing a flexible and reconfigurable distributed platform which supports plug-and-play automation systems based on agent-based technologies. The agent-based approach is implemented with real-time control agents, and information transfer between agents is implemented on PLC's Logix™ control (i.e. its flagship product). The controls interface for the agents can store and share data using tags. Outside resources access these tags using an OPC communication bridge and their proprietary Java-based interface. The existing industrial visualisation solutions based on operator interface panels also operate with these PLC tag values using their proprietarily implemented solution [100, 101]. As identified in [102], drawbacks of tag-based representation requires re-compiling and reloading of the HMI application when reconfiguration process is undertaken.

In terms of HMI system, shop-floor operations can be monitored over the web using their FactoryTalk™ studio proprietary package. While this enables configuring and runtime of web-based HMI that monitors machines having consistent screens and navigational support, it is not integrated with the machine lifecycle process in terms of providing early operator training and machine validation. Furthermore, this product is tightly integrated with Rockwell's Logix control platform, ultimately limiting and locking its users to its flagship products [103].

3.1.3 ITEA SIRENA and SOCRADES

ITEA SIRENA was an award winning collaborative project with Schneider Electric. It proposed a novel approach of using web services, based on a SOA standard, to create an open, flexible and agile environment with plug-and-play connectivity at the device level. It applied the XML-based web services paradigm for interconnecting distributed components through the use of Ethernet TCP/IP, which demonstrated the possibility of a universal, platform, and language-neutral connectivity among various shop-floor components. ITEA SIRENA proposed the idea of building advanced functionality, embedded into

devices, to enable new distributed application paradigms based on self-reliant smart devices [42, 104].

The results of ITEA SIRENA were used as foundation for the SOCRADES project. This project's aim was to develop new methodologies, technologies and tools for modelling, design, implementation and operation of networked hardware and software systems in industrial automation by exploring SOA at both, the device level as well as the application level as shown in the figure 3.2 [27, 87]. Initial exploitation of SOA-based web services was carried out using distributed control devices called FTB (designed by Schneider Electric) in the car engine manufacturing domain (using the Component Based automation approach described in the section 3.2 of this chapter) through the use of an orchestration engine (further information in the chapter 8.2.5). These FTB devices were programmed to communicate using web services. The future map of SOCRADES is to implement SOA-based web services in traditional PLC-based control devices or completely replace these conventional controllers with fully-distributed high-efficient control devices [27, 87].

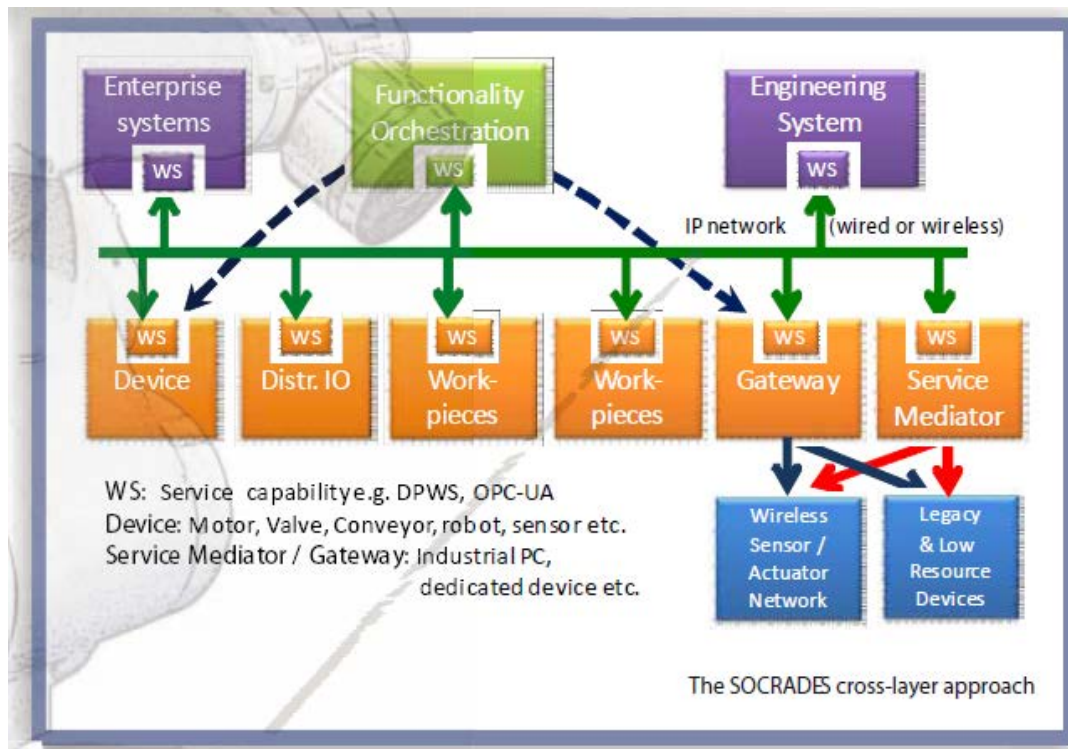


Figure 3-2: SOCRADES Approach

Implementing a SOA web services-based system within a PLC environment can lead to problems in its real-time control capabilities due to excessive slowness of the SOCRADES technologies. These technologies represent an increase in the communication flow between controllers programmed using web services. Owing to the openness of web services standard (compared to the proprietary closeness of the PLC's), currently, it provides a heavy communication mechanism at shop-floor levels. HMI system implementation along the lines of SOCRADES technologies would be very beneficial for having a truly open support framework across the entire supply-chain, however, to successfully adopt and implement a web service model throughout vertical levels of plant support would need further protocol compression mechanisms which are currently being addressed [27].

3.1.4 Other Miscellaneous Research in the Academia

This section reviews examples of some academic researches currently undertaken within the manufacturing industry. These research approaches are organised into their respective contribution sections to identify the gap in the existing manufacturing automation domain.

Remote Control, Monitoring and Maintenance of Machines using Open Web Standards

When reviewing development of remote control, monitoring and maintenance of manufacturing machines, open standards (to a certain extent through web technologies) are usually employed. A number of examples of successful applications and researches can be found in the literature and in industry. Among them, Muto [105] presents @factory XML system that provides remote machine surveillance for information display, video camera monitoring and data analysis for machines. Sahin [47] implemented an approach to remotely control and monitor DOPC (Distributed OPC) system where data from multiple controllers are propagated through the internet in its pure I/O format. Shi [106] has developed a remote monitoring system for diagnosing faults using an

expert system implemented on the standard web architecture (through ActiveX control mechanism for performance improvisation). Kirubashankar [107] proposes a remote monitoring system for a manufacturing plant where security of data transfer is managed via VPN (Virtual Private Network) where as Campos [108] provides an extensive review of application of ICT in the field of remote monitoring and maintenance of systems. These systems are not integrated with lifecycle engineering process, thus their usefulness can only be realised after a machine (and its associated operator interface system) has been physically implemented at the end user's site.

Early Training and Machine Validation using 3D Representation

In the manufacturing industry, training is usually realised on the job where experienced operator or the machine builder engineers train novice operators after machine commissioning phase using the implemented HMI. New machines are more likely to have process faults in the early phases and operators need to be trained for these scenarios. In fact, training is insufficiently carried out in some instances [109]. Pantforder [110] evaluates a process to training machine operators using historical records with 3D representation integrated within the HMI. While this approach enables early training (i.e. through commissioning phases), it is solely based on combination of old machine transactions with scenarios which may not reflect the new machine specification, for example, the sequencing and interlocking logic are unique to each machine implementation therefore operators need to be trained on the actual machine that is going to be implemented for them to understand its operation and any recovery practice. Moreover, there is neither a simulation process nor integration with the lifecycle engineering process equipped with real-time data connectivity in the current practices (which is a must for next-generation HMI [111]). This approach requires machine data from a comparable machine programme to be available for training the operators, which may be difficult to obtain.

3.1.5 MSI Research Institute

The MSI Research Institute at Loughborough University has focused on the lifecycle support of distributed automation systems by replacing the traditionally

centralised PLC control solution with distributed vendor-independent solution using a component based (CB) design approach (discussed in the section 3.2), where the control functionality is embedded into the components [59]. The CB design has been evaluated using industrial system case studies to create the design of generic and modular device components, and to determine industrial feedback regarding its performance and capabilities [51]. This distributed system implementation has been conceived as a key approach towards an agile manufacturing system.

The work carried out within this research domain has been usually funded by the United Kingdom's Engineering and Physical Research Council (EPSRC), Innovative Manufacturing and Construction Research Centre (IMCRC) and European grants, and spans numerous projects such as COMPAG [85], BDA [112] and SOCRADES [87]. The major industrial collaborators within these projects have been Ford Motor Company, ThyssenKrupp Krause, Schneider Electric and Bosch-Rexroth. The ultimate aim of these projects has always been to research, develop and implement next-generation engineering tools that support powertrain manufacturing machines throughout their lifecycles.

3.2 Component Based (CB) Automation

3.2.1 General Description

To increase flexibility and reconfigurability in manufacturing systems, modularity is typically introduced as an agile enabler as discussed in the chapter 2.2.1. Modularity concept can be found in many related manufacturing areas such as RMS. A number of research approaches have developed modular manufacturing production techniques, specifically focusing on rapidly adaptable and reusable machine systems that support their lifecycle needs [11, 51, 113-115]. Implementing modular techniques in industrial control engineering enables dissection of the automation system solution into a set of mechatronic modules (also known as components) such that if pre-developed and pre-validated, can be reused within a system with reduced efforts as long as their interface specification is agreed upon [59, 116]. In the field of software engineering, this

concept is usually referred to as Component Based (CB) development which structures a solution around components and their interfaces.

A component is a piece of software which is self-contained and reusable in the design of large distributed system solution. The construction of any system can be undertaken through integrating various components using their well-defined interfaces, which may contain services, attributes, events and times to show what the component can deliver [116, 117]. Using CB is advantageous since systems can be developed faster within reasonable budget, and can provide better usability and encapsulation of best practices. In powertrain manufacturing, CB approach has been researched and implemented through a novel engineering method which supports lifecycle requirements of manufacturing machines, as described next.

3.2.2 CB Application

With respect to machine lifecycle support in the field of manufacturing automation, research at MSI Institute (Loughborough University) has investigated (through the COMPAG project [85]) and implemented a CB automation framework to engineering industrial machine's control (in the SOCRADES project [27, 82]) using modular mechatronic devices (i.e. components) [118]. This automation framework aims to replace the existing PLC-based (or a PC-based) system architecture (described in the section 2.3.2) by developing control systems using components which reside within component libraries. CB automation approach promotes control engineering through configuration data represented in a uniform format, which ultimately becomes the application logic of a control system, rather than through programming the application code using any of the IEC 61131 / 61499 languages (section 2.3.2) for example, ladder diagram [59].

The basic idea of this approach is that new control system is composed from components that have already been developed, tested, validated and implemented in the past using an engineering environment (explained later in this section). Any new system development process requires developing and

testing only new components while using majority (almost 70%) of the pre-built and pre-tested components from the library to rapidly construct a system. This reduces the costs and efforts in developing a system. Furthermore, this reduces the time to market a new product to meet customer demands quickly [11, 29, 59, 119].

A simplified representation of this approach is shown in the figure 3.3, where a “system” corresponds to a complete production machine which consists of one or more “components”. A component in CB automation hierarchy is any input or output device (for example, sensor, actuator or a complex drive) which can be configured with a unique finite state machine that defines its “control logic” within a system. The lowest granularity level for this approach is a component which can be configured to operate in new circumstances using an engineering environment (described next), investigated at the author’s premises [1, 29, 54, 59, 120, 121].

Major elements that facilitate implementation of CB automation approach are:

- An engineering environment: This consists of an engineering toolset that can be used by globally distributed supply-chain partners throughout the entire machine lifecycle. This toolset (called Core Component Editor (CCE) [122]) supports the production process planning, machine design and sequencing, interlocking logic and provides simulation tools to validate behaviour of the entire machine system prior to its actual build. The toolset enables creation of a new component (or reconfiguration of an existing one), simulation of its behaviour and storage of it in a library for future re (use). Therefore, a new machine system can readily be constructed through configuring, combining and installing these components to drive a physical (i.e. a real) or a simulated machine.
- A common machine data model which consistently stores all the machine information avoiding its fragmentation across different systems (such as machine control system and support system such as operator interface). This acts as a central repository housing all the configurations that are shared throughout a typical machine lifecycle.

- Machine components which can be either real or modelled. Modelled components are simulated using 3D VRML representation. Real machine components are physical modular machine parts (such as sensors and actuators) containing embedded interlocking and sequencing logic.

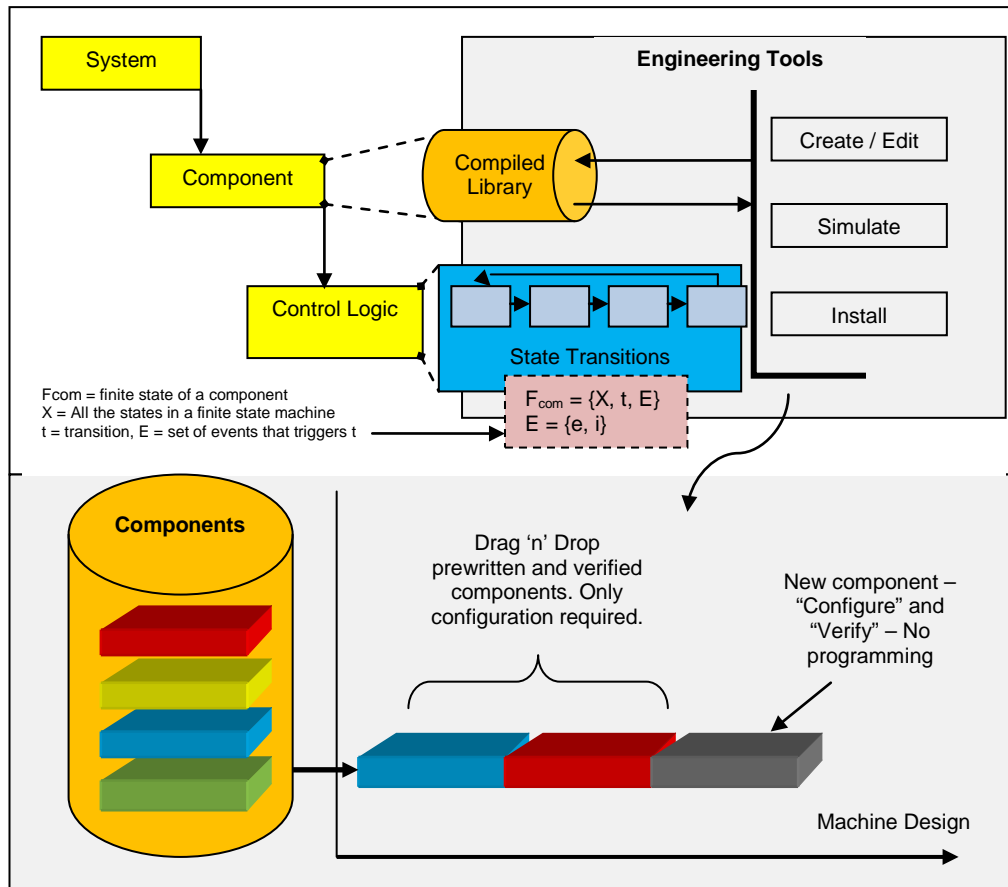


Figure 3-3: Component Based Automation Approach

- Runtime support environment that facilitates close control and monitoring of machines through their lifecycles using vendor-independent operator interface system templates, reconfigured at run-time to present real-time machine status, regardless of machine's geographic location, or its implementation type or its state (i.e. real, simulated or hybrid machine). This integration aspect of the CB automation approach is the main focus of research covered within this thesis. Figure 3.4 shows author's research involvement with respect to contributing towards CB automation implementation.

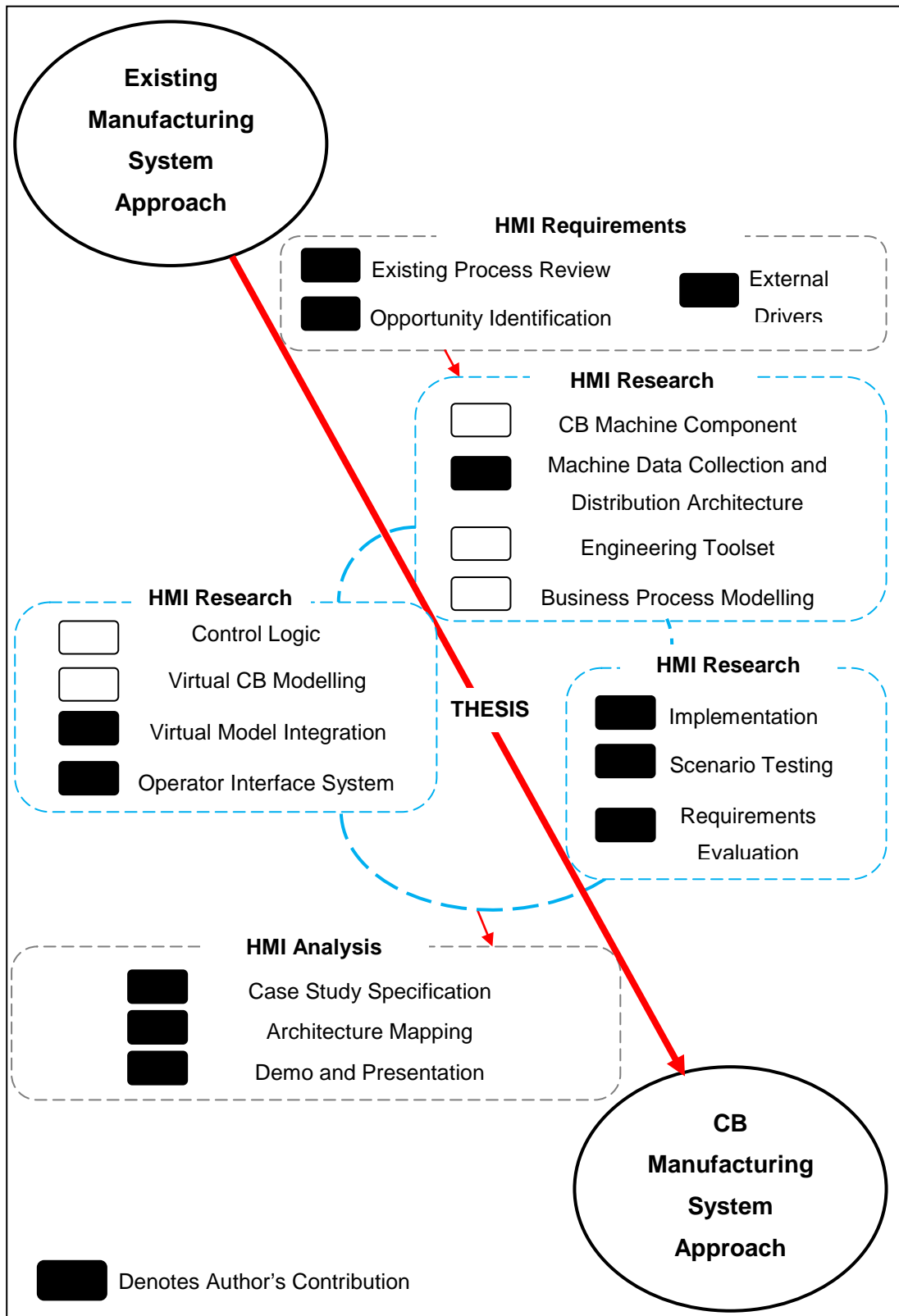


Figure 3-4: Author's Involvement in CB Automation Implementation

3.3 Focused Attributes

All the major requirements identified from the state-of-the-art in powertrain manufacturing and operator interface system's review (chapter 2) have presented with a set of desired attributes which drive a need to provide an operator interface system that supports them. These focused attributes have been summarised in the figure 3.5. Each attribute has one or more "implementation feature (s)" associated with the approach employed to satisfy identified attributes. Description of these features is as follows:

- (i) Reconfigurability and Reuse Support: This attribute requires both, industrial machines and their associated operator interface systems to be reconfigurable and reusable across various machine programmes and throughout their lifecycles. Automation approach to accomplishing this identified attribute is through modularisation of control components using CB automation approach (described in the section 3.2). Furthermore, developing generic reconfigurable operator interface screens (chapter 7.5), implementing the overall system within a suitable control and monitoring system architecture (chapter 5.3), and support for real and virtual machines are additional essential implementation features.

- (ii) Information Transparency and Mobility: This attribute requires addressing a number of things. Firstly, scalability is required by allowing machine changes to be made more efficiently by simply adding or removing machine components (for example, sensors) at will without causing any disruptions to data control and monitoring applications. Secondly, the research solution must accommodate different types of support systems regardless of their source vendors thus avoiding any licensing issues. Thirdly, legacy machine system support should be incorporated within the information exchange architecture. Automation approach to accomplishing this identified attribute is through better business to production information system integration using an open (vendor-independent) and distributed system architecture implementation that supports "plug-and-play" components integration functionality.

Furthermore, representing data in a uniform format (such as an XML standard) and separation of machine control and monitoring functionality are essential implementation features for data transparency and its mobility (chapter 5.3.2).

- (iii) Loose Mapping of HMI to Actual Machine or its Control Logic: This attribute requires the HMI to be de-coupled from the machine type or its control logic such that any changes to the machine (or its logic), or the entire machine replacement should not require any re-programming to be carried out to the operator interface systems. Automation approach to accomplishing this identified attribute is through modularisation of control components using CB automation approach. Furthermore, developing generic operator interface template-based screens which are simple, cost-effective and machine-neutral (or control logic-neutral), and implementing the overall system within a suitable control and monitoring system architecture that supports this attribute requirement are additional essential implementation features.

- (iv) Real-time Remote Machine Control, Monitoring and Maintenance: This attribute requires operator interface systems to display machine information at real-time on the HMI screens regardless of their geographical locations. Furthermore, operator interface systems should support machine maintenance at real-time. Implementation features such as web-based technologies (i.e. web-based HMI), suitable control and monitoring system architecture, 3D machine representation support and various non-functional system requirements (evaluated in the chapter 9) such as security, performance, reliability and safety are essential features addressing this attribute.

- (v) Virtual Machine Validation: This attribute requires operator interface systems to support machine validation in a virtual environment such that, the HMI system can be driven from the virtual machine model to verify its behaviour prior to the actual build. Automation approach to

accomplishing this identified attribute is through modularisation of control components using CB automation approach. Furthermore, integration of 3D VRML simulation functionality and suitable system architecture are additional essential implementation features.

- (vi) Early HMI Verification: This attribute requires HMI system to be verifiable before the actual machine has been built. Automation approach to accomplishing this identified attribute is through modularisation of control components using CB automation approach. Furthermore, integration of 3D simulation functionality and suitable system architecture are additional essential implementation features.

- (vii) Early HMI Training: This attribute requires HMI system to enable operator training prior to the actual machine build. Automation approach to accomplishing this identified attribute is through modularisation of control components using CB automation approach. Furthermore, implementing the overall system within a suitable control and monitoring system architecture, and support for real and virtual machines are additional essential implementation features.

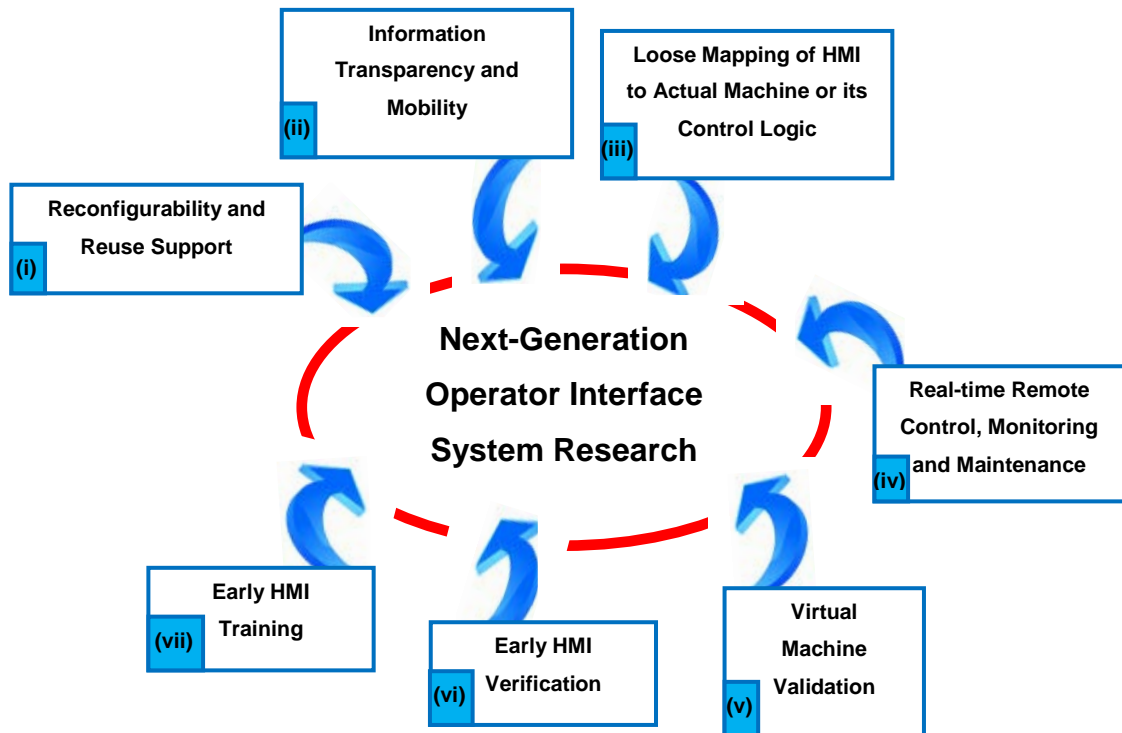


Figure 3-5: Focused Attributes Addressed within this Research

3.4 Research Aim and Novel Contributions

3.4.1 Aim

The primary aim of this research is to design, develop and evaluate next-generation operator interface solution that can be applied to locally as well as remotely control and monitor machine lifecycle phases. The research outcome has to be integrated to the Component Based automation approach for agile automation systems to allow different classes of users through the supply-chain, to efficiently interact with the manufacturing systems.

3.4.2 Novel Contributions

To accomplish the primary research goal as identified in the section 3.4.1, major contributions established during the course of this research would be:

- Review of the current approaches to operator interface system implementation and its utilisation in the context of lifecycle support of powertrain machines, determining limitations and inadequacies in supporting next-generation industrial requirements.
- A novel system architecture that provides new ways of using operator interface system for control and monitoring of machine lifecycle phases, and incorporates all functional and non-functional industrial requirements within a well-defined framework (i.e. CB automation approach).
- Description of the design and development of the operator interface system solution for implementing this research to powertrain manufacturing machines engineered using the CB automation approach.
- Evaluation of this research in an attempt to demonstrate a proof-of-concept system, identifying its benefits and any significant improvements when adopting CB operator interfaces system solution.

Chapter 4 : Enabling Technologies and Methods

Chapter Contribution to this Thesis:

The main contribution of this chapter is to identify various technological opportunities which can effectively support local / remote control and monitoring of manufacturing machines through their lifecycle phases using state-of-the-art methods in operator interface system design.

4.1 General Overview

This chapter reviews the current state-of-the-art in HMI system engineering and implementation. Moreover, it describes the enabling technological opportunities for remote support and next-generation operator interface system design and development, which can satisfy the identified manufacturing requirements established in the chapter 2 of this thesis, providing a platform onto which this research can be materialised.

Although web-based operator interface system is considered as a key enabler for reconfigurable manufacturing systems (as in chapter 2, table 2.1), no published literature within the RMS domain has concentrated on the engineering aspect of the operator interface system that addresses the need of HMI user roles to support machine lifecycle requirements. Furthermore, with frequent reconfigurations to the manufacturing machines, their associated operator interface systems have a requirement to be rapidly created and deployed. The next two sections focus on the modelling, engineering and implementation technology which is not necessarily dedicated to only manufacturing machine applications.

4.2 User Interface Modelling and Engineering

Over the last decade, a substantial amount of research has been carried out in the area of user interface systems, addressing various aspects of interface modelling and system engineering.

Interface Modelling

User interface modelling provides a notation which represents an end user’s functionality and describes the user-machine interaction process [123]. This process is determined by the machine behaviour, user’s operational goals and expectations, and the actual interface system [124]. Modelling allows the designer to think about the underlying concepts of the interface design instead of just focusing on the appearance of the interface [125], for example, a machine operator may require certain types of machine error information, current machine state and interaction with the mode control of the machine. This information must be comprehensively specified using models so an end user interface can be developed. A well documented specification using various modelling notations is therefore required to ensure consistency in the end user requirements being met in the design and implementation of the user interface system. Different aspects of user interface are described using different declarative models [126-128]. Table 4.1 presents four major types of declarative models with a superficial description of each.

Declarative Model	Description
Task Model	This model describes how users do their tasks in a certain application, as well as the relationship between various tasks in user interface interaction [127]. Producing a usable interface requires a thorough understanding of the underlying user goals [128]. Task analysis is used to determine a task model as it contributes to the design of the interface interaction by understanding the relevant tasks in a system domain using techniques such as interviews, observations, documentation and training [129]. In software engineering, task analysis is useful to gathering user requirements where as task model is useful in the design and evaluation of a system. Traditionally, interfaces are ineffectively started by describing the static visual screens (i.e. interfaces) with certain structural or control objects often called widgets [130]. Designers must think the description in functionality context by starting to think about the user’s task and subtasks which can better support the idea of user-centred design. If

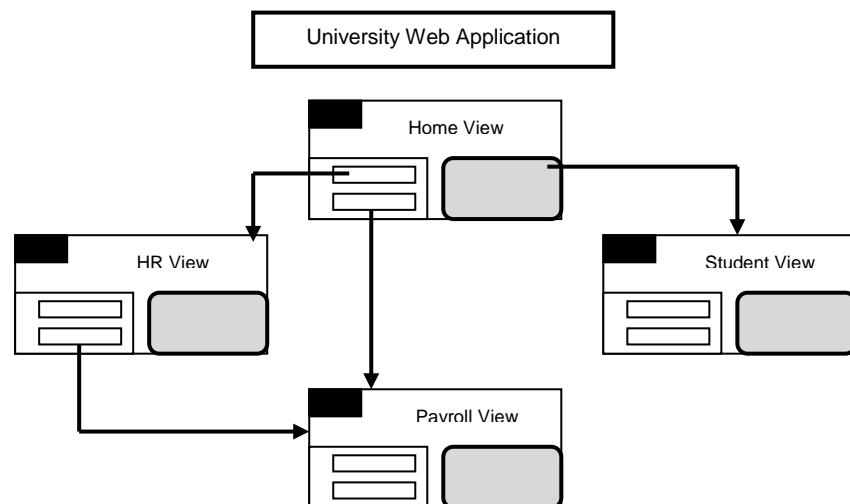
	<p>the flow of the tasks and its subtasks is clear, it is easier to choose the appropriate visual layout and widgets to provide a user with concrete tools to complete a task [131].</p> <p>For describing the requirements which satisfy the needs of the user, use cases are commonly used [132]. One use case can call upon the services of another use case achieving a hierarchical order to system design. The following example shows a website use case for project management within an organisation [133]. The use case contains 2 users, an external user and an employee. The external user's task and its interaction are just to read about projects. The employee can read, add, remove and edit projects. This example illustrates that use cases can be used to describe the user tasks and its interaction but it does not capture the user's task flow.</p> <div data-bbox="582 851 1308 1288" data-label="Diagram"> <pre> graph LR subgraph PM [Project Management] R([Read]) A([Add]) Re([Remove]) E([Edit]) end EU[External User] --- R Emp[Employee] --- R Emp --- A Emp --- Re Emp --- E </pre> </div> <p>Similarly, use cases have been used within this research to capture low-level operator-interface system requirements as described in the chapter 5.2.2.</p>
<p>Dialogue Model</p>	<p>This model describes the particular objects within a user interface and their possible states (i.e. sequence or flow) [134]. It represents the actions that the user may initiate through the interface, as well as the reactions that the application may execute. In simple terms, the description depicted by a task model is refined by a dialogue model by specifying the behaviour of the user interface in terms of user interactions and system feedback [135]. State transition diagrams are commonly used to capture the sequence of either a complete or particular aspect of a user interface system [125]. These diagrams model objects as finite state machines. They show the various states of an object, the conditions that trigger the transition from one state to the other, and the actions that result from this transition, as shown below. They only provide a model of the user input (action) and the result (interface states), and do not</p>

	<p>provide any information regarding the visualisation or the layout of the user interface.</p> <div data-bbox="630 331 1109 745" data-label="Diagram"> <pre> graph TD subgraph Object_A [Object "A"] S1[State 1] S2[State 2] Sn[State n] S1 --> S2 S2 -.-> Sn end Cond[Condition(s)] --> S1 S1 --> Act[Action(s)] </pre> </div> <p>Within this research, state transition diagrams are used to model various object's states such as navigational buttons, component state activations, etc.</p>
<p>User Model</p>	<p>This model describes the characteristics of various types of users. Its purpose is to create personalised interfaces at design time [136]. Firstly, for each type of user, it defines a set of tasks he/she can perform. Secondly, for each type of user, a projection on the actions within a concrete task that he/she can perform is established. Finally, depending on a user's attributes (skill level, experience); the interaction information provided by a dialogue model to show the information in the domain model is adapted to the user [137]. In summary, it provides an approach to model user interface preferences by defining attributes and roles of specific users of an interface system. User models are described vaguely in the available literature and are present in very few user interface implementations [123, 127].</p> <p>This technique is usually preferred in profiling user interfaces and thus is not directly applicable to HMI system research covered in this thesis, however, various user role requirements (as described in the chapter 2.3.5) are catered for using user modelling to generate one HMI system layout (as described in the chapter 5.2 and chapter 7.5).</p>
<p>Presentation Model</p>	<p>This model represents the visual layout and navigational structure provided to the user by the interface [128]. It is basically a static collection of elements with attached stylistic attributes such as font size, colours [138]. Software modelling techniques such as Unified Modelling Language (UML) notation seems to have no support to specifying the layout of user interface systems [139],</p>

although researchers at Ludwig-Maximilians-University (Munich) have proposed using collaboration diagrams for modelling the navigation and presentation of the user interfaces [140]. In designing a web application, a web designer usually proposes a sketch of each interface view which shows rough drawings of a couple of relevant elements of each navigational object. Although this informal sketching technique is frequently used by designers, it does not have a precise notation of representing the layouts and its navigational links [140].

Storyboarding, on the other hand, is used to provide users with a “walkthrough” of the interface system by creating a series of screens and widgets, and depicting the screen navigation using arrows. A storyboard model only defines the structural organisation of the presentation given by interface objects such as texts, images, forms and menus, and not the layout characteristics, in terms of fonts, special formats, colours which are determined at the implementation phase of the user interface development. However, it may still be able to provide a hint on the position and the size of the interface objects relative to each other [141-143].

After designing the interface views they are combined using navigational links in the storyboarding model, showing sequences of user interface in the order in which a user can navigate from one view to the next. This helps in visualising the organisation of a web application structure in a more intuitive manner than using standard UML notations. Furthermore, storyboard models provide a useful means for communicating between stakeholders involved in the lifecycle of a user interface system, enabling them to be validated with the use cases identified during the analysis phase of the interface development lifecycle [142, 143]. An example storyboard model is illustrated below.



	Within this research, storyboarding technique has been used to deduce navigational structure of HMI system screens as described in the chapter 5.2.2.
--	-------------------------------------------------------------------------------------------------------------------------------------------------------

Table 4.1: User Interface Modelling Components

Interface Engineering

The main goal of engineering a user interface is to make a user's interaction experience with a machine as enjoyable as possible, in terms of satisfying identified requirements. A well engineered user interface system increases the productivity of the user and the machine. Furthermore, it increases the system uptime and provides a consistent product quality output. Therefore, it is critical to engineer interfaces that take the “usability” criteria into account when coordinating user interface development [144]. A number of engineering methods have emerged trying to contribute new ideas in this field. Mayhew [145] proposes a “Usability Engineering Lifecycle” structured in three stages namely requirements analysis, design/testing/development and installation. This development process follows the waterfall lifecycle and thus cannot be considered as an iterative approach. Owing to the sequential development approach, this method focuses on user interface software development and not user centred design [137].

Constantine and Lockwood [146] propose a user centred design approach which consists of a set of usability-oriented coordinated activities. They include task analysis and user modelling (as described in the interface modelling section of this chapter). The fundamental concept of applying user centred design to user interface engineering is that it is an iterative process where the tasks and user requirements are defined, analysed, implemented in the form of the user interface and then evaluated. Once evaluated through usability testing, user reactions can be feedback into the user requirements and domain analysis. Feedback is performed on each development process by introducing the information collected in the tests performed by the users in the user interface usability testing [137]. This way, the usability of the designed

interfaces improves notably [147]. Figure 4.1 shows the iterative development process which is fundamental to user centred design approaches [137].

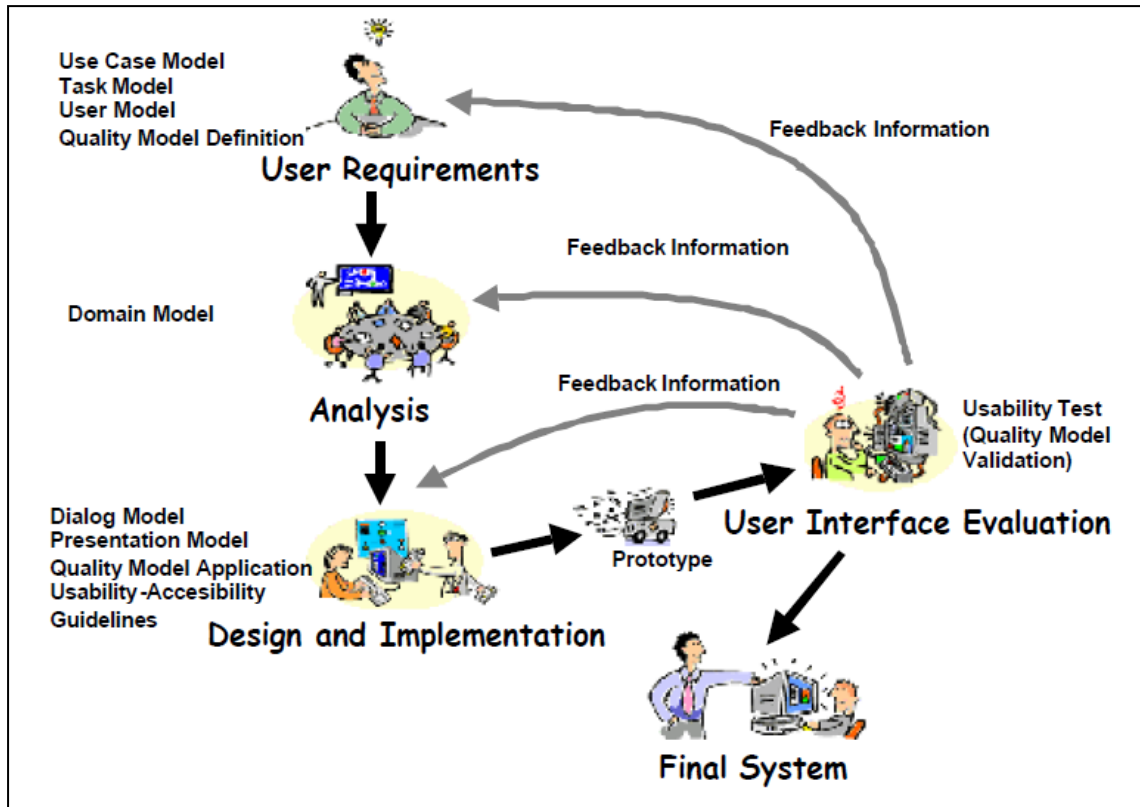


Figure 4-1: Iterative User Interface Design Approach

Taken from [137, 148]

4.3 User Interface Implementation

Obtaining a desired interface system requires the use of appropriate implementation method which can transform an interface model into a piece of software system that meets an end user's functional and non-functional expectations [149]. In this section, brief description of different tools used for implementing user interface is covered. These tools are primarily used to support the interface design, implementation, evaluation and maintenance processes. The two main approaches to implementing a user interface system are shown in the figure 4.2 below:

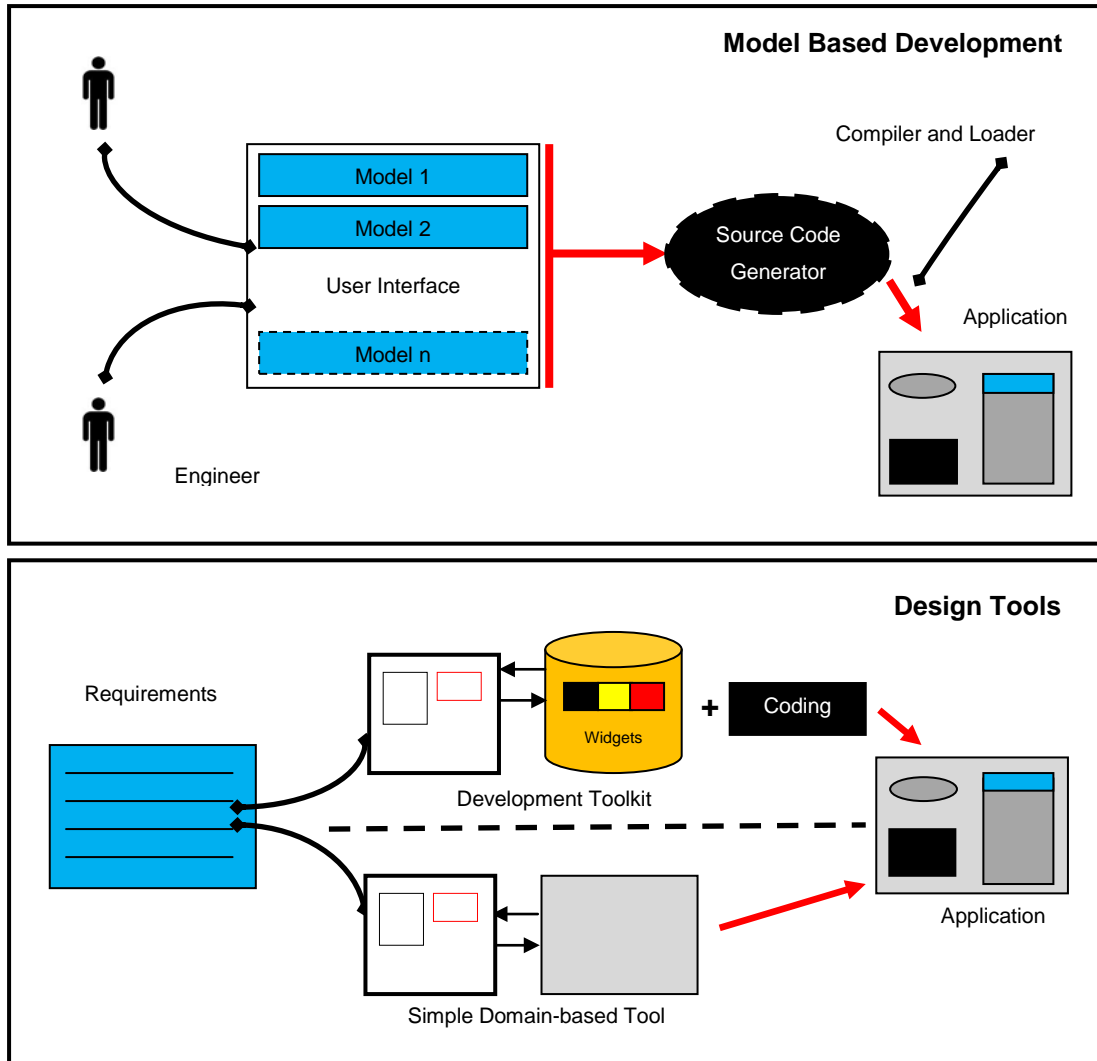


Figure 4-2: User Interface Implementation Tools

Model Based Development Approach

Using this approach, a user interface is automatically generated from a series of user interface models [116]. The process of developing a user interface within this approach is through iterative development and refinement of a set of declarative models using graphical editor tools or high-level specification languages. Once these models are developed, they are transformed according to the ergonomic rules and / or style guidelines into an interface specification. Consequently, this specification is coupled with the underlying application code to generate an executable application [150, 151]. Table 4.2 summarises major attributes and shortcomings associated with the model based user interface development approach [116, 123, 128, 144, 151].

Attributes	Shortcomings
<p>The higher level of abstraction for developing user interface allows a designer to concentrate on the design issues rather than low level system code generation.</p>	<p>With user interfaces increasing in functionality, the complexity of the models and their required notation is considered as a major limitation for these systems.</p>
<p>It is a well structured development method followed by a user interface designer. This strictly enforces best practices that the designer must follow, for example, applying ergonomic design rules and using formal notation to specify the interface ensures that there is no obscurity in the requirements.</p>	<p>It is difficult to model relationships between models (i.e. mapping problems) using this approach.</p>
<p>User interface system is quickly developed due to the system architecture automating the lower level implementation details. This enables a designer to develop user interfaces without relying on software engineers' input.</p>	<p>Mostly, this approach only supports a very simple user interface system, typically a form based interface system. This is the reason why it has not been widely adopted within the commercial sector.</p>
<p>The interface is always consistent with the requirements specified in the model due to the direct relationship the system provides between the user interface model and the implemented user interface.</p>	<p>The capacity of the user interface developed using this approach is often very limited and does not support a broad enough scope to be applied successfully in practical commercial areas.</p>

Table 4.2: Attributes and Shortcomings of Model Based User Interface Development

User Interface Design Tools

Schneiderman [147] defines user interface design tools as a software development environment that simplifies and assists implementation of a user interface by programming through graphical or textual based approaches. These developer-centred environments provide sufficient support in using and managing widgets, organising and arranging interface layouts, and evaluating

interfaces [128]. User interface design tools can further be divided into two different categories; user interface toolkits and user interface application tools.

- Toolkits

These provide software libraries containing common widgets such as windows, scroll bars, pull down menus, buttons, dialog boxes, etc. Though programming languages with these common widgets can provide great flexibility in the user interface design, skilled programmers are usually required to develop the user interface. The main advantage of using them is the provision of extensive control and flexibility in creating the required interface system [116, 147].

Development environments such as Microsoft's Visual Basic / C# are easy to get started with yet they have excellent set of features. Visual Studio[®], as well as Visual Web Developer tools, coupled with dot Net framework, provides a remarkable distributed object oriented application platform which is robust, easy to learn, cost-effective and performance driven. The other example of toolkit is the Java Development Toolkit[™], which supports implementation of Java based technologies in user interface implementations [147, 152]. These toolkits provide a general purpose programming environment to prototype and develop equivalent packages found in commercial HMI system design, thus used in this research (chapter 7).

With respect to designing and implementing HMI within automotive domain, commercial off-the-shelf (COTS) packages such as Siemens WinCC[®] and Schneider's Vijeo[®] products are available to support the process of designing and implementing operator interfaces; however, these COTS are proprietary, encrypted (i.e. machine control dependent), expensive (i.e. need licenses) and difficult to learn and troubleshoot without formal training. Furthermore, they support HMI implementation in vendor-specific operational panels only and not on standard computer monitors / touch screens [153-156].

Toolkits such as Visual Studio[®], is not comparable to the COTS in this regard as application engineers at manufacturing plants never use programming toolkits, instead they use COTS to develop operator interface systems. With the flexibility offered by these programming toolkits such as Visual Studio[®], HMI configuration tool (equivalent to COTS) can be developed and implemented in manufacturing machine programme to overcome shortcomings identified in the COTS packages.

- Application Tools

Application tools provide more constraints than toolkits. Using these design tools, user interfaces are developed for a particular application area which maybe for example; a web page developed using Adobe Dreamweaver[™] or Microsoft Expression Web[™], or the National Instruments Labview[™] where a user interface is designed for an engineering application.

This type of user interface application tool provides faster development due to the constraints of developing the user interface for a particular application area. Often they are What You See Is What You Get (W.Y.S.I.W.Y.G) based and very little or no programming is required (generally scripting level languages are used) which supports lower skilled designers and leads to faster development time.

4.4 Real-Time Machine Data Transmission Options and Issues

As established in the chapter 2, remote control and monitoring support is an essential requirement when addressing existing machine lifecycle issues, this instigates a need to transmit data from manufacturing machines to locally / remotely implemented operator interface systems. This section reviews various approaches available to transmitting data for real-time control and monitoring of production machine's processes. Furthermore, it describes number of issues that needs to be addressed when propagating information at real-time to remote locations. In this thesis, the term "real-time" corresponds to soft real-time (or event-based having no strict time deadlines).

Transmission Options

In manufacturing environment, large amount of data is continuously generated from operating production machines, which can either be transmitted in “raw form” to drive remotely implemented engineering tools (for example HMI), or through accessing locally implemented HMI screens using “remote sharing sessions”.

Raw Data Transmission: This transmission corresponds to distributing unprocessed data directly from an operational machine to remote locations. Any processing on this data is carried out remotely using supply-chain partner’s engineering tools or third-party resources. This is more efficient approach to transmitting data over a network as only raw machine status is actually sent enabling direct data exchange between systems in (soft) real-time, beneficial for high-performance and flexible applications. Any resource, for example, 3D visualisation tool or a HMI system at machine builder’s site can be deployed and operated remotely (with additional functionality and performance this approach provides) [157]. Nonetheless, this approach to transmission requires relaxing security of an existing network infrastructure to tunnel raw data from local site to remote locations. A typical example for this would be to open up additional firewall ports, which is dangerous to organisational security if not carefully thought. Implementing this approach can be outsourced; however, organisations prefer to carry out its in-house implementation.

Screen Sharing Sessions: This corresponds to a technology which enables a user interface screen to be shared with collaborating partners in a remote session. Each participant (for example, a machine builder and / or a controls vendor) can remotely view the local screens (for example, an end user HMI screen) and possibly control it (if given necessary permissions to do so by the source initiator, in this case the end user). This technology is also known as desktop sharing approach [158]. The process requires software to be loaded or accessed from the local point of termination. From a remote location, a user can navigate to a specific web address that functions as an access portal through which it is possible to use a set of login credentials to reach the local screen.

Screen sharing is a less efficient approach owing to the large overheads associated with distributing an entire graphical screen over a network compared to lightweight raw machine data. Furthermore, it requires local execution and support of engineering tools such as 3D visualisation and other applications. Nevertheless, this approach requires a standard internet browser at remote locations to view (and possible control) local screens. This is useful when working with limited resources remotely. This approach can be outsourced as well as developed in-house; however, most organisations prefer outsourcing owing to data transfer compromise safety delegated to a solution provider.

A number of desktop sharing solution providers exist in the market offering various features. A detailed review of providers is described in the appendix section B of this thesis [158-164]. One provider in particular, known as Cisco WebEx is a market leader offering impressive remote sharing performance thanks to its MediaTone network [165]. Furthermore, this provider offers integration API's (Application Programming Interfaces) where external hooks to its remote services can be established from any third-party system. In the light of this provision, a support tool called RemoteIMS (Issue Management System) is implemented at Ford Motor Company, UK that provides web-based real-time desktop sharing interface (using WebEx API calls) to collaborate on various machine issues using multimedia tools like SMS and email [166] as shown in the figure 4.3 below.

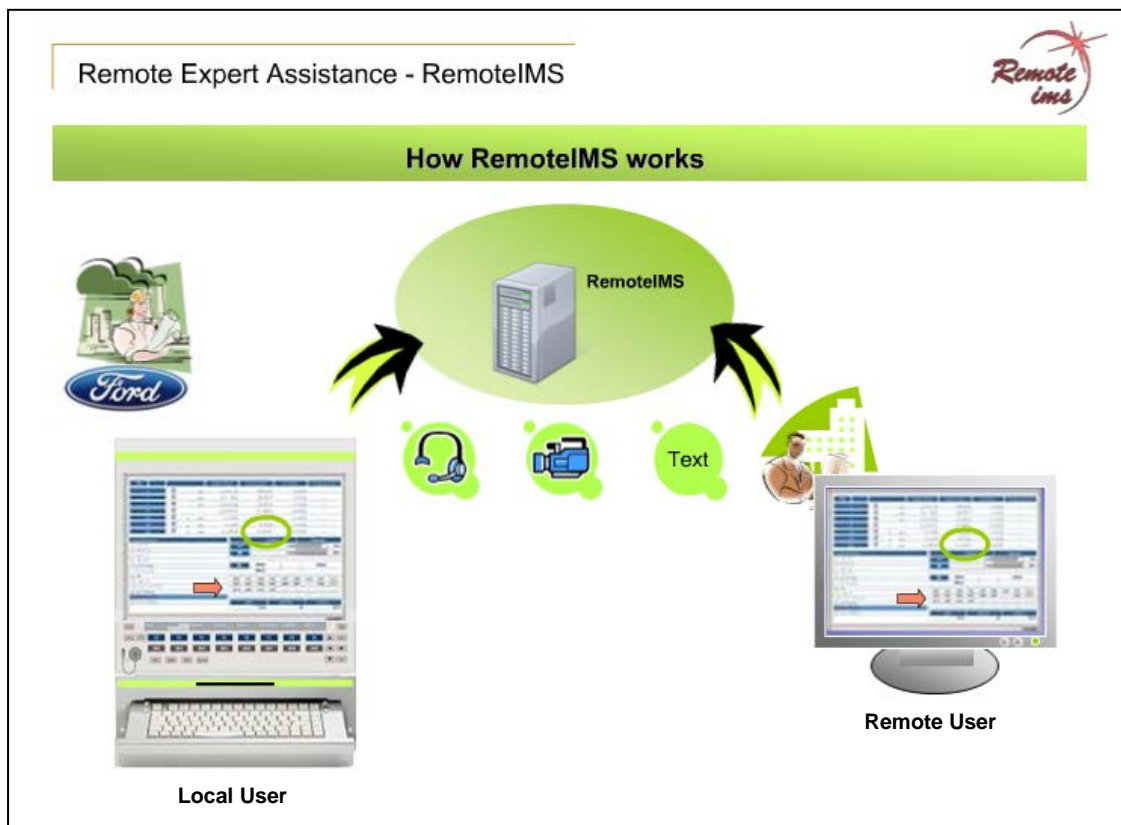


Figure 4-3: RemotelMS Tool Implementation at Ford Motor Company, U.K

Taken from [166]

Transmission Issues

Improved operator interfaces are required in the context of local production support and of remote support from supply-chain partners [1]. In order to transmit machine operational information to remote locations for driving or sharing with supply-chain partner's tools such as operator interface systems, visualisation tools or commercial support packages provided by third-party such as SAP business suite [167], a number of issues must be considered as discussed next (evaluation of these issues is covered in the chapter 9).

Security: With increase in web-based remote data transmission approaches being adopted by manufacturing automation partners, security of business information is the primary concern as it increases vulnerability of industrial processes [3, 19]. Concerns like data authenticity and confidentiality are part of this data transmission issue. Providing remote access to partners like machine

builder to controlling and monitoring end user's production plant using web-based HMI, needs implementation of good security strategies [79]. In industrial setups, mechanisms like firewalls (for example, Scalance S612 security switch [168]), secure protocols (for example; VPN, HTTPS over TCP/IP) and access controls (for example, username / password credentials, audit trails) are usually used to authenticate and manage incoming and outgoing traffic [169].

Safety: Transmitting real-time data to and from remote locations requires safety procedures to be strictly followed to protect against hazards [3, 170]. Enabling remote HMI to control industrial machines is a very risky issue. The control logic of a production machine needs to be intelligent enough to establish the current state of the machine and decide on the execution of control commands issued using remote operator interfaces. In a recent interview with Ford production engineering team, they stressed the importance of implementing control safety procedures and layers of access control within next-generation operator interface systems, for example only one instance of the HMI should remotely control its corresponding machine at a time to ensure safety of both the technological and human resource [79].

Robustness and Reliability: To address measurable issues like robustness and reliability, information transmission architecture should support features like failure recovery and data transmission handshakes [169]. Some of the frequently experienced evils of transmission affecting a system's reliability and robustness are packet losses and data corruptions, which can be resolved using connection-oriented protocols like TCP [171]. Handshaking using acknowledgement messages is a widely used approach to ensure the reliability of data transfer; however, these mechanisms add to the required network overhead and thus decrease the overall network performance [99]. It is obvious that the reliability is governed by different types of implemented hardware and networks, the designed software and the machine operator's input to the system [116].

Data Uniformity: Representing data in the format that promotes openness is a very important issue to be aware of. If data is encoded and presented to engineering partner resources in a proprietary format, which cannot be

deciphered or decoded by third-parties, it discourages lifecycle partner's involvement and complicates the machine support process. XML is a uniform de-facto transmission format that can be decoded by any system as long as interface specification is agreed and followed upon [42, 172] (see its evaluation in the chapter 8.2.5).

Performance: Achieving high-performance data transmission is vital to today's agile business requirements. It is greatly affected by network bandwidth and design choices made during system development, for example choices like data size and its representation [169, 173]. Features like data latency and system throughput are key measurement performance metrics for transmission systems [116]. Performance in propagating machine information within powertrain system domain is evaluated by comparing the implemented solution against an existing response time benchmark of half a second for local HMI and one second for a remote HMI [174].

Scalability: The architecture adopted for transmitting data should be scalable enough to support high volatility of engineering tools, as these tools join and leave the network on ad hoc basis [79, 80]. This is usually achieved through implementation of a flexible system architecture that supports simultaneous nodes' access on-the-fly.

4.5 Communication Mechanisms

4.5.1 Communication Queues

Implemented software components (i.e. distributed applications such as HMI) exchange machine data over industrial networks. With complexity of distributed system components increasing, careful attention should be given when communication queues (or channels) for information sharing among these network nodes are implemented. Four main categories of these logical pathways to communication have been identified such as point-to-point, publish / subscribe, guaranteed delivery and message bus.

Point-to-Point

This provides a dedicated one-to-one communication channel that links two systems or processes over a network. It ensures that only one receiver consumes any given message. If the channel has multiple receivers, only one of them can successfully consume a particular message. If multiple receivers try to consume a single message, the channel ensures that only one of them succeeds, so the receivers do not have to coordinate with each other as shown in the figure 4.4. The channel can still have multiple consumers to consume multiple messages concurrently, but only a single receiver consumes any one message. This is a simple approach that provides high-bandwidth but scales poorly with multiple nodes [175]. A typical example of such a communication model is implemented in file transfer applications where the request for information is directed from one sender to only one receiver at the other end [176]. Within this research, the Marshaller system component implements this channel to pass on the machine control tokens to various clients (as described in the chapter 7.4.4).

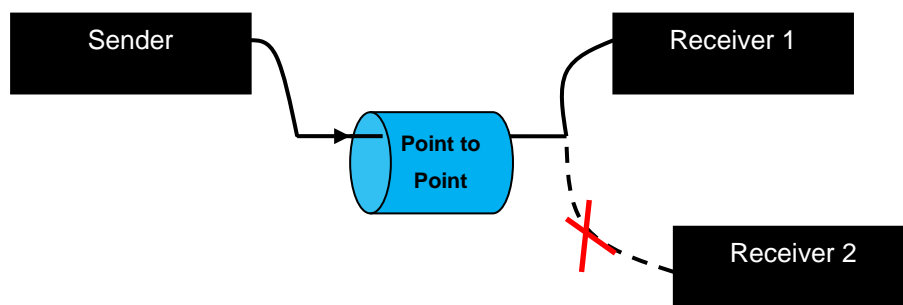


Figure 4-4: Point-to-Point Communication Queue System

Publish / Subscribe

This provides one-to-many, many-to-many or many-to-one communication channel where a source can transmit message to interested receiver (s) without the knowledge of the number or the location of receiver (s). Receivers (s)

specify their interests by subscribing to asynchronous notifications of events generated by the source (i.e. publisher). The publisher produces a message (i.e. event) which explicitly specifies its type and if it matches with the subscriber's interest, it receives the message without knowledge of what, if any, publishers there are. A broker acts an intermediary which forwards each message from the publisher to the interested subscriber as shown in the figure 4.5. Three variations of the Publish/Subscribe system one can use to create a mechanism that sends messages to all interested subscribers are List-Based Publish/Subscribe, Broadcast-Based Publish/Subscribe, and Content-Based Publish/Subscribe [116, 157, 177]. Broadcast-Based Publish/Subscribe is the adopted mechanism when propagating machine events within the Broadcaster system component (as described in the chapter 6 and chapter 7 of this thesis).

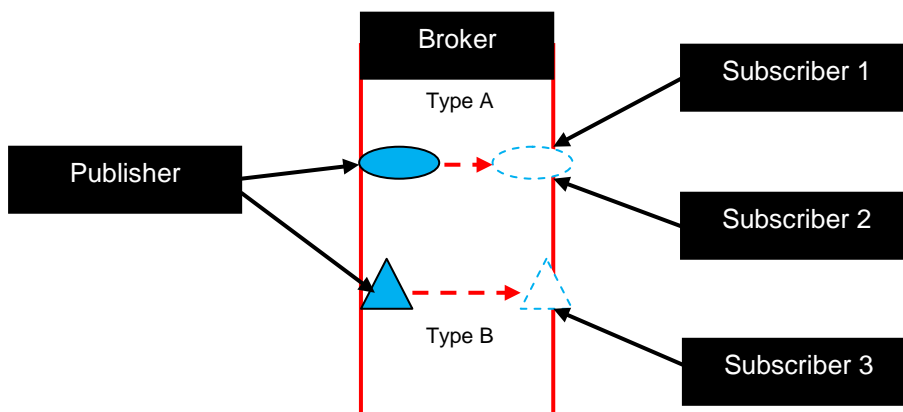


Figure 4-5: Publish / Subscribe Communication Queue System

Guaranteed Delivery

This provides a persistent approach to communication transfer by using a built-in data store to persist messages in each participant node on which the system has been implemented. The default system storage media (i.e. memory) works well as long as the queue works reliably, but if the system crashes, all the stored messages are lost from the memory. With guaranteed delivery, local storage disk space safely stores messages until they are successfully delivered. This queue system increases reliability, but at the expense of performance as it

involves considerable numbers of I/O and consumes a large amount of disk space [157, 177].

Message Bus

This provides an integration solution where applications from different vendors coexist and transfer messages asynchronously among themselves using a logical component called a message bus. An application that sends a message no longer has individual connections to all the other recipient applications that must receive the message. Instead, the sender merely passes the message to the message bus, and the message bus transports the message to all the other receivers that are listening for bus messages through a shared infrastructure such as a message router.

Likewise, an application that receives a message no longer obtains it directly from the sender; instead, it takes the message from the message bus as shown in the figure 4.6. An application that sends messages through the bus must prepare the messages so that the messages comply with the type of messages the bus expects (i.e. common data model and command messages). Similarly, an application that receives messages must be able to understand the message types. An application can be added or removed without affecting communication to other applications when using this queue system [116, 157, 176].

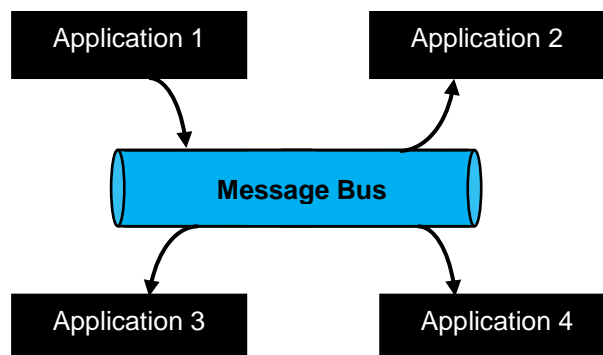
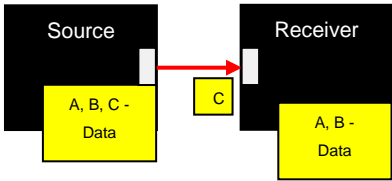


Figure 4-6: Message Bus Communication Queue System

4.5.2 System Interaction Styles

Nowadays, systems are developed and implemented in more and more distributed ways, scattered over an industrial network. Table 4.3 briefly describes two most common examples of interaction styles found in systems and automation technology from the functional and non-functional system requirements point of view [116, 178, 179]. This description does not make assumptions about application design or its implementation. Their design can be governed by any architectural pattern (described in the section 4.5.3) and their implementation can use any of the communication channels (described in the section 4.5.1).

Interaction Style	Description
<p data-bbox="331 949 663 1025">Continuous data distribution mechanism</p> 	<p data-bbox="753 949 1406 1532">A connection between a source and a receiver is “wired up” in this interaction style. The receiver always maintains an up-to-date copy of the data and the source can handle any number of receivers. An illusion of this wiring within a system implementation is created by making the source transmit fresh data cyclically to receiver(s). This is an efficient approach if small size data values change frequently. The other approach of implementing this mechanism is to transmit new data after an application has exceeded a specific threshold value (i.e. when the data value changes significantly). This approach is preferable if the data structures are large.</p> <p data-bbox="753 1592 1406 1989">The required frequency of data transfer depends on the process dynamics for example, in discrete automation many tasks have frequency periods of several seconds. Usually, multiple receivers of the same data are serviced by the data source using either data broadcasting or point-to-point transmissions. In this thesis, interaction between system components uses data broadcasting services (more information is described in the chapter 5 and 6).</p>

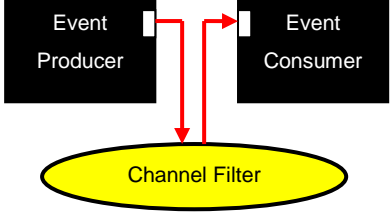
<p style="text-align: center;">Event-driven data distribution mechanism</p> 	<p>Data ports between a source and receiver(s) are wired up as in continuous data distribution style; however, the transmitted data is an event instead of a data message (or a signal). An incoming event usually triggers the execution of an algorithm associated with the input data port. This algorithm may usually process some input data which might be carried by the event message or wired to other input ports. Missing an event (due to intermittent network connections) may have serious consequences, in the worst case; a critical task would never be executed since it waits forever for an event that was lost in translation.</p> <p>An updated flavour of this mechanism implements event notification and acknowledgement style where events (in the form of alarms or critical messages) require handshaking at the application level. The major benefit of this flavour is that it de-couples the event producer (i.e. source) and the event consumer (i.e. receiver) by sending the events to a channel or a manager, from where the receiver(s) can retrieve or filter events as required. If the receiver is unavailable, the manager can store the event for future use (depending on design constraints). Sometimes, the channel can be substituted by a common data repository (i.e. shared memory area) from where notifications can be sent about new events. This is mostly applicable in repository pattern (see section 4.6.3).</p>
---------------------------------------------------------------------------------------------------------------------------------------------------------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Table 4.3: Examples of Common Interaction Styles

4.6 Architectural Patterns

Very few systems are designed totally from scratch; in general, systems are designed using a single or collection of architectural patterns. Architectural patterns provide the fundamental structural organization or schema for software systems. It is a way of presenting, sharing and reusing best practices about

software systems where rules and guidelines for designing subsystems and organising relationships between them are provided [180]. This section briefly covers relevant architectural patterns commonly used in different types of systems within this research implementation.

4.6.1 Model View Controller (MVC)

This pattern is the basis of managing interactions in many web-based applications [116]. It modularises a user interface of an application by separating the business logic (i.e. the domain logic), the presentation logic (i.e. the display for the user) and the actions based on the user input (i.e. the application logic) into three separate objects namely; model, view and controller respectively as shown in the figure 4.7. The “model” manages data and behaviour of the application domain, responds to requests for information about its state (usually from the view), and responds to instructions to change state (usually from the controller). The “view” manages the display of information by defining how the data is presented to the user where as the “controller” converts the actions from the user, informing the model and/or the view to change as appropriate [179, 181].

It is important to note that both the view and the controller depend on the model, in fact, the view is created from the model data. However, the model depends on neither the view nor the controller. This is one of the key benefits of the separation. This separation allows the model to be built and tested independent of the visual presentation resulting into better application management and faster development [116]. Web-HMI system component within this research has been designed using this pattern (illustrated in the chapter 7.5.2).

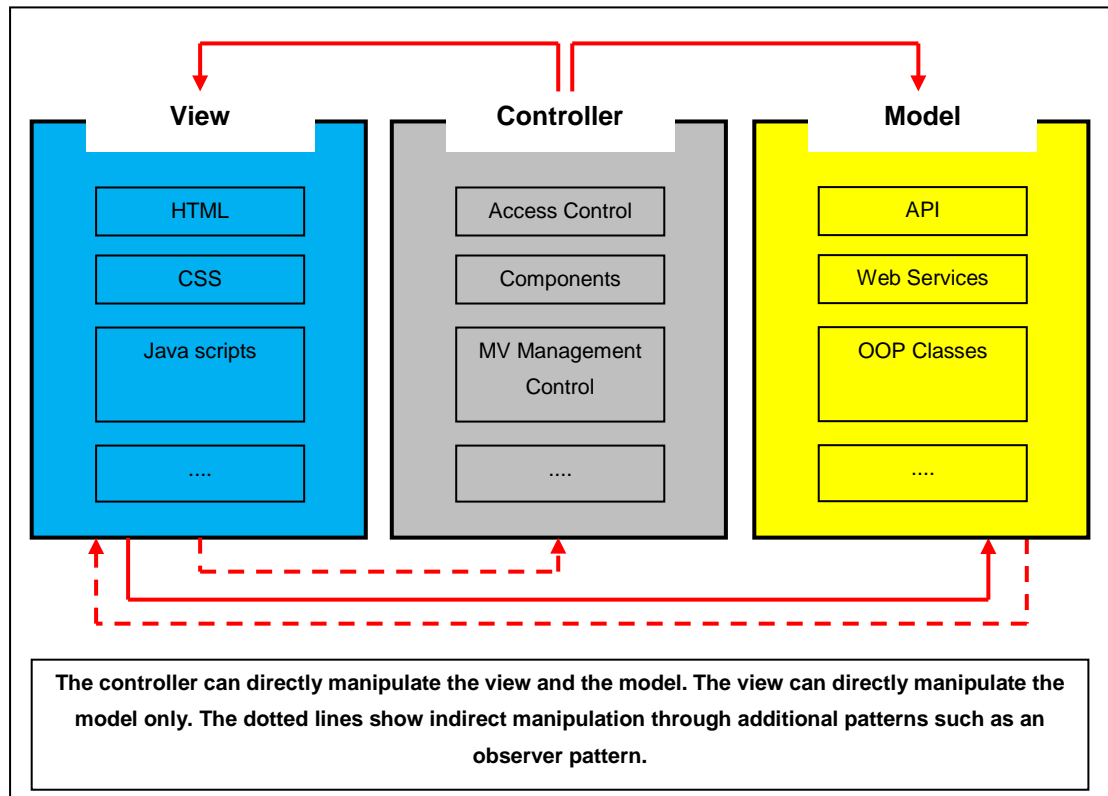


Figure 4-7: MVC Pattern

4.6.2 Layered Architecture

In this pattern, a system's functionality is organised into individual layers. Each of which relies on the facilities and services offered by the layer immediately beneath it [116]. This promotes incremental development such that once a layer has been developed; some of the services provided by that layer can be released to users. Furthermore, it is changeable and portable where a layer can be replaced by another, equivalent layer without any issues so long as the interface is unchanged. When layer interfaces change or new services are added to a layer, only the adjacent layer is affected and not the entire system. The most beneficial intent of this pattern is to divide a task into groups of subtasks such that each group of subtasks is at particular level of abstraction having its own set of responsibilities [180]. Within this research, layered architecture is used to organise the required security mechanism in the Web-HMI system component (chapter 9.3). As an example of this pattern implementation, OSI (Open Systems Interconnect) seven layer networking

model breaks the overall network communication task into a number of subtasks, each is carried out by an individual layer as shown in the figure 4.8 below [182].

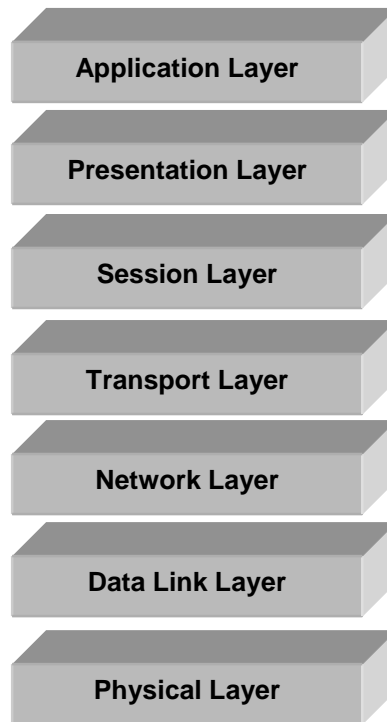


Figure 4-8: OSI Layer Architecture

4.6.3 Repository

The above two previously described patterns deal with presenting the conceptual organisation of a system but give no indication of how the system components could share data. Repository pattern describes how a set of interacting system components can access data in a shared fashion. This pattern is suitable to applications in which data is generated by one source and used by another. Majority of systems that use large amount of data are organised around a shared data source or a repository. An example of this system is a management information system. In a practical implementation, it is usually difficult to distribute a repository over a number of computer systems; however, it is possible to divide a logically centralised repository though data redundancy and inconsistency issues may arise in doing so. To overcome this,

data can be shared between systems using a controller mechanism which generates and propagates events to various components when data becomes available. Systems do not interact directly, only through the repository. This pattern should be applied to data-driven systems where the inclusion of data in the repository triggers an action or a tool. This mechanism of triggering components using a shared repository structure is also known as blackboard model (described in the chapter 6) [116, 179].

4.6.4 Client – server Model

A repository pattern does not demonstrate run-time organisation of a web-based system but only shows its static design structure [116]. In distributed applications, client – server model is a very commonly used pattern showing its run-time organisation and it is useful when implementing a data sharing system where the actual data has to be accessed from a range of locations. In this pattern, a set of clients with limited functionality, access a set of servers which manage data. In fact, this model is often a generic umbrella term for any application model that divides processing among two or more processes, often implemented on two or more computers. In a client – server model, a client (implemented as a program) initiates a communication session with a separate server program (usually on a different computer) for a specific function or purpose through a network. This model is the basis for WWW applications where the client (i.e. a web browser such as an Internet explorer[®]) accesses remote web servers where web pages are browsed from. Typically on the server, information is stored in a database that the web server accesses to retrieve information for the user. The user can write information to the database by submitting this information using the HTTP request / response mechanism which is a standard internet technology [183].

Figure 4.9 shows a widely implemented three-tier (also known as multi-tier), web application client – server model where client(s) implement the presentation logic (i.e. thin client front-end). The business logic is implemented on an application server and the data resides on a database server. The Web-

HMI system component has been implemented using this pattern as described in the chapter 7.5.2.

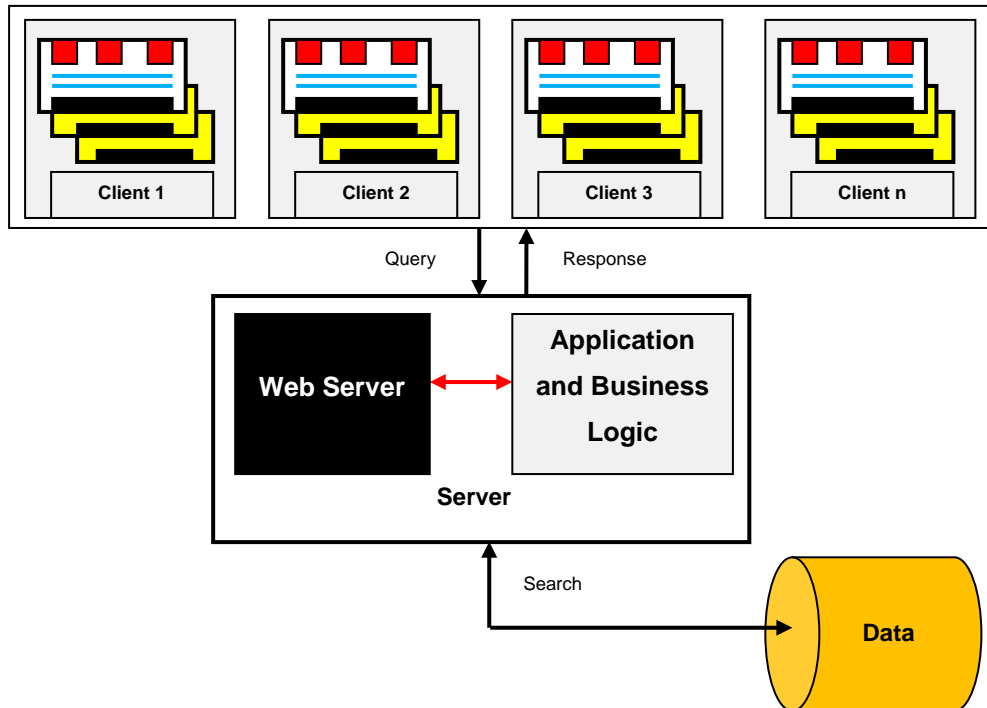


Figure 4-9: Client – server Model

4.7 Technological Review Analysis

In this chapter, major methods for the operator interface system design and development aspect of the automotive research were examined. Since no published literature within the RMS domain has concentrated on the user interface engineering aspect of the HMI system which addresses the needs of the identified HMI user roles involved within a machine lifecycle, a section on modelling and engineering (i.e. combining the models together) was covered. Moreover, the engineered model can be transformed into a workable system using various implementation approaches such as model based development and design tools.

Model based development suffers from relationship modelling complexity where as design toolkits provide extensive control and flexibility in creating the required user interface system. Since HMI systems are currently developed

using proprietary toolkits (for example, Siemens WinCC®), they are expensive, difficult to learn and cannot be transferred to other vendor's machine control systems (i.e. they are machine control dependent). There is a need to have a system platform that allows integration of different vendors' machine components.

A review of machine data transmission options and their associated issues (such as security, safety, robustness and reliability, data uniformity, performance and scalability) were documented in this chapter. Two approaches to remote data transmission identified are raw data transmission (which drives remotely implemented HMI) and remote sharing sessions (which copies the entire HMI screen from the local site to the remote site). Both approaches to remote data transmission can be employed depending on the supply-chain partner's data propagation implementation model.

Information sharing requires careful understanding of the underlying communication mechanisms. This chapter identified four major communication pathways to exchanging data in distributed applications such as point-to-point, publish / subscribe, guaranteed delivery and message bus. In addition to this, two common examples of data interaction styles are described to be continuous data transmission and event-driven data distribution. The discussion of this chapter ended with a review of relevant architectural patterns commonly used in distributed systems, such as MVC (for separation of the presentation view from the underlying domain model), layered architecture (for managing system access levels), repository (for describing how a system shares data) and client – server model (for simultaneous remote access to resources over the web). Next chapter details the architectural design for next-generation operator interface system implementation supporting machine lifecycle.

Chapter 5 : Architectural Design

Chapter Contribution to this Thesis:

The major contribution of this chapter is the evolution and layout of a novel system architecture that enables remote control and monitoring of production machines by supporting a set of operational requirements identified and justified within this chapter.

5.1 General Overview

This chapter describes general design guidelines which govern the operator interface system screen structure and describes the design capture process undertaken within this research. The design capture process gathers a set of operational requirements which drive a novel control and monitoring system architecture, an evolution of which is also covered within this chapter. Lastly, a number of benefits with respect to the system architecture implementation in powertrain manufacturing domain are detailed.

5.2 Requirements Design

5.2.1 Design Guidelines

The purpose of design guidelines is to define and promote consistency and completeness in the operator interface. These guidelines provide rigid standards and accepted practises for the design of the user interface within the desired domain of its implementation [147]. A good operator interface screen design requires careful use of layout logic, colours and contents (such as navigational routes, icons, menus, buttons, etc) that provide the correct level of usability. If something is inappropriately presented on the HMI screens, the operator may miss critical production information, business can lose money, or worse, someone may get injured [184].

For operator interface system to be industrially accepted, it needs to follow a particular set of display standard [185, 186] to provide the familiar “look and feel” experience to operators who have been accustomed to traditional vendor-

specific HMI systems [79, 80]. Since the human machine interface design within this research is driven by supply-chain partner requirements (summarised in the chapter 2.4 and chapter 3.3), it makes perfect sense to adopt useful principles from a set of design guidelines (presented as standard) directly from the powertrain manufacturing domain. Consequently, various features from the Siemens Transline standard [187] are utilised within this research when designing the operator interface system screens for control and monitoring machine lifecycle, as described next.

Siemens HMI Transline Standard

Siemens Transline [187] is a proprietary design guideline for Siemens PLC / HMI products. It supports traditional shop-floor functionalities such as operation, visualisation and diagnostics in their Simatic family of products using their HMI panels. The operation functionality typically supports general screen setup, operational modes, navigation routes and standard controls for a production machine. The visualisation functionality provides process specific display (in this research, process is usually termed as a machine component), and diagnostics provide message display, machine components and hardware diagnostics. An example of an implemented Transline HMI screen at Krause plant, Germany is shown in the figure 5.1. While detailed explanation of various features available within this guideline standard is outside the scope of this thesis, adopted principles from this standard and other ergonomic principles are detailed next [79, 80, 184-188].

- **Background Colour**
Muted toned colours such as grey and blue are best for backgrounds as they provide good contrast for the brighter colours such as red, yellow and green which may be used for important dynamic data such as machine component state change. To immediately identify different groups of screens on the HMI display, various shades of the muted toned colours can also be used.

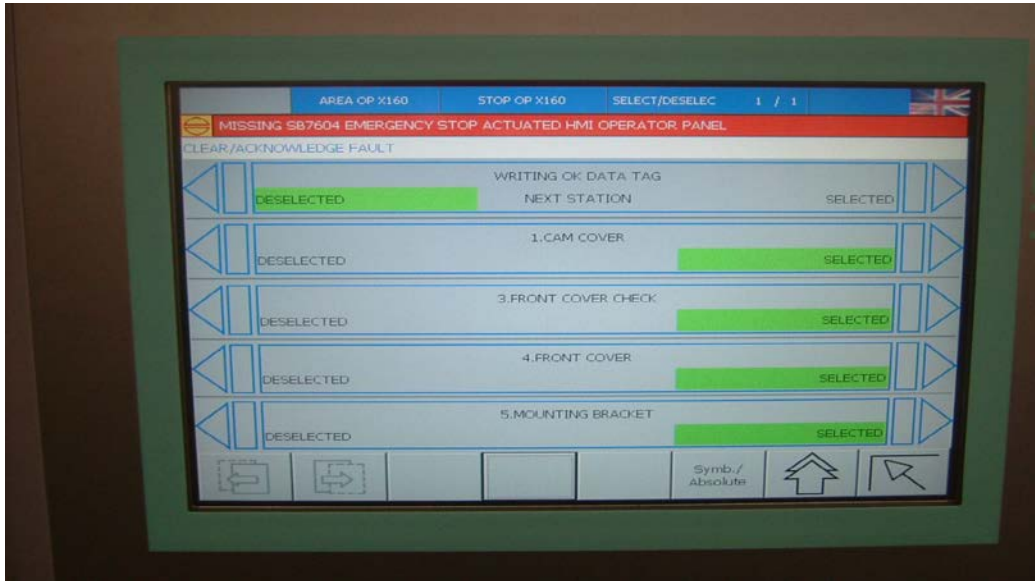


Figure 5-1: Implemented Transline HMI Screen Example

- Screen Structure and Textual Information

The screen should be structured such that a dedicated section for component status change and fault detection is always at the top and highly visible. Any critical machine operational mode information should be at the upper section too along with any universal screen icon / logo. Any component specific status change details, faults and diagnostics descriptions should appear at the central section of the screen. The navigational routes and machine control actions should be at the very bottom as shown in the figure 5.2.

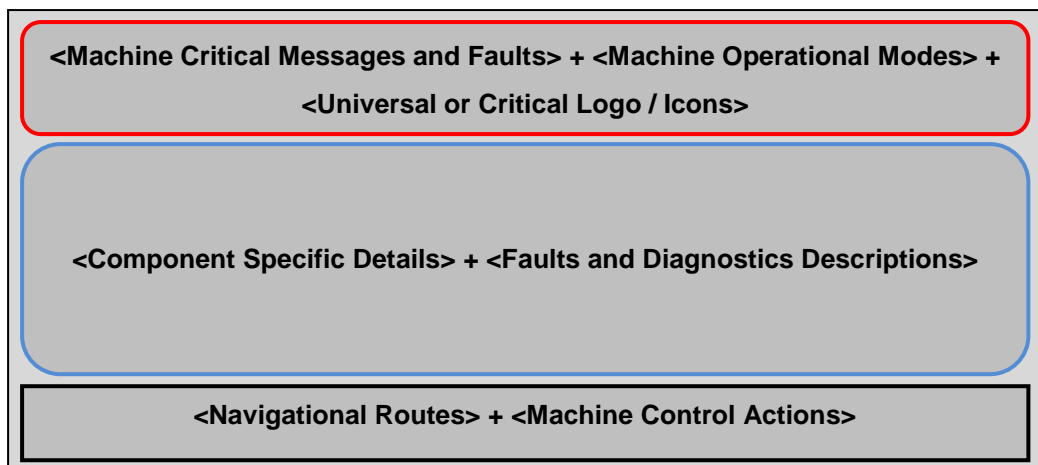


Figure 5-2: HMI Screen Structure Division - Transline Standard

The font style and size selection are critical aspects of the HMI screen design. A san-serif font style such as Arial is better owing to its lower resolution requirements which can easily be rendered by wide range of computer displays. Font size should be selected in a way which can enable reading machine critical information from several feet away such as size 12 points and onwards.

- **Faults and Component Status Data Display Conventions**

Dynamic machine data is the key aspect of the HMI display. There are two types of dynamic data such as faults and component status data. Fault status for the overall machine, preferably organised into groups, should be visible on every operator interface screen and there should be a simple navigational route to access the screen containing additional information about the fault. The fault colour should have a red background making it visible to an operator several feet away. As 1 in 12 men have some form of colour-blindness, which can affect the perception of red, colour cannot be used as the sole indicator of faults. Any colour change must be supplemented with a pictorial change such as appearance of an indicator. Whatever convention is used, faults should be placed where they can be easily seen, preferably along the top of the operator screen.

Component status data display should follow the traffic lights colour convention where a red / grey colour means an out-of-state message status (or stopped), yellow colour corresponds to an intermediary or transitional phase and green corresponds to in-state message status (or running). Apart from the colour display, the organisation of all the components should follow a standard display convention as shown in the figure 5.3. Since the number of components varies according to the nature and complexity of a manufacturing machine, the machine can only be monitored and controlled through a series of successive representations, one after the other, using screen browsing mechanism.

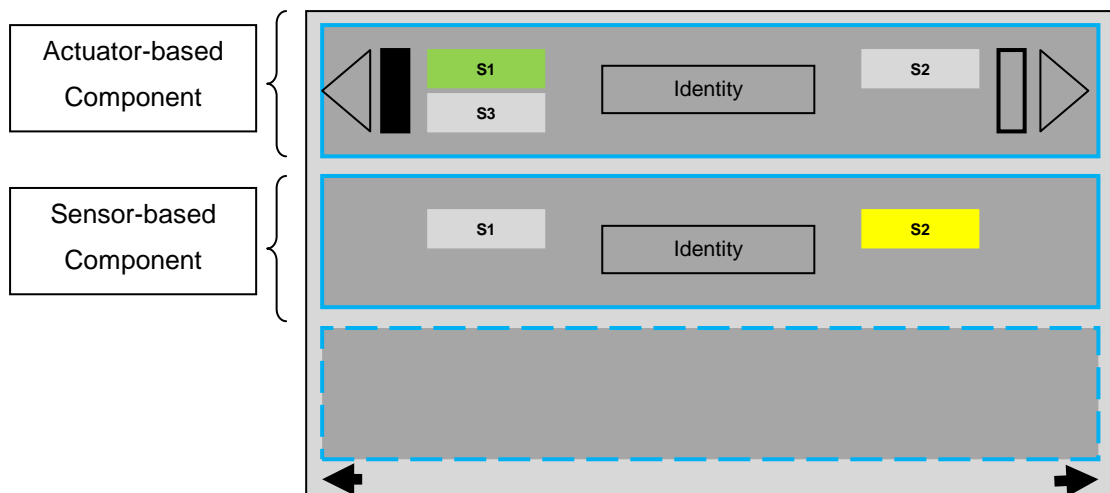


Figure 5-3: Machine Component Screen Convention - Transline Standard

- Navigational Routes and Control Actions

An operator must be able to navigate from one HMI screen to the next one quickly and easily using a logical tree structure presentation consistently located at the lower part of the screen. A navigation link to browse to the home screen should always be available in all the screens preferably located at the far bottom right of the interface. Next section explains the navigational route design capture using the storyboarding technique.

Control action buttons should be hidden and displayed only when machine control is initiated and acquired by the appropriate operator interface screen. Any control command execution should have action confirmation mechanism to avoid its accidental activation.

5.2.2 Design Capture

This section describes the process of capturing the design in terms of identifying various end user HMI roles' operational requirements and the navigational structure (i.e. navigational routes) for the operator interface screens. Some operational requirements have been captured using use case descriptions (as introduced in the chapter 4.2) while others have been obtained through numerous end user / machine builder interviews, demonstration,

meetings, site visits, and the available literature. Navigational structure which enables transition from one HMI screen to the next is captured through the storyboarding technique (concept described in the chapter 4.2).

Operational Requirements

Operational requirements correspond to functional and non-functional requirements for the required operator interface system implementation. Functional requirements describe what the HMI system implementation should achieve where as the non-functional requirements are related to the emergent system properties such that they define constraints on the overall design of the HMI system implementation [116]. The operational requirements are shown in the table 5.1. As shown, functional requirements are further categorised into research domain requirements (i.e. high-level requirements) and the HMI user role to machine interaction requirements (i.e. low-level requirements). Operational requirements collectively drive the HMI design, and the overall control and monitoring system architecture.

Operational Requirements	
<u>Functional Requirements</u> – drive the application control and monitoring architecture	<u>Non-functional Requirements</u> – drive the technical control and monitoring architecture
<p>High-level requirements:</p> <ul style="list-style-type: none"> • Real-time control and monitoring support. • Separation of control and monitoring functionality. • Global accessibility of the operator interface system (i.e. remote access). • Reconfiguration and reuse support. 	<ul style="list-style-type: none"> • Reliability and robustness of machine information i.e. provide it accurately through the use of a common repository which is not fragmented across multiple systems with poor integration. • System performance i.e. real-time machine information must be propagated to the user in a deterministic manner, preferably within 1.5 seconds through separation of

<ul style="list-style-type: none"> • Machine lifecycle support. • Legacy system support for control and monitoring applications. • Support machine-to-machine communication over the network. • Three-dimensional simulation support. • Cost-effective HMI system implementation (i.e. no associated licensing costs and auto-screen version update features). <p>Low-level requirements:</p> <ul style="list-style-type: none"> • Request machine commands. These commands can control the machine's position or operational mode. • Request real-time machine information, for example, current machine error information. • Display real-time machine events. These events are time-critical and must be delivered in real-time in order for the operator interface system to reflect the current state of the machine. • Display machine configuration information. This information may be the number of machine components with their states, and diagnostic information for identifying machine faults. 	<p>interface.</p> <ul style="list-style-type: none"> • Safety against common hazards should be implemented through correct level of procedures especially when supporting remote control operations. • Scalability of the architecture such that it supports different machine automation applications for example, engineering tools and third-party analysis packages. • Security of machine information transmission is of paramount importance. Remote data transmission over the web should be implemented through proper security strategies.
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Table 5.1: Operational Requirements

Low-level functional requirements have been captured using use case descriptions which describe the high-level interaction between the HMI user roles and the HMI system. Use case diagrams based on standard UML notation [189] are applied to represent use cases. Scenarios are then applied to each use case which describes generalised sequences of how the system processes the use case.

- Use cases

Use case illustrates HMI user role to machine interaction. As defined in the chapter 4.2, use cases capture a set of interactions between external actors and the system under consideration. Actors are outside the system, in this case HMI user roles. It captures who (actor) does what (interaction) with the system and for what purpose (goal) without dealing with the internal working of the system itself [133, 189]. In the use case diagram figure 5.4, only one actor (i.e. general HMI user role) is shown, the purpose of which is to outline all the interactions that all the user roles may have with the operator interface system to enable its implementation to meet these requirements.

As shown in the figure 5.4, there are four types of low-level functional requirements (i.e. use cases) identified which describe all the possible common interactions HMI user roles have with the operator interface system. The machine can be either simulated or real, transferring real-time machine information or machine configuration information to the HMI system to be displayed on its screens. User role requests machine commands which can be propagated to either the real or the simulated machine.

- Scenarios

Scenarios are applied to each of the use case. These scenarios provide a walkthrough of what the operator interface system is expected to do and how it responds for each use case. Extracted information from this practice can enable operator interface system implementation (chapter 7) to support the functional requirements identified in the table 5.1, and supports designing of the overall control and monitoring system architecture (chapter 5.3). Table 5.2 details four set of scenarios (one for each use case).

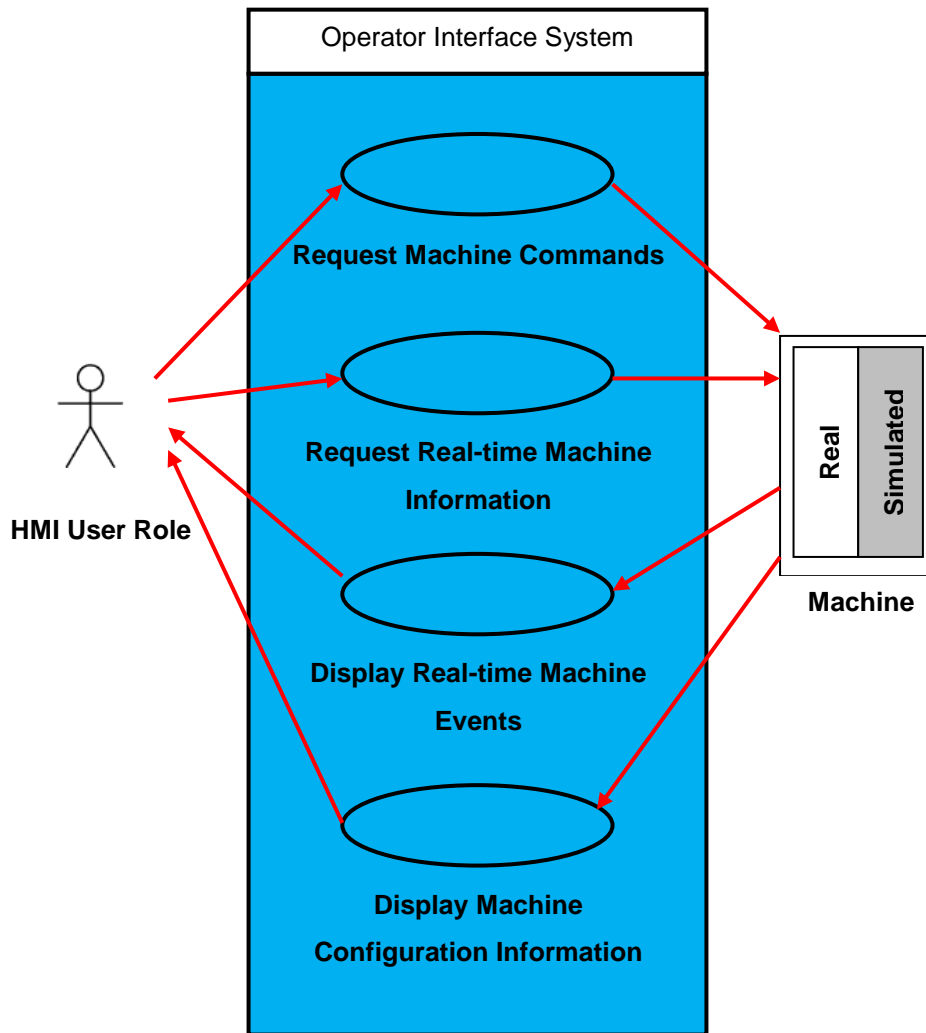


Figure 5-4: Use Cases Supporting HMI User Roles

Assumption: Some general elements are assumed to exist to explain an initial abstract executable process. These elements are a machine server (that contains application logic associated with the user role to machine interaction) and an operator interface screen (i.e. client browser) which enables interaction with the machine.

Scenario Use Case	Use Case Example	Scenario Description
Request a	Request to	(i) HMI user role requests a machine command using the

Machine Command	change machine operational mode	<p>operator interface screen.</p> <p>(ii) This request is transferred to the machine server.</p> <p>(iii) Machine server logic checks if the command can be executed by the requester.</p> <p>(iv) If yes, the command is processed and propagated to the machine for its execution (i.e. either real machine or simulated machine). The real / simulated machine logic decides the validity of the command based on the current state of it.</p> <p>(v) Response is returned to the requester to confirm command execution.</p> <p>NB: Multiple levels of safety measures can be implemented as shown in the chapter 7.4.4.</p>
Request real-time machine information	Request actual machine error information	<p>(i) HMI user role requests particular type of machine information using the operator interface screen.</p> <p>(ii) Machine server retrieves the requested real-time machine information from its live repository and configuration repository.</p> <p>(iii) The requested information is processed and returned to the operator interface screen where it is displayed to the user role.</p>
Display real-time machine events	Display a machine component state change	<p>(i) When a machine component's state changes, it generates an event.</p> <p>(ii) The event is propagated to the machine server that stores real-time machine events.</p> <p>(iii) The machine server broadcasts this event in real-time to all the operator interface screens.</p> <p>(iv) Operator interface screens receive the event and display it for the HMI user role.</p>
Display machine configuration information	Display all the machine components for a	<p>(i) An operator interface screen makes a request to the machine server for a particular type of machine configuration information to be displayed.</p> <p>(ii) The machine server retrieves the requested information</p>

	particular machine station	from its configuration repository. (iii) The requested machine configuration information is processed and returned to the requester operator interface screen, where it is finally displayed.
--	----------------------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Table 5.2: Scenario-based Design Capture

Navigational Structure

The correct level of navigational structure is vital to enable HMI user roles to efficiently carry out their day-to-day tasks. This requirement is specified using a storyboarding technique [190, 191] which comprehensively describes the operator interface system’s functionality in terms of its navigational structure (see chapter 4.2). This approach to specification can provide sufficient information to design and implement operator interface system’s navigational flow. By definition in this context, a storyboard is a way to tell a story through the use of discrete static diagrams (represented on individual boards or sub boards) that can be strung together to tell a story of the HMI system. These diagrams express the structure of the operator interface screens with their potential navigational routes, providing a roadmap of the HMI system’s usability. Expected navigational routes are detailed using arrows with a textual description of the action or method that associates the link between the screens.

The generic functionality required within all the HMI screens is the screen to screen navigation. With reference to the Transline layout as in the figure 5.2, example storyboard screen navigation is shown in the figure 5.5. In this example, four operator interface screens have been shown where the navigation from a source screen (i.e. machine fault details screen) to the respective target screens (i.e. current fault details, solved fault details and fault history details) is represented using an arrow with the description of the navigational action written on it. The direction of the arrow shows the corresponding navigational route. In this figure, current fault details screen can be navigated from the machine fault details screen using current fault navigation action. The same is applicable to other navigational routes.

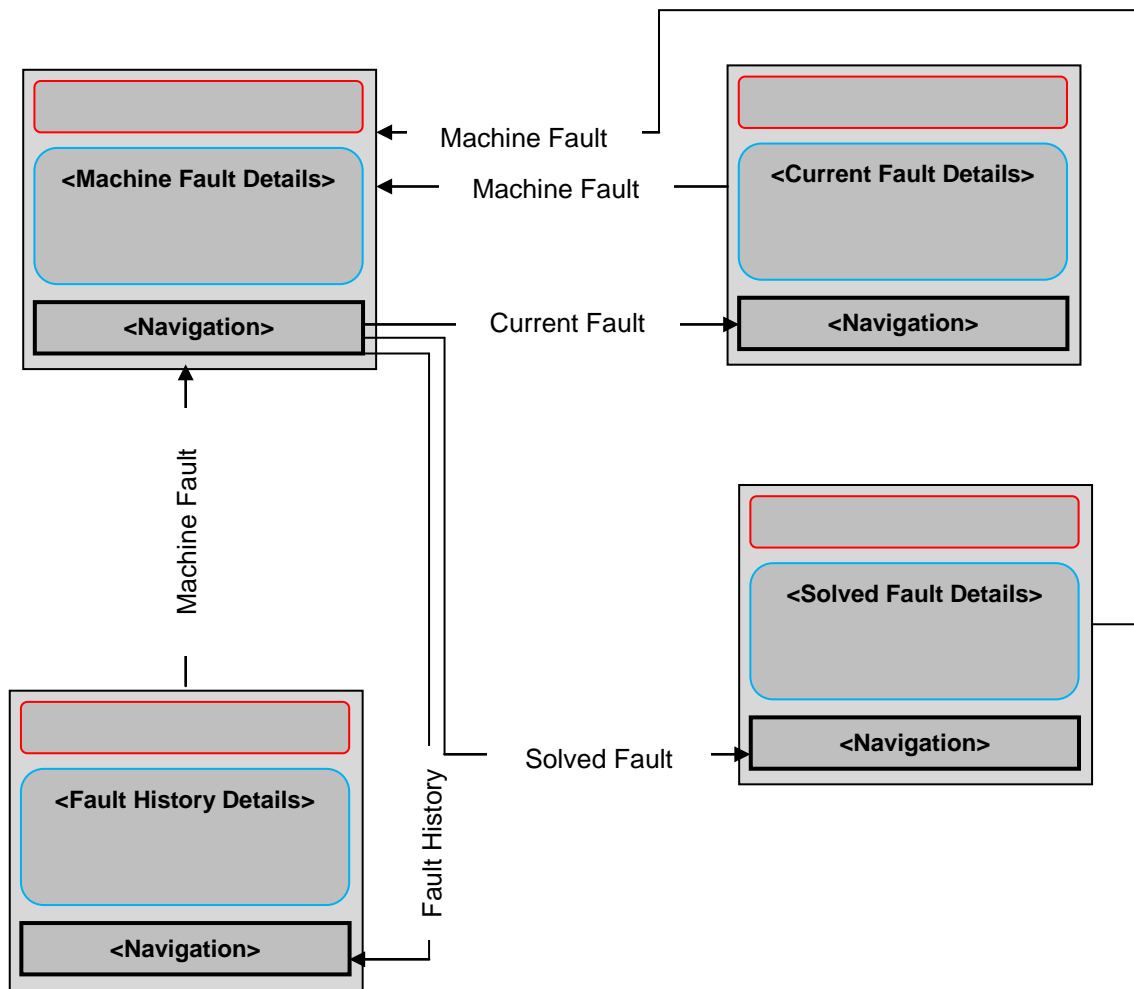


Figure 5-5: Example Storyboard Capturing Fault Details Navigational Structure

This section of the chapter has captured operational requirements and identified the navigational structure to design operator interface system. For the HMI system implementation to successfully support manufacturing machine lifecycle, the rather lengthy list of requirements (described in the table 5.1) must be satisfied. This actually boils down to a novel system architecture suitable for complimenting CB automation approach implementation to supporting machine lifecycle (described next).

5.3 System Components Architecture

5.3.1 Architectural Evolution

In this section, the evolution of novel system architecture for operator interface system implementation to support the identified requirements is described. The architecture of a system provides a well defined structure of the different components in the overall system solution and their interconnections [183]. This overall arrangement helps development of software components (incorporating the rules and regulations) in the detailed design stage of system engineering [116].

Figure 5.6 shows the process through which the system architecture has evolved within this research to support the operational requirements described in the table 5.1. For ease of understanding, this figure is divided into five architecture versions with varying complexity levels such that level 1 is the most elementary system architecture (with its limitations) whereas level 5 is the evolved architecture (highest level) supporting all the identified requirements. The lower the level, the further it is away from satisfying the actual research requirements.

The simplest architectural solution is shown in the level 1 where HMI screens are directly linked to the manufacturing machine (i.e. real and simulated machine components). Inherent with the simplicity of the solution, there are functional limitations such that only direct proprietary link is supported where by machine components (implemented through its logic) are physically mapped to the operator interface screens' interaction elements. In this case, interaction elements are used for transferring information from the machine components to the HMI screens for example, set of action or display buttons on the screens.

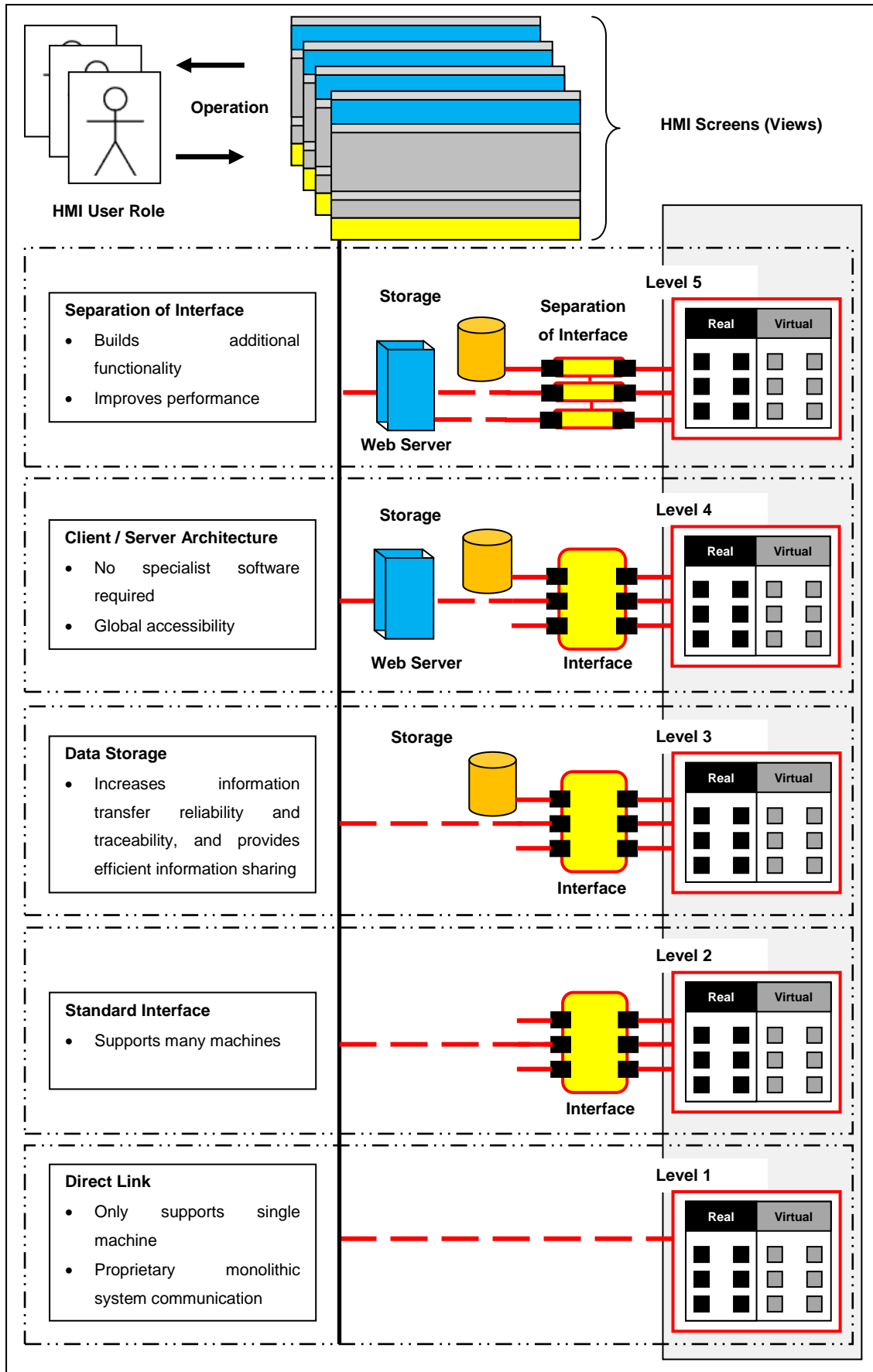


Figure 5-6: Evolution of Control and Monitoring System Architecture in this Research

Changing a machine (or its logic) requires modification of the screens. With an additional system tier i.e. a standard interface between the HMI screens and the machine components (as in the level 2), any machine component that meets the interface standard can be incorporated within the system architecture. Modification to machine logic does not necessitate change of individual screens as the interaction is through a defined interface standard. With an additional tier supporting storage of configurations and information in a repository (as in the level 3), it increases the reliability of the information transferred within this architecture as the information is retrieved, stored and then used instead of retrieving and using it directly. Furthermore, this enables efficient sharing of information between various machine and system components, and provides traceability of the information for both HMI user roles and machine component status which can be useful for future use for example running simulation of machine components using stored configurations.

Linking individual screens to the actual machine is limited in terms of supporting portability and global accessibility. Level 4 shows an additional system component (i.e. a web Server component) which partitions the system such that distributed services can be called upon by HMI clients (i.e. HMI screens running on standard browsers) and processed at the server side. The server component contains the necessary logic to control the execution of operational commands from various HMI clients on the manufacturing machine and propagate information from the machine to the HMI clients.

To build additional functionality and improve the performance of the overall implementation, a separation of the standard interface is shown in the level 5. Each separated interface can be implemented as a standalone system component (i.e. sub-system) to support required control and monitoring functionality. The system architecture described in the level 5 of the figure 5.6 is elaborated and represented as a block diagram in the figure 5.7. This shows the implementable system architecture which can be taken by system engineers to develop and implement it in the powertrain manufacturing domain (as shown in the chapter 7 and 8) to support remote control and monitoring of machines throughout their lifecycles. It has to be noted that the architecture assumes the

machine control to be engineered using the CB automation approach (as described in the chapter 3.2).

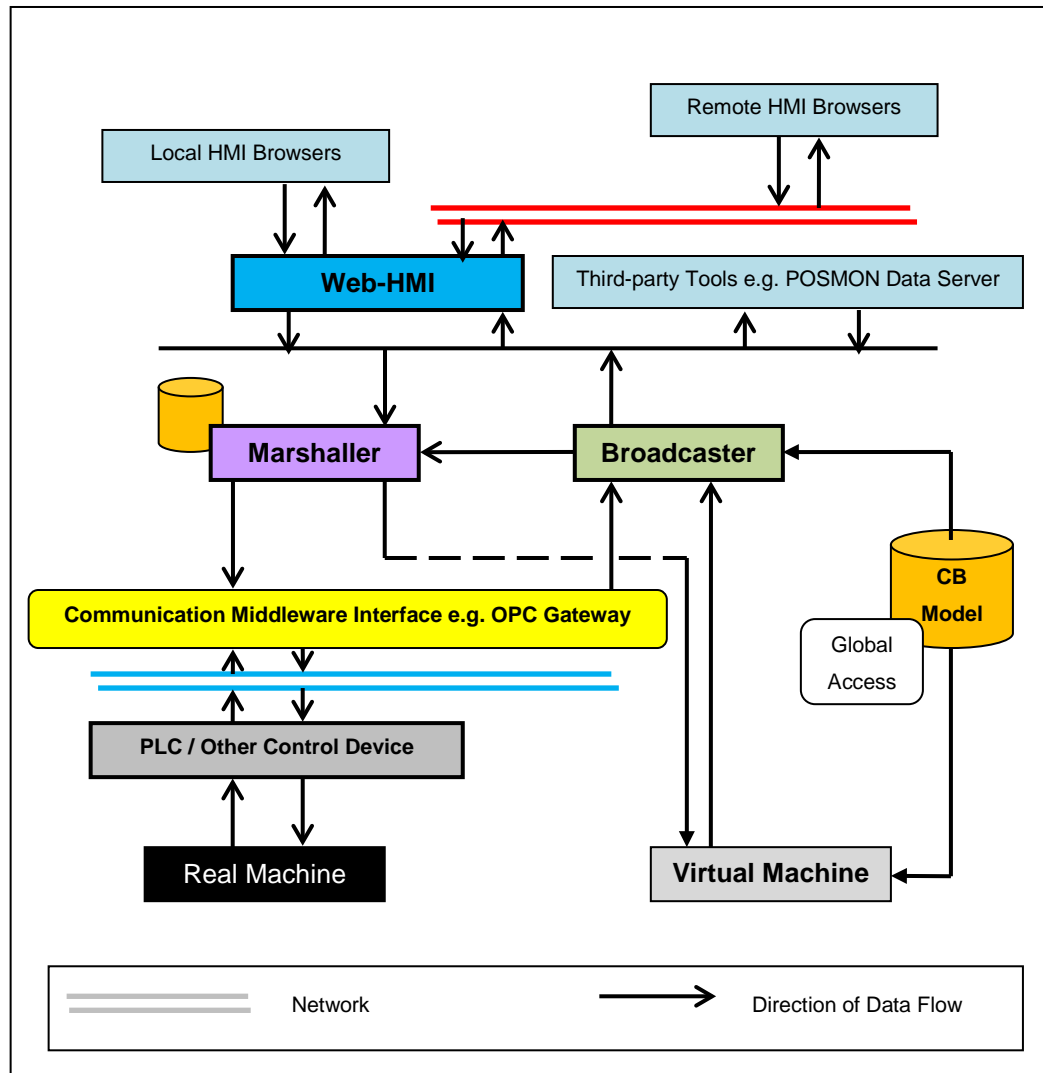


Figure 5-7: Proposed Control and Monitoring System Architecture

The system architecture in the figure 5.7 shows a decomposition of the overall system into three architectural system components for data control and monitoring. It is inherently distributed i.e. Broadcaster, Marshaller and Web-HMI. In a typical configuration, these components may physically be deployed either on a single PC, on networked computers, or within actual control devices. A superficial description for each of these system components is given next, prior to justifying the architectural layout in the next section of this chapter. A

detailed explanation of its implementation is covered in the chapters 6 and chapter 7.

Broadcaster: a system component responsible to continuously collect status data from shop-floor machines / virtual machines, process and propagate this information, in timely fashion, to range of distributed resources, regardless of their system mechanisms or their geographical locations. This system component is independent of the machine control type or the distribution strategy adopted by supply-chain partners. It acts as a central hub serving data monitoring functionality within the system architecture where real-time machine component's status information is held regardless of the machine's implementation state (i.e. real, simulated or hybrid (part real and part simulated)). Any third-party tools for example, POSMON used by Ford Motor Company can be linked to the Broadcaster component on-the-fly to gather machine's current operational status.

Marshaller: a system component acting as a channel (or bridge) responsible to controlling the communication between either the Web-HMI system component (serving various HMI client browsers), or other resources (for example another machine on the same production line) and the shop-floor machine / virtual machine. Furthermore, it is responsible to managing historical transactions associated with the machine events and its control operations using a static data repository.

Web-HMI: a server-side system component which serves many HMI client browsers regardless of their locality, to enable them to control and monitor production machines (real / simulated) using the Marshaller and the Broadcaster system component's assistance. The Web-HMI collects real-time machine information published by the Broadcaster system component and displays it on various connected HMI client browsers. This enables an HMI user role to monitor the status of real / simulated machines at real-time. Furthermore, Web-HMI supports control functionality through the use of its application logic which is directly (but internally) linked to the Marshaller system component's logic in order to implement necessary safety control procedures in a shop-floor

setup. This component has VRML model integrated to enable various simulation features such as machine live mimic, playback support, etc.

Broadcaster and Marshaller enable the Web-HMI to carry out its functionalities and are a value added benefit to the overall system architecture. These components collectively resolve limitations with the existing architecture (chapter 2.3.2), and satisfy focused system attributes and operational requirements identified earlier (chapter 3.3 and section 5.2 respectively). The justification of adopting this system architecture in powertrain manufacturing domain is detailed in the next section.

5.3.2 Architecture Justification

This section justifies benefits offered by the proposed control and monitoring system architecture (shown in the figure 5.7) over existing architecture (shown in the figure 2.4) through a summarised description of its uses as demonstrated in the table 5.3.

Benefits	Justification
Reconfigurability and Reuse Support. For example, reconfiguring machine by adding a new component such as an actuator.	<p><u>Assumption:</u> Applicable to CB engineered systems only.</p> <p>Machine configurations can be shared throughout the system architecture using Broadcaster system component. Any modifications to the machine (or its control logic) governed through CB engineering tools, is dynamically reflected by the HMI screens through the Web-HMI system component. Since the HMI screens are generic, any changes are easily accommodated at run-time as evaluated in the chapter 8.2.4.</p>
Machine Lifecycle Support. For example, control and monitor virtually simulated machines before their actual build.	The architecture is independent of the machine's implementation state (i.e. it does not differentiate between real, virtual or a hybrid machine). The Web-HMI has an integrated 3D VRML model which supports virtual validation of machines to identify their behaviour prior to their build process as shown in

	<p>the chapter 8.3.4. Furthermore, the same features are useful during maintenance process where virtual machine representation can be used to identify actual faults (described in the chapter 8.2.7).</p>
<p>Real-time Remote Control and Monitoring Support. For example, an end user remotely monitors production machine evaluated at a machine builder's site.</p>	<p>Broadcaster takes real-time feed from a machine and propagates it to Web-HMI component. Since Web-HMI supports remote HMI clients, remote monitoring is easily supported. Any operational commands (i.e. control commands) are propagated from remote HMI clients to Web-HMI system component which in turn needs Marshall's assistance to execute the command on the machine. This enables remote machine control as shown in the chapter 8.2.7.</p>
<p>Information Transfer Transparency and Mobility. For example, support third-party engineering tools.</p>	<p>Broadcaster transfers information in uniform XML format to its clients thus legacy systems (and third-party systems) can easily be plugged-in (on ad-hoc fashion) within the system architecture as shown in the chapter 8.2.5. This is beneficial in terms of extending the lifetime of a legacy system as it can be integrated with the existing infrastructure.</p> <p>Furthermore, since Web-HMI supports web standards, there are no licensing costs associated with version updates; in fact any version update is automatically propagated owing to the distribution nature of the system components within this architecture.</p>
<p>Rapid Machine Design and Early Machine Ramp up. For example, train machine operators before the actual machine implementation at the end user's shop-floor.</p>	<p>As stated earlier, the architecture is independent of machine's implementation state (i.e. it does not differentiate between real, virtual or a hybrid machine). The Web-HMI has an integrated 3D VRML model which supports virtual validation of machines to identify their behaviour prior to their build process. Therefore, the Broadcaster takes feeds from a virtual machine and updates Web-HMI during the training process (evaluated in the chapter 8.3.6). Any control operation is supported by the Marshall component which interacts with the virtual machine.</p>
<p>Separation of Control and</p>	<p>The system architecture decomposes the control and monitoring</p>

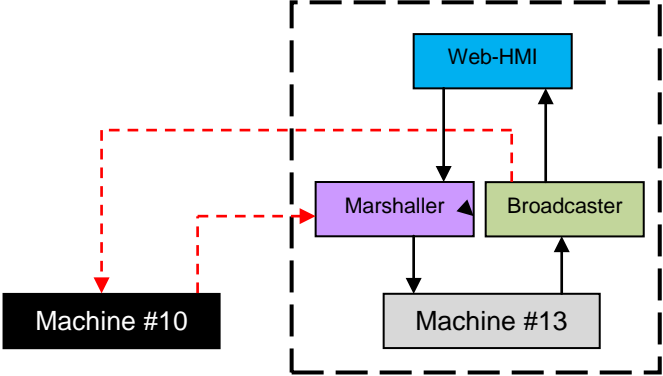
<p>Monitoring Functionality. For example, implement one production machine with only monitoring support (and avoid control functionality).</p>	<p>functionality into two distributed system components; namely Marshaller (for control) and Broadcaster (for monitoring). Each solution (i.e. control and monitoring) can be implemented independently based on the support required by an individual machine implementation.</p>
<p>Reactively Support Machine to Machine Communication. For example, if a production machine has some fault within a production line, it should be possible to immediately connect to another machine so that the product can be redirected to it (if they have common operation). In other circumstances, a machine can be instructed by another machine to carry out a specific operation by reading the RF tag.</p>	<p>For example, the machine # 10 in the diagram below can immediately connect to the Marshaller and Broadcaster system component of a machine # 13 to instruct it accordingly. Based on the available machine's operational status, a new product can be directed to it from the machine #10 and monitored at real-time.</p>  <p>The diagram illustrates a system architecture. A dashed box encloses a 'Web-HMI' (blue box) at the top, a 'Marshaller' (purple box) on the left, and a 'Broadcaster' (green box) on the right. Below these is 'Machine #13' (grey box). Solid arrows show the Web-HMI connected to both the Marshaller and Broadcaster, and the Marshaller connected to Machine #13. Dashed red arrows show Machine #10 (black box) connected to both the Marshaller and Broadcaster, indicating communication between the two machines.</p>

Table 5.3: Summary Justifying the Proposed System Architecture Benefits

In a bigger picture, this system architecture supports integration of widely accepted industrial standards like OPC, XML, HTTP, HTTPS, Ethernet, web services, etc. Furthermore, it enables heterogeneous co-existence of various industrial controllers such as PLC, FTB, Lonworks, etc. The operational requirements capture and its specification described within this section deals with the type of information and interactions HMI user roles require, and do not specify the detailed design of how the operator interface system implementation processes machine data to support the overall control and monitoring functionality through the lifecycle. This aspect of the research is described in the chapter 6 using the blackboard-based methodology of processing data within the system components.

Chapter 6 : System Components Detailed Design

Chapter Contribution to this Thesis:

The major contribution of this chapter is a detailed design for each of the system components using the blackboard-based methodology. The described models validate that the overall system has correct data interaction and processing mechanism to provide required control and monitoring functionality.

6.1 Blackboard-based Methodology

Nowadays, operator interface application requirements are getting more and more complex and their design is not getting trivial either. Their design requires a model which can enable a smooth transition from the engineered requirements to a detailed solution that can easily be transformed into a working system [149]. This chapter describes the blackboard systems design models for the system components identified in the chapter 5.3 that offer a new approach to next-generation operator interface system implementation.

6.1.1 General Description

This section introduces the concept of blackboard-based methodology and the benefits observed from adopting this methodology within this research. The subsequent sections describe each of the major components of this methodology in detail.

Concept

This methodology is a task independent architectural design approach which can be explained using a problem scenario providing a simple metaphor that gives insights to how this approach actually works, adopted from Corkill [192].

“Imagine a group of human specialists seated next to a large blackboard. The specialists are working cooperatively to solve a problem, using the blackboard as the workplace for developing the solution. Problem solving begins when the problem and initial data are written onto the blackboard. The specialists watch the blackboard, looking for an opportunity to apply their expertise to the

developing solution. When a specialist finds sufficient information to make a contribution, he or she records the contribution on the blackboard, hopefully enabling other specialists to apply their expertise. This process of adding contributions to the blackboard continues until the problem has been solved”.

The above group problem solving metaphor is a natural way for teams to solve problems. In general, the blackboard methodology involves the design of three principal components (known as BB-components in this thesis) namely blackboard, knowledge sources and controller. Each of the BB-components can easily be identified from the scenario metaphor described above [193]. In the scenario, the workplace onto which the initial set of problem, intermediate solutions and the final solution is written to corresponds to the blackboard BB-component of this methodology. The specialists who are responsible to apply their expertise in solving the problem correspond to the knowledge sources of this methodology. In order to control the flow of the problem solving process and schedule the contributions of each of the specialist onto the workplace, the controller BB-component of the blackboard methodology is required.

Figure 6.1 shows an overall blackboard systems model where a set of resources (i.e. knowledge sources (KS)) share a common global database (i.e. the blackboard) and the access to this shared resource is managed by a control shell (i.e. controller or manager). As shown, the knowledge sources can be internal (e.g. functions, procedures, modules, etc) or external (e.g. complete systems such as an operator interface (HMI)). The blackboard can be a single publicly accessible region or subdivided into regions or panels (explained in section 6.1.2). The controller can be implemented as a separate entity (centralised) or can be partly implemented in the blackboard and partly in the knowledge sources (distributed).

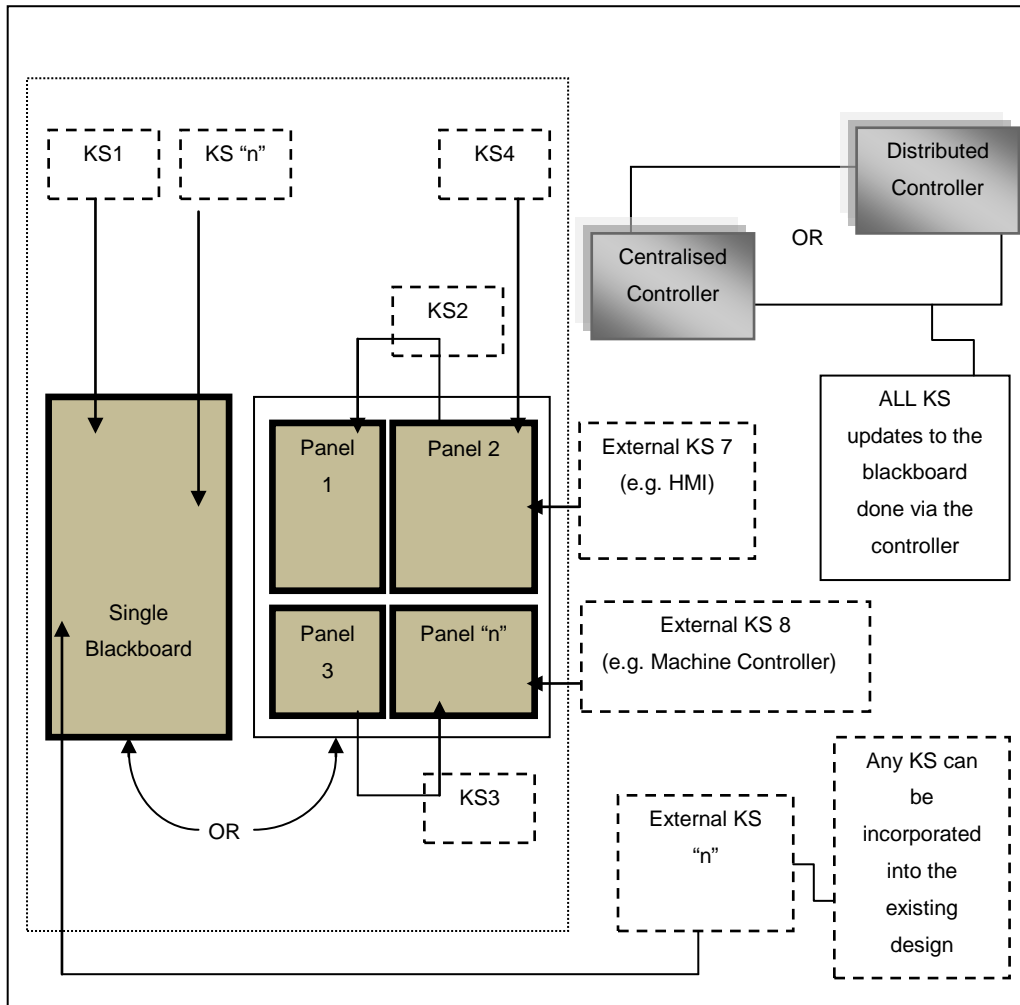


Figure 6-1: Overall Blackboard Systems Model

Benefits

This research requires distributed partner resources to efficiently interact with shop-floor machines. This poses a need to have a methodology which supports soft real-time communication and implements loose coupling of various systems, while providing up to-date view of the machine status, regardless of what mechanisms the partner resource systems implement or where the systems are actually located. Since large numbers of loosely coupled components require access in shared fashion to the published machine data, repository is the best approach to modelling the architecture in software engineering as mentioned in the chapter 4.6.3. In the field of software architectures, blackboard-based systems are also known as repositories [116, 194].

Furthermore, the distributed partner resources and the communications among them are usually not static; therefore the associations between them are indirect and cannot be predetermined until specific data values become known at run-time. As identified by Corkill [195] any direct interaction among resources may encourage the use of private communication protocols between the distributed resources. Though these private protocols can be made efficient, any changes to the communication strategy and / or addition of an extra resource may require changes to the existing communication strategy. The approach needed in this research needs to provide a model which provides reconfigurability and indirect communication among resources via an intermediary such as a blackboard data repository (as shown in the figure 6.2). This methodology allows parallel execution of processes to improve system's effectiveness over traditional message-based transfer systems.

Blackboard-based approach is a highly modular way of building problem solving systems [116]. This is because modularising the BB-components allows interactions between them to be regularised [196]. Furthermore, it allows clear and rigid interfaces to be defined through which the BB-components can be accessed. Firstly, the knowledge sources are independent of each other and communicate only via posting and modifying entries on the blackboard. Any functionality addition or modification to an existing knowledge source can have no effects on the other BB-components of the system. In addition to this, any new knowledge source BB-component can be added to the existing model without causing adverse effects to the system as shown in the figure 6.2. Thus it allows dynamic reconfiguration as the resources can join or leave the systems at runtime, promoting a system to be more reusable and maintainable [197].

Secondly, as the blackboard BB-component is organised into abstraction levels, problem solving activity at one level is independent of the reasoning at the other. This is because objects on two levels are not the same and interactions between these levels can only be indirect. Since the hierarchy of abstraction exists, this implies that the inferences made at one level do not interact with those at another unless they are required to do so by some knowledge source. One can regard the contents of one abstraction level as being separate from the

contents of another. Moreover, the architecture requires a controller to be a special component which is independent of the knowledge sources that represent the problem solving functionality. Therefore given the same set of knowledge sources, a number of different control strategies can be implemented, and a number of different controllers can be constructed. This makes the control behaviour easy to be modified or replaced without affecting other BB-components. This justifies the choice adopted by the author to select the blackboard systems methodology to tackle this research problem.

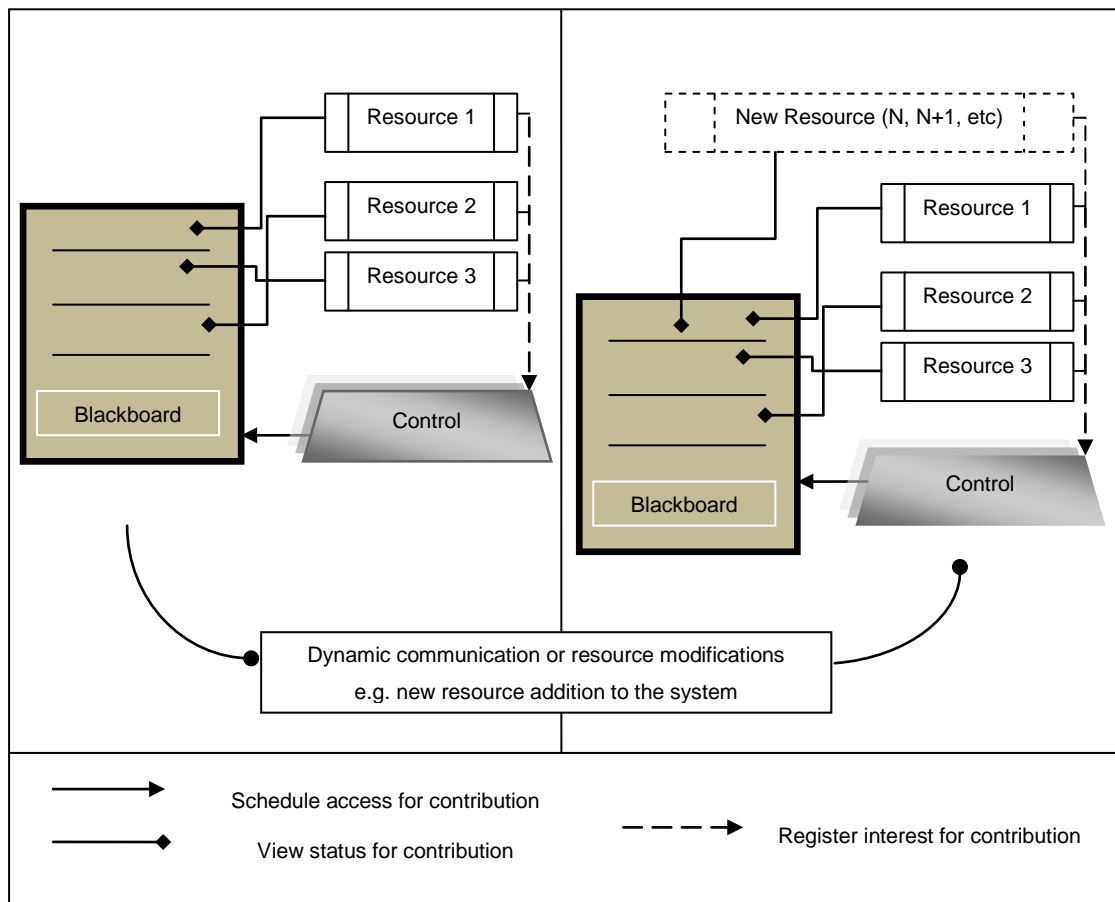


Figure 6-2: Indirect Communication and Ease of Reconfigurability in Blackboard Methodology

6.1.2 Blackboard

Blackboard is a globally accessible region in a system organised into linear hierarchy composed of abstraction levels that forms an outline plan for problem solving [193]. Blackboard mediates all the communication within a system and

holds intermediary results by recording the machine information in the form of entries. In simple terms, any item placed or modified on the blackboard is called an entry. These entries are often expressed as sets of attribute-value pairs, placed onto the repository (blackboard) by the knowledge sources. Each entry is linked to the next in the form of a graph structure which represents intermediate results' configurations bearing some relation to each other [198]. A complete solution to the problem under consideration consists of a collection of entries which reside at different levels of abstraction.

If the problem being solved by the knowledge sources is complex and the number of contributions made on the blackboard begins to grow, quickly locating relevant information becomes a problem. In a good design, a knowledge source should not scan an entire blackboard to see if a particular item has been placed on it by another knowledge source [192]. Efficient retrieval mechanisms are normally needed to support the use of a blackboard as a group memory, for contributions generated by earlier knowledge sources' executions.

It is advised in the literature to divide the blackboard to not only abstraction levels, but also into non-overlapping regions or panels as shown in the figure 6.1 earlier. A panel is a distinctive region of the blackboard which has its own set abstraction levels. Each panel is dedicated to solving a different part of the original problem. Depending on the problem tackled, a panel can be designed to support a read only, write only or read / write operation. Any communication between the panels is carried out by the knowledge sources. This is because these panels do not share storage when information from one panel is to be passed to another. This makes a blackboard system an extremely flexible model for problem solving. In addition to panel division, pre-programming the flow of machine information through the blackboard system can achieve substantial performance enhancements [199].

6.1.3 Knowledge Source

Knowledge sources represent the problem-solving knowledge essential to operate a system [196]. In contrast to the blackboard BB-component, a knowledge source corresponds to the long-term memory of the system as it possesses the necessary knowledge / expertise about some aspect of the domain, which can be applied and re-used as necessary. They respond to the blackboard state by adding new entries or modifying existing ones. When a particular state changes on the blackboard, an event is raised that can be used to trigger the knowledge sources for their contributions. Knowledge source can then carry out concurrent reads and writes on the blackboard BB-component.

As mentioned earlier, they may only communicate with each other (e.g. pass parameters, results, etc) via the blackboard. Knowledge sources are anonymous as they neither communicate directly with each other nor know what other specific knowledge sources are present in the system, instead they communicate indirectly by altering the contents of the blackboard database. The knowledge sources can be widely diverse in their computational techniques as they can be represented as a procedural routine, function or an entire complex external system in any implementation language [197]. Each knowledge source in the model can be designed and programmed using different techniques such as object-oriented technique, web services technology, etc. Knowledge sources can be designed to execute on separate threads or on separate processes on the same or different computers. If knowledge sources are supposed to be executed using separate threads, each threaded knowledge source can access the blackboard because the threads share the same address space. If it executes on separate processes (possibly different computers), technology like Remote Procedure Calls (RPC) etc, can be used for communication between the knowledge source and the blackboard [200].

Knowledge sources have two major parts (besides other) in its configuration as shown in the figure 6.3, a precondition and an action. A precondition monitors the state of the blackboard to determine the circumstances in which it can make a contribution to the existing problem solving process. Only those knowledge sources, whose preconditions have been satisfied by blackboard events, can be

executed. When the precondition gets satisfied (i.e. preconditions equals to true), its action(s) is / are used to either add or modify an entry on the blackboard. This may lead to a transformation of entries of one level into entries at another. The process of adding entries by various knowledge sources onto the blackboard continues until the overall problem has been resolved. It is possible for a knowledge source action to alter the blackboard at more than one abstraction level (as illustrated in the section 6.2.5), e.g. add an entry to level one and modify an entry at level two of the blackboard simultaneously, when executed.

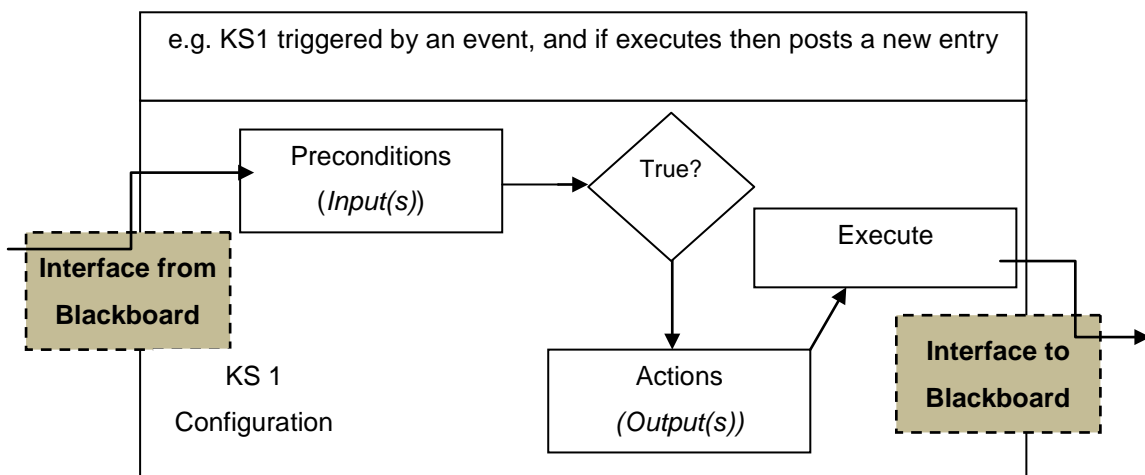


Figure 6-3: Knowledge Source Basic Configuration

6.1.4 Controller

In order to effectively converge on a solution, the methodology requires a complex controller implementation. This is because the blackboard model uses different kinds of knowledge, and to determine the application of a specific knowledge source at a specific time poses a challenge to the design of the system. The controller handles this complex runtime problem-solving strategy in order to allow a smooth flow of events in the implemented model. This is done by examining the state of the solution on the blackboard (which is represented by the state of the blackboard database at any one time) and selecting the available knowledge source that can be applied at that particular blackboard state.

Summarising, the controller provides a mechanism of organising the use of knowledge sources in most effective and coherent fashion. The controller can adapt a range of strategies based upon the task under consideration. It might, for example, schedule knowledge sources in a bottom-up manner, top-down fashion, or a combination of both.

Controller can be centrally implemented in the blackboard system or distributed. If the blackboard is divided into panels then a separate controller can exist for each panel or a controller can manage the executions across all the panels. When an event occurs on the blackboard, the controller uses the event parameters (such as abstraction level, panel, event type, etc) to traverse through the lists of registered knowledge sources for this particular event and redirect their activities to different regions of the blackboard as illustrated in the section 6.2.6. Furthermore, it triggers the registered knowledge source(s) for this specific event in order to execute the knowledge source. The next section details individual components of the control and monitoring system architecture by describing their respective blackboard design models.

6.2 System Components Design

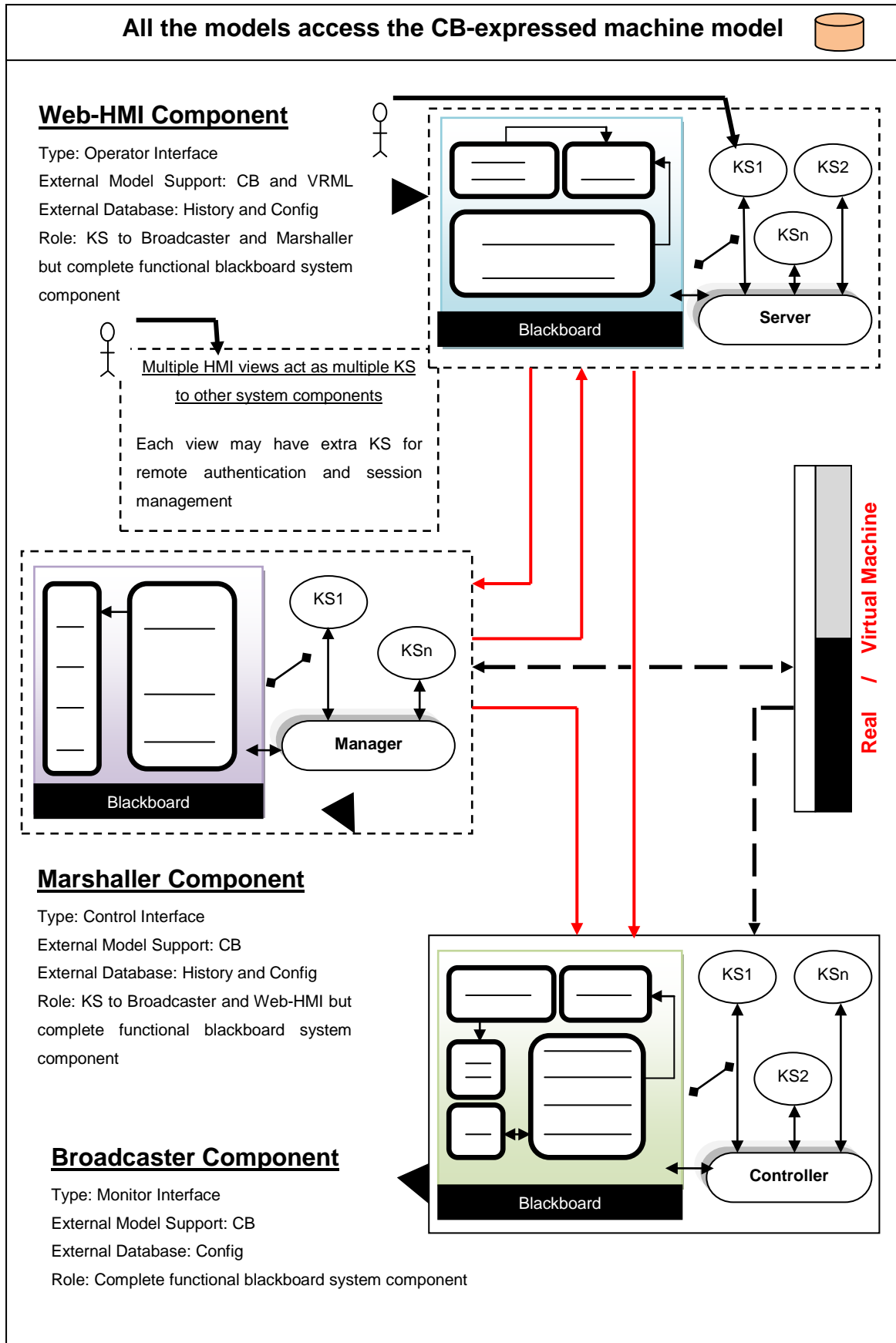
6.2.1 Overall Design Description

Each system component within the system architecture (shown in the figure 5.7) is architecturally designed and described using Blackboard-based methodology. The result of which is a detailed description for each component which validates that the system has correct data interaction and processing mechanism to provide overall control and monitoring functionality. In fact, these models show how these components are operating to process machine data in powertrain manufacturing systems. These models can be directly transformed into workable systems using various implementation techniques (for example, object-oriented implementation as shown in the chapter 7).

These three system components (i.e. Broadcaster, Marshaller and Web-HMI) can be overall identified working as a distributed multiple-blackboard system

model, where each major external knowledge source to other system component is a complete system node in a network, incorporating its own set of internal knowledge sources, blackboard and controller to function correctly. The central system component (i.e. the Broadcaster) provides access as a main public blackboard system enabling various other system components (major external knowledge sources such as Web-HMI and Marshaller) to monitor the state of Broadcaster's blackboard. The overall model design is shown in the figure 6.4. It has to be noted that the internal structure of the blackboard and movement of entries within it, as shown in the figure 6.4, does not correspond to the actual model design. Each system component design model is described in detail in the subsequent sections with the exact type of knowledge sources, blackboard structure, input and output data flows and control strategies adopted.

Since the research aim is to offer a new approach to operator interfaces which support control and monitoring functionality, the overall designed model shown in the figure 6.4 decomposes the research task and delegates it to three separate system components, each carrying out its own responsibility to solve the overall problem in a distributed environment. Author's approach in this research is to divide the problem so that an architecturally intact blackboard system can be discovered at each node in the distributed problem-solving network. This causes the grain-size of each node on the network to be a substantially large complete blackboard system. If the unit of distribution is a complete blackboard system, each node can solve complicated problems efficiently [201] and the problems tackled by each node can be completely different. It has to be noted that when the unit of distribution is smaller (unlike the model shown in the figure 6.4), nodes are frequently forced to cooperate in solving subtasks and the communication requirements are increased accordingly. Since operator interfaces can operate in a geographic location or with mechanisms, which can have some network communication constraints such as limited bandwidth, the approach to divide the nodes into smaller units is practically unrealistic. Next section describes each system component's blackboard model with their KS and control structure providing the overall control and monitoring support throughout production machine lifecycle.



6.2.2 Broadcaster Blackboard Model

Broadcaster system component model is divided into five distinctive panels to avoid performance bottlenecks. Each panel deals with a completely different task delegated to its respective hierarchy. The entire blackboard acts as a gateway through which all the events in the system pass. This blackboard model is shown in the figure 6.5. The required functionality within this system component is to collect machine data (in soft real-time) and propagate it to a number of system components (locally and remotely implemented) for monitoring purposes.

Reflecting to this functionality, the blackboard is organised in such that the incoming machine information is progressively transformed into a simpler uniform message structure ready for transmission. Blackboard division into various panels enables the system to deal with variety of manufacturing information which needs to be filtered accordingly, based on the type and nature of the received data. In reasonable terms, these panels could not be combined into a single blackboard due to data filtering, preparation and propagation strategy. Within each panel a number of levels exist, where each level represents a specific problem to be solved. Any interaction between panels (through knowledge sources) is purely event-based implemented within the controller of the Broadcaster system component. Following is an explanation for each panel with their associated knowledge sources within the Broadcaster model.

In-Load Panel

This panel is responsible for collecting and analysing input data from a machine and propagating that data to the appropriate panel for further processing. The hierarchy of this panel consists of two abstraction levels namely, “Boots” and “Filters”. KS1 is a dedicated source whose “Signal” is never turned to “No” as it is responsible to receive data from machine source continuously (more information on this “signal” parameter is given in the section 6.2.5). Furthermore, the same KS1 is responsible for loading configurations (i.e. CB

and system configurations) and initiating Broadcaster system component. Since any reconfiguration to a manufacturing machine is engineered by and represented in the CB approach, dynamic changes can easily be accommodated within the Broadcaster system at run-time. Any incoming machine data is always written (i.e. posted) to the “Boots” level by this KS1, creating a new “Add” event type.

Posting on the “Boots” level triggers KS2 to read the message format and check if it conforms to the CB configurations. Furthermore, KS2 discriminates data according to its type, i.e. any incoming message can either be a machine state, machine error, machine operation or corrupt / unfiltered information. KS2 distinguishes this information at this level and posts the resultant outcome onto the “Filters” level. This creates a new event on the “Filters” level as new information is being written on it. KS3 is responsible to propagating the filtered information to “Machine-State” panel (machine states), “Diagnostic” panel (machine errors), “Operation” panel (machine operational commands and modes) or “Unfiltered Logs” (corrupt information) based on the message type. When data appears on the “Filters” level, the logic of the KS3 directs it to its corresponding section of the system component. KS3 plays a role of transmitting data from In-Load panel to the most suitable message formatting panel. Each of these panels deals with completely different type of machine data. Within each panel, each level accepts information (i.e. entries) in a specific format where each entry can only reside at a specific level if its object has the properties required by that level.

Machine-State Panel

This panel is responsible for preparing machine state information delivered by the KS3 in a format ready to be stored within the Broadcaster system. The hierarchy of this panel consists of three levels namely “ST-Extracts”, “ST-Refers” and “ST-Prepares”. As soon as information is posted on “ST-Extracts” level, KS4 reads and extracts the necessary machine state information, and

collects any necessary reference from the CB configurations. It posts the referenced information to the next level below.

Once the extracted machine state information is posted on the “ST-Refers”, KS5 is triggered. Since the original obtained machine data needs to be propagated further to remote sources, it has to be stored within the system. This level prepares the machine state information in a specific format which makes it an ideal unit to be stored efficiently within the system. The prepared unit of storage is posted to the “ST-Prepares” level. KS6 is responsible to propagate the ready-to-be stored state unit to the “Out-Load” panel. The information flow in this panel is top-down in nature as information on the higher level is decomposed, and then extracted and prepared into a lower simpler state unit ready to be stored within the system runtime.

Diagnostic Panel

This panel is responsible for preparing machine fault information delivered by the KS3 in a format ready to be stored within the Broadcaster system. The hierarchy of this panel consists of three levels namely “ERR-Extracts”, “ERR-Refers” and “ERR-Prepares”. As soon as information is posted on “ERR-Extracts” level, KS7 reads and extracts the necessary machine fault information, and collects any necessary reference from the CB configurations. It posts the referenced information to the next level below.

Once the extracted machine state information is posted on the “ST-Refers”, KS8 is triggered. Since the original obtained machine data needs to be propagated further to remote sources, it has to be stored within the system. This level prepares the machine fault information in a specific format which makes it an ideal unit to be stored efficiently within the system. The prepared unit of storage is posted to the “ERR-Prepares” level. KS9 is responsible to propagate the ready-to-be stored fault unit to the “Out-Load” panel. The information flow in this panel is top-down in nature as information on the higher level is decomposed, and then extracted and prepared into a lower simpler fault unit ready to be stored within the system runtime.

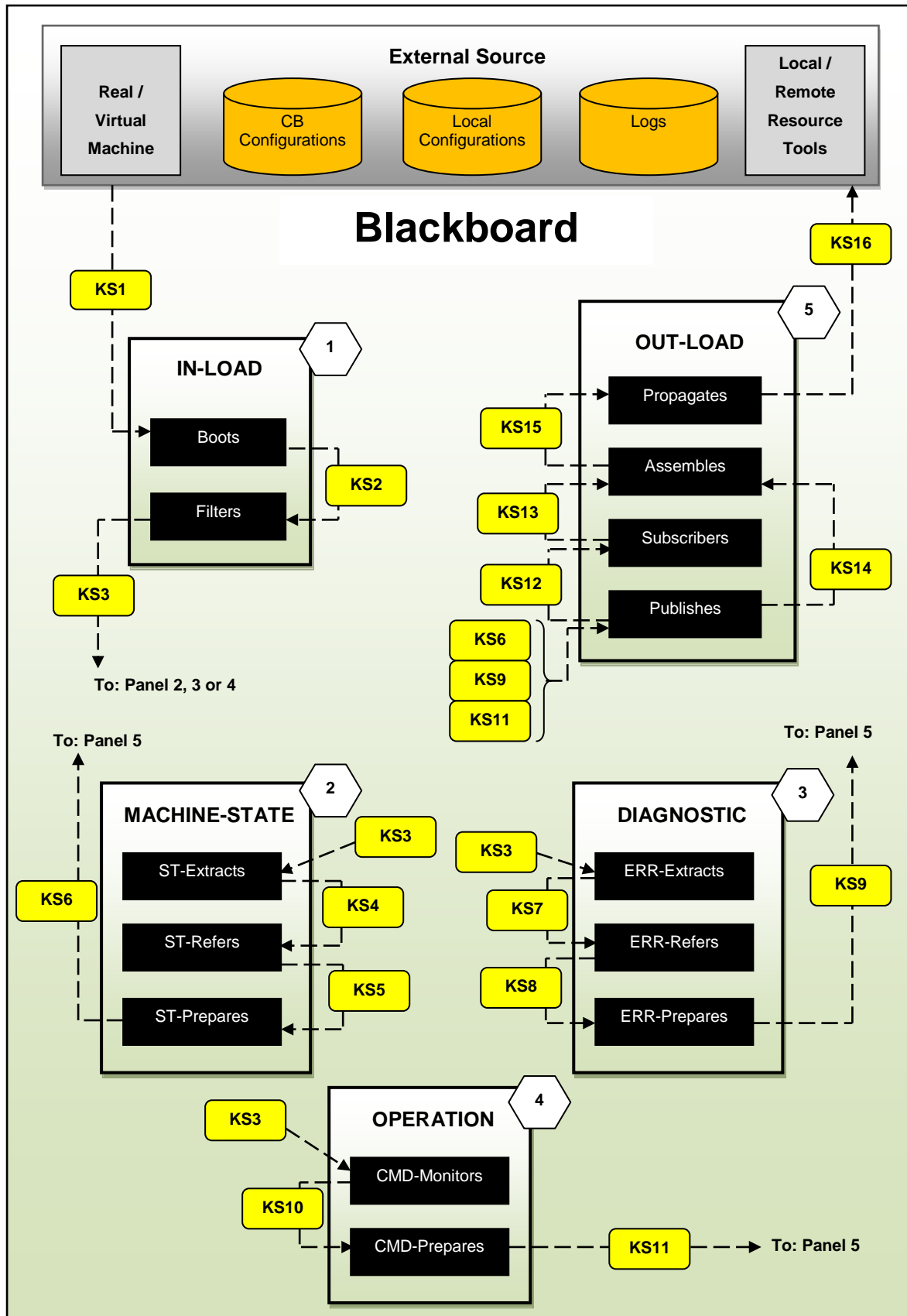


Figure 6-5: Broadcaster Model

Operation Panel

This panel is responsible for preparing machine operation information delivered by the KS3 in a format ready to be stored within the Broadcaster system. The hierarchy of this panel consists of two levels namely “CMD-Monitors” and “CMD-Prepares”. As soon as information is posted on “CMD-Monitors” level, KS10 reads and extracts the necessary machine operational information, and collects any necessary reference from the CB configurations. It also refers to the local configurations for necessary processing. It posts the referenced and processed information to the next level below i.e. “CMD-Prepares”.

Once the prepared machine operation unit is posted on the “CMD-Prepares” level, KS11 transmits it to the “Out-Load” panel for necessary storage. The information flow in this panel is top-down in nature as information on the higher level is processed into a lower simpler operation unit ready to be stored within the system runtime.

Out-Load Panel

This panel is responsible for storing machine information posted by KS6, KS9 or KS11 and propagating it to a number of remote clients on soft real-time basis. Its hierarchy consists of four levels namely “Publishes”, “Subscribers”, “Assembles” and “Propagates”. The “Publishes” level is the storage level where any incoming data unit is checked by the KS12 before it is stored in a circular buffer location. All the information units (i.e. machine states, machine faults and machine operation) are stored in this reconfigurable structure for easy and prompt access by remote partner resources. In order to provide a high-performance data access system, information needs to be written to the memory and not to a disk space [116]. Using a main memory as a repository rather than a traditional database allows KS contributions to be made at memory rather than at disk speed (more information on this storage mechanism implementation is provided in the chapter 7.3.4).

As soon as KS12 stores information units, two types of events are raised. One event triggers KS13 by informing it of the new information stored within the buffer. KS13 activates all the currently connected partner resource tools. All the remote clients' connections are checked and activated at "Subscribers" level by KS13 and necessary client information is posted onto the "Assembles" level. The second event triggers KS14 to transmit the current state / fault / operation information of the machine from the buffer to the "Assembles" level. At this level, KS15 converts all the stored information units and CB configurations (obtained from the KS14's execution) into a uniform XML structure ready to be transmitted to all the resource tools (information obtained from the KS13's execution).

The prepared XML machine information and connected resource tools status details are posted onto the "Propagates" level by KS15. As soon as the machine information becomes available, KS16 manages and propagates this data to all the partner resources connected to the broadcaster system component including the Web-HMI and Marshaller system. This process enables the current machine information to be monitored by remote parties regardless of their mechanisms or geographic locations. The movement of information across this blackboard panel follows the bottom-up progression because the information stored as a simple unit in the lower level is converted into an XML structure to be propagated to the clients in the highest level of the hierarchy.

6.2.3 Marshaller Blackboard Model

Organisation of the Marshaller system component model is influenced by two major target requirements supporting the required level of functionality within this system. These are:

- The component is mainly targeted to provide control functionality in the system architecture by managing a communication channel between the partner resource tools (for example, operator interface system) and the machine. Furthermore, it is responsible to managing this control

functionality by implementing a logic that governs any safety conditions under which a particular resource tool and machine can communicate.

- The component provides a repository service by logging all the machine transactions in a static structure for future use, for example, provision of machine playback functionality on the operator interfaces.

These requirements clearly identify two main features this system component has to support i.e. control and storage. Based on this, the gross structure of Marshaller's model is divided into two different blackboards namely "Historic" and "Channel" as shown in the figure 6.6. The "Historic" blackboard is used to perform buffering and repository processes on the captured data from the Broadcaster system component. The "Channel" blackboard is further subdivided into two distinctive panels responsible to managing communication and control processes within the system. Since both of these panels deal with a separate task, it seems logical to avoid combining them into one. Within each panel a number of levels exist, where each level represents a specific problem to be solved. Following is an explanation for each blackboard and its panels with their associated knowledge sources within the Marshaller model.

Historic Blackboard

This blackboard is responsible for capturing machine transaction information from the Broadcaster system component and buffering this prior to storing it into a static database. The hierarchy of this blackboard consists of three abstraction levels namely, "Collects", "Constructs" and "Buffers". KS1 is a dedicated source whose "Signal" is never turned to "No" as it is responsible to receive data from the Broadcaster source continuously (more information on this "signal" parameter is given in the section 6.2.5). Any machine transaction message is always written (i.e. posted) to the "Collects" level by KS1, creating a new "Add" event type.

This triggers KS2 to interpret the message, extract important information and sort it based on a time stamp accompanied within the received data. Incoming

data can be machine states, machine faults or machine operational information. Any transformation of this data to comply with the database design format is carried out at this level (database schema shown in the chapter 7.4.3).

KS2 posts the resultant outcome of these processes onto the “Constructs” level. This creates a new event on this level, where by the data is buffered in a FIFO structure by the triggered KS3, based on the timestamp of every machine message. As soon as the data is buffered up, an event is triggered on the “Buffers” level. KS4 monitors the buffer and saves this machine information to a static database based on its scheduler’s process preconfigured time. In order to reduce any communication overheads involved when performing necessary database IO operations, KS4 can be preconfigured with a specific time which manages the transfer of machine information from the system buffer to the static database.

Channel Blackboard

This blackboard is responsible for managing communication channel between partner resource tools and a manufacturing machine. Furthermore, it is also responsible to processing the safety logic which enables only one resource tool to control the machine at one time (algorithm described in the chapter 7.4.4), while enabling simultaneous machine querying from all the tools. This functionality is handled through two panels namely “Client” and “Machine” panel.

Client Panel

As the name suggests, this panel deals with the communication and control processes associated with the clients (i.e. partner resource tools). The hierarchy of this panel consists of four abstraction levels namely, “Inputs”, “Filters”, “Verifies” and “Propagates”. KS5 is a dedicated source whose “Signal” is never turned to “No” as it is responsible to receiving data (i.e. client’s token) from client resource continuously. KS5 carries out necessary authentication of

resource clients in order to implement required security regime. Any incoming information is always written (i.e. posted) to the “Inputs” level by KS5, creating a new “Add” event type.

KS6 is triggered to interpret this incoming client token and discriminate it into its type, i.e. any incoming token from a client resource can either be a control request, control message or a unit query. A control request corresponds to the request sent by any client to the Marshaller component with an intention to controlling a machine. A control message corresponds to the message a client can send once it acquires the control of the machine. A unit query corresponds to the query message that can be sent by any number of clients to query a specific machine unit (i.e. component).

As soon as KS6 performs necessary interpretation, it posts its outcome onto the “Filters” level. If the token type is a unit query then KS7 is triggered which prepares and transfers the query information to the “Machine Panel”. If the token type is a control request, KS11 is triggered which applies a safety logic on the token request and attempts to verify the control process. KS11 posts the outcome onto the “Verifies” level. Since the token type is a control request, KS12 gets triggered and performs its actions and prepares the response message for the control request token prior to posting an entry to the “Propagates” level. Anything posted on this level, is always propagated to the client it is destined for, by the KS5. If the KS12 posts a reply for the control request token at this level, KS5 propagates it to the client which generated the request.

If the token type is a control message, KS11 first verifies whether the client has already acquired control of the machine and posts the outcome on the “Verifies” level. If the client controls the machine and needs to propagate this token then KS7 works on it and transfers the control message to the “Machine” panel. If the client does not control the machine (KS11 decides this) then KS12 prepares the reply message and KS5 propagates the denial message to the original client source.

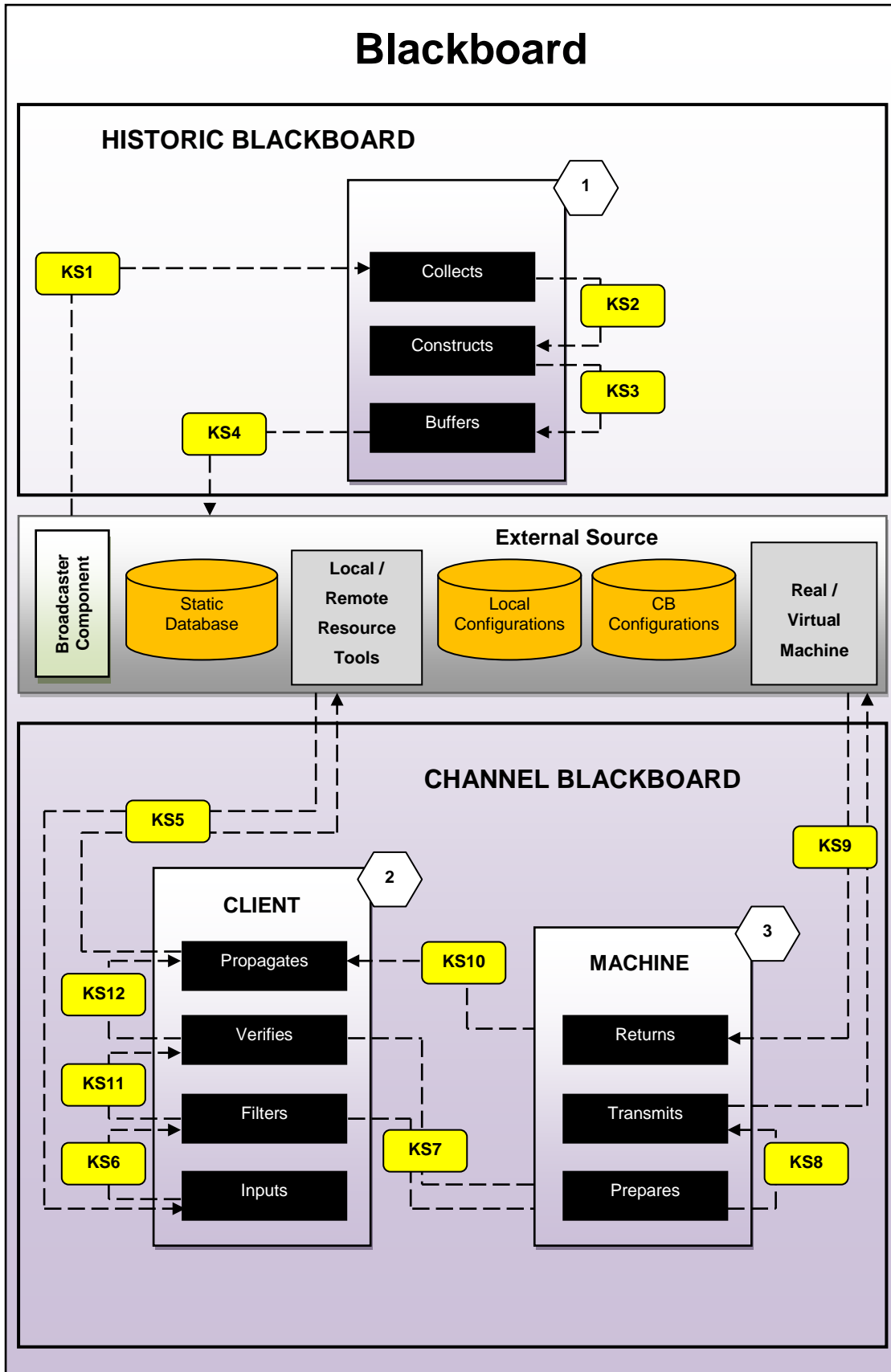


Figure 6-6: Marshaller Model

Machine Panel

As the name suggests, this panel deals with the communication and control processes associated with the machine (i.e. real and / or simulated). The hierarchy of this panel consists of three abstraction levels namely, “Prepares”, “Transmits” and “Returns”. KS7 is responsible to transferring information from the “Client” panel to this panel. There are two types of information transferred by KS7 onto the “Prepares” level, either a control message or a unit query. Presence of both this information triggers KS8 which distinguishes them and posts the entry on the “Transmits” level of the hierarchy accordingly.

KS9 is responsible to transmit these tokens from “Transmits” level to the machine source in their required format. Machine source replies with information related to the type of request message sent by KS9. The same knowledge source collects the responses and posts them back onto the “Returns” level. Since responses need to be transmitted to the client resource tool that generated a request, KS10 transfers the token response to the “Client” panel - “Propagates” level. As mentioned earlier, KS5 keeps track of the client’s requests on this level, therefore, it makes the decisions as to which client generated the request and thus replies accordingly. All the request-response tokens are transmitted in the uniform XML format making the implementation independent of the actual targets (i.e. clients and machine).

6.2.4 Web-HMI Blackboard Model

Organisation of the Web-HMI system component model is also influenced by some target requirements supporting the required level of functionality within this system. These are:

- The component is mainly targeted to provide machine playback support by integrating its logic with a simulation model (represented using VRML techniques). The simulation model provides a 3D representation of a manufacturing machine.

- The component seeks to monitor machine status by subscribing to the live feeds published by the Broadcaster system component. Any updates to the machine's status need to be reflected on the operator interface screens.
- The component controls access of various HMIs to the Marshaller system component by implementing local access logic, and handling any bidirectional communication involved in doing so.

Based on this, the structure of Web-HMI's model is divided into three different panels namely "Monitoring" panel, "Control" panel and "View" panel as shown in the figure 6.7. Following is an explanation for each panel with its associated knowledge sources within the Web-HMI model.

Monitoring Panel

This panel is responsible for handling all the tasks associated with machine monitoring by subscribing to the live feeds published by the Broadcaster. The hierarchy of this panel consists of three levels i.e. "Inputs", "Processes" and "Updates". KS1 is a dedicated source whose "Signal" is never turned to "No" as it is responsible to receive data from the Broadcaster source continuously (more information on this "signal" parameter is given in the section 6.2.5). Any machine transaction data is always loaded (i.e. posted) onto the "Inputs" level by the KS1, creating a new "Add" event type. This triggers KS2 to carry out necessary processing by interpreting and extracting important real-time information from the received machine xml structure. Incoming data from the Broadcaster component can be machine states, machine faults, machine operational information and CB configurations. The resultant simplified form of the data is written onto the "Processes" level by the KS2.

Availability of the processed data triggers KS3 to further sort it in order to update the corresponding Web-HMI operator interface screen(s) or its records. Based on the type of the simplified data and the internal system logic, KS3 arranges data for its corresponding HMI screens. The outcome is posted on the

“Updates” level by KS3. KS4 updates all the operator interface client screens simultaneously when any information is written on the “Updates” level. The information flow in the “Monitoring” panel is top-down in nature as machine information on the higher level is processed, extracted and prepared into a lower simpler form ready to update the corresponding HMI client screens.

View Panel

This panel supports a 3D view of a machine using a VRML simulation model integrated within the Web-HMI component, and any request to display historical information on the screens. Its hierarchy consists of two levels namely, “Instructions” and “Representations”. In order to view machine simulation representation (for example, view machine fault), an operator initialises a session on the operator interface client screen and KS5 loads the necessary instruction type and additional request parameters on the “Instructions” level. The instruction type can be a machine fault view, machine playback view, machine live mimic view or historic information view. Machine fault view displays 3D machine fault location on the simulation model. Machine playback view supports playback of machine transactions within this system. Machine live mimic view displays machine’s execution state at real-time on its 3D machine simulation model. Historic information view displays historic fault information on the operator interface screens, for example error history. As explained earlier, simulation functionality is a very useful feature which supports maintenance process and early machine system verifications.

Presence of new instructions triggers KS6 to refer to the corresponding information from the static database and write the resultant outcome on either the “Representations” level of this panel or “Monitoring” panel – “Processes” level. If the outcome is posted on the “Representation” level, KS7 reads the extracted information and passes necessary display parameters to the VRML simulation model for display. If the instruction is of the type historic information, it gets propagated to the “Machine” panel – “Processes” level only to be processed by KS3 and sent to the operator interface client screens using KS4.

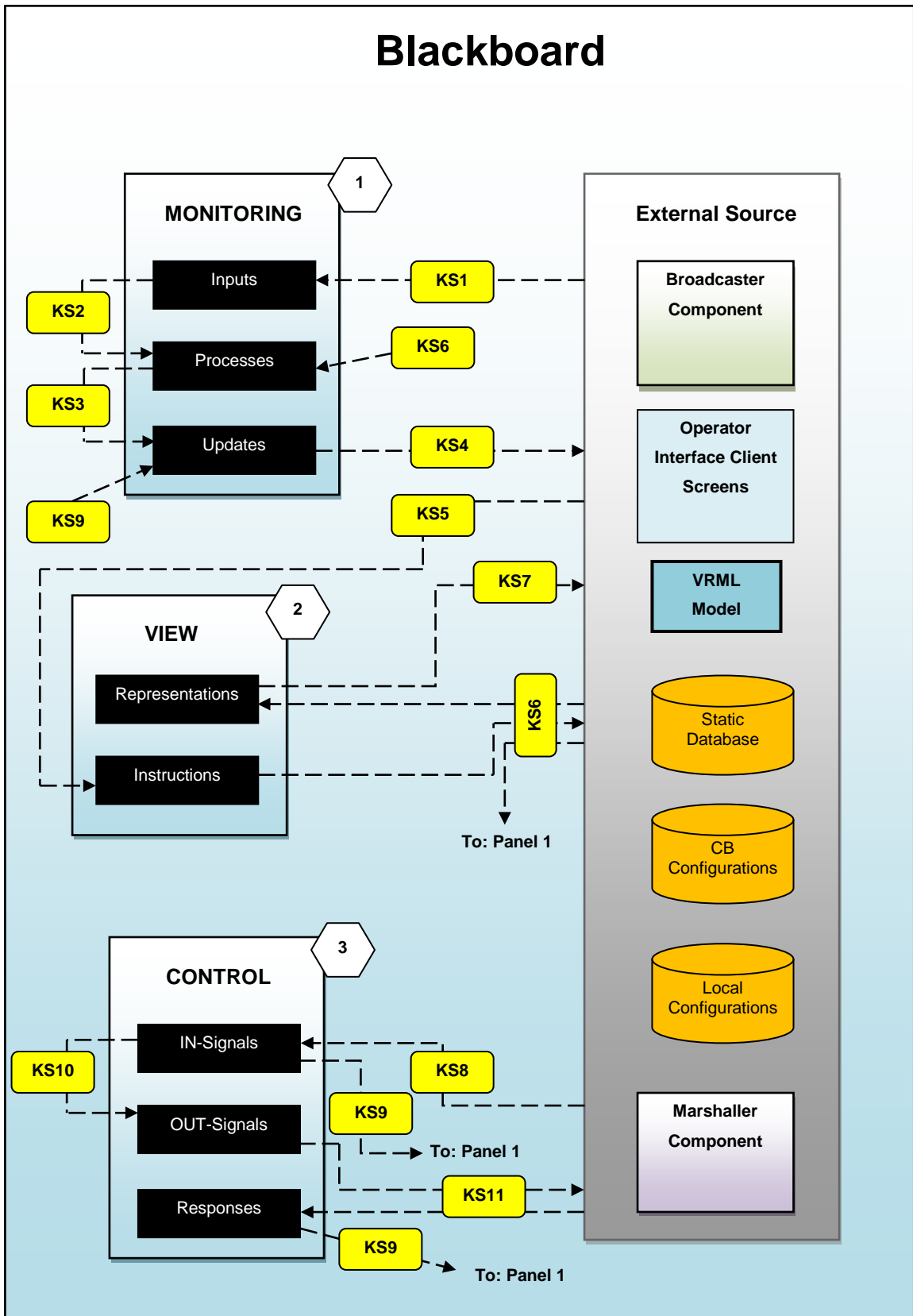


Figure 6-7: Web-HMI Model

Control Panel

This panel handles the required control mechanism for this system component. All the operator interface client screens (i.e. clients) are served by the Web-HMI component, which acts as a server processing many simultaneous connections, regardless of the nature of the clients or their locality. This panel plays an important role in controlling the access of all the clients, by managing any conflicts which arise when a client expresses interest to the Marshaller for controlling and querying a machine, and acquires necessary responses in return.

Its hierarchy consists of three levels namely, "IN-Signals", "OUT-Signals" and "Responses". Any incoming signal in the form of a token received from an operator interface client screen is processed by KS8. This knowledge source applies the authentication regime on the clients in order to determine the permissions associated with them. As described earlier in section 6.2.3, communication messages between the Web-HMI component (or any other client resource tool) and the Marshaller component can be of three types, i.e. control request, control message or unit query.

If the incoming signal is of the type control request then KS8 applies a local logic to determine if the operator interface client screen can control a machine or not. If not, in a situation where other operator interface client currently holds the machine control, then KS9 gets triggered. KS9 transfers the necessary denial information to "Monitoring" panel – "Updates" level, KS4 updates the client screen which generated the request. If none of the connected operator interface screens control the machine, KS10 is triggered by an event on "IN-Signals" level which in turn processes the request and posts it onto the "OUT-Signals" level with the control request parameters. KS11 propagates the message to the Marshaller component. Any response from the Marshaller is posted by KS11 onto the "Responses" level. This triggers KS9 to transfer the response to "Monitoring" panel.

Suppose the incoming signal from the operator interface screen is of type control message, KS8 checks if this HMI controls the machine. If it does then

KS10 propagates the message to the next level and KS11 transfers to the Marshaller. If this HMI does not control the machine then KS9 gets triggered (instead of KS10) and the resultant denial is sent to “Monitoring” panel by KS9.

Suppose the incoming token is of type unit query then KS10 is triggered and the output is sent to the Marshaller via KS11. Any replies from the Marshaller will be handled in the “Responses” level by KS11. KS9 transfers the responses to the “Updates” panel for propagating the message to the HMI client which generated the token.

6.2.5 BB Component – Knowledge Source Structure

Sections 6.2.2, 6.2.3 and 6.2.4 covered BB components design model for each system component from the blackboard BB component perspective, but did not demonstrate how the knowledge sources have been structured to enable machine data to be transmitted from one blackboard level or system to the other. This section shows the generic configuration (structure) of a knowledge source which acts as a template with which all the knowledge sources (within all the system components) have been designed and implemented.

Since every KS is specialised to contribute knowledge for a specific task, its implementation is either procedural (i.e. procedure or function) or a complete system. The generic structure of KS is shown in the figure 6.8 where its configuration is divided into two groups, accommodating six fields in total. The description of each field is given as follows.

The first field (“**UType**”) is universal type identification for the knowledge source which specifies its domain, and a unique id made up of its name and a unique tag number, for example, WKS1. WKS1 shows that it works on a Web-HMI component and its name is KS with an id of 1. The second field (“**Signal**”) is an activation switch which corresponds to either being “Yes” or “No”. If the signal switch value is “No”, it means that this knowledge source cannot be executed and if “Yes” then vice versa. These first two fields make up group1 and are declared only once in the knowledge source’s configuration. Group1 cannot be duplicated across the system.

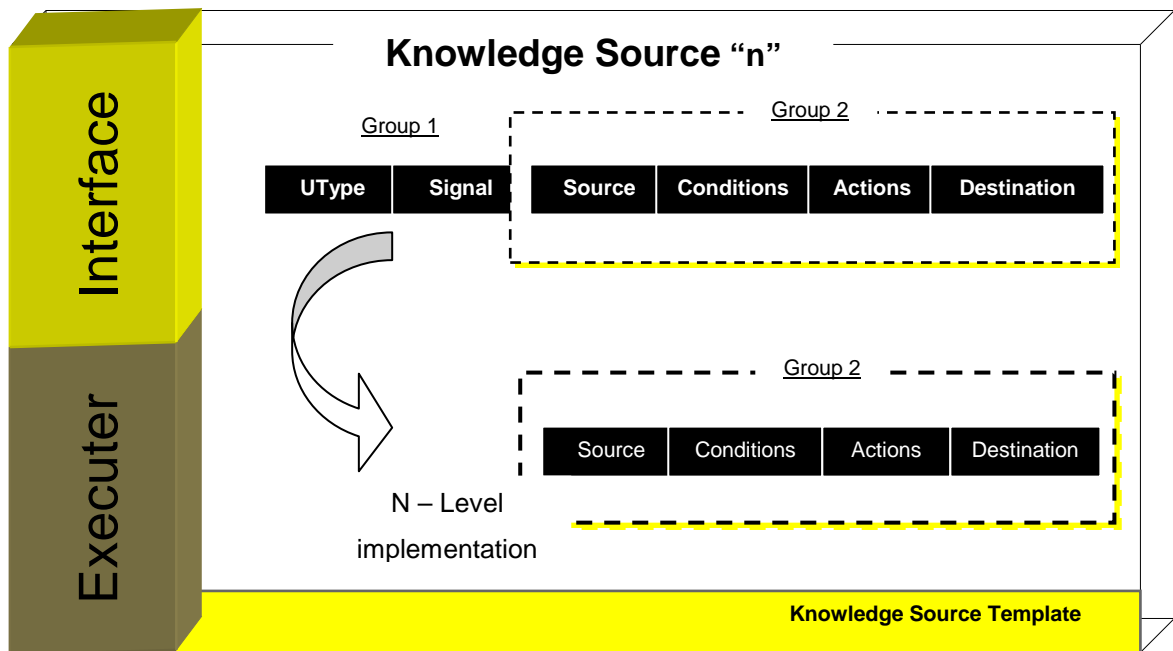


Figure 6-8: Knowledge Source Structure

The third field (“**Source**”) specifies the source address of the blackboard where this knowledge source is interested. For example, 132.217.83.209/B1/P2/L3 can be a source address which shows that this knowledge source is interested in the blackboard B1 panel P2 level L3. Every source address is preceded by an IP address of the system where the system component is operating. This promotes distributed operation of the knowledge source over a Wide Area Network (WAN) such as the Internet.

The fourth field (“**Conditions**”) comprises of the conditions and sub-conditions joined by Boolean operators like AND, OR, etc, if needed. The fifth field (“**Actions**”) correspond to the procedures and / or functions that must be executed when the conditions are satisfied. The seventh field (“**Destination**”) specifies the address of the destination where the output of the computation will be written to. The format of this field is similar to the previously described example for the source field.

For every group1 within the knowledge source “n”, there exist a number of further groups 2 consisting of four fields each. Since a knowledge source can

operate on 1 or more levels therefore every knowledge source has an N-level implementation of group 2 fields. This means that a knowledge source “n” can for example, operate on 2 different blackboard sources executing different actions at once. Every knowledge source has a well-defined “Interface” which promotes to its modularity. Each knowledge source has an “Executer” logic element which is explained in the section where the controller’s operation for the blackboard system is detailed.

6.2.6 BB Component – Controller Operation

Sections 6.2.2, 6.2.3 and 6.2.4 covered BB components design model for each system component from the blackboard BB component perspective, but did not demonstrate how the controller operates to schedule knowledge sources to contributing their speciality at the required blackboard panel’s level. Controller of each system component (i.e. namely server for the Web-HMI, controller for the Broadcaster and manager for the Marshaller) operates using event-invocation scheduling [202] (using publish / subscribe communication mechanism as described in the chapter 3). Since this is an open-ended problem where the goal cannot be determined owing to the continuous execution nature of the application, event-invocation scheduling technique is the best suited approach [116, 197]. With this approach, the publisher and the subscriber (both being KS BB component) are decoupled using the broker (i.e. blackboard and controller BB component) [157].

Structurally, the controller mechanism has been separated into three logic elements namely “Catalogue”, “Scanner” and “Executer” as shown in the figure 6.9. In order to clarify the responsibility of each logic element of this controller mechanism, and to understand the interaction of knowledge sources with a blackboard BB component, an example has been given in the figure 6.10.

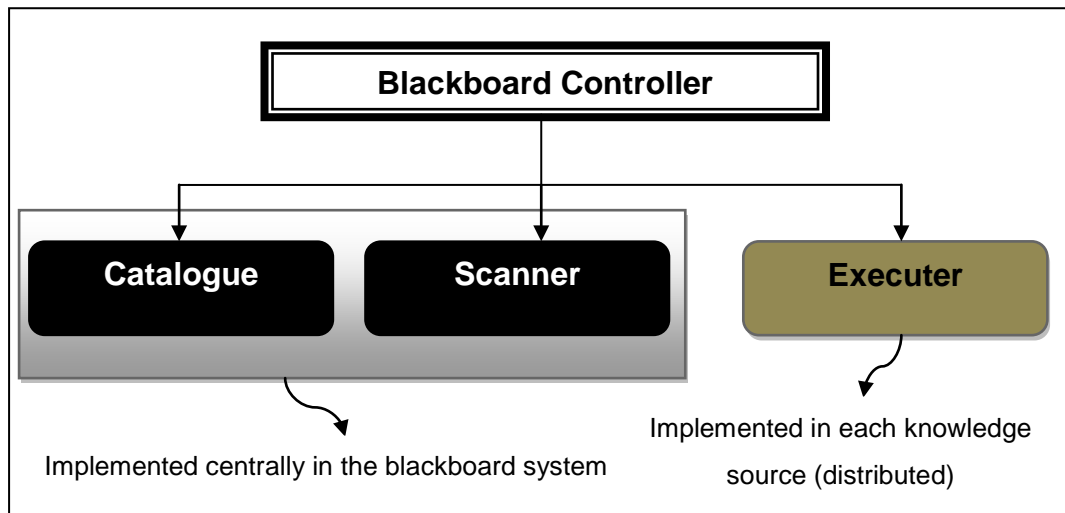


Figure 6-9: Controller Division

Example in the figure 6.10 assumes that that a blackboard BB component with multiple panels (each having a well-defined distinctive hierarchy) exists. Furthermore, a number of unique knowledge sources also exist in the system component. When a blackboard system starts operating, the “Catalogue” logic element’s responsibility is to register all the KS in a dynamic tabular record with their details at runtime as shown in the figure 6.10.

When the state of a blackboard changes (i.e. a new entry appears), an event is generated and this can be used to trigger the KS. Instead of all the KS scanning the blackboard continuously, each KS registers an interest by informing the “Scanner” indirectly (i.e. via “Catalogue”) on the type of information (i.e. event) it would like to contribute to. The “Scanner” has the responsibility to continuously monitor the blackboard and record all the events being raised at runtime, and populate its dynamic tabular record as shown in the figure 6.11. When a specific event type is detected by the “Scanner”, it is registered in its dynamic tabular record (figure 6.11). Its logic immediately scans the catalogue’s record (figure 6.10) to find all the corresponding knowledge source(s) registered for this event based on the “Source” address. As soon as it finds the corresponding knowledge source(s), “Catalogue” immediately changes the knowledge source’s “Signal” to “YES”. This triggers the knowledge source causing their respective “EXECUTER” logic element to read any necessary parameters (i.e. entry represented as objects) from the source blackboard.

Runtime Tabular Record – Catalogue Element

Cat #	UType	Source	Signal
CT1	WKS1	132.217.83.209/B1/P2/L3	NO
CT2	WKS1	132.217.83.209/B1/P1/L1	YES
CT3	WKS2	132.217.83.209/B1/P2/L3	NO

Note:

- One knowledge source can operate on one or more levels as shown in CT1 and CT2. In this case, group 2 of knowledge source “WKS1” will have 2 configurations, each corresponding to each source.
- One or more knowledge source(s) can operate on the same level as shown in CT1 and CT3. In this case, CT1 and CT3’s corresponding knowledge source’s group 2 configurations may or may not be similar.
- By default, all the knowledge sources’ signals are switched to “NO”, meaning they cannot execute at this time. The only KS whose signal is always “YES” deals with posting the original entry on the blackboard when new data arrives from an external source as shown in CT2. The main KS will never switch its signal to “NO” as it needs to collect data from external source for the system to continue processing.

Figure 6-10: Runtime Catalogue Logic Element's Record

The executer’s logic applies its conditions on these parameters. If the condition(s) is / are satisfied (based on the event type), the corresponding action(s) are executed by the executer logic and the output is written to the destination field’s address. When this output is successfully written to the blackboard, the executer switches its “Signal” back to “No” and updates this in the catalogue’s record shown in the figure 6.10. When the output is written to the blackboard, new event occurs (based on the entry type) and the same cycle continues until the system is halted. This approach is beneficial in terms of addressing system-level properties like performance, reusability, reliability and security. This overall process is clearly shown in the figure 6.12 for incoming entry of the type X¹.

Runtime Tabular Record – Scanner Element

Scan #	Source	Entry	Event Type
S1	132.217.83.209/B1/P2/L3	X ¹	e1
S2	132.217.83.209/B1/P1/L1	X ²	e2
S3	132.217.83.209/B1/P2/L3	X ³	e3
...

Note:

- An entry can only exist on one level at a time.
- One or more entries can exist at one level at the same time as shown in S1 and S3.

Figure 6-11: Runtime Scanner Logic Element's Record

A simplified expression for comparing the “Source” and switching the signal to “Yes” is pseudo-coded as shown below:

```

For each record in "Catalogue" corresponding to "Source"
  If Runtime.Source = Catalogue.Source then
    SWITCH Catalogue.Signal="YES"
  End If
Next
    
```

All the knowledge sources in the system execute in parallel and can read entries from the blackboard simultaneously. In order to avoid deadlock, only one knowledge source is allowed by the controller to write data (i.e. an entry) in the same panel's level at one time. The concept of executing knowledge sources in parallel applies to the overall blackboard system but not to individual panel, meaning, the controller executes the knowledge source in sequence at each panel but in overall, multiple knowledge sources can execute simultaneously provided that they do not operate and write on the same blackboard panel. If they operate on the same panel then sequential execution constraints are

applied onto them and write access is given to only one knowledge source per panel. Next chapter describes the overall runtime operation for the system components corresponding to a runtime architectural description of the solution, and their respective implementations.

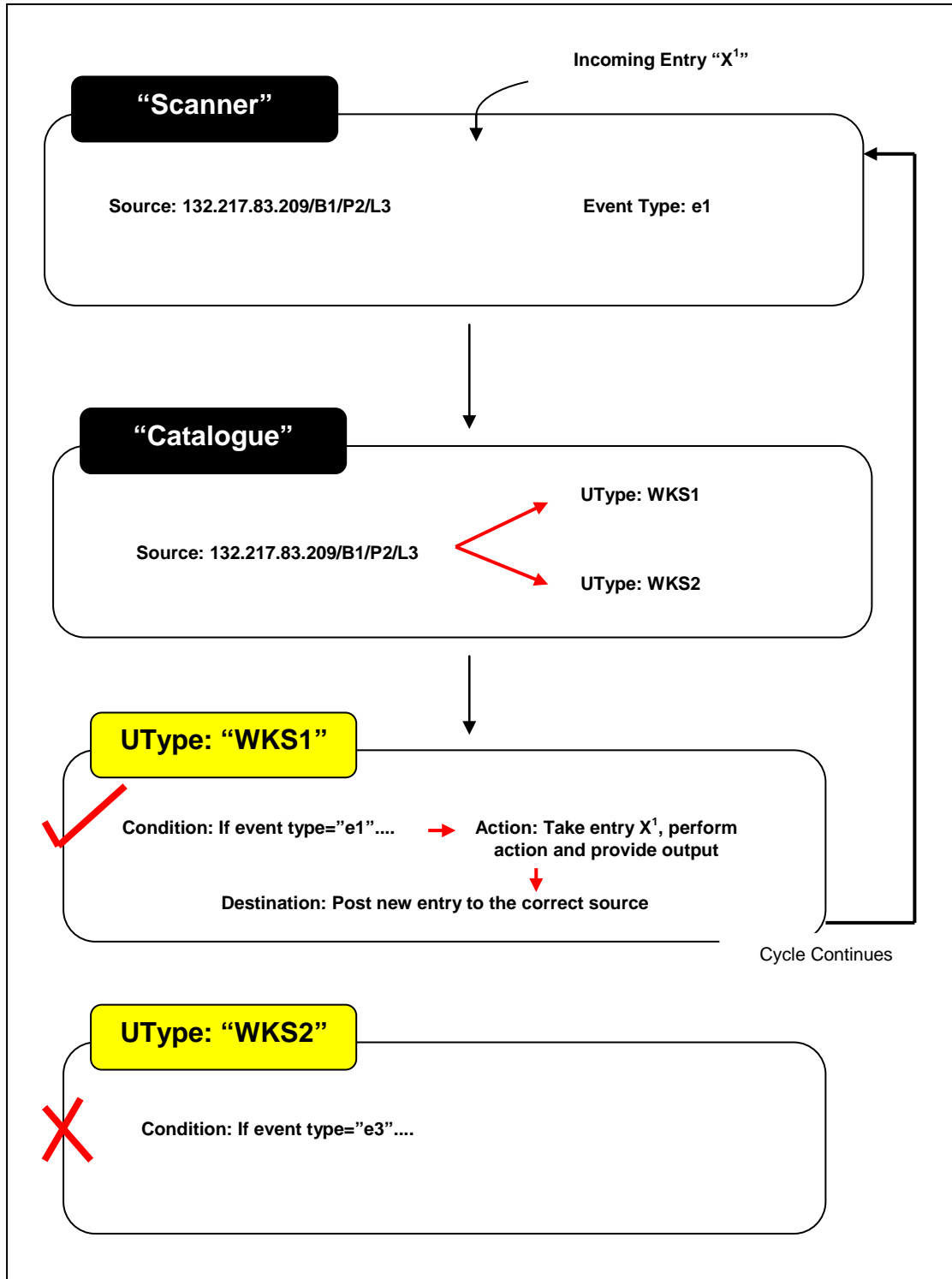


Figure 6-12: Blackboard Processing Example

Chapter 7 : System Components Implementation

Chapter Contribution to this Thesis:

This chapter's main contributions are the process runtime implementation description for each system component and the Web-HMI's templates representation layout that maps to a machine's CB model description at runtime, providing reconfigurable and consistent display to all the stakeholders through the machine lifecycle.

7.1 General Overview

This chapter describes a detailed runtime implementation process for the Broadcaster, Marshaller and Web-HMI system component. An overall system runtime operation is outlined first followed by detailed implementation descriptions for each system component, satisfying functional and non-functional requirements identified in the chapter 5.2. Furthermore, the design models described in the chapter 6 are implemented using object-oriented techniques and communicate using socket-based links, a representation of which is also covered within this chapter.

7.2 Overall System Runtime Operation

For supporting close control and monitoring activities throughout a machine lifecycle, an implementation of the overall system runtime operation has been illustrated in the figure 7.1. Initially, the previously referenced CCE process engineering toolset (chapter 3.2.2) [122] manages creation of machine components (with their behaviour definitions) and stores them in a library for future use (ref 1 in the figure 7.1). For any reconfiguration requirements or machine design and build process, these components can be edited using the same toolset (ref 2 in the figure 7.1), and either be installed (for a real machine) or be simulated (for a virtual machine) (ref 3 in the figure 7.1). One of the outputs from this engineering operation is the CB machine model description which can be shared throughout the control and monitoring system architecture

(chapter 5.3) with the help of the Broadcaster system component (ref 4 in the figure 7.1).

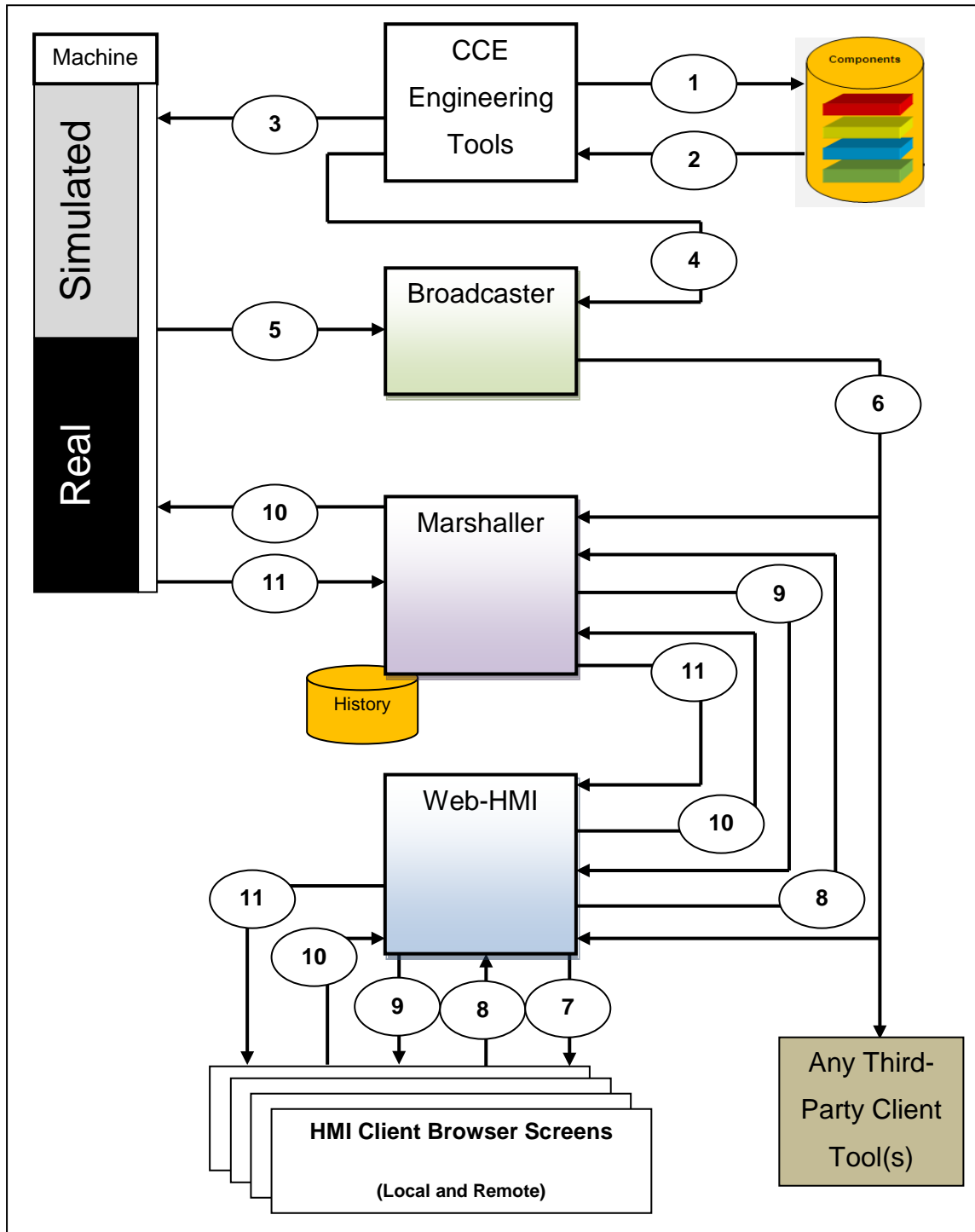


Figure 7-1: System Components Runtime Operation

When machine operates (whether in real, simulated or hybrid mode), its status is continuously monitored using the Broadcaster (ref 5 in the figure 7.1) which in turn processes, prepares and streams it to various clients (for example, the Web-HMI system component) using TCP/IP socket communication [203] in XML based data format (ref 6 in the figure 7.1). Web-HMI serves various HMI client browsers by returning standard template-based HTML pages populated with these CB machine model descriptions and initial current machine status data. Furthermore, any machine status updates are propagated to various HMI client browsers using XML-based objects as described in the section 7.5.2 (ref 7 in the figure 7.1).

When an HMI client browser intends to control a machine, it transfers a control request to the Marshaller via the Web-HMI (ref 8 in the figure 7.1). The Marshaller permits (or denies) machine control to the HMI client browser which initiated the control request (ref 9 in the figure 7.1 through applying its machine control sharing mechanism as discussed in the section 7.4.4). If the HMI client browser obtains machine control, it can start controlling the machine operation by sending control commands (ref 10 in the figure 7.1), for which the machine acknowledges their receipt (ref 11 in the figure 7.1). Depending on the current state of the machine, these commands can be executed by machine control logic and updated status is propagated to the Broadcaster system component (ref 5 in the figure 7.1). Next sections detail implementation description for each system component designed within this research.

7.3 Broadcaster Implementation

7.3.1 I / O Outline

This section outlines Broadcaster's operational boundary showing major inputs and outputs supporting its monitoring functionality within CB implementation. Figure 7.2 shows I / O outline where only external links are presented. As an input, CB model is serialised and prepared to be propagated to clients when they connect. Any machine status changes such as state change, emergence of errors or operational information are collected, processed and transmitted to

clients using well-defined interfaces as described in the section 7.3.3 of this chapter. To appreciate its internal workings, a description of its process runtime implementation is provided next.

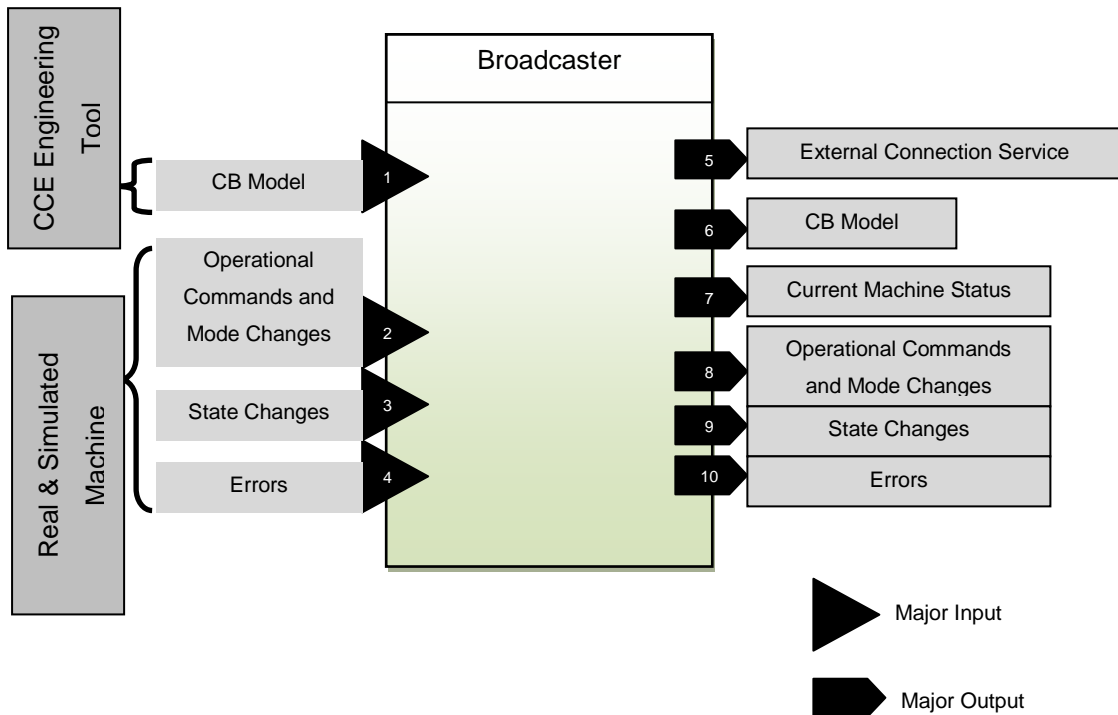


Figure 7-2: Broadcaster I / O Overview

7.3.2 Process Runtime Implementation

The principal functionality of the Broadcaster is to simultaneously propagate machine events to a number of clients regardless of their geographical locations or their system implementation types. Figure 7.3 shows a process runtime implementation for this system component. It has two TCP/IP socket connections permanently open for listening to connection requests, one from a machine and the other from client(s). Each socket operates with a different configuration as specified by a user using its graphical interface (illustrated in the section 7.3.5). When a connection request is detected by the respective sockets, individual thread is created to handle each connection type to enable either client subscriptions or machine publications to raise a corresponding event in the Broadcaster's blackboard. If the corresponding event is of the client

request type, instances of client socket(s) are created by their listening socket upon connection and subscribed to the published streams. If the corresponding event is of the machine type, data entries are processed and stored with the help of management interfaces (described in the section 7.3.3) into reconfigurable memory buffer (described in the section 7.3.4).

There are separate read and write pointers to access this memory buffer. There is only one single pointer that writes to the buffer where as there are multiple read pointers, each created for every client request and its reference is provided to the corresponding dedicated client socket instance. Data (in the form of CB model description, current machine status, state changes, errors, or operational commands and modes) is prepared and streamed to the connected clients.

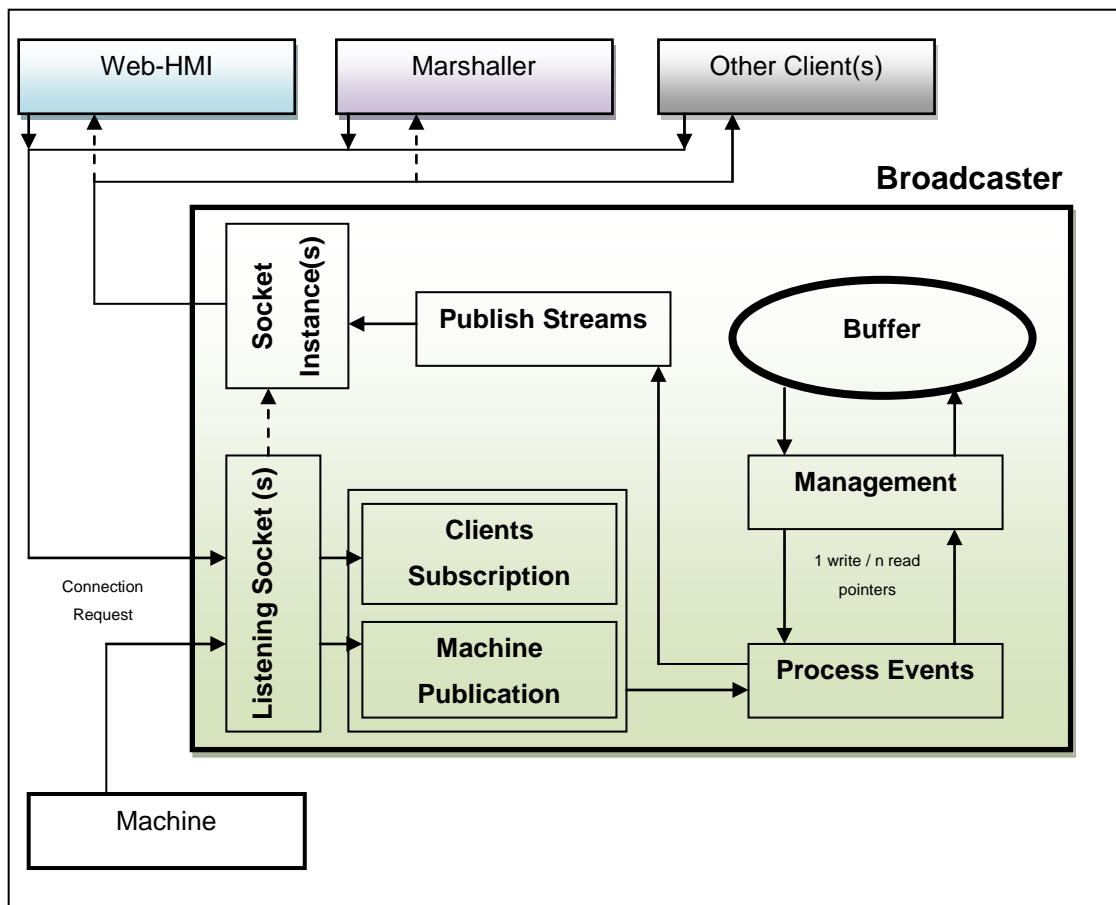


Figure 7-3: Broadcaster Process Runtime Implementation

7.3.3 Interface Description

Broadcaster system component model described in the chapter 6.2.2 has been implemented using object-oriented techniques [204], a description of which is covered within this section of the chapter. KS being a machine data carrier and processor, it can either be represented using set of functions or procedures as mentioned in the chapter 6.2.5. These representations in object-oriented programming (OOP) field correspond to objects; therefore a single KS responsibility can be managed by implementing one or more object representations depending on its processing burden, improving the overall modularity and performance of the system.

For demonstrating its implementation process, KS1 of the Broadcaster blackboard model is taken as a reference example. KS1 is responsible for carrying out various tasks such as:

- Loading local system configurations entered by users,
- Loading CB model configurations created by CCE engineering tools,
- Initialising communication with external sources such as production machine and clients like Web-HMI,
- Collecting continuous incoming machine data and
- Posting the collected data onto the “Boots” level of the panel “In-Load”.

Each of these tasks has been represented as “objects” in the form of methods as shown in the figure 7.4. These methods form the object’s interface with the outside world [205] and carry out all the responsibilities required by the KS1 in the “Boots” level. This is a logical representation which bears no resemblance to the actual implementation of the Broadcaster system. In actual implementation terms, each of these objects has been implemented using a class within the system, an example of “SystemLoader” class is shown in the figure 7.5 using the UML notation [139]. Declaration of this class includes attributes such as machine name, machine port, system IP, client port and memory queue size configurations. Likewise, all the major methods have been implemented as individual or grouped classes using Visual Studio® toolkit and dot Net programming framework [45]. These KS classes implement Broadcaster’s

controller interfaces (described next) which provides a level of abstraction within the system code, improves code organisation as well as supports the system growth when new features will be desired [205].

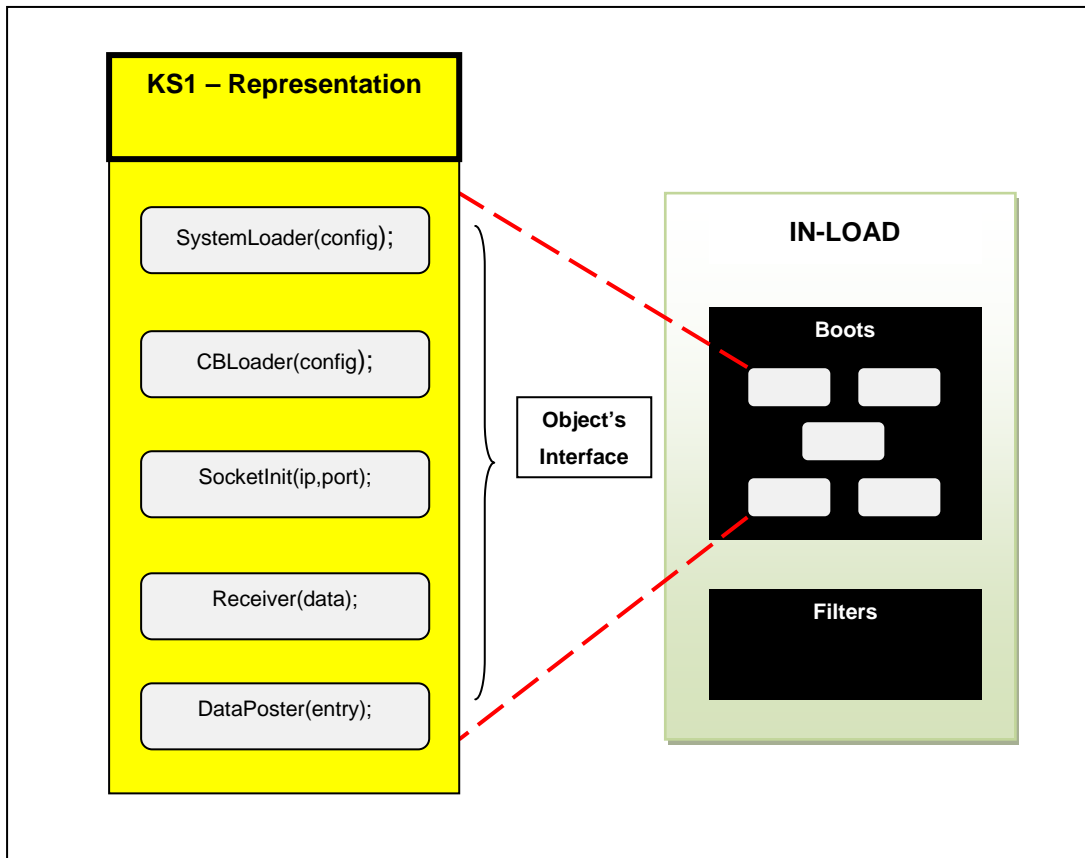


Figure 7-4: Broadcaster KS1 Interface Representation

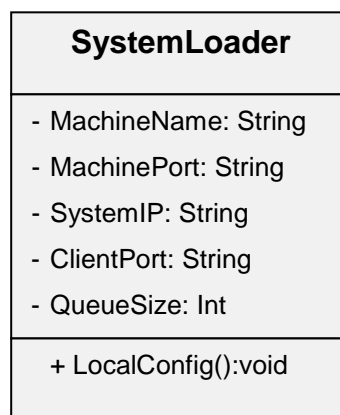


Figure 7-5: Broadcaster KS1 "SystemLoader" Class

Broadcaster Control

Broadcaster's controller (known as "Broadcaster Control") is actually implemented using a set of interfaces. They manage and schedule all the execution activities for Broadcaster system component using multi-threading techniques. They are organised into logical grouping and implementable interface groupings as shown in the figure 7.6. Implemented interfaces enable encapsulation of various KS objects (instantiated from their respective classes), for example, an object instantiated from the class "SystemLoader" (as described previously) is encapsulated inside "Configuration" interface of the "Registry" group. There are four logical groups as follows:

- Mechanism Management: This logical group has four implemented interfaces within the system component, namely ClientManager (deals with client lifetime management), MachineSet (deals with machine management), GUIView (deals with user interface updates) and Abstract Machine (deals with base communication instantiations).
- Dictionary: This logical group has two implemented interfaces within the system component, namely GlobalDictionary (deals with data definition management) and ControlParser (deals with parsing machine messages).
- Blackboard Management: This logical group has three implemented interfaces within the system component, namely QueueData (deals with data propagation management), CircularQueue (deals with dynamic memory storage – described in the next section) and FrozenState (deals with client state management).
- Registry: This logical group has four implemented interfaces within the system component, namely Validation (deals with data validation), Configuration (deals with configuration management), Translation (deals with necessary internal translations) and Instance (deals with system component instance management).

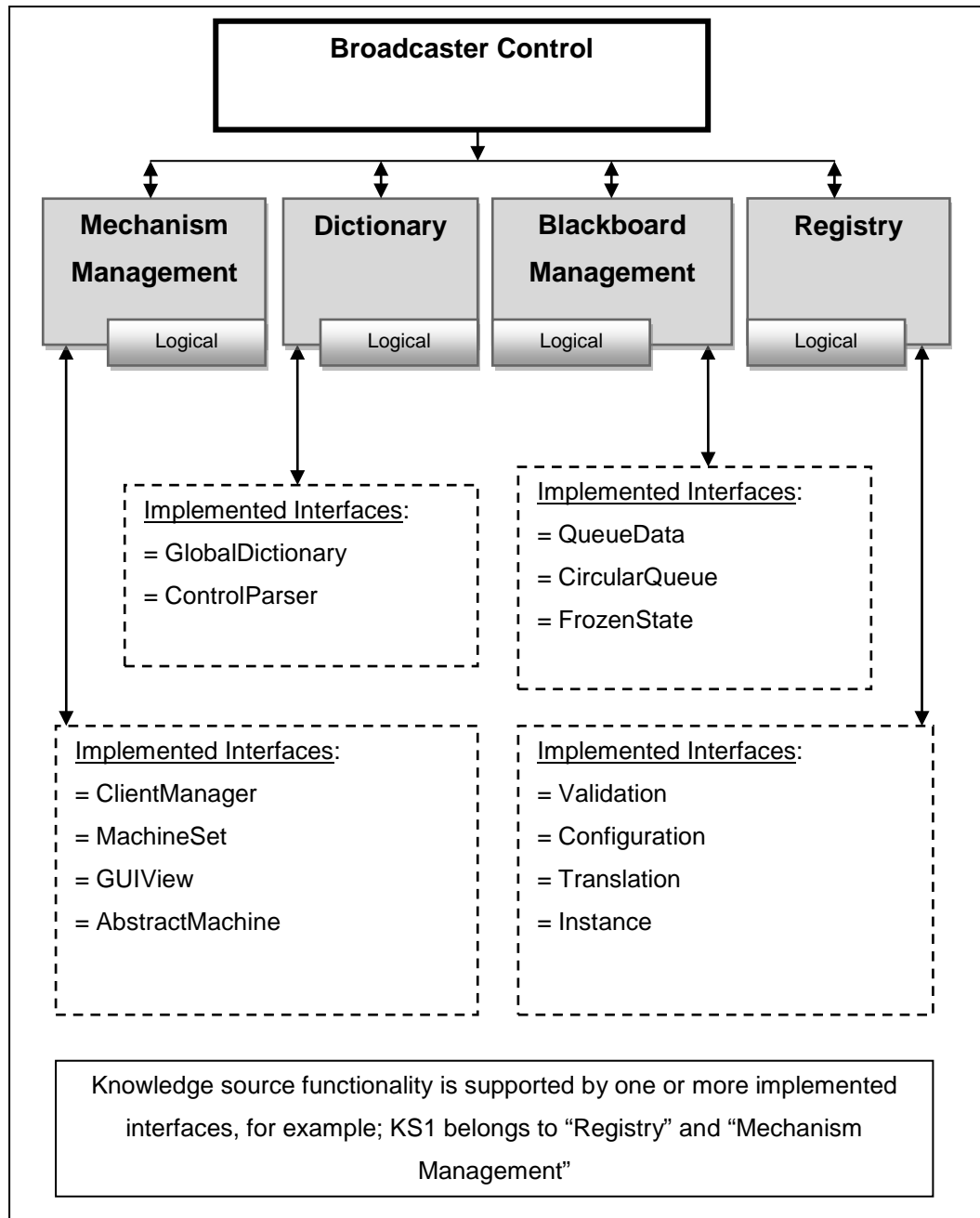


Figure 7-6: Broadcaster Control Interfaces

Propagation of collected machine data, after its processing has been carried out requires efficient data buffering mechanism. A reconfigurable buffer is thus employed within Broadcaster's implementation as discussed in the next section.

7.3.4 Reconfigurable Memory Buffer

As mentioned in the chapter 6.2.2, all processed machine data within the system component is stored in a reconfigurable memory for its easy and prompt access by clients. Since data transfer requires soft real-time access in applications of this type, there is always a risk of collecting data faster than actually processing and propagating it further, or vice versa [116]. To smooth out any speed differences, a circular data storage acting as a buffer queue is implemented. Using main memory rather than a traditional database as a repository allows KS contributions to be made at memory speed, significantly improving system performance when transmitting data from one source to the other at soft real-time basis.

This circular structure acts as a reconfigurable First in First out (FIFO) queue where data can quickly be saved and retrieved from by respective KS. Three types of data can be saved with their timestamps i.e. machine states, machine errors, and machine operational commands and modes. Current implementation converts incoming machine data (in textual string format) into a compressed integer-based unit, saving it into the buffer (with the help of “Blackboard Management” controller group).

Any incoming data is added to the next available slot in the buffer until it gets full; at this stage oldest unit in the buffer gets overwritten. When a client connects to the Broadcaster, it gets an instant access to the existing pointer location in the buffer storing current operational state of the machine. This leads to propagation of the current state of machine from the Broadcaster to its clients such as Web-HMI as shown in the figure 7.7. The circular buffer’s size is reconfigurable using a configuration user interface view within the Broadcaster system as discussed in the next section.

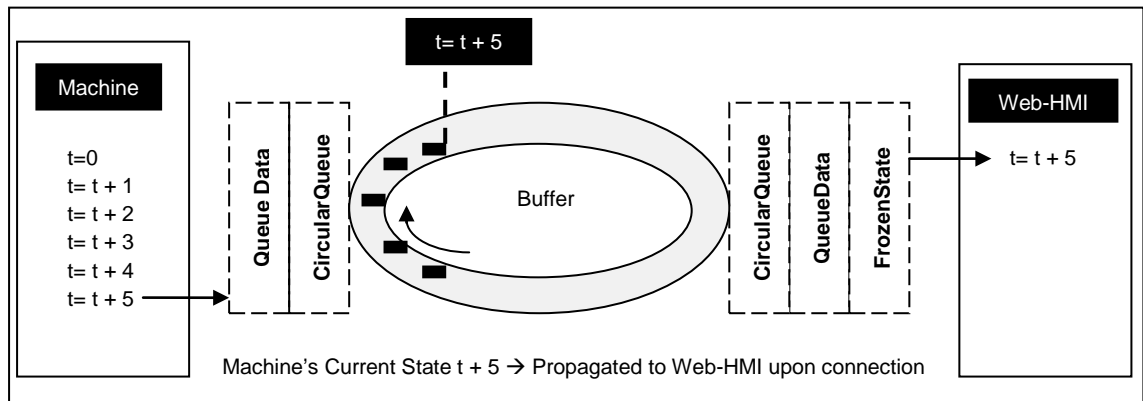


Figure 7-7: Reconfigurable Circular Queue

7.3.5 Graphical User Interface View

Currently, Broadcaster receives machine data in textual string format, processes it and transmits it in XML-based uniform representation to its clients upon their connections. Incoming data from machines can be monitored and necessary system configurations can be made using Broadcaster's graphical user interface. As soon as any machine data is collected and processed, it is displayed on its incoming messages section of its graphical user interface using a colour coded representation to distinguish data based on its type as shown in the figure 7.8.

Machine states, and machine operational commands and modes are presented in green colour, machine errors are in red colour and unfiltered / incorrect or corrupt data in purple colour. Furthermore, the same figure shows its configuration interface view which enables a user to configure system settings like IP addresses, Queue size, Machine name and Port details. Next section describes Marshaller system component implementation details.

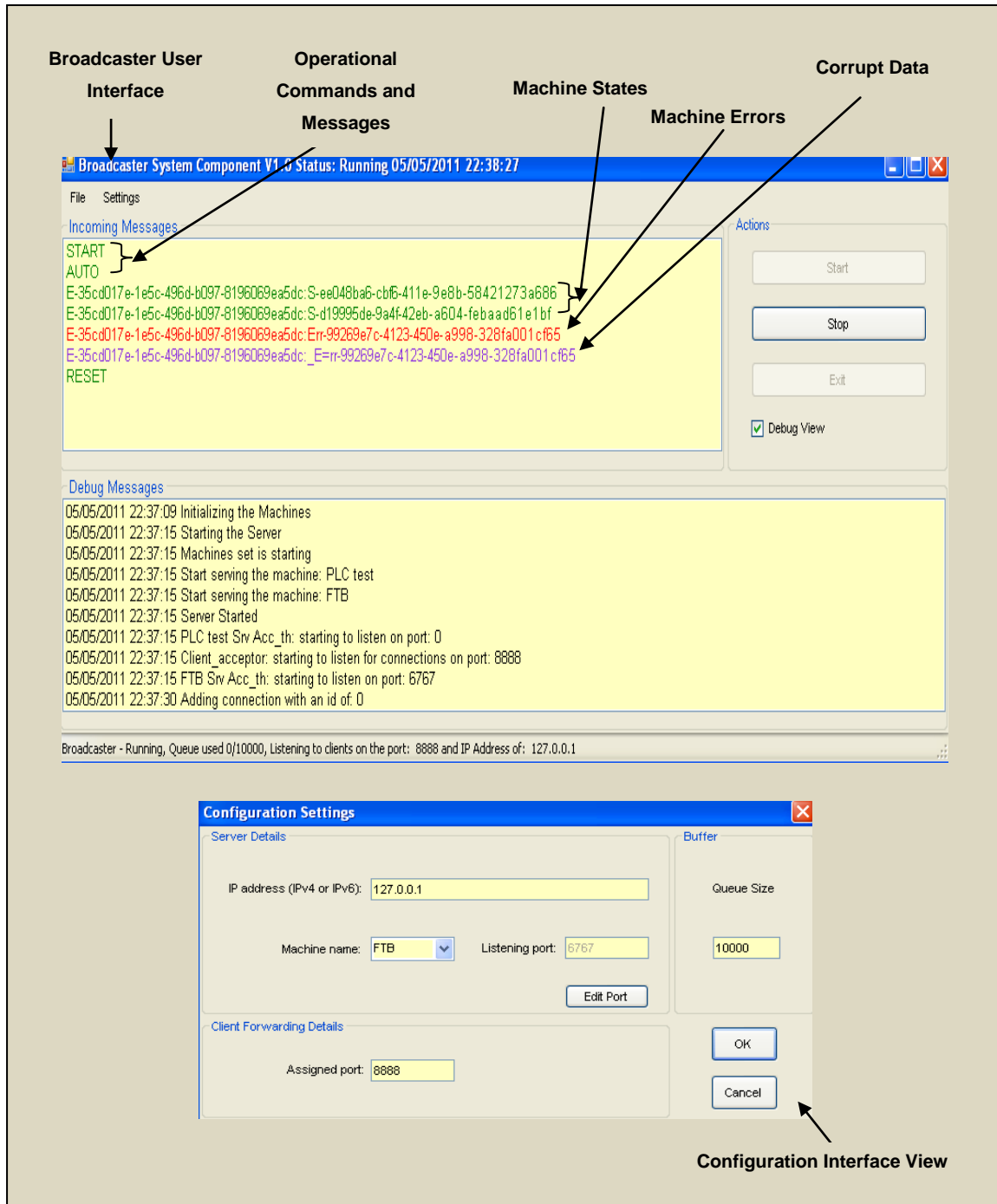


Figure 7-8: Broadcaster System Component Graphical User Interface

7.4 Marshaller Implementation

7.4.1 I / O Outline

This section outlines Marshaller's operational boundary showing major inputs and outputs supporting its control functionality within CB implementation. Figure 7.9 shows I / O outline where only external links are presented. As an input, CB model is received and de-serialised for supporting its marshalling functionality. Any machine status changes (i.e. state change, emergence of errors or operational information) are collected from the Broadcaster, processed and stored in a static data repository described in the section 7.4.3.

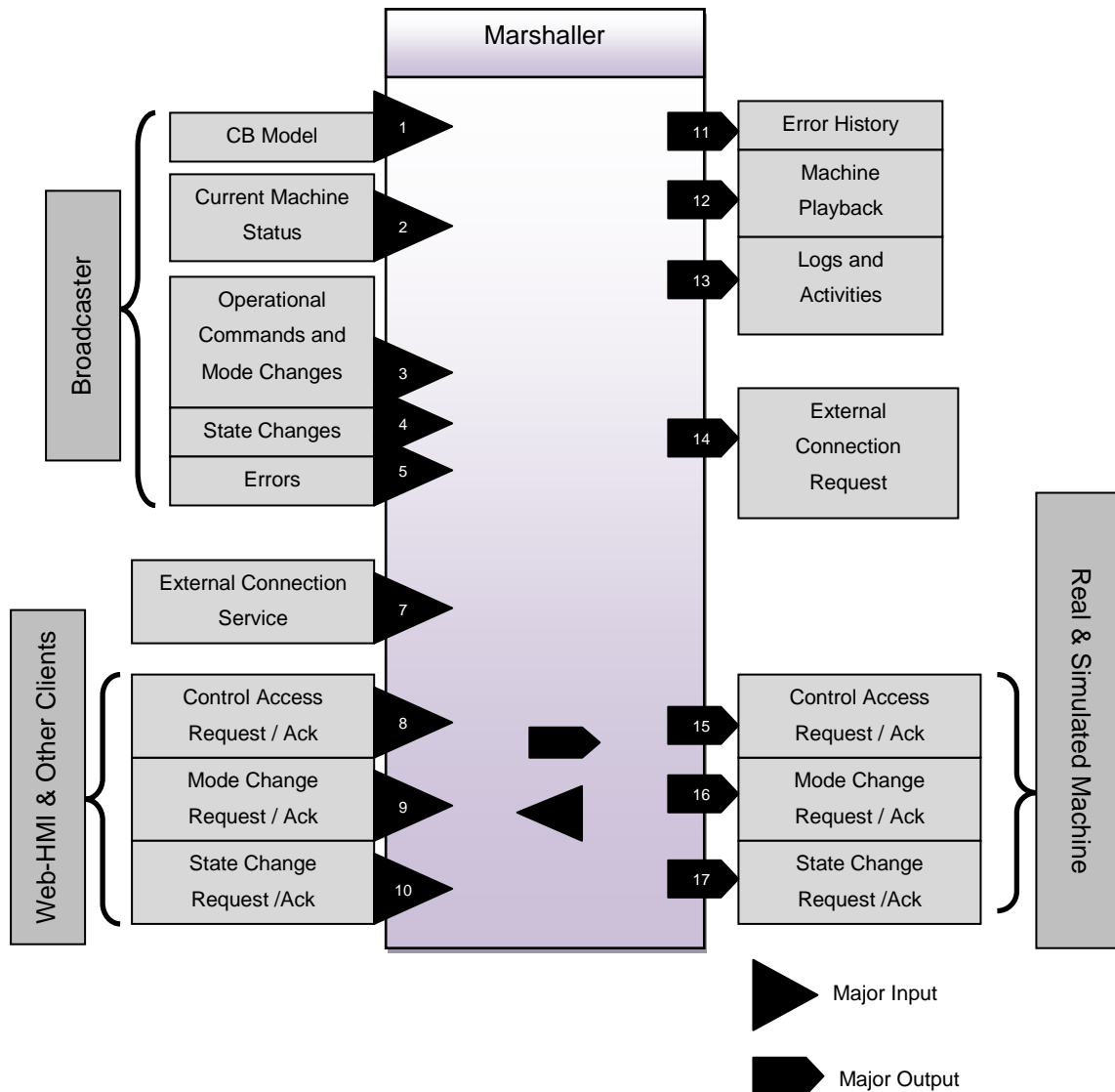


Figure 7-9: Marshaller I / O Overview

7.4.2 Process Runtime Implementation

The principal functionality of the Marshaller is to provide permission mechanism for operator interface system to control a machine, and to manage historical transactions for supporting additional functionalities within the system such as playback support, real-time machine mimic presentation, etc. Figure 7.10 shows a process runtime implementation for this system component. It has two communication channels where the first channel deals with interaction with the Broadcaster and the other deals with interaction between a machine and various clients for example, Web-HMI. These communication channel settings can be configured using Marshaller's graphical interface (illustrated in the section 7.4.5).

A TCP/IP connection is requested by Marshaller's caller sockets to the Broadcaster which in turn streams machine data over the established link. A separate thread is created handling this connection type to enable its published data to raise a corresponding event in the Marshaller's blackboard. This causes the received data to be processed, buffered and stored in a static SQL database (whose schema is described in the section 7.4.3).

Marshaller has a permanently open TCP/IP socket for listening to client requests. When a client such as Web-HMI establishes a socket connection with the Marshaller, a separate thread is created to handle client subscriptions to raise an event in the blackboard. Furthermore, instance of this client socket is created by this listening socket and subscribed to the published streams. These published streams can be control responses, mode responses or state responses from a machine.

To control a machine, the connected client requests permissions through the same established link. This request is managed by the machine management, which houses logic mechanism for machine control limiting only one client to control a machine at a time (described further in the section 7.4.4). If a control can be granted, a caller TCP/IP socket connection is requested with the machine, which in turn upon approval, streams necessary data to the Marshaller. A corresponding event is generated which streams data to the connected controlling client. Data can freely be exchanged between the

controlling client and the controller machine using the established socket link with the aid of machine management and process events.

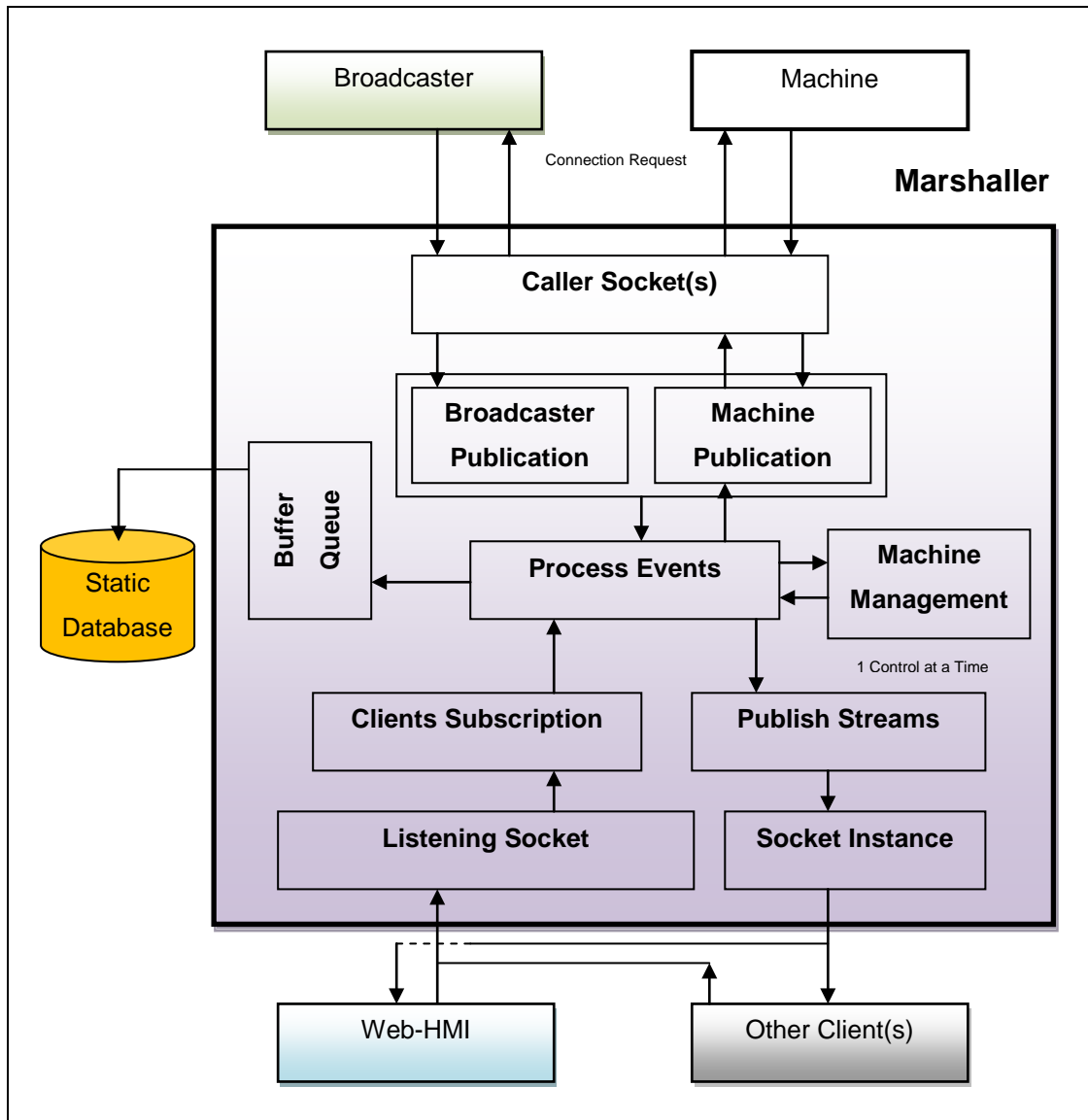


Figure 7-10: Marshaller Process Runtime Implementation

7.4.3 Top-level Database Schema

A static data repository called “MessageData” is created and implemented using Microsoft SQL server 2008 to support historical transactional storage and provide other functionalities within the system architecture such as data for

machine playback, error history, etc. KS4 of the Marshaller system component model is responsible for this storage functionality (chapter 6.2.3). Since the amount of incoming machine data is expected to gradually increase database size, an agent script which truncates data older than 7 days is also implemented. This script's parameters can be reconfigured using the SQL browser. Figure 7.11 illustrates a top-level schema of the database implemented in this research.

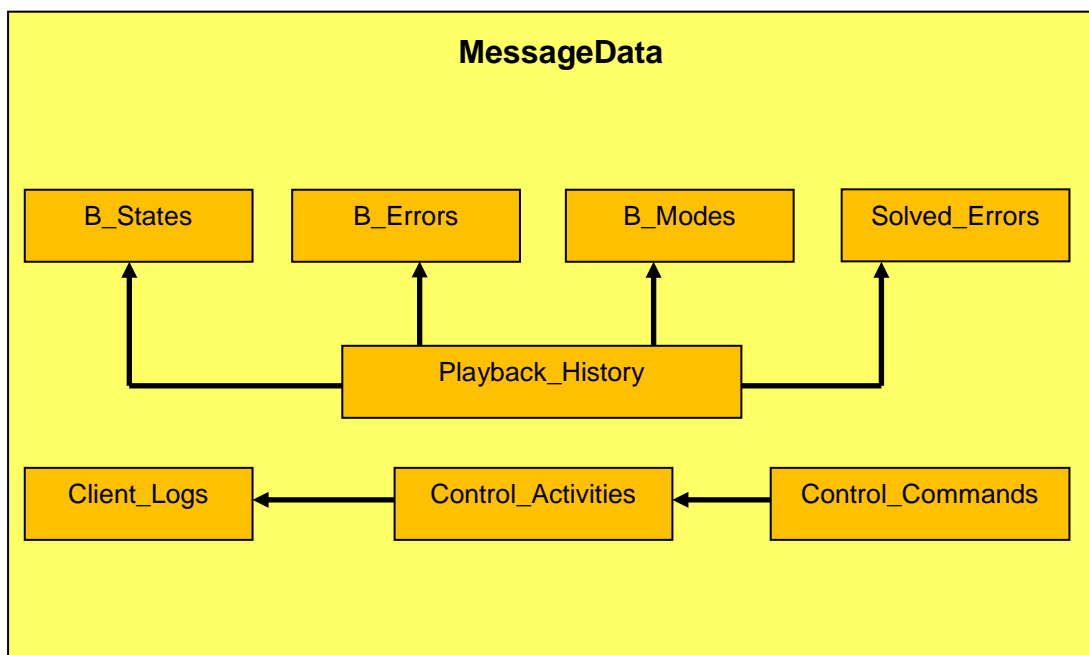


Figure 7-11: Top-level Database Schema

This database consists of a set of tables where the description for each table is briefly provided as follows.

- B_States: This table contains real / simulated machine state transactions data with their associated date/time stamps.
- B_Errors: This table contains real / simulated machine error transactions data with their associated date/time stamps.

- **B_Modes:** This table contains real / simulated machine operational commands and modes data with their associated date/time stamps.
- **Solved_Errors:** This table contains real / simulated machine solved errors data with their date/time stamps.
- **Playback_History:** This table contains a sequential record of machine transactional data with their date/time stamps. This table provides possibilities to the Web-HMI component to support playback functionality using VRML-based techniques.
- **Client_Logs:** This table contains all the connection-specific details of clients who aim to obtain control of a production machine. For example, Web-HMI component clients.
- **Control_Activities:** This table contains data pertaining to all the machine control-specific requests / responses made by various clients after their successful connection with the Marshaller.
- **Control_Commands:** This table contains data associated with exchange of messages between client and production machine once it successfully obtains control of the machine. A machine control mechanism is an integral part of the Marshaller which enables safe management of machine control activities within the system architecture. This control mechanism is described next.

7.4.4 Machine Control Sharing Mechanism

Since Ford Motor Company has stressed the importance of having control safety procedures when implementing this research solution [79], a control sharing mechanism is implemented at both levels of the system architecture (i.e. Web-HMI and Marshaller). At the Web-HMI level, its server control (i.e. BB controller component) prohibits more than one HMI client browser to control a

machine at a time. At the Marshaller level, its manager control (i.e. BB controller component) prohibits more than one client (for example Web-HMI, any other machine acting as a client, etc) to obtain machine control at a time. This mechanism ensures dual-level safety within the system architecture (as shown in the figure 7.12) of both, technological and human resource, at the shop-floor.

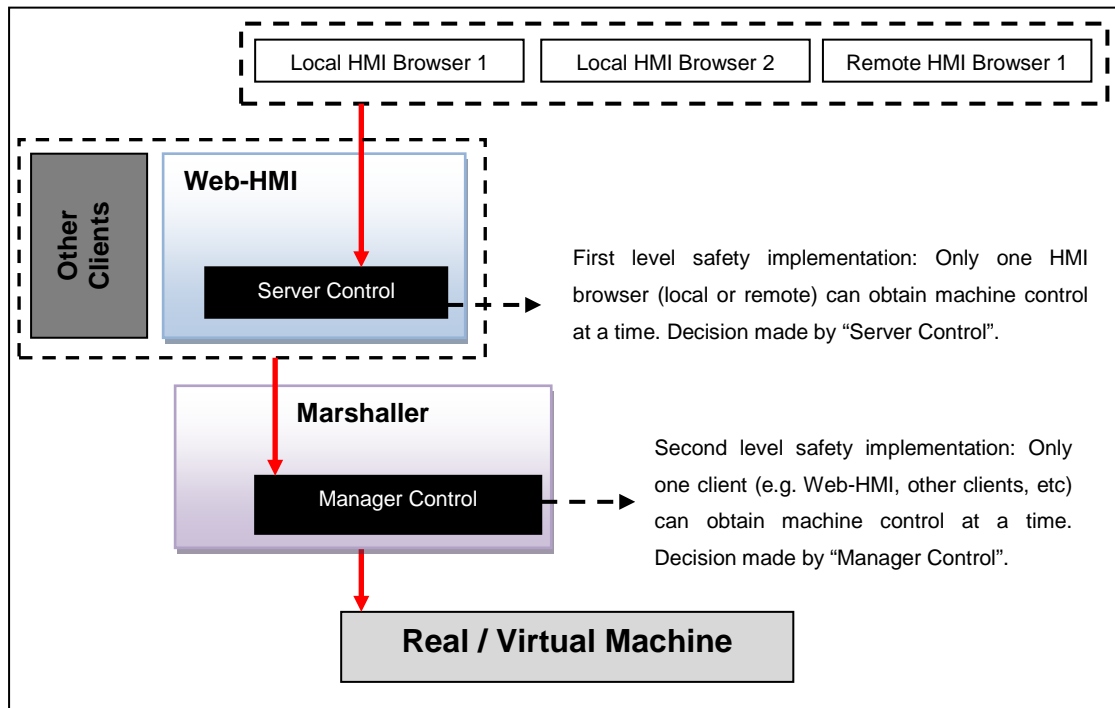


Figure 7-12: Dual-level Control Safety Mechanism

This control mechanism is shown as a flowchart in the figure 7.13 and its corresponding explanation is as follows:

- (a): Marshaller can be preconfigured with details of all the clients aiming to connect with it. These clients may be Web-HMI, other machines, other third-party clients, etc. For the sake of simplicity and ease of explanation, an assumption of their unique id's can be made to be C1 (for Web-HMI client) and C2 (for a web service-based third-party client tool that aims to control a machine). In addition to these Ids', as Web-HMI serves many HMI client browsers, their corresponding Ids' can be assumed to be C1H1 (for a local HMI browser) and C1H2 (for a remote HMI browser).

By default, no control is given to any of these clients at Marshaller's initial start up.

- (b): When a client wants to request control of a machine, it sends its unique Id to the Marshaller. If the client is a HMI browser, it transmits C1H1 and if it is a web service-based client, it transmits C2W1.
- (c) and (d): Assuming C1H1 makes a control request, Marshaller performs Id screening by checking its internal records where Id's of the clients allowed to control a machine are whitelisted. If C1H1 is restricted to control the machine, its control request is dropped.
- (e) and (f): If C1H1 is in the control white list, the Marshaller checks whether a control token is available for the machine. An availability of the token corresponds to availability of the machine control. This token can only be consumed by one client at a time. If the token is not available (owing to C2W1 controlling the machine or the machine is unavailable for some reason), then C1H1's control request is denied.
- (g): If the control token is available, C1H1 consumes the control token, which locks it to the control mechanism such that any control requests from other clients such as C2W1 will be denied. Any control requests from C1H2 will immediately be blocked at the Web-HMI level as it maintains a local control token (i.e. first level safety as described earlier).
- (h): If by any chance C1H1 releases the machine control, the Marshaller makes its control token available for other clients such as C1H2 and C2W1 to initiate the process from (a) to (g).
- (i) and (j): When C1H1 is controlling the machine, it can start sending messages to the Marshaller, only for it to be propagated to the machine. These request messages are related to mode changes, state changes or any component queries. Upon their receipt, Marshaller screens

messages against its CB description (received from the Broadcaster system component). If message(s) are not confirming to the CB model description, Marshaller propagates a denial message to C1H1. In this case, C1H1 can either send further messages to Marshaller or release machine control to make the token available for other clients.

- (k): If the message confirms to the CB model descriptions, firstly, Marshaller establishes its connection to machine and upon a successful connection, propagates this message to it. Secondly, the machine acknowledges its receipt which is propagated by Marshaller back to the client that generated original message request, in this case C1H1. In this way, C1H1 gets an assurance of its successful message propagation to the machine through the Marshaller system.

Marshaller system component consists of a set of graphical user interfaces which provide various functionalities such as system configurations, communications and debugging status views as illustrated in the next section.

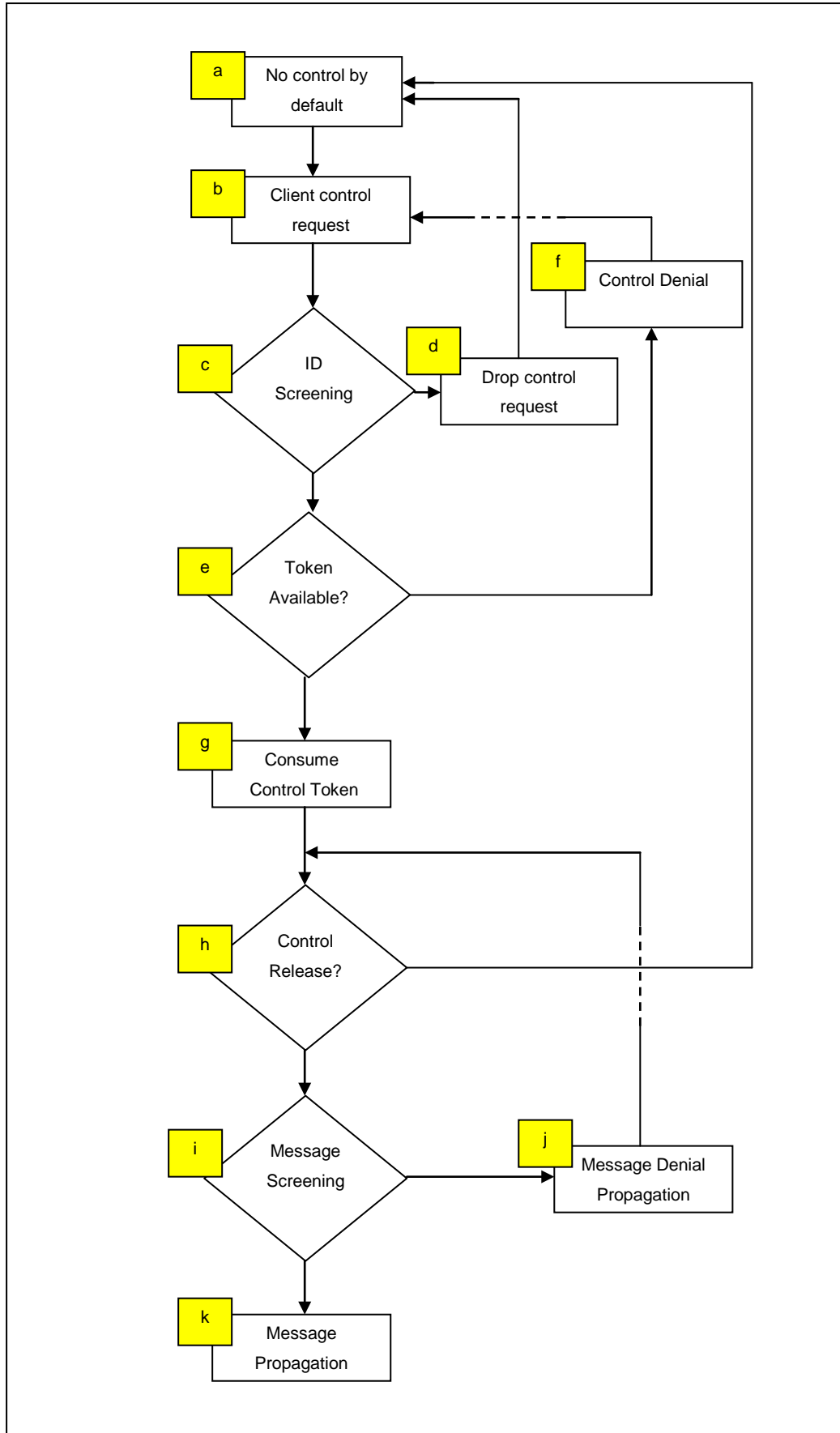


Figure 7-13: Machine Control Sharing Mechanism

7.4.5 Graphical User Interface View

Currently, Marshaller receives CB model descriptions and machine data in XML format which is processed, stored and displayed in textual string format to a user. It offers a set of graphical user interface views for providing various levels of functionality required within the system. Figure 7.14 shows two major views provided by Marshaller system component.

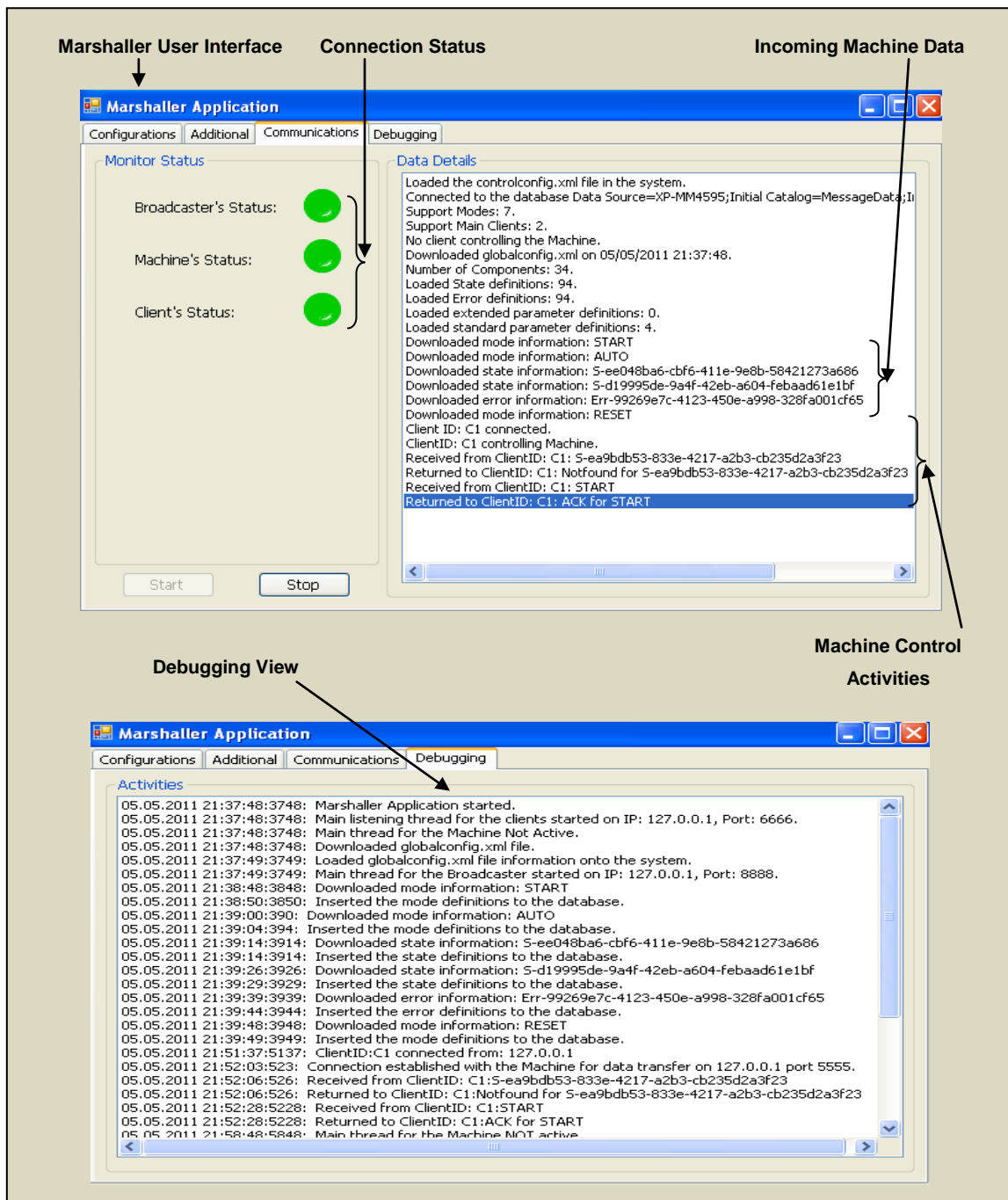


Figure 7-14: Marshaller System Component Graphical User Interface

In the figure 7.14, two major views are illustrated such as communications view and debugging view. The communications view shows connection status with other resources within the system architecture such as Broadcaster, clients and machine. Furthermore, it shows incoming data from the Broadcaster and any machine control related activities in the data details section of the graphical user interface.

Debugging view shows any system critical information and database related activities. It acts as a logging section which saves Marshaller's activities in an external textual file for future debugging process, if needed. In addition to these two views, Marshaller has two additional graphical user interface views such as configurations view (for system related configuration settings) and additional view (for database and control client's settings). These views are not shown in the figure 7.14. Next section describes Web-HMI system component implementation details.

7.5 Web-HMI implementation

7.5.1 I / O Outline

This section outlines Web-HMI's operational boundary showing major inputs and outputs supporting the overall operator interface systems' monitoring and control functionality within CB implementation. Figure 7.15 shows I / O outline where only external links are presented. As an input, CB model and machine status is received and de-serialised by the Web-HMI server for supporting its monitoring functionality. Any machine status changes (i.e. state change, emergence of errors or operational information) are collected from the Broadcaster, processed and propagated to HMI client browsers. Furthermore, machine control operations are supported with the aid of the Marshaller. To appreciate this client / server implementation, a description of its process runtime is provided next.

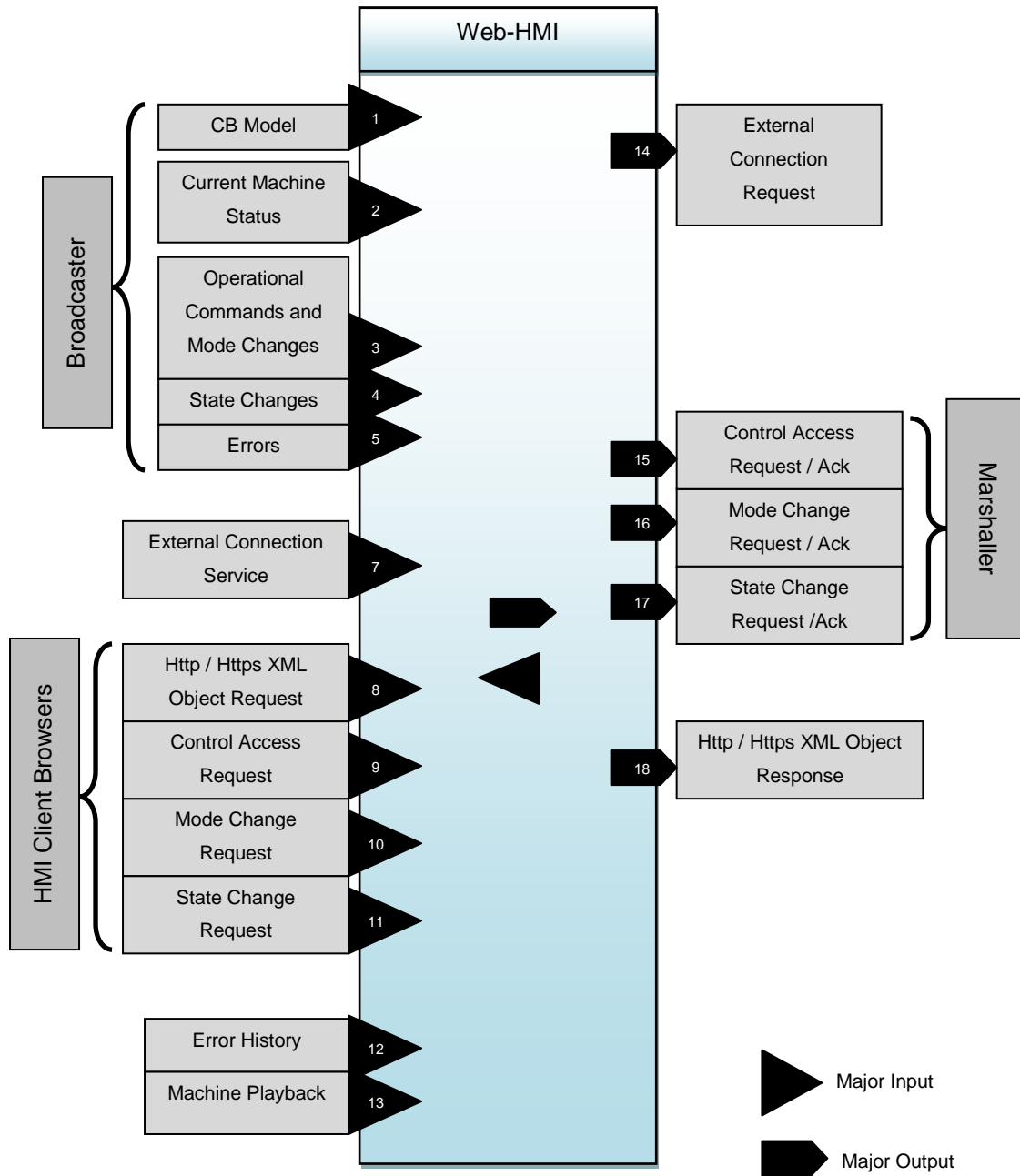


Figure 7-15: Web-HMI I / O Overview

7.5.2 Process Runtime Implementation

Since the Web-HMI system component supports various distributed HMI client browsers in control and monitoring operations, BB knowledge sources communicate with the blackboard using Client - server model (as described in the chapter 4.6.4). This approach enables anytime, anywhere accessibility to the operator interface system by supporting multiple client views, serving real-

time information (using XML-based objects, described later in this section) and processes client's control operations, overall satisfying the functional and non-functional requirements identified in the chapter 5.2.2.

Figure 7.16 illustrates Web-HMI's process runtime implementation from the client – server model's perspective. The Web-HMI system is internally organised using the MVC pattern (described in the chapter 4.6.1) and security is implemented using HTTPS protocol and layered architecture (described in the chapter 4.6.2). Initially, an HMI client browser requests a required operator interface screen (i.e. webpage) from the server using a URL (Uniform Resource Locator), which is nothing but the address of resources serving the Web-HMI system component over the WWW. With the help of its logic, a TCP/IP socket connection from the Web-HMI caller sockets to the Broadcaster is established (if it is not already initiated) to receive CB model and current state of a machine. Upon their receipt, the logic updates the required html page (represented as a template – further explanation in the section 7.5.3) by populating it with the CB model description and current status of the machine. Furthermore, any necessary navigational information is provided by this logic and the requested html page is returned to the HMI client browser.

To avoid the traditional client – server communication issues where a client waits for response from a server after every task, an AJAX (Asynchronous JavaScript and XML) technique is implemented within this system component to improve performance, interactivity of the overall application [206] and provide real-time machine information. Any update request from the HMI client browser takes the form of a JavaScript call to an AJAX engine (written in Javascript and tucked away as a hidden frame) instead of a standard HTTP / HTTPS call.

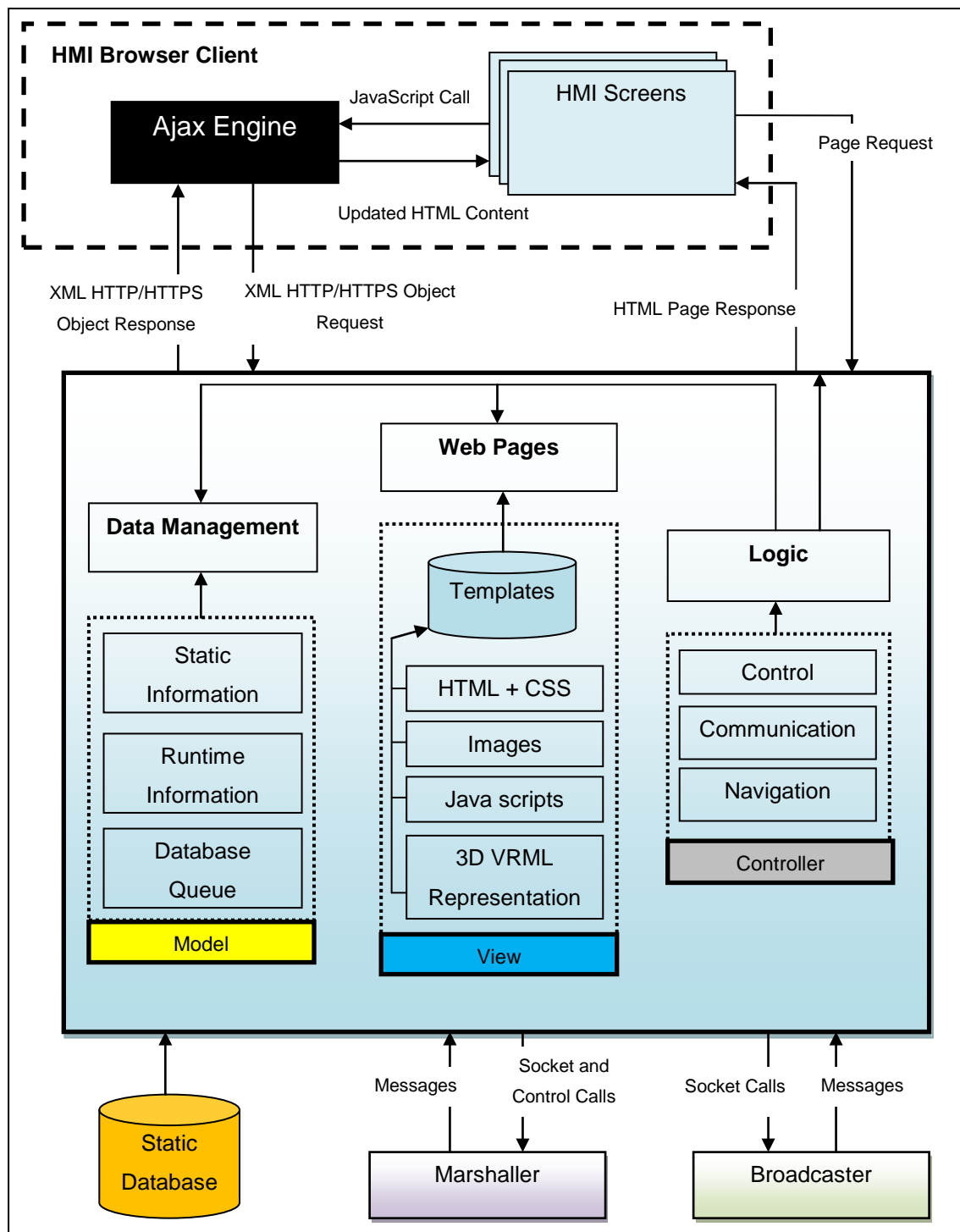


Figure 7-16: Web-HMI Process Runtime Implementation

This AJAX engine renders the operator interface screens and communicates asynchronously with the server in the client's behalf. If the engine requires any updates from the server to refresh certain HTML content on the browser (for example, retrieving real-time machine status information such as a component

state change), the engine makes asynchronous XML HTTP/HTTPS based object calls to the server without stalling operator's interaction with the HMI browser screen. This technique provides responsive operator interface to HMI roles whilst updating various contents on the HMI screens. Depending on the request, the Web-HMI logic responds to the XML object request, which becomes the updated HTML content of the HMI client browser screen. In this way, various HMI screens and their contents can be requested without experiencing any visual interruptions.

When the HMI client browser aims to obtain a machine control, a request token (represented as an XML object) is propagated to the server where the logic control permits or denies passing the machine control to the client browser that generated the object. If control is permitted, a bi-directional communication channel is established from the HMI client browser to the Marshaller (via the server). Any updates that do not require requesting a complete webpage is handled by the AJAX engine. The overall look and feel of the HMI screens is consistent throughout, thanks to the template-based implementation described in the next section of this thesis.

7.5.3 Operator Interface Template to Configuration Mapping

As described previously, Web-HMI serves numerous web pages which are reconfigurable and consistent to displaying machine configurations at the browser side through the use of template-based representation. This section describes the exact approach utilised to represent these templates and the process through which they are populated with various configurations at runtime. Figure 7.17 shows an overall representation process illustrating generation of HMI template with editable regions that map to various configurations using the Web-HMI system component logic. As shown, an operator interface template's layout is governed by the representation layout which is decomposed into three levels namely; panel view, zone view and link points. These templates are inspired from Siemens Transline layout (as described in the chapter 5.2.1).

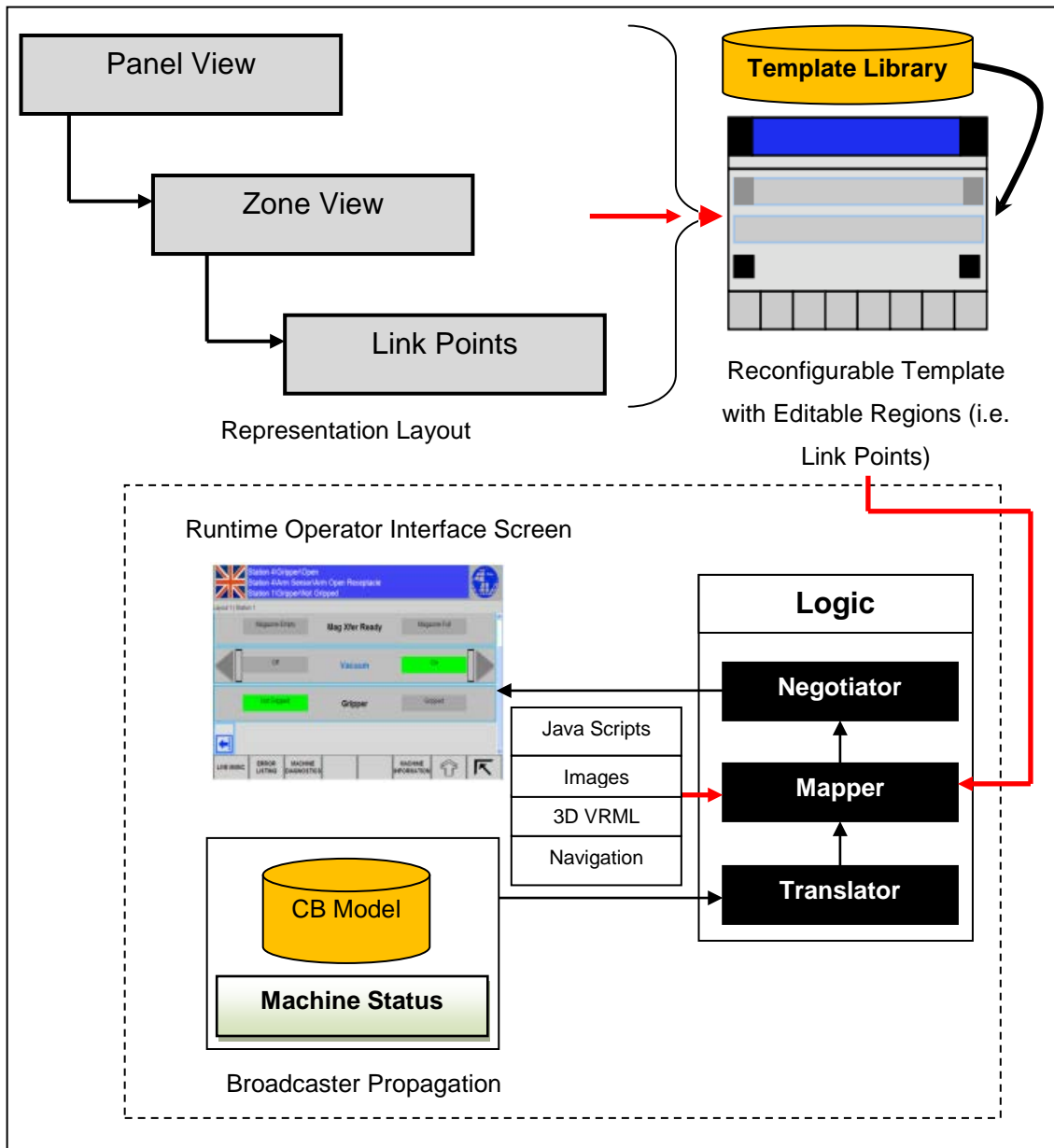


Figure 7-17: Web-HMI Template Representation and Configuration Population Process

An HMI screen can have any number of panels where each panel can have multiple zones. Within each zone, numerous link points can be defined, which are nothing but editable regions (dynamic array items acting as a placeholders) that get populated at runtime with various configurations. This representation layout provides reconfigurable templates once description of panels, zones and link points have been carried out. It has to be noted that no external configuration tool is yet available within this research to enable these parameters to be configured outside the Web-HMI system code. Presently, this

representation layout has been described within the system code and expressed using CSS (Cascading Style Sheet) templates.

The Web-HMI logic's translator interprets received CB model description from the Broadcaster and handles the responsibility of populating these descriptions to a mapper. The mapper creates an instance of the required reconfigurable template and updates instance's editable regions with the available configurations such as CB model, machine data, images and VRML description. Furthermore, it attaches executable scripts and navigational information to this template instance. Negotiator's responsibility is to manage communication session with the requestor (i.e. HMI client browser) and thus return complete html page (i.e. template instance) representing machine configurations. With any modifications to a machine, corresponding sections of the HMI screen are updated using AJAX objects (as described in the section 7.5.2).

To clarify the above explanation of representing and populating HMI screens, an example has been illustrated in the figure 7.18, describing a layout of the operator interface system's component browser screen within this research. In the representation layout described earlier, the panel view forms a boundary within which zones exist. Within a zone, multiple items can be configured using their respective link points. The screen is populated from its template with the help of the mapper logic which updates various link points with their respective configurations as shown in the example in the lower section of the figure 7.18. Within the Web-HMI logic, the control (i.e. server logic) decides which link point is of which category type. If the link point needs to house an image, it instructs the mapper to load the required image at runtime.

The operator interface system can be implemented to run as a stand-alone terminal communicating on its own to a machine controller (such as a PLC), or as a server collecting data from various controllers and providing display screens to numerous clients at various locations. Next chapter describes proof-of-concept application of this CB operator interface systems approach by validating it through three industrial case studies.

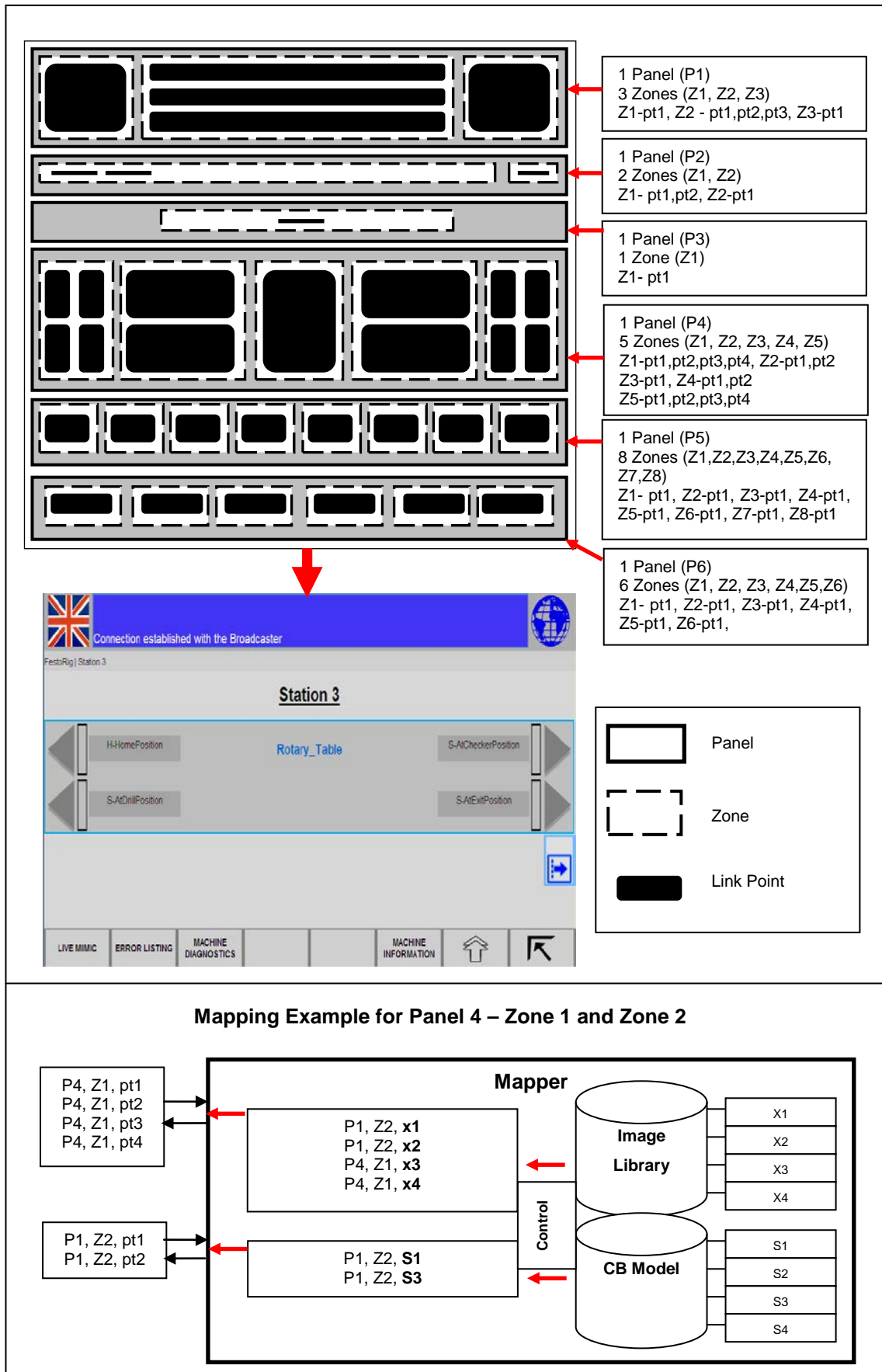


Figure 7-18: Operator Interface Template to Configuration Mapping at Runtime

Chapter 8 : Industrial Case Studies

Chapter Contribution to this Thesis:

The main contribution of this chapter is the assessment of the CB operator interface system using three industrial case studies, verifying the proposed research solution by fully supporting the identified manufacturing automation requirements.

8.1 General Overview

This chapter describes the practical suitability of the CB operator interface system approach in industry by assessing the research requirements focused and identified as attributes in chapter 3.3. The development and implementation of this research has been undertaken through various research projects such as COMPAG [85], SOCRADES [27] and BDA [112]. Initially, the operator interface system's scope within the CB machine lifecycle is identified prior to the verification of this research approach using three case studies.

8.1.1 Operator Interface Context within CB Machine Lifecycle

The aim of the CB automation approach is to fully support machine lifecycle requirements. The idea is to define information once but to use it many times by all the stakeholders throughout the machine's lifecycle. As indicated in the chapter 3.2.2, the fundamental concept of this approach is to compose a complete automotive machine from modular machine components. The major elements that facilitate implementation of CB approach are the CCE engineering environment, a common machine data model (CB configurations), real or simulated machine components and runtime support environment assisting in close control and monitoring of machine components using the web-based operator interface system.

From the CB approach perspective, the lifecycle of a machine involves machine design, installation, operation and reconfiguration. Design and installation phase includes intermediate phases such as build and try-outs where as operation

includes maintenance. Reconfiguration phase encompasses all machine modification activities from minor machine interlocking changes to major machine re-compositions.

During the machine design and installation, process engineers plan the operation of a machine from the manufacturing process requirements that are to be supported by it. Subsequently, engineers select machine components that meet the specification of the new machine's requirements. Using the CCE engineering tools, the machine's sequencing and interlocking logic can be inputted by control engineers using a high-level state-based representation that defines the machine operation. This definition becomes the control logic of the machine which can be validated using machine simulation. Simulation involves a 3D model execution of the machine logic that can be evaluated by the operator interface system prior to the physical machine build phase. The output of the CCE machine design activity provides a machine behaviour definition in the CB configuration form that can be shared within the system components architecture (described in the chapter 5.3).

Within this research implementation no operator interface configuration tool is currently available therefore default operator interface system screens can be reused by process engineers within the machine program. Since the machine behaviour is already available in the form of CB configurations, operator interface system can be used to validate various machine tasks prior to its build (as studied in the section 8.3.4). Furthermore, operator interface system's functionality can be verified (as studied in the section 8.3.5) and machine operators can be trained (as highlighted in the section 8.3.6) using the same machine simulation. This enables a complete operator interface system to be available to support design, build, try-out and installation phases of the machine lifecycle.

After the machine installation at the end user's site, operator interface system can be used to support day-to-day control and monitoring operations, and any maintenance activities (as studied in the section 8.2.7). Any machine reconfiguration can be carried out to meet new product requirements using the CCE engineering tools by modifying existing components from the CB library or

creating and integrating new components in the design. The corresponding machine modifications are stored in CB configurations. This single common machine data model can be shared throughout the machine lifecycle using the Broadcaster system component, enabling operator interface system to support reconfiguration activities (as studied in the section 8.2.4).

8.2 Stage 1 Case Study: Ford-Festo Test Rig

The Ford-Festo test rig is a table top style test bed located at the MSI laboratory implemented to facilitate the advancement of the state-of-the-art in manufacturing machine control technology and collaborative demonstrations. The test rig (illustrated in the figure 8.1) was developed in conjunction with Ford Motor Company of the UK, and mimics a general assembly automation machine line (described in the section 8.3).

The main objective of using this rig is to investigate new research approaches to powertrain automation in isolation and combining assembly stations in a coordinated fashion. The rig provides a flexible environment where new ideas can be verified without the cost of lost productivity, safety issues or scrap inevitable in using it in a real manufacturing environment. A description of this rig is covered in the next section.

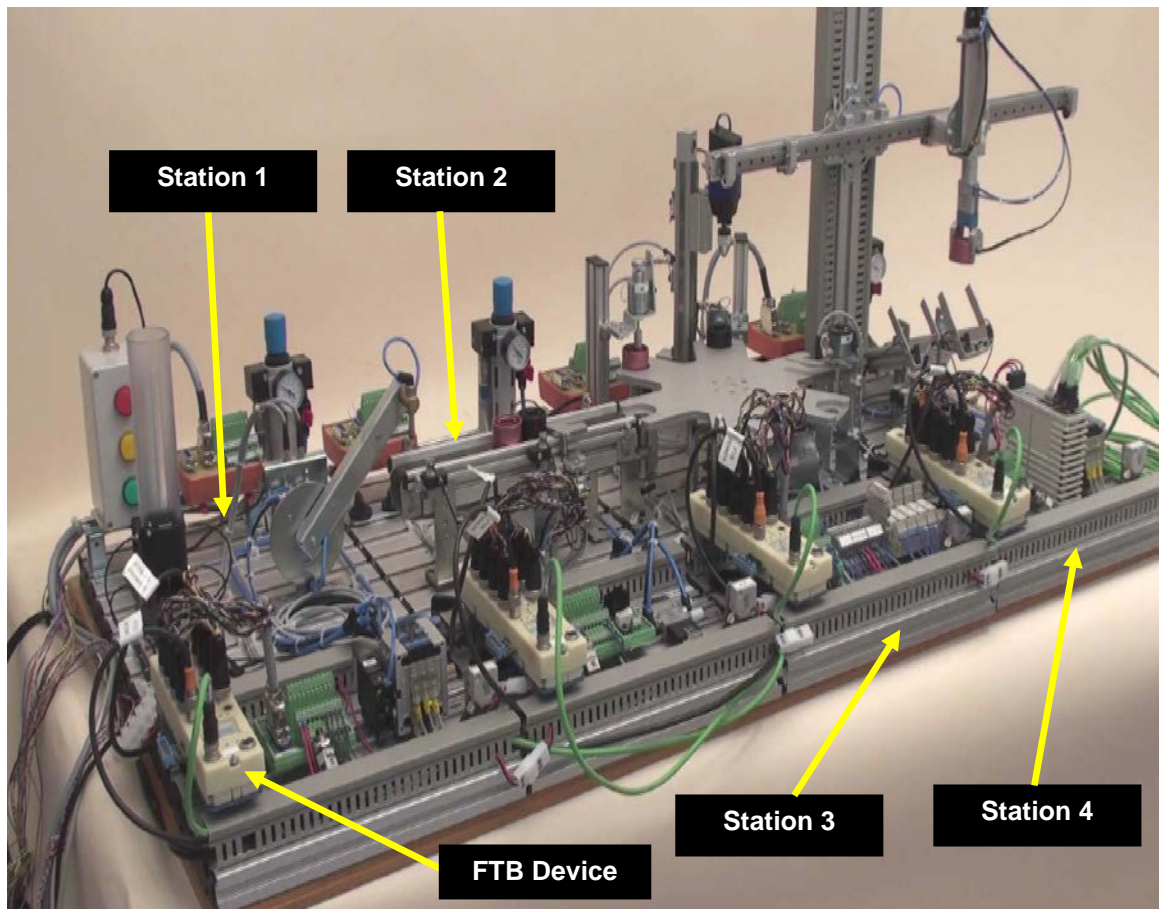


Figure 8-1: Ford-Festo Test Rig

8.2.1 Test Rig Description

This study is based on the application of CB operator interface systems approach by validating it through the Ford-Festo test rig. The test rig consists of four subsystems (i.e. stations) as illustrated in the figure 8.2:

- Station 1: This is a subsystem consisting of a distribution hopper unit.
- Station 2: This is a subsystem consisting of a buffer unit.
- Station 3: This is a subsystem consisting of a processing table unit.
- Station 4: This is a subsystem consisting of a handling arm unit.

Each of these subsystems has one or more mechanical components that are connected to field devices (i.e. sensors and actuators) through a distributed FTB-based control module from Schneider Electric.

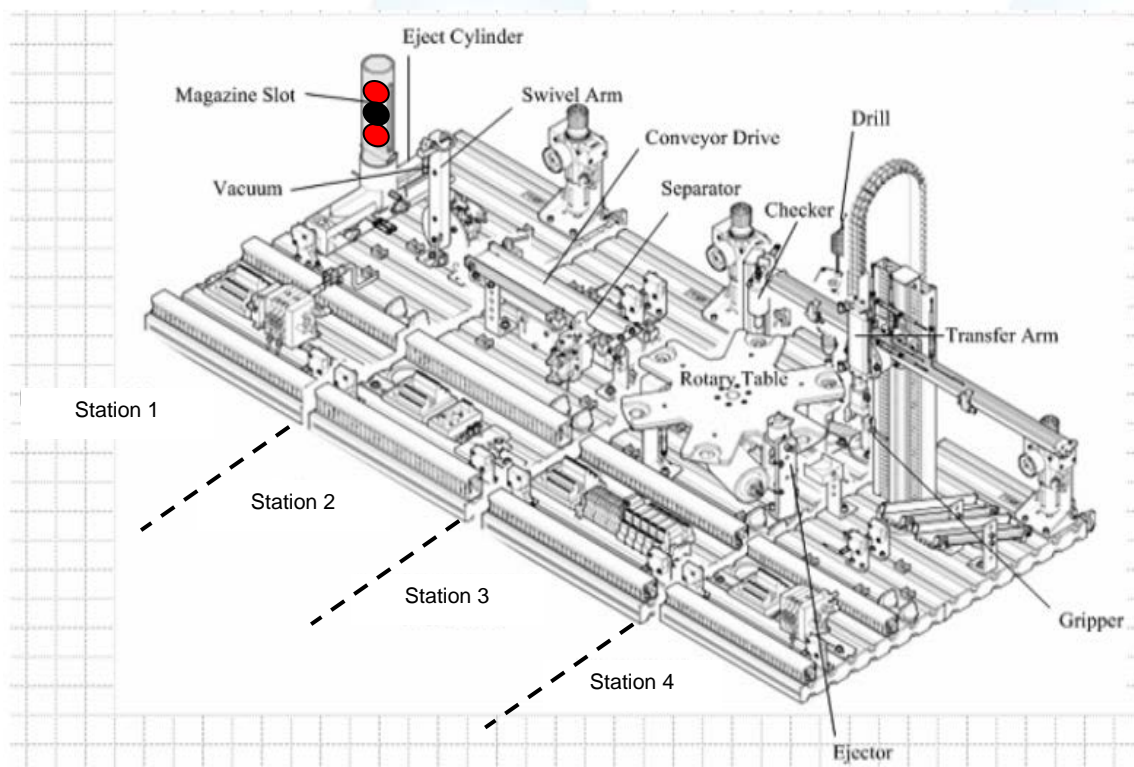


Figure 8-2: Ford-Festo Test Rig Illustrating Major Components

As previously described, this test rig's functionality represents a typical powertrain machine line operation which is used to assemble vehicle engines. In the assembly process, parts are inserted to the main body of the engine block (or head) at each station along the processing line. In this test rig the engine block is represented as a plastic workpiece (or part) that undergoes various assembly tasks such as transferring, buffering, slot checking, drilling and sorting.

Its assembly sequence is such that the workpieces are loaded into the rig in the magazine slot and the eject cylinder pushes each part from it at the station 1. Swivel arm picks each one up and transfers them to the conveyor drive at the station 2. With the help of the separator, parts are controlled to flow onto the rotary table one at a time. At the rotary table in the station 3, parts move through different locations such as checker, drill and ejector. The checker is used to confirm that the workpiece is positioned correctly prior to drilling operation. If it is not then it will skip the drill operation and raise an alarm to the

machine operator. If it is positioned correctly then drilling occurs and the part gets moved to an ejector which pushes the workpiece to the buffer of the transfer arm. The transfer arm at the station 4 uses the gripper to pick the parts to place them in either goods part bin or goods reject bin depending on whether a good / bad drilling operation is carried out earlier. This differentiation is implemented using coloured workpieces.

Adoption of Web Services in CB Control Description

The control description within this test rig has been engineered using the CB approach (as described in the chapter 3.2.2). In the test rig implementation, there are four controller nodes, each responsible for the control tasks of one subsystem (i.e. station) resulting in a fully distributed control environment. Each controller node is a prototype embedded microprocessor device called FTB which has been designed and provided for research by Schneider Electric. Each component is enabled with a web services interface as a result of the SOCRADES EU research project. Web services enable distribution of test rig status information to widely used higher resources; in this case a web-based operator interface system. This approach has been demonstrated through its integration in a control and monitoring system architecture using a service orchestration engine [82] developed at Loughborough University. This engine is implemented within the application logic as a finite state machine and orchestrates various services on the components [31, 82].

8.2.2 Case Study Setup

Figure 8.3 illustrates the overall case study setup using the Ford-Festo test rig controlled using web services-based FTB devices.

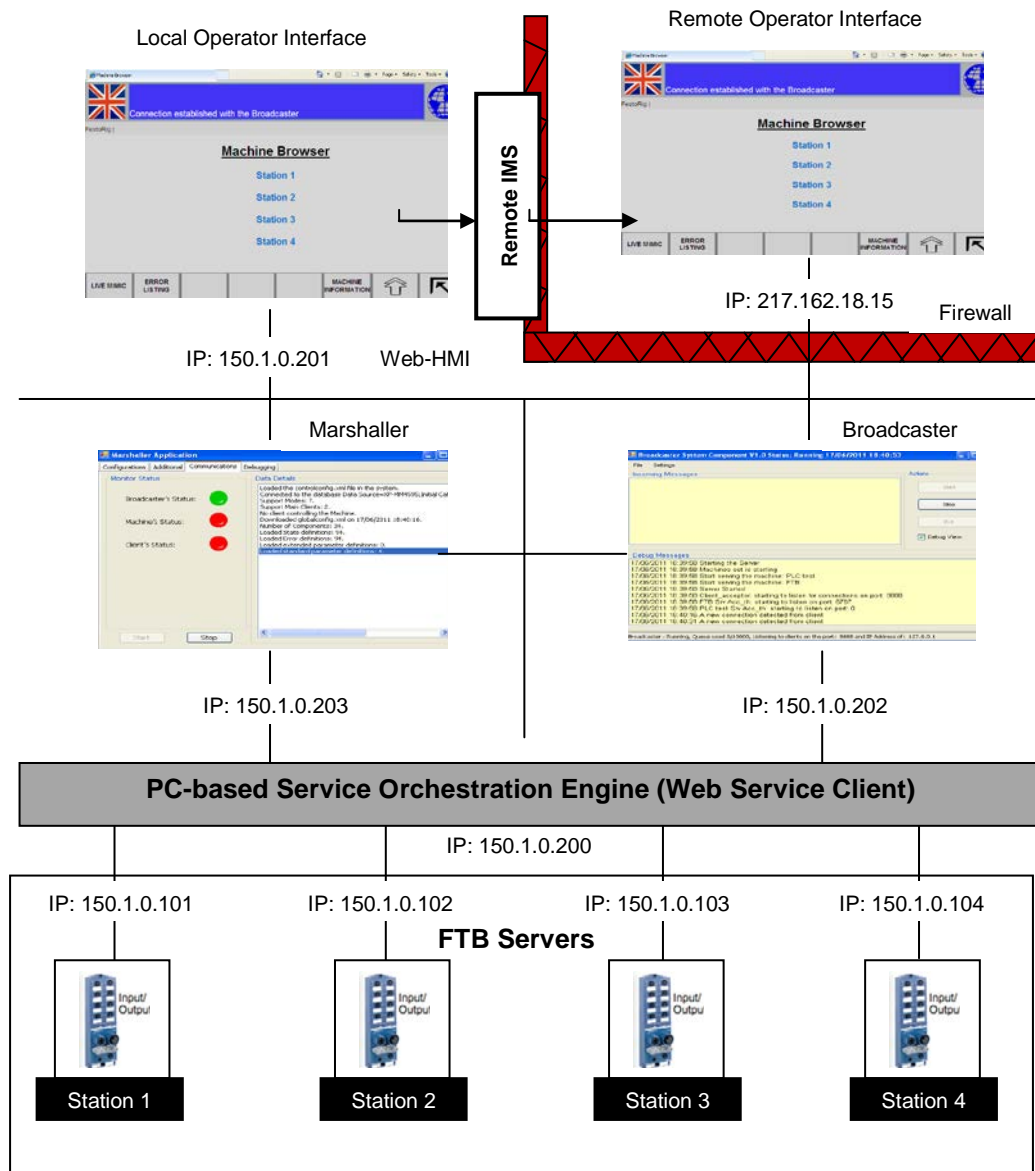


Figure 8-3: Ford-Festo Test Rig Case Study Setup

To support fully distributed heterogeneous web services-based manufacturing automation, a PC-based orchestration engine is implemented, acting as a client to a service on the FTB control device for device operation. The orchestration engine subscribes to the machine events published by the web service control logic at the FTB server side and in doing so establishes a TCP/IP socket connection with the Broadcaster to propagate a machine’s published status (in this case, the test rig) and accepts incoming control commands from the Marshaller via an alternative socket TCP/IP link. Machine status is broadcasted

to all the connected clients (including the Web-HMI and the Marshaller system components). Web-HMI, in turn, serves local operator interface client browser(s) as well as remote browser(s). An alternative remote connectivity link can also be established by initiating a RemoteIMS screen sharing session (concept described in the chapter 4.4) from any local HMI client browser's PC.

The system components are deployed in a PC-based environment communicating over the Ethernet / Internet. Each system component has been given a specific IP address enabling them to operate in a distributed fashion. The HMI system console consists of a standard computer touch screen (i.e. non vendor-specific) operating using the Microsoft Explorer (standard web browser) with internet connectivity.

8.2.3 Research Attributes Assessment

Some of the identified requirements of this research (summarised as attributes in the chapter 3.3) are assessed using scenarios and demonstrations appropriate for this case study. The benefits of utilising a CB operator interface system in addressing these desired attributes have already been discussed in the chapter 2.4. The four main attributes tested in the subsequent sections are as follows:

- Reconfigurability and Reuse Support.
- Information Transparency and Mobility.
- Loose Mapping of HMI to Actual Machine or its Control Logic.
- Real-time Remote Machine Control, Monitoring and Maintenance.

8.2.4 Reconfigurability and Reuse Support

As a requirement of agile automation, it is vital to be able to reconfigure and reuse production machine (and its associated engineering support tools) throughout its lifecycle. From the perspective of operator interface system, implementation of their screens must cater for any dynamic changes exercised

on the machine modules (i.e. mechanical and control components) owing to the support required to facilitate new products.

In order to enable this functionality within the proposed operator interface system, two major implementation features have been incorporated:

- Generic operator interface template screens are populated at runtime with the machine configuration. Templates specify the screen layout according to the representation layout as described in the chapter 7.5.3. Since the operator interface's view is separated from its logic (as described in the chapter 7.5.2), the same generic screens can be utilised within many machine programmes having different configurations.
- Presence of the Broadcaster system component enables propagation of machine configurations (including any dynamic updates) to the operator interface system at runtime. Since any change to a machine during its reconfiguration activity is captured in the CB model (i.e. machine configurations), their propagation to Web-HMI system component supports dynamic screen updates to match machine changes.

Scenario Description: Modifying a Process Workflow

To assess the capabilities of the system in this respect, a scenario on the test rig is setup as illustrated in the table 8.1. This scenario represents a real life reconfiguration example that occurs at Ford Motor Company corresponding to a process workflow change in an assembly line. In this scenario, the station 2 (buffer unit) is removed from the existing test rig's configuration in order to bypass the workpiece straight to the station 3 (processing table unit). The configuration steps are illustrated in the figure 8.4:

Scenario Details				
Implementation Target: <u>Ford-Festo Test Rig</u>				
Control: <u>Web Services-based FTB Device</u>				
Ref	Station ID	Configuration Type	Operator Interface Client Type	Application Re-programming
SN1	Station 1,2,3,4	Existing Machine Configuration	Local and Remote HMI Screens	None Required
SN2	Station 1,3,4	New Machine Configuration	Local and Remote HMI Screens	None Required

Table 8.1: Reconfiguration and Reuse Scenario

The configuration work flow is as follows:

- The Process engineer uses the CCE engineering toolset to re-design the Ford-Festo test rig model, and mirror those changes through mechanically rearranging the stations by removing station 2 and linking the station 1 straight to the station 3. This replaces the current configuration of having four stations (station 1/2/3/4) by a new configuration having three stations (station 1/3/4). The new design can be used to configure the control software as per the CB approach.
- Restart the Broadcaster system component after replacing CB model description output provided by the CCE engineering toolset.
- Refresh the Internet explorer browser running the operator interface system. Any configuration changes are dynamically reflected onto the HMI screens (through the AJAX objects) and thus a machine operator does not need to wait for a new version of the HMI system to be deployed at the HMI panel (in this case a standard touch screen). The output is a consistent machine information display that reflects the actual machine configuration.

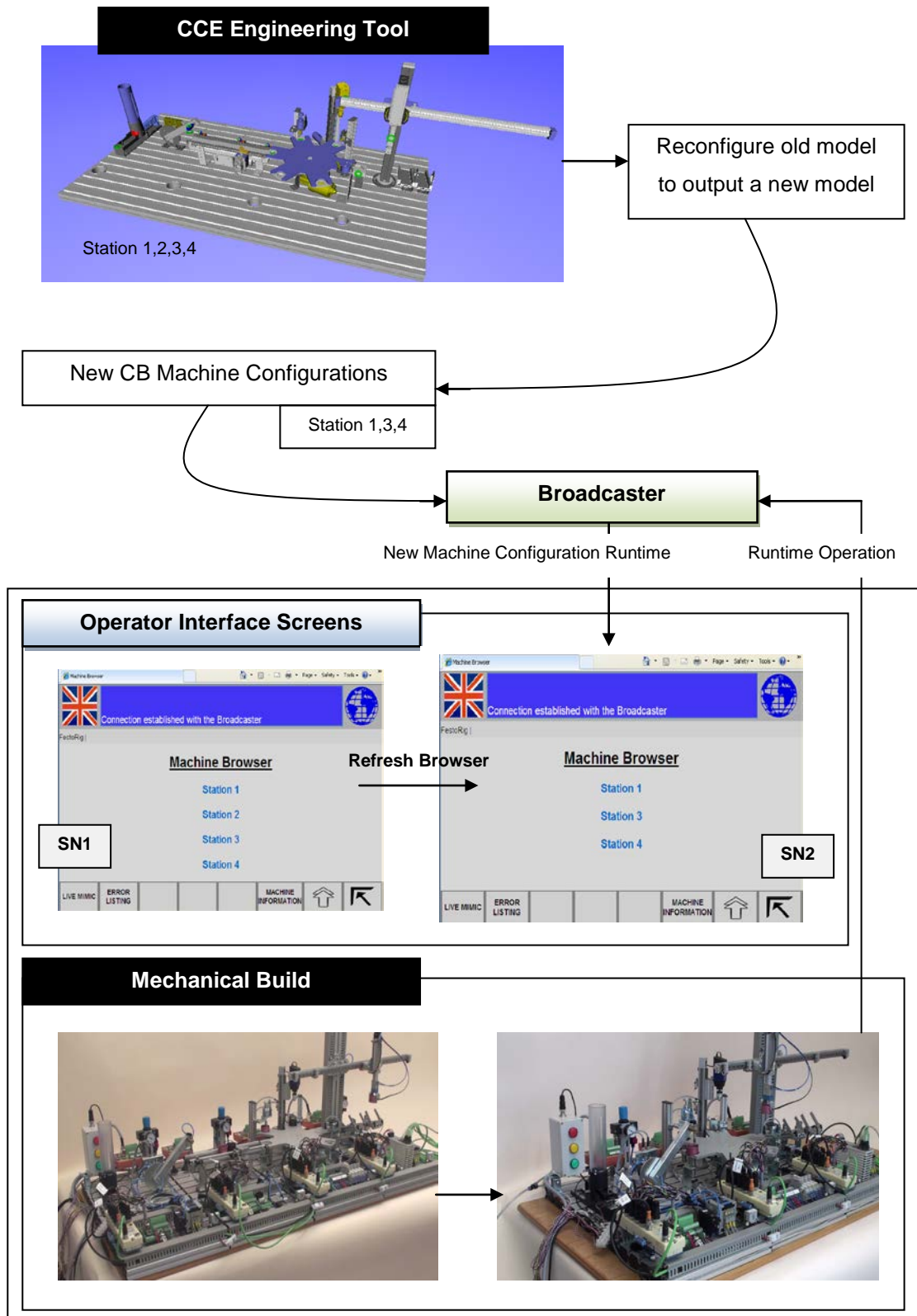


Figure 8-4: Reconfigurability and Reuse Scenario Case Study

8.2.5 Information Transparency and Mobility

A co-operation between business and shop-floor levels must be consistently represented through standard interfaces that enable various support tools to be integrated within the engineering and runtime processes of a machine's lifecycle. In fact, it is essential to improve the transparency of the connection between back-end systems and shop-floor production. From the perspective of this research, the control and monitoring system architecture must provide a "plug-and-play" integration framework, legacy system support and operate using open standards formats.

In order to support these functionalities within the proposed operator interface system implementation, two major enabler features are implemented:

- A distributed system architecture that operates using standard TCP/IP communication link, enabling third-party tools to connect to the Broadcaster and Marshaller on ad-hoc fashion.
- Broadcaster system component represents machine operational status data in uniform XML open format which can be collected and decoded by any third-party (or legacy) system for further analysis, storage and propagation.

Business Application Integration Demonstration: SAP Tool

To verify this attribute, a demonstration has been undertaken with SAP GmbH's SAP xMII (Manufacturing Integration and Intelligence) for shop-floor activity monitoring and fault diagnosis [207] as part of the SOCRADES project. This application enables business systems to obtain a real-time view of industrial processes, supporting business activity monitoring, maintenance optimisation and overall equipment effectiveness.

An overall setup for this demonstration is schematically shown in the figure 8.5. As shown in this schematic, SAP xMII application is configured to interact with the Ford-Festo test rig using the service orchestration engine (via the

Broadcaster's TCP/IP link). The service orchestration engine has a state publication utility and a service invocator. These enable SAP tool to subscribe to the required device's states and to invoke the operations on the FTB devices [31].

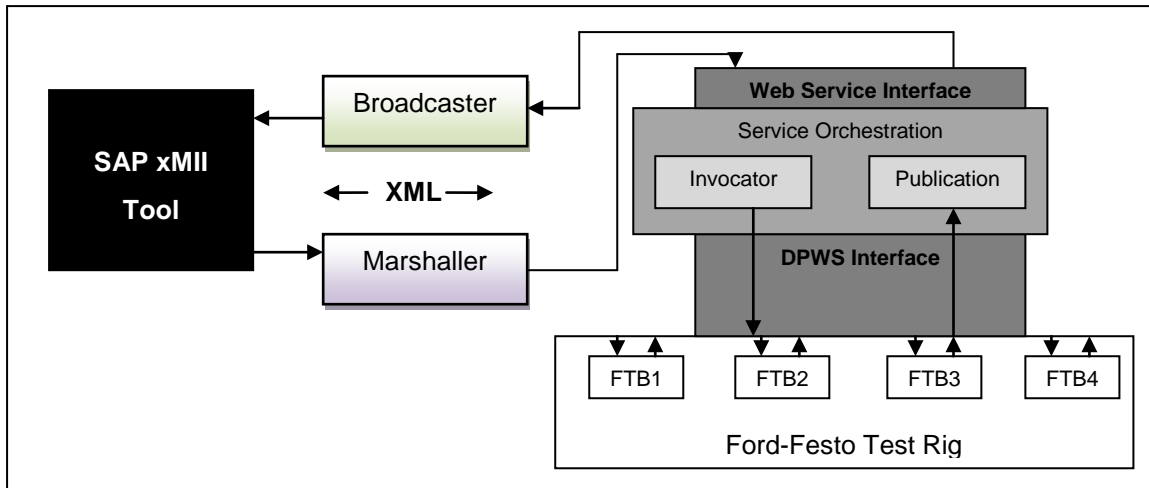


Figure 8-5: Schematic Setup for SAP xMII Demonstration

In this integration demonstration, the test rig provides control device data (i.e. real-time device status), work piece status, the number of processing units, and the machine operational and idle times for data manipulation and analysis as shown in the figure 8.6. The operational status of the test rig is provided by the state variables determined from the logical state of local sensors. Based on the event-based communication model, these variables are transmitted to the SAP tool through the Broadcaster system component as and when a component state changes.

By collecting this status information, businesses can have a global view of the entire shop-floor manufacturing process. This enables strategic decision making and provides an integration framework where third-party (or higher-level ERP) systems can be plugged into the existing architecture.

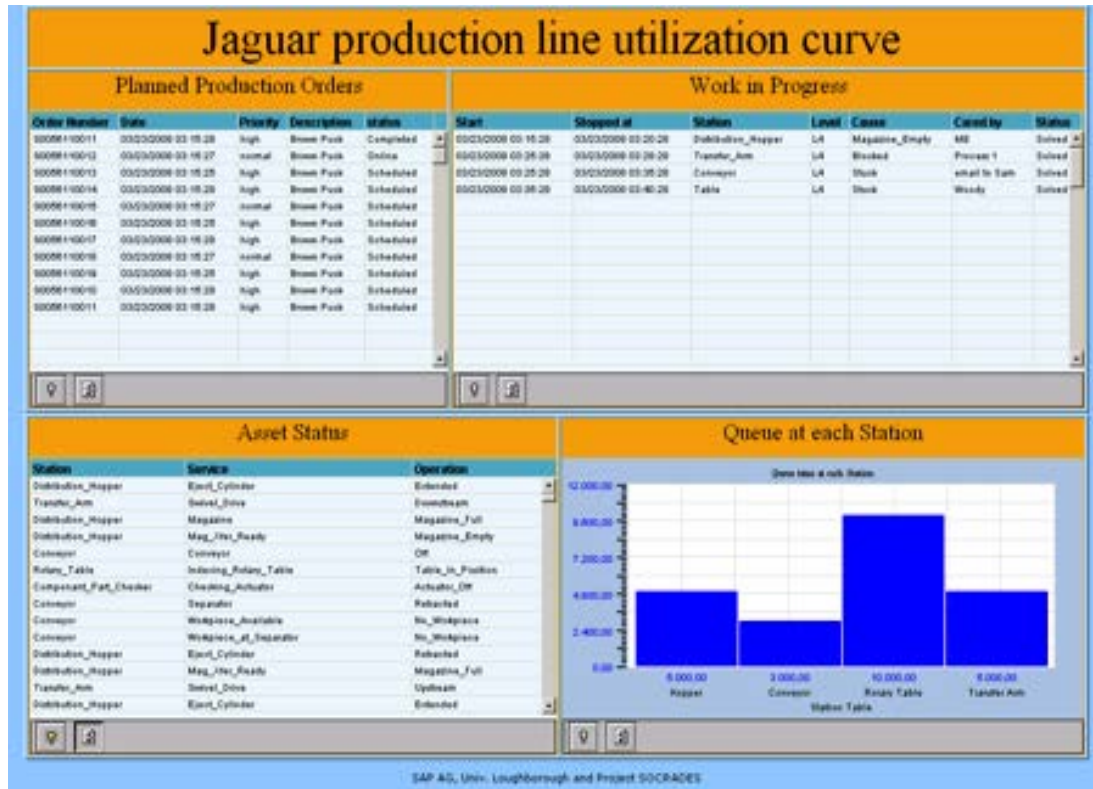


Figure 8-6: SAP xMII Tool Integration in SOCRADES

8.2.6 Loose Mapping of HMI to Actual Machine or its Control Logic

It is of paramount importance to have operator interface systems that are machine or control logic independent, requiring HMI to be loosely mapped to both. Any reconfiguration activity or supplier selection process during the machine lifecycle should not be influenced by the existing resources at the shop-floor. Furthermore, any modification to a machine (eventually leading to changes in the control logic) should not require any programming at the operator interface system side. In fact, regardless of the machine type or the controls vendor, operator interface systems should be integrated to the machine engineering process in an open fashion, providing a truly heterogeneous existence of systems at the shop-floor.

From the operator interface system's research perspective, HMI screens should be generic enough to enable same set to be deployed across various machines in a production programme. Author acknowledges the fact that the same set of screens cannot be practically utilised across various programmes having

different screen layout requirements (for example, transfer line machines and assembly line machines may have different screen layout requirements), however, these requirements within this research approach can manually be integrated using programming the operator interface system codes and modifying its representation layout (chapter 7.5.3). This setback is owing to the lack of an available HMI configuration tool that can easily be used by process engineers to quickly churn out new layout screens to fit new screen requirements (as discussed in the future work, chapter 10.2).

In order to enable this functionality within the proposed operator interface system, three major implementation features are provided:

- Machine configurations (based on CB approach) are independent of the operator interface system owing to the separation of HMI's logic from the actual view (as described in the chapter 7.5.2). The logic maps machine configurations to the HMI views only at runtime, enabling operator interface screens to be generated with updated machine configurations. A machine of type "A" (having its specific configurations downloaded to an industrial controller from vendor 1) and a machine of type "B" (having its specific configurations downloaded to an industrial controller from vendor 2) can use the same operator interface system.
- Generic operator interface template screens specify the screen layout based on the representation layout as described in the chapter 7.5.3. The generic screens can be utilised within many machine programmes having different configurations.
- A suitable control and monitoring system architecture that does not differentiate between the machine type and the control type as it is purely data transmission-oriented (which is represented in a uniform XML format). Furthermore, the architecture supports hybrid machine control and monitoring, where a partly real and partly simulated machine can coexist and drive the HMI system.

Scenario Description: Simulated Machine-Independence

To verify this attribute, a scenario demonstrating the machine independence feature of the operator interface system implementation is carried out (shown in the figure 8.7). Two simulated machines are used for this scenario, first one being the Ford-Festo test rig and second one is an Oil Pan Rundown machine (described in the section 8.3). In this scenario, both machine models drive the same operator interface system implemented in a PC, one after the other.

Initially, the Ford-Festo test rig model is run using the CCE engineering tools, (its corresponding machine configuration is imported to the Broadcaster system component prior to this). The Broadcaster is started, followed by the Marshaller system component. Web-HMI's client browser is refreshed to download the rig's configurations. The HMI browser can be used to control and monitor the test rig's model execution process, thus corresponding messages are displayed on the screens.

The Ford-Festo test rig's model is stopped and the Oil Pan Rundown machine model is started (having imported its machine configurations to the Broadcaster, following its restart), Web-HMI's client browser needs a simple refresh action from a machine operator. The HMI browser can be used to control and monitor Oil Pan Rundown machine's model execution process, thus corresponding messages are displayed on HMI screens.

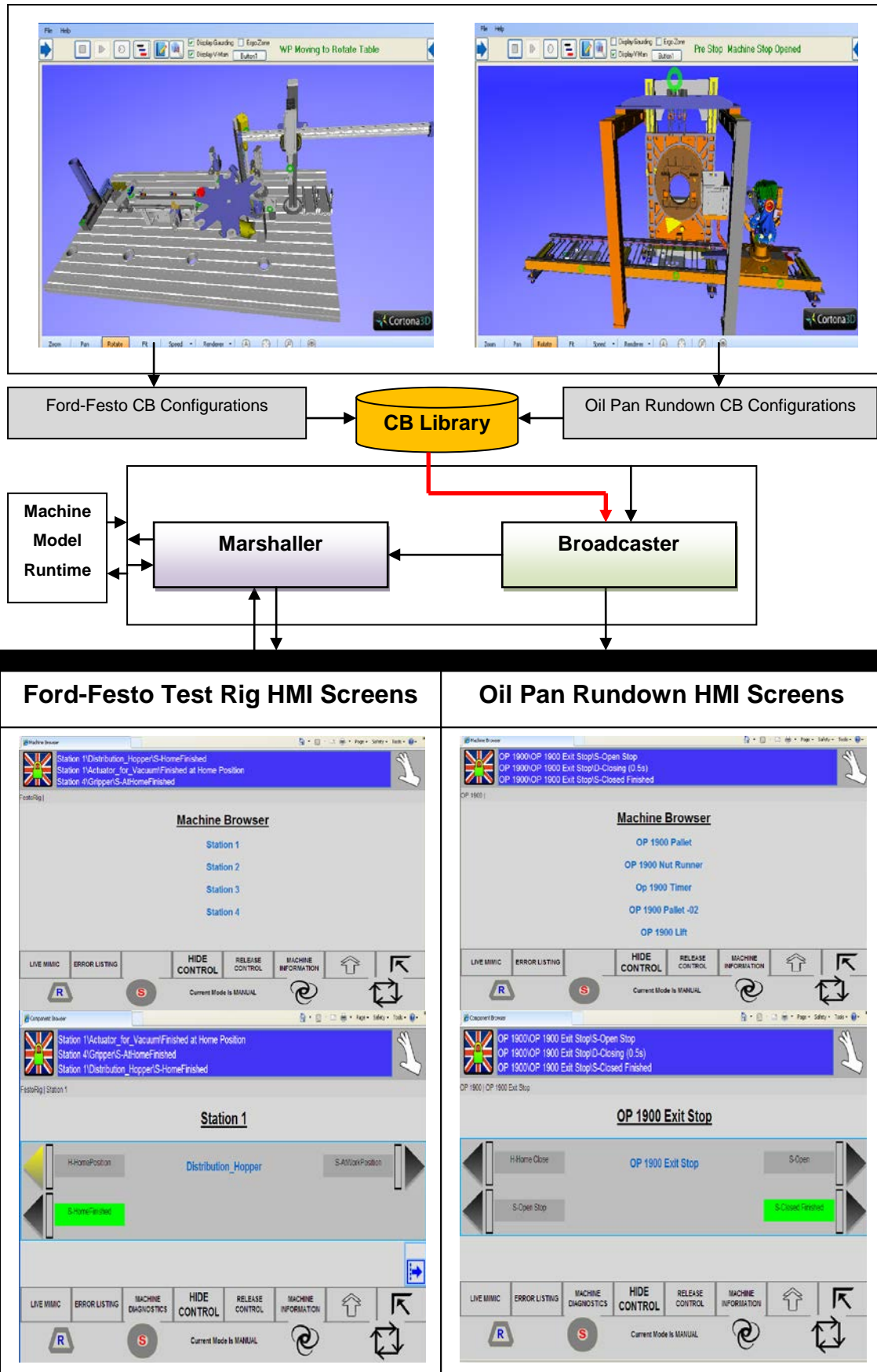


Figure 8-7: Machine Independence Scenario Case Study

8.2.7 Real-time Remote Control, Monitoring and Maintenance

Factors such as globalisation, environmental concerns and ICT demand changes to existing practices for machine control, monitoring and its maintenance in terms of remote connectivity. The web-based operator interface system must be able to remotely support lifecycle requirements to overcome shortcomings of the existing processes described in the chapter 2.

To enable these functionalities within the proposed research approach, three major implementation features are provided:

- Web-based technology adopted in implementing operator interface systems enable anytime, anywhere connectivity to manufacturing machines regardless of their geographical locations.
- A three-dimensional VRML machine simulation model integrated within the Web-HMI system component enable engineers to visualise the exact machine status and not rely on the mental machine models as traditionally practiced. This is very useful during machine maintenance process.
- A distributed system architecture having Broadcaster (to support real-time monitoring) and Marshaller (to support real-time control), enabling machine configurations to be shared throughout its lifecycle.

For verification of these attributes, two scenario cases are setup. The first scenario aims to investigate remote control and monitoring functionality offered by the CB operator interface system, where as the second scenario aims to investigate remote maintenance capabilities of the system.

Scenario Description: Remote control and monitoring

Remote control can be very useful feature when evaluating a machine at the builder's site during the try-out phase of a lifecycle to reduce costs and save

time. Furthermore, end users may require remote control of for example, an inspection test machine in a production programme. To verify the remote control and monitoring aspect of the operator interface system, a scenario has been setup where a remotely located HMI client browser (used by a remote operator) can supply operational commands (such as start, stop and reset) to the Ford-Festo test rig. On the successful execution of these commands, the remote HMI client browser can monitor rig's status at real-time as shown in the figure 8.8.

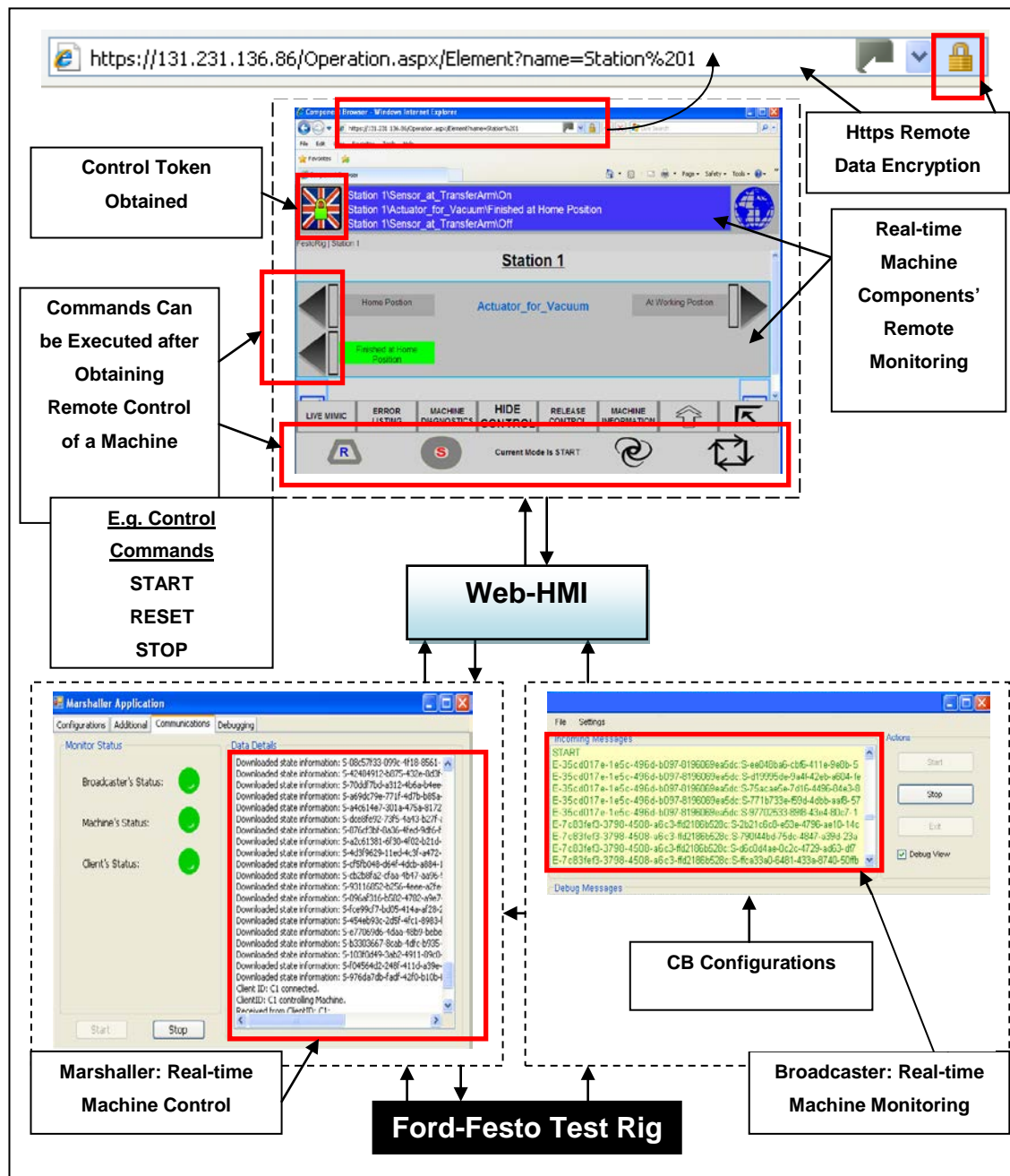


Figure 8-8: Real-time Control and Monitoring Scenario Case Study

Initially, remote HMI obtains rig's control token from the Marshaller and sends the "START" operational command. Marshaller propagates this command to the rig's control (via orchestrator), which in turn acknowledges the receipt of it and simultaneously starts operating the rig (if there are no pre-existing errors). Test rig's operation generates machine status data which gets propagated to the Web-HMI through the Broadcaster. Every component of the test rig has one or more states defined during its design phase of the lifecycle using the CCE engineering tool. Once the test rig is started, machine's state transitions can be monitored in the operator interface system. These state transitions are populated on the HMI screens at runtime from the CB configurations distributed by the Broadcaster system component, enforcing consistent machine information display across various connected HMI client screens.

Similarly, commands like "STOP" and "RESET" are also issued by the remote client's machine operator when needed. If the machine is operating in a manual mode, individual components can also be controlled by executing the required state using the operator interface client browser. Within this case study setup, the test rig's control only supports automatic mode. Once the control is locked by a client browser, the Web-HMI system provides the ability to change a machine mode from manual to automatic and vice versa. Any other remote or local HMI client browsers cannot obtain test rig's control token although simultaneous monitoring is possible by all of them.

To address security concerns when remotely connecting to the Web-HMI, user authentication mechanism (in terms of a combination of IP control whitelisting and account credentials) is strictly applied. Furthermore, data transmission over the web (i.e. from the Web-HMI server to the HMI client browser) is encrypted using the HTTPS protocol implementation as shown in the chapter 9.3.

Scenario Description: Remote Maintenance

A scenario for machine maintenance has been setup where end user's diagnostic engineer needs machine builder's diagnostic engineer's support from a distant location to diagnose machine faults using the operator interface

system. Two different types of errors (i.e. faults) are introduced into the Ford-Festo test rig such as:

- Parts are jammed on the conveyor drive (station 2) and
- Sensor pairs check at the transfer arm (station 4).

While remote control and monitoring aspect of this research has already been verified previously, this scenario looks at the practical applicability of the operator interface system to support machine maintenance at Ford's manufacturing plant, using approved and acceptable approach. RemoteIMS tool (approved by Ford Motor Company, UK) is therefore used to manage these support sessions as shown in the figure 8.9. This tool allows a local operator interface screen browser at the end user's site (with integrated 3D machine model functionality) to be shared over the internet with the remote diagnostic engineer through acceptable maintenance standards.

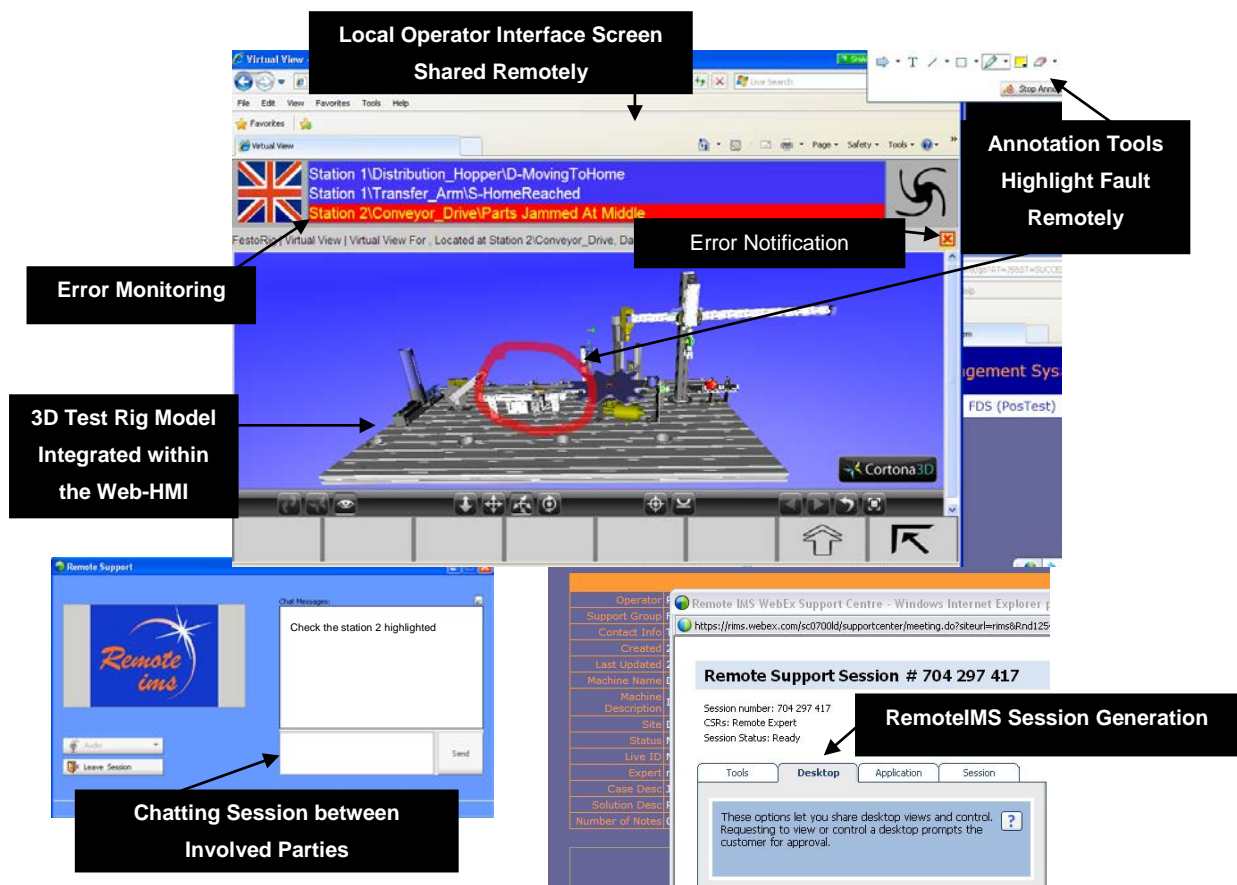


Figure 8-9: Remote Maintenance Scenario Case Study

In the above case study, once the fault occurs at the local site (for example, parts get jammed at the conveyor drive), a RemotelIMS session is initiated by the end user's diagnostic engineer and the corresponding machine builder's diagnostic engineer is contacted (via SMS / Email) automatically through the RemotelIMS system. The machine builder's engineer can join the support session and ascertains that the cause of the problem occurred in a manufacturing machine (thousands of miles away) is identified and resolved.

With the help of the 3D model integrated within operator interface screen, the machine diagnostic engineer is able to highlight the location of the error and provide clear instructions to the end user's engineer on how to go about resolving this issue. The 3D machine model facilitates verification of parts location and their position on the machine. While in traditional maintenance procedure, lots of time is usually spent on establishing the cause of a problem owing to lack of machine visualisation functionality within the HMI systems, this research approach provides a detailed view and a good description of the problem to remote engineers.

8.3 Stage 2 Case Study: Oil Pan Rundown Machine Simulation

Oil Pan Rundown (referred to as Op 1900 in this thesis) is an assembly automotive machine recently implemented at Ford's Fox three-cylinder gasoline engine program. Assembly automation involves the design and manufacture of machine that enables physical components to be assembled into complete manufactured parts or sub assemblies. Figure 8.10 shows a typical layout of an engine block assembly plant (adapted from the machine builder J. A. Krause Machinenfarik GmbH). It consists of a transport system that links assembly and test stations. Engine blocks are loaded onto empty pallets on the transport system (at the "Start") before being carried to different assembly stations where various engine parts such as pistons, conrods and cylinder head are assembled to the block. The assembly stations (i.e. machines) operate independently of each other in a modular fashion. At the end of the assembly process, fully assembled engines are tested in hot test cells prior to being unloaded and sent

to vehicle assembly plants [59, 80]. Op 1900 is one of the automatic assembly machine implemented within the Ford's Fox engine program (described next).

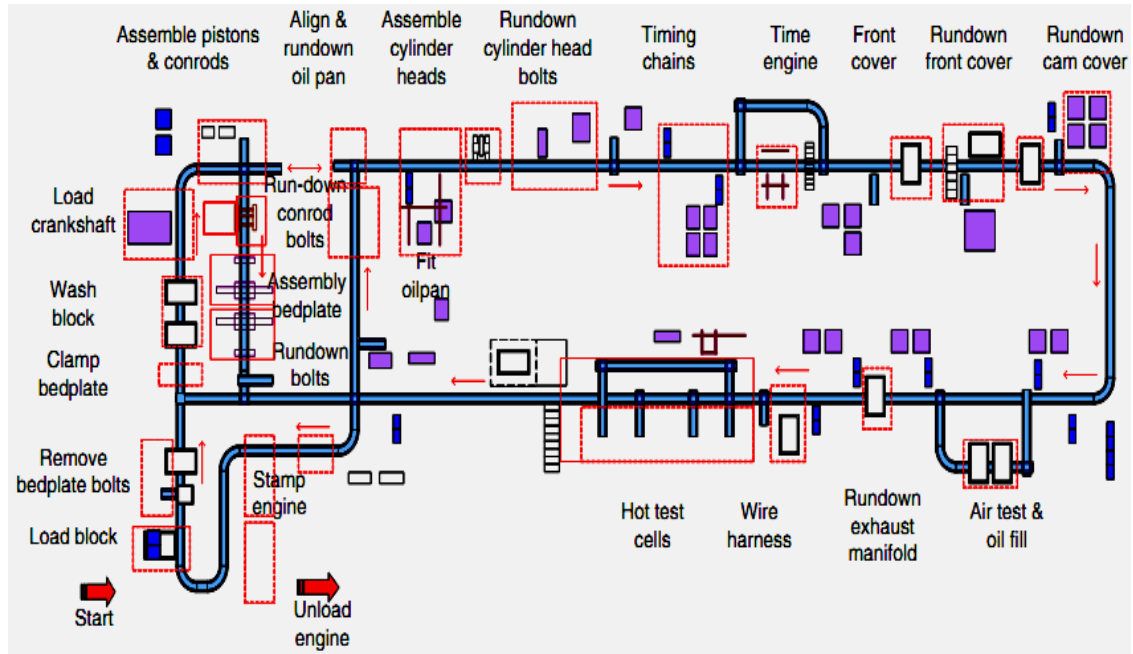


Figure 8-10: Example of Engine Assembly Plant

8.3.1 Oil Pan Rundown Description

This case study is based on application of the CB operator interface system concept by validating it through the Oil Pan Rundown machine used within the Fox engine assembly program, specifically for nut running purposes. At this phase, only a simulation for Op 1900 machine is available for this research study. This machine model (as shown in the figure 8.11) has been engineered during the design phase of the Op 1900 machine lifecycle using the CCE engineering toolset at Loughborough University. The machine consists of major components (as shown in the table 8.2) which are simulated based on a process timing sequence chart (as shown in the figure 8.12) provided by Krause (i.e. the machine builder).

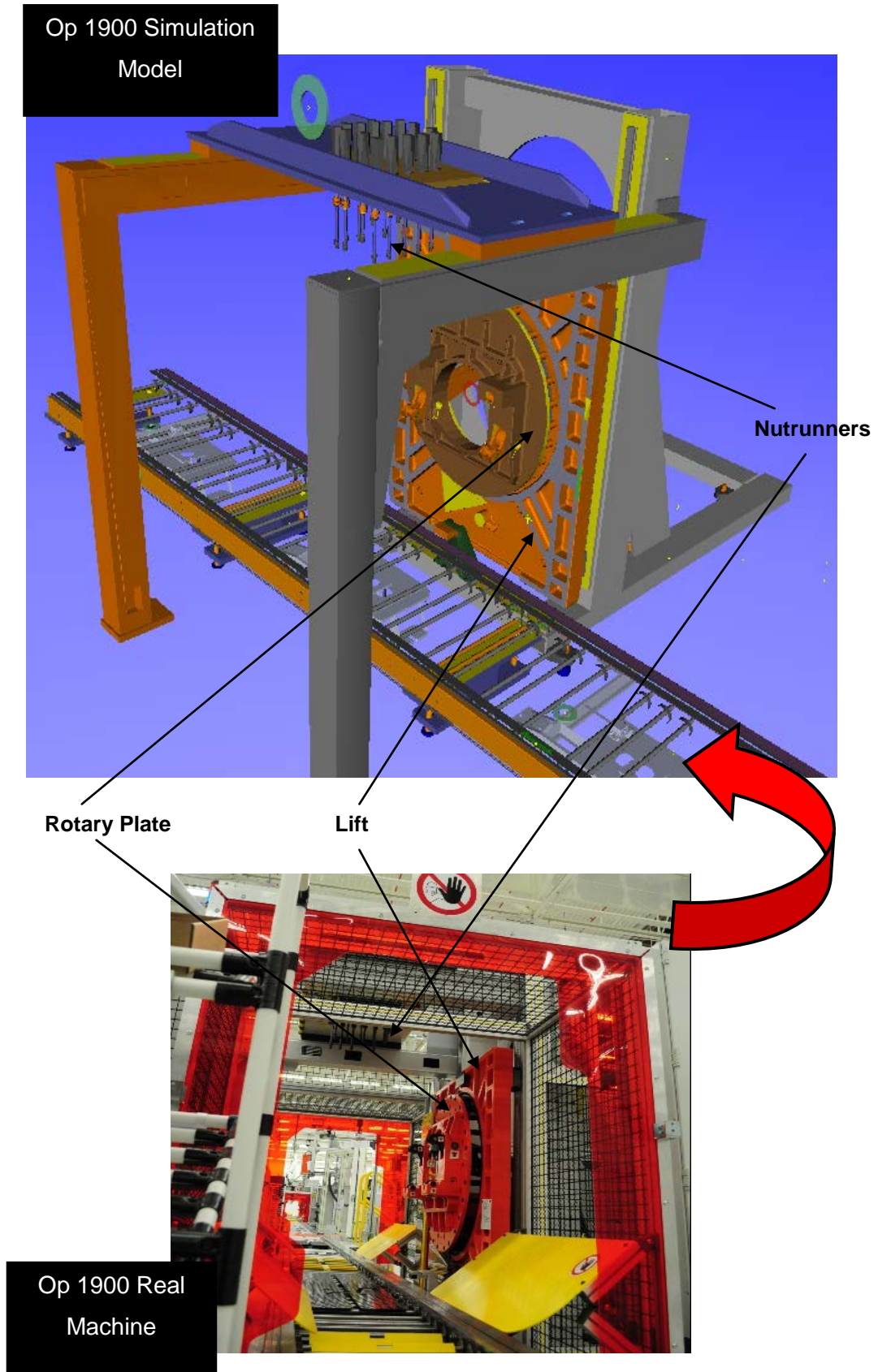


Figure 8-11: Oil Pan Rundown Machine

During the assembly process, a pre-stop opens and a pallet (with a gasoline engine) arrives at the Op 1900 machine. Pallets have RFID (Radio Frequency Identification) tag information mounted underneath which serves for various purposes such as sorting the pallets, and reading / writing process data. As the pallet arrives, it initiates a data tag reading process while an engine plate is clamped on the rotary plate. The engine is then raised to a tightening position where bolts are tightened by the Nutrunner spindles (where 15 oil pan bolts are rundown to a final torque).

On completion of this process, the engine is lowered to a rollover position by the lifting unit where it is rotated counter clockwise by 180 degrees. The engine is further lowered to the clamping position while data is written to the pallet's tag. The engine plate is then unclamped and when the stop opens, the engine leaves this station.

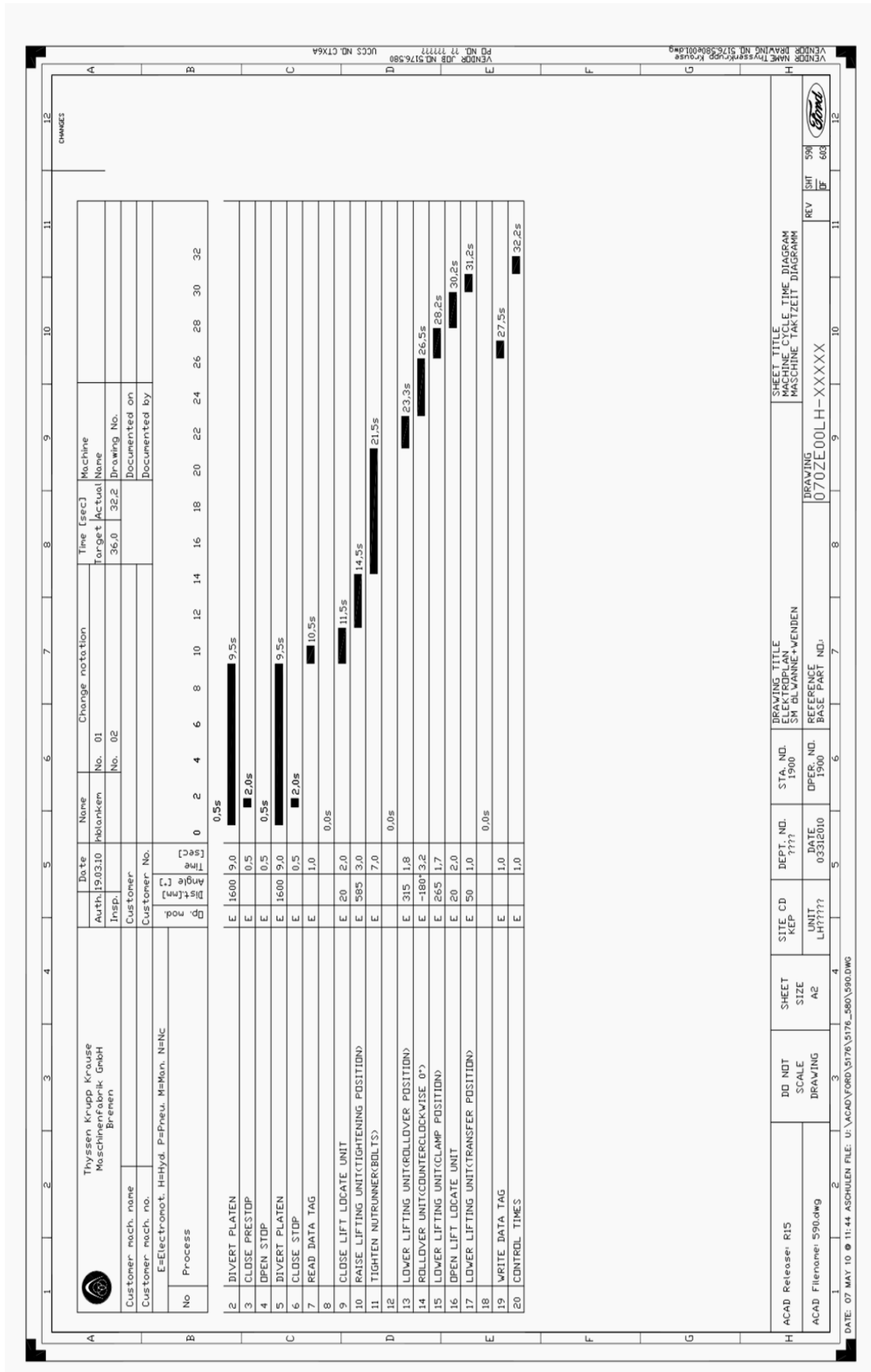


Figure 8-12: Op 1900 Process Time Chart

8.3.2 Case Study Setup

Figure 8.13 illustrates a schematic of the overall case study setup when validating this research using the Oil Pan Rundown machine simulation, designed using the CCE engineering toolset. On starting the machine simulation, a TCP/IP socket connection with the Broadcaster is established by the CCE environment to propagate machine's published status (in this case, the Op 1900 machine). The machine model accepts incoming control commands from the Marshaller via an alternative socket TCP/IP link. Machine status is broadcasted to all the connected clients (including the Web-HMI and the Marshaller system components) through the Broadcaster. Web-HMI, in turn, serves local operator interface client browser as well as a remote browser. An alternative remote connectivity link can also be established by initiating a RemoteIMS screen sharing session (concept described in the chapter 4.4 and example case study demonstrated in the section 8.2.7) from any local HMI client browser's PC.

The system components are deployed in a PC-based environment communicating over the Ethernet / Internet. Each system component has been given a specific IP address enabling them to operate in a distributed fashion (similar to the Ford-Festo test rig case study described in the section 8.2.2). The HMI system console consists of a standard computer touch screen (non vendor-specific) operating using the explorer (standard internet web browser).

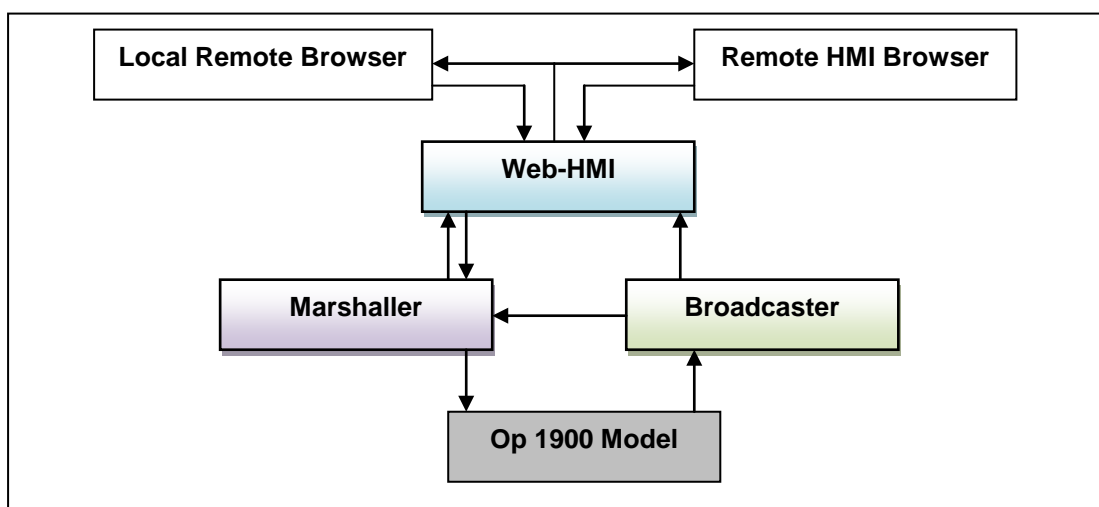


Figure 8-13: Op 1900 Machine Case Study Setup

8.3.3 Research Attributes Assessment

Some of the identified requirements of this research (summarised as attributes in the chapter 3.3) are assessed using scenarios and demonstration appropriate for this case study. The benefits of utilising CB operator interface system in addressing these attributes have already been discussed in the chapter 2.4. The three main attributes tested in the subsequent sections are as follows:

- Virtual Machine Validation.
- Early HMI Verification.
- Early HMI Training.

8.3.4 Virtual Machine Validation

Validating a machine virtually prior to its build is essential in identifying faults beforehand during its lifecycle to avoid unnecessary surprises. To enable these functionalities within the proposed research approach, three major implementation features are available:

- A three-dimensional VRML machine simulation model integrated within the Web-HMI system component enable supply-chain partners to visualise and understand the machine behaviour prior to its build process (regardless of its geographical location).
- Machine design information is stored in CB configurations that are shared using the distributed system architecture. These configurations can drive machines (regardless of their implementation state).
- System architecture supports real, simulated and hybrid machine operation through sharing CB configurations. Operator interface system can therefore be driven by either a real, simulated or a hybrid machine.

Components Clashing Demonstration: Ford’s Fox Machine Simulation

To verify this attribute, the Op 1900 machine simulation model is operated and inspected for any clashes that may be occurring during its design to avoid any potential issues during its build. The table 8.2 (and accompanying figure 8.14) summarises the clash type and its location identification in the model during the design phase of the lifecycle. Solving this clash is extremely essential for safe and economic operation of this machine (and the gasoline engine).

Problem Overview	Description and Location	Proposed Solution and Outcome
<p>Clash between the engine plate and the pallet is identified during the simulation process.</p>	<p>Occurs when an engine is rotating 180 degrees counter clockwise, after completing the nut running operation.</p>	<p><u>Proposed Solution 1:</u> Rotate the engine clockwise instead.</p> <p><u>Outcome 1:</u> The engine still clashes slightly.</p> <p><u>Proposed Solution 2:</u> Raise the position of the machine lift unit where it stops to rotate the engine. This position of lift unit is called “Rollover Position” in the Krause’s process time chart shown in the figure 8.12.</p> <p><u>Outcome 2:</u> No clash identified.</p>

Table 8.2: Virtual Machine Validation Demonstration

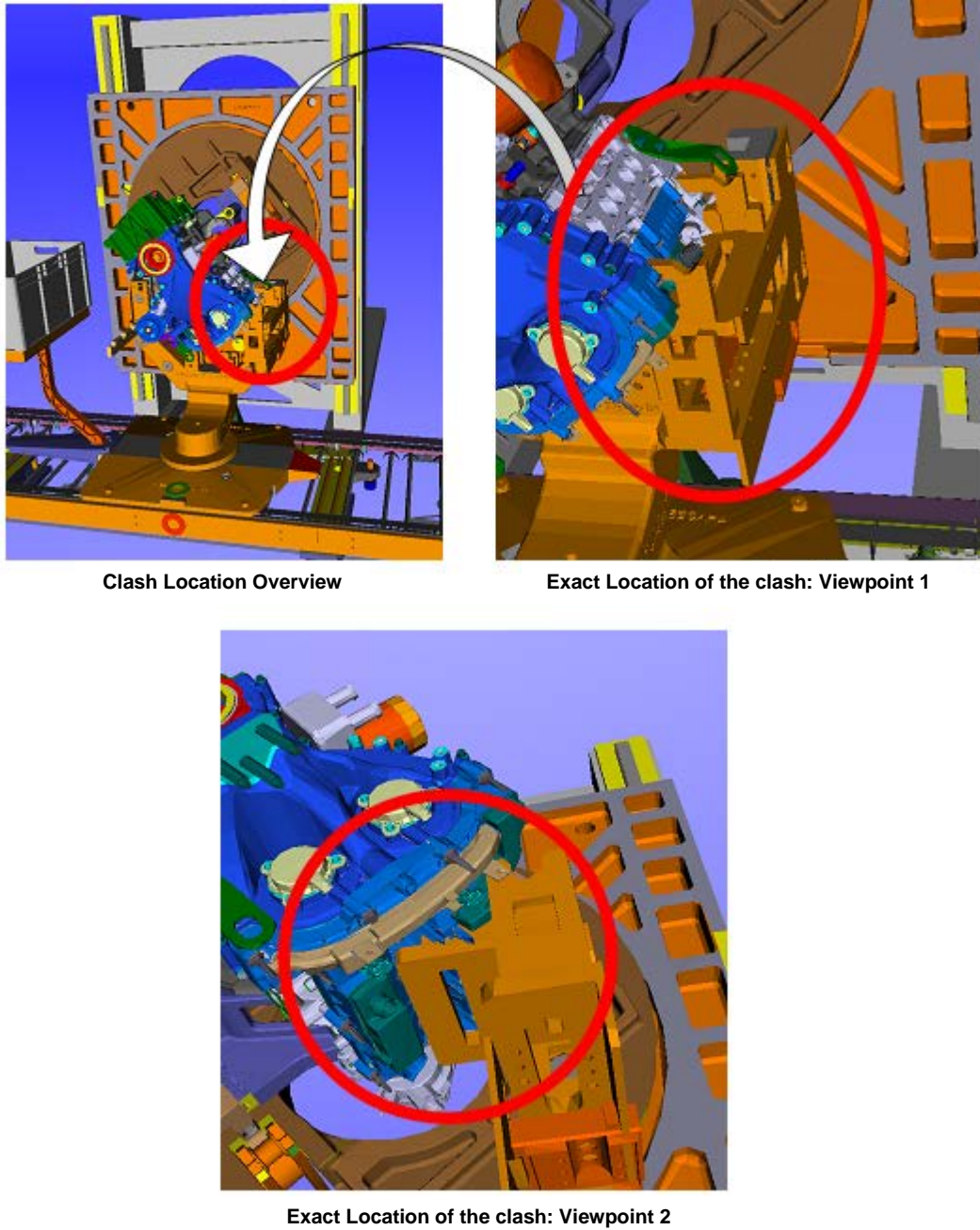


Figure 8-14: Virtual Machine Validation Case Study Demonstration

8.3.5 Early HMI Verification

While machine needs to be validated earlier on, likewise, the accompanying HMI needs to be verified too. An early verification of the operator interface system highlights any inconsistencies or operational issues on the HMI side of the system implementation prior to the machine ramp up. To enable this

functionality within the proposed research approach, three major implementation features are available:

- A three-dimensional VRML machine simulation model operation verifies operator interface system prior to the build and commissioning phases.
- Machine design information is stored in CB configurations that are populated to the operator interface screens at runtime, providing consistent display to the machine operators.
- System architecture supports real, simulated and hybrid machine operation through sharing CB configurations. Operator interface system can therefore be verified by either a real, simulated or a hybrid machine.

Figure 8.15 illustrates ramp up activities for a new production machine. The tools within the CCE engineering environment support the machine's mechanical design. All the machine configurations are housed in the common data model (i.e. CB configurations) that resides with the Broadcaster system component. Web-HMI system component de-serialises these CB configurations upon connection with the Broadcaster and thus all the necessary information required by the operator interface system is downloaded to the browser screens at runtime. This enables HMI to be verified prior to the machine build and commissioning phase, and machine operators to be trained earlier.

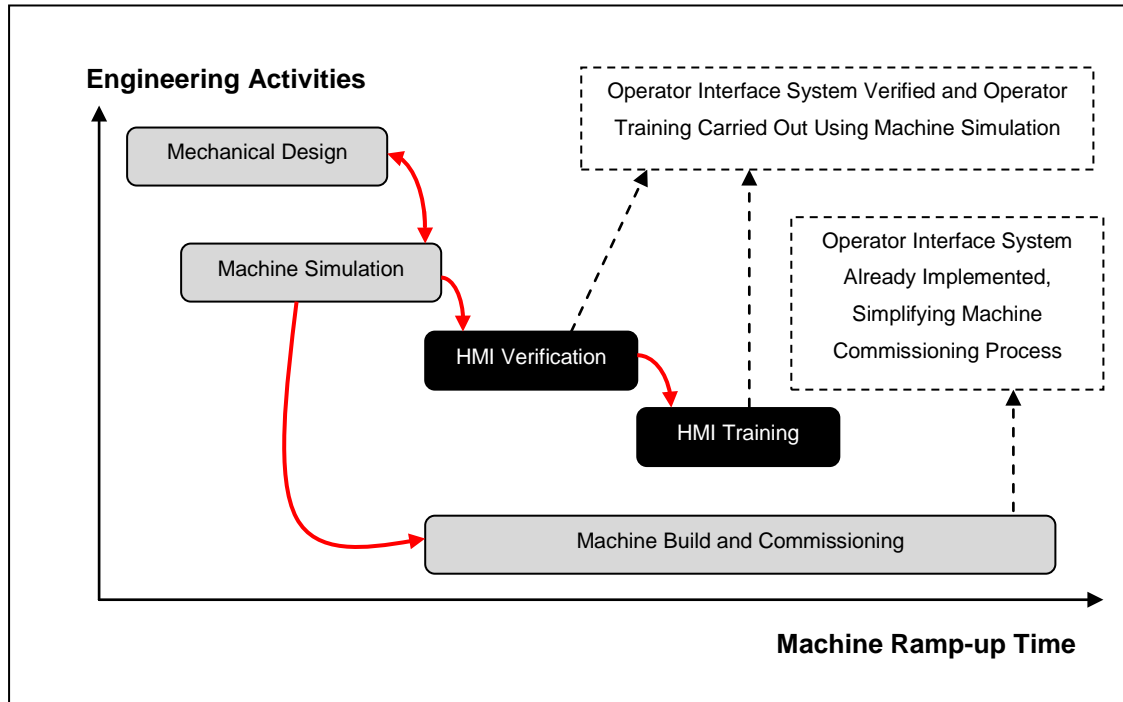


Figure 8-15: Machine Ramp up Activities Within this Research

Scenario Description: Web-HMI Consistency and Operation

Two aspects of the operator interface system are addressed in this scenario. The first one is the consistency and the second one is the operation of the HMI (in the absence of a commissioned (i.e. real) machine). A scenario is setup (as shown in the figure 8.16) where the Op 1900 machine simulation model is initially operated with a default set of components (having their unique names defined during the machine design using the CCE engineering tools).

Operator interface screens are populated at runtime from the CB information that reflect the same configurations as defined within the machine model. Later, machine model is modified (for example, an additional component is introduced in the machine, or an existing component's name is changed). The output CB configuration (for the latest machine model) is reflected consistently at the HMI side, and any machine model operation updates the corresponding component states on the HMI screens. Furthermore, other essential operational aspects such as navigation links are dynamically updated based on these CB configurations.

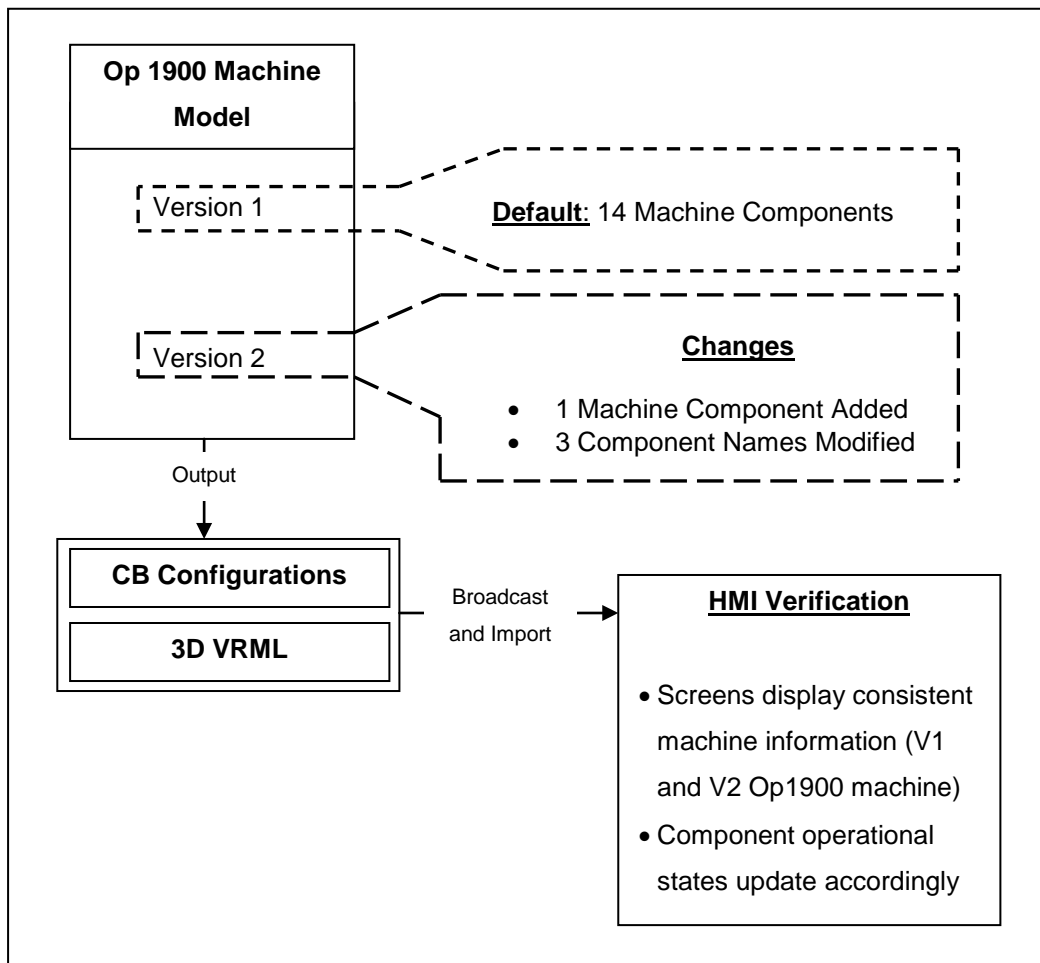


Figure 8-16: Early HMI Verification Case Study

8.3.6 Early HMI Training

Machine operators are required to be trained earlier using the implemented HMI to enable them to understand the machine behaviour effectively, to familiarise themselves with the HMI interaction and to speed up the machine ramp up process. Implementation features to achieve this functionality are similar to the ones described in the section 8.3.5. Using the CB operator interface system, both real and simulated machines can be driven, enabling engineers to be trained on the operation of the machine prior to its commissioning process.

Operators need to be trained on various aspects of the HMI system. These are:

- Identifying changes to the operator interface screens with every machine reconfiguration process (highlighted in the scenario of the section 8.2.4).

- Remote monitoring and control of the machine (highlighted in the scenario of the section 8.2.7).
- Maintenance support process (highlighted in the scenario of the section 8.2.7).
- Supporting machine validation process (highlighted in the scenario of the section 8.3.4).
- Familiarisation with the operator interface screens and their corresponding navigational structure (highlighted in the scenario of the section 8.3.5).

8.4 Stage 3 Theoretical Case Study: Fox Assembly Plant Layout

Practical application of the CB operator interface systems by assessing it through industrial case studies has been covered in the sections 8.2 and 8.3; however, this section studies the Ford's Fox engine plant architecture and theoretically demonstrates the possibilities of implementing the system components architecture (described in the chapter 5.3) by mapping its components to the actual production plant layout. This study is essential in identifying the physical locality of Broadcaster, Marshaller and Web-HMI to support control and monitoring operations for different types of machines.

8.4.1 Production Plant Architecture Description

Figure 8.17 shows a simplified version of the Fox production plant architecture where the overall layout is broken down into zones, having various stations. A zone corresponds to a logical decomposition of the Fox production line into various manageable sections. There are approximately 10 zones in total, having approximately 10 to 12 stations each. A station in this context corresponds to the actual production machine with its associated devices like PLC, HMI, etc.

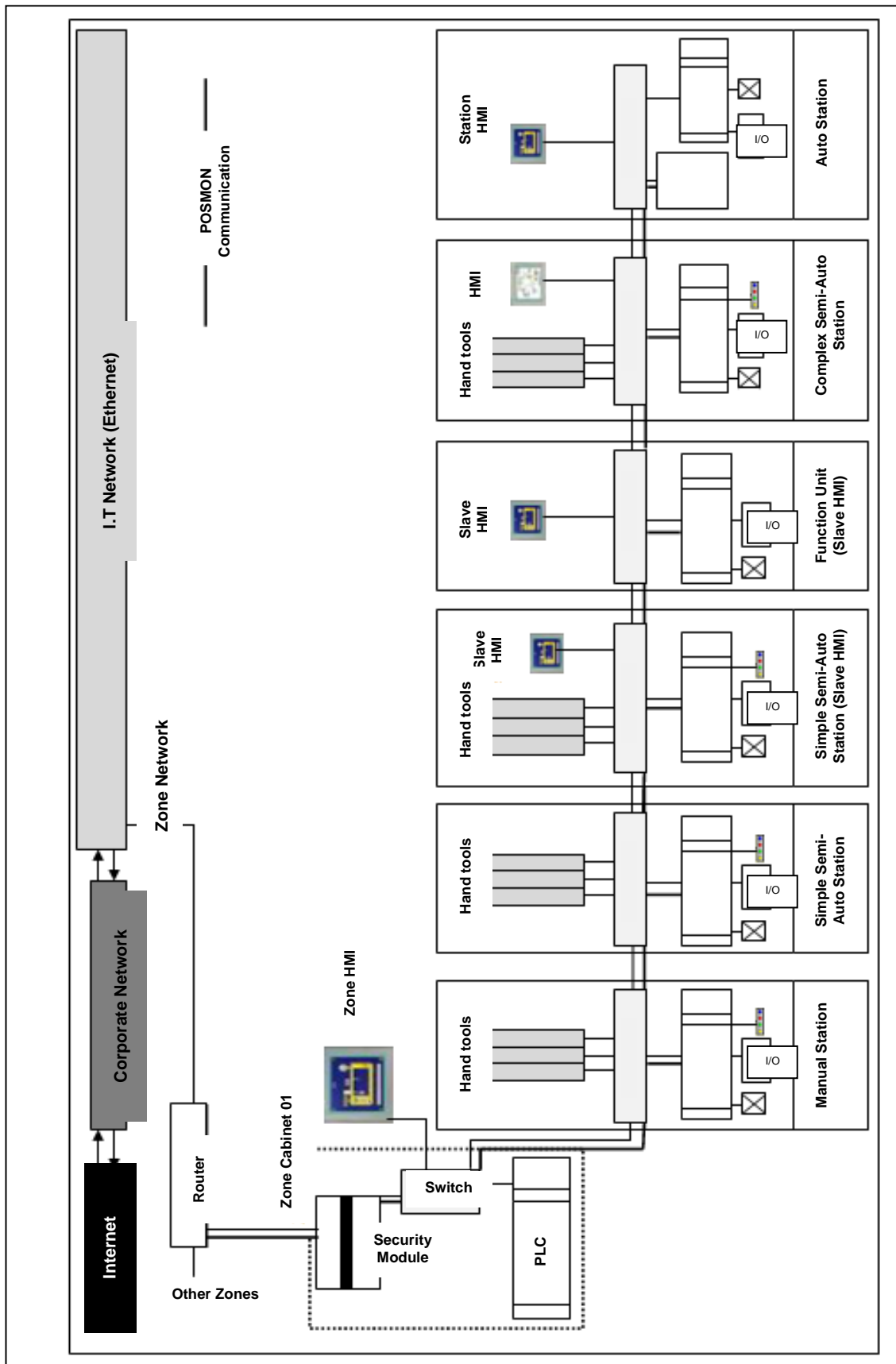


Figure 8-17: Fox Production Program Plant Architecture

A zone can have a mixture of different types of stations namely manual, auto and semi-auto. With reference to a recently implemented Fox line in Craiova, the total number of stations amount to 112, out of which manual stations add up to 80, auto stations add up to 12 and semi-auto stations add up to 20. Each zone has a zone cabinet where a master HMI panel exists. Furthermore, every zone has a zone PLC, security modules and switches for control and network management. In each zone, at least 50% manual stations exist.

Each station type has a different configuration when implemented at the shop-floor as follows:

- Auto Station

An auto station's machine operational sequence is automatically executed following a cycle start request from an operator. Likewise, it can be stopped at the end of the cycle using a cycle stop request. Physically, auto station consists of its own station PLC and a station HMI. A zone HMI merely controls an auto station besides supplying a global start and global stop. Op 1900 machine (described in the section 8.3.1) is an example of an automatic machine station.

- Manual Station

A manual station's machine provides individual movements (regardless of its sequence) and other control actions can be made by using operator push buttons. Physically, manual station consists of its own PLC but does not have a HMI. A machine operator uses push button control box to move machine components. A manual station is monitored using its zone HMI.

- Semi-Auto (Simple) Station

Physically, a semi-auto simple station is controlled by a zone PLC and has local push buttons. It does not have a local HMI (by default) as it is monitored by the zone HMI. Sometimes, Ford may request a slave HMI to exist in this station to control an independent operating area.

- Semi-Auto (Complex) Station

Physically, a semi-auto complex station is controlled by its own local PLC and has its own HMI. Sometimes, this station is implemented as a device of a standard manual station where by the PLC of the manual station becomes the master controller and the semi-auto station's PLC acts as a slave device. No clear information has been obtained from the end user's regarding the possible physical configuration for this station.

8.4.2 System Components to Plant Architecture Mapping

The system architecture components can be mapped to the Ford's plant architecture using an implementation strategy which takes performance and cost-effectiveness into account. The proposed strategy is to have the Broadcaster and the Marshaller in the same location as the HMI (either in a single PC or different PC's) to avoid any network performance bottlenecks, and to make the system implementation cost-effective. As identified in the section 8.4.1, various different types of stations exist, each having a different configuration; mappings for system components are can be applied as follows:

- Auto Station

For an automatic station, Broadcaster, Marshaller and Web-HMI can possibly be installed in an auto station PC. A number of local HMI browsers (using vendor-independent standard touch screen panels) can monitor and control the automatic station using the Broadcaster and the Marshaller respectively. Any remote connectivity can be initiated using a standard internet explorer browser from a zone cabinet location, or from the I.T network, corporate network or even from a distant location over the internet (if needed) as shown in the figure 8.18. POSMON data can also be collected at the I.T network directly from the Broadcaster.

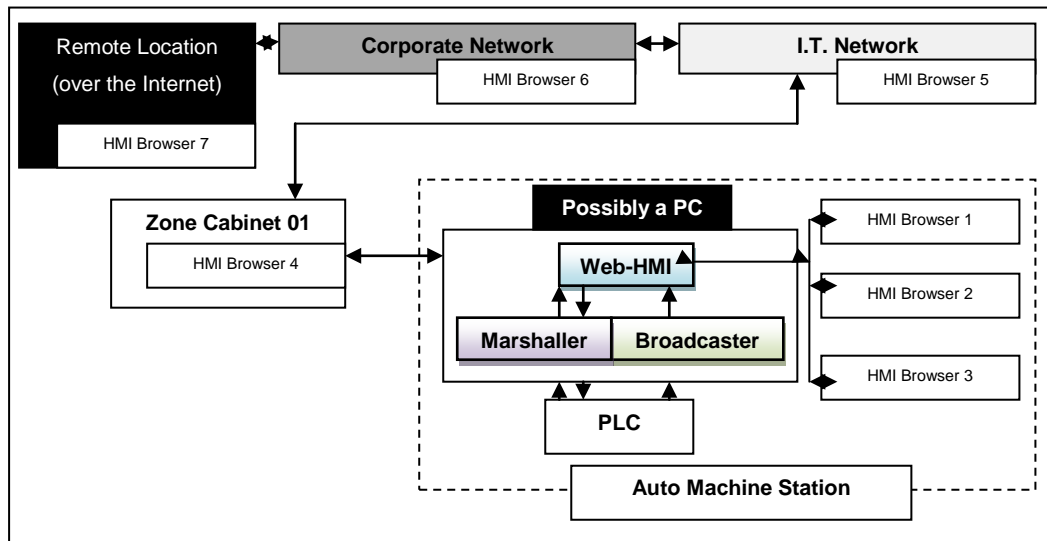


Figure 8-18: System Components to Auto Station Mapping

- **Manual Station**

For a manual station, Broadcaster, Marshaller and Web-HMI can possibly be installed in a zone cabinet PC. A number of local HMI browsers (using vendor-independent standard touch screen panels) can monitor and control entire zone's manual stations using the Broadcaster and the Marshaller respectively as shown in the figure 8.19. Should a need arise to remotely connect to the manual station (or locally connect from an immediate vicinity of the machine), a standard internet browser is sufficient. POSMON data can also be collected at the I.T network directly from the Broadcaster.

- **Semi-Auto Station**

For a semi-auto station (simple), Broadcaster, Marshaller and Web-HMI can possibly be installed in a zone cabinet PC (similar to the manual station mapping described earlier). Since complex semi-auto station's configuration details are incomplete / inconsistently obtained from the end user's engineers, no specific analysis of the system components mapping for a complex semi-auto station is provided in this thesis.

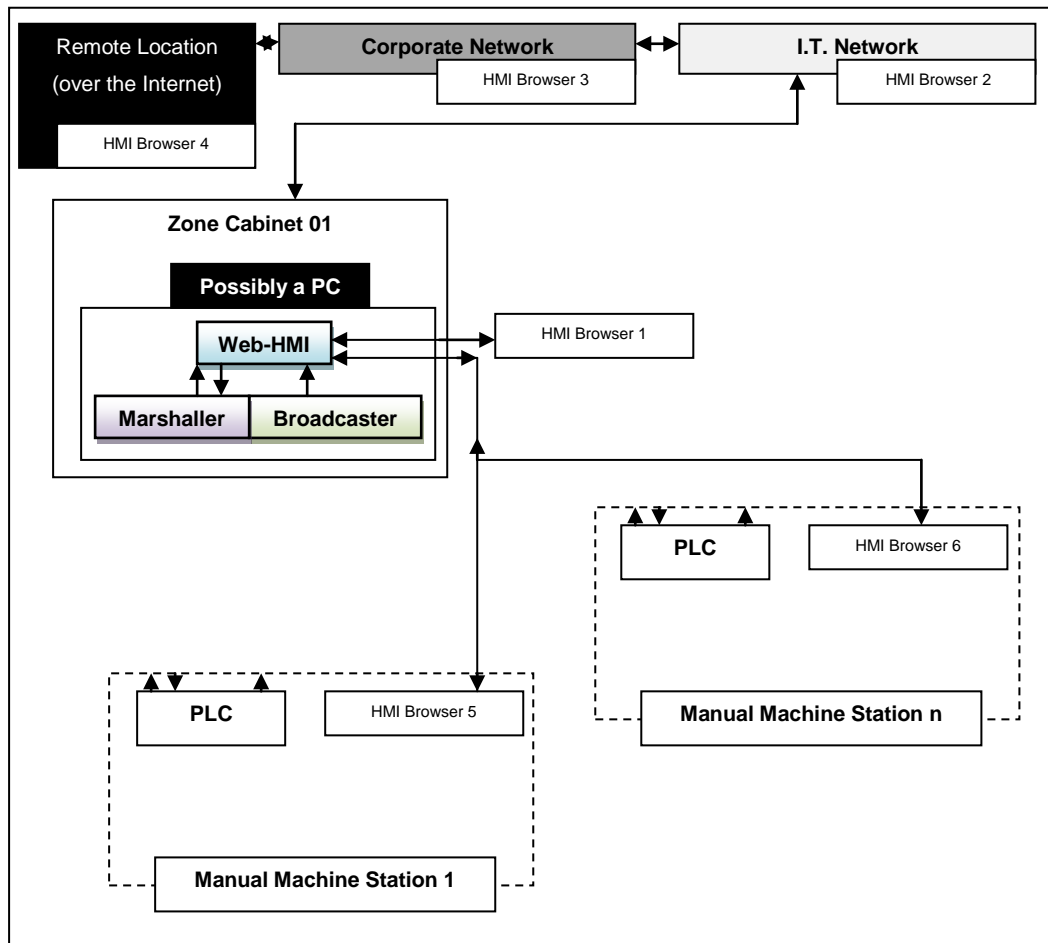


Figure 8-19: System Components to Manual Station Mapping

8.4.3 Considerable Issues

To practically apply and support the system components to plant architecture mapping discussed in the section 8.4.2, some runtime issues need to be considered. These issues are instigated by raising various research questions regarding the practicality of the implementation strategy described earlier. These are described in the table 8.3.

Research Considerations	Addressing Issues
<p>How many machines can the system architecture support for the manual station strategy discussed earlier?</p> <p>This is owing to fact that the system</p>	<p>Scalability of the Broadcaster and the Web-HMI needs to be evaluated. This is carried out in the chapter 9.7 of this thesis.</p>

<p>components are attempting to support an entire zone having at least 50% manual stations.</p>	
<p>Is the communication between the system components operating in a fail-safe manor?</p>	<p>Communication robustness between the system components needs to be evaluated. This is carried out in the chapter 9.5 of this thesis.</p>
<p>What is the message propagation delay within the system architecture?</p>	<p>Performance in terms of response times need to be evaluated. This is carried out in the chapter 9.6 of this thesis.</p>
<p>How do supply-chain partners ensure that their own engineering tools can be plugged into the system components architecture to support various functionalities they need, through the security model existing at the Ford's Fox plant?</p>	<p>A remote serving client needs to be implemented at higher I.T levels, such that it propagates Broadcaster's published machine status messages at real-time in uniform format, by establishing a secure channel (through the Ford's security model) to remote parties. This uniformity can enable third-party vendors to decode and utilise remotely collected information accordingly. This is acknowledged within the future work section in the chapter 10.2 of this thesis.</p>
<p>How does the Broadcaster and the Marshaller communicate with a PLC?</p>	<p>This can be achieved through OPC connectivity. A practical evaluation of this is undergoing and has been acknowledged in the chapter 10.2 (future work) of this thesis.</p>

Table 8.3: Considerable Issues when Implementing System Architecture at Ford's Plant

Chapter 9 : System Components Evaluation

Chapter Contribution to this Thesis:

The main contribution of this chapter is the behavioural evaluation of the system components using their non-functional but critical system properties for their industrial acceptance and application, specifically in the powertrain manufacturing automation.

9.1 General Overview

As described by Somerville [116], emergent properties of a complete system can only become apparent once the system components have been integrated in their operational state. The functional emergent properties have already been evaluated through case studies in the chapter 8; however, the non-functional emergent properties relating to behavioural evaluation of the system components in its operational state is carried out in this chapter. The importance of non-functional emergent properties cannot be underestimated as failure to achieve some minimal defined level in these properties can make the system unusable [116]. The critical system properties focused for evaluation are safety, security, reliability, robustness, performance and scalability.

9.2 Safety

For an industrial acceptance of this research, evaluation of the safety operation strategies implemented within the system components is very essential. Safety in this context corresponds to the protection of the system's environment and its users against hazards caused by incorrect functioning of the system components [170]. The essential safety criteria that need to be evaluated for the system components are:

- Regulating machine control through implementation of industry best practice i.e. implementation of machine control safety procedures.

- Avoiding accidental execution of operational commands through their confirmation accomplishment.

To satisfy the above criteria, a number of safety strategies have been implemented and evaluated as discussed next.

9.2.1 Configuration Whitelisting and Token Sharing

The industrial requirement is to grant only one operator interface system the ability to control the machine at a time to avoid any operational disasters, for example, one operator may use a HMI client browser to issue a command to move an actuator while another operator directs the machine to return to its initial position using another HMI client browser; these commands confuse the machine and crashing is certain. To avoid such circumstances, the system architecture applies the control sharing mechanism as described in the chapter 7.4.4. The underlying safety strategy implements configuration whitelisting and token sharing.

Configuration whitelisting supports an approved list of IP addresses (as shown in the figure 9.1) and client Id's which are privileged enough to request machine control. Any client outside the whitelist is automatically blacklisted thus is unrecognised and denied to even initiate a machine control request session with the Marshaller system component. Token sharing creates a key value pair within the system such that only one control token is available for all the client connections. If the token has been consumed, the returning value turns to false, meaning the token is not currently available.

To evaluate this safety strategy, three HMI client browsers are connected from different computers all at once to the Web-HMI and a control request is initiated from each, one at a time. Results obtained from this evaluation are shown in the table 9.1.

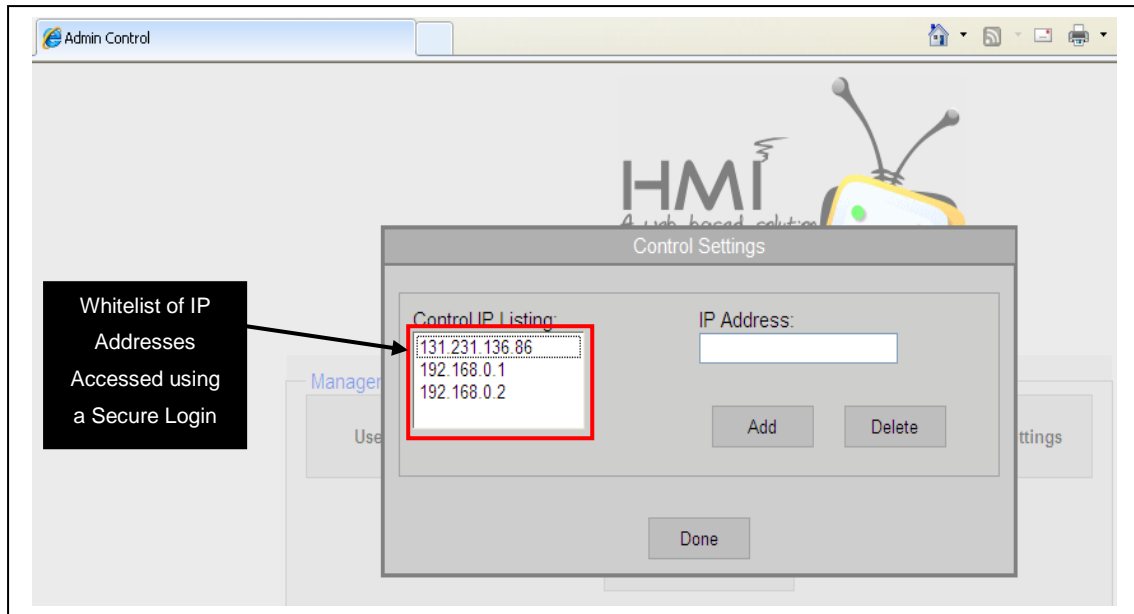


Figure 9-1: Web-HMI IP Whitelisting Strategy Implementation

Time	HMI Client 1	HMI Client 2	HMI Client 3	Comments
t1	Connects	-	-	-
t2	Idle	Connects	-	-
t3	Idle	Idle	Connects	All the three clients are connected at this point of time
t4	Idle	Request Control	Idle	-
t5	Idle	Controlling	Idle	-
t6	Request Control	Controlling	Idle	-
t7	Request Denied	Controlling	Control Attempt	Reason: Token Consumed by the Client 2
t8	Idle	Control Released	Not Initialised	Reason: Client 3 IP address not whitelisted
t9	Idle	Idle	Idle	Control Token Available

Table 9.1: Whitelisting and Token Sharing Strategy Evaluation

9.2.2 Command Execution Confirmation and Handshaking

Once the machine control has been obtained, it is extremely important to avoid executing any unintentional operational commands using the HMI client browser, and have some sort of handshaking mechanism to ascertain commands have been successfully propagated to the machine. To implement these safety criteria, the strategy adopted is to confirm every command execution on the HMI screen through double click capture events. These events force the machine operator to click a control command button twice to successfully propagate any instructions to the machine.

The first click event activates the command button and changes its background colour to yellow, indicating the corresponding activation. The second button click event propagates the control command to the machine and changes its background colour to green, confirming the command execution success at the HMI side. When the machine logic acknowledges the receipt of this command, the corresponding blue colour change is displayed on the screen, confirming that the message has been successfully delivered from the operator interface system to the machine [187].

To evaluate these strategies, a simple commissioning test scenario is conducted where a manual machine system's components have to be driven by the operator interface system. The main functionality to evaluate is to ensure that the operator can manually select and move all the machine components to their desired states safely using the HMI screens. The process through which this is evaluated is as follows (shown in the figure 9.2):

- The machine control is acquired by an HMI client browser.
- The machine is set to be in “Manual” operation mode by pressing the manual command button on the screen.
- Since the machine components' states are displayed as buttons on the screen, clicking them twice propagates the corresponding machine move request to the machine logic.

- Acknowledgement is received and displayed at the client browser screen. Depending on the machine logic's decision, the desired components' states can be safely moved.

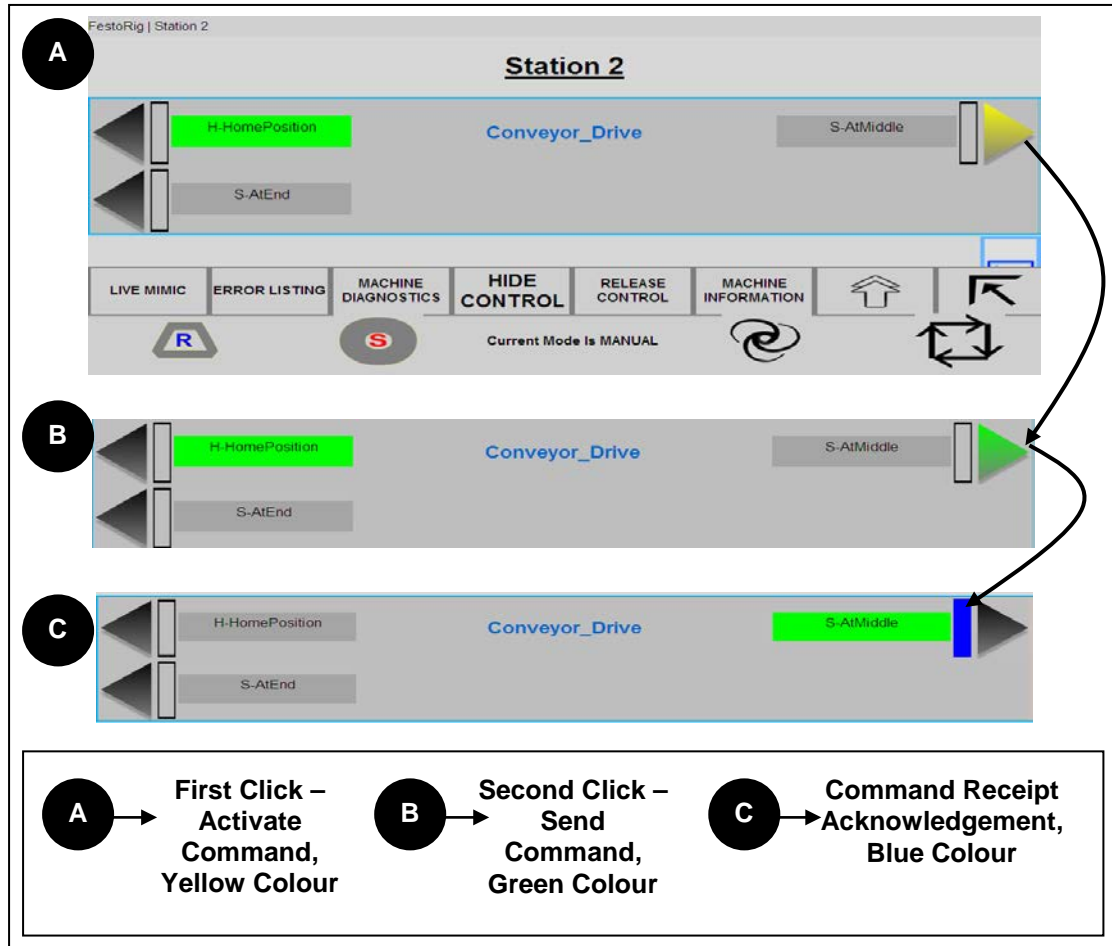


Figure 9-2: Command Execution Confirmation and Handshaking Strategy Evaluation

9.3 Security

Implementing web-based operator interface system solution with remote connectivity (for example, within the case study described in the chapter 8.2.7) raises important security concerns. Remote connectivity requires internet access, which may in turn trigger possibility of unauthorised access to manufacturing process data, raise safety concerns when working with machines and ultimately lead to financial loss for an automotive company. The essential

security criteria that need to be evaluated for the system components are [116, 170]:

- Protecting manufacturing data from unauthorised third-party access over the Internet i.e. confidentiality of information through identification of users and their entitlements.
- Avoiding any accidental or deliberate unauthorised manipulation / damages to machine message contents i.e. integrity of information through application of data ciphering.

To satisfy the above criteria (i.e. confidentiality and integrity) within the system architecture, two security strategies have been implemented and evaluated, one for each criteria. The first one is authentication followed by authorisation, and the second one is data encryption. With respect to the availability of the Web-HMI system, local mechanisms outside the system can be deployed such as firewalls and antivirus protection to ensure that the system is available to authorised users.

9.3.1 Authentication and Authorisation

Authentication through proof of HMI users' identity is implemented and evaluated. Only when HMI users are authenticated using the Web-HMI system component, they are authorised to perform the required control and monitoring operation. This strategy has been implemented using login credentials approach (employing layered architectural technique as described in the chapter 4.6.2) where a user needs to input a correct username and password to request the Web-HMI system component to serve HMI screens on the HMI panel.

The Web-HMI system component uses ASP.NET administration tool to manage rules for securing specific resources of the application. ASP.NET uses a security system that restricts access to specific user accounts or the roles to which the user accounts belong. Two roles have been created such as "Administrator" and "Operation". The administrator role can make critical

alterations to the Web-HMI system such as configuration settings, control settings and account settings, where as operation role enables one to request HMI screens and download machine data. A number of users can be created; however, one user for each role has been created in the current implementation i.e. “Admin” user (belonging to “Administrator” role) and “Ops” user (belonging to “Operation” role) as shown in the figure 9.3.

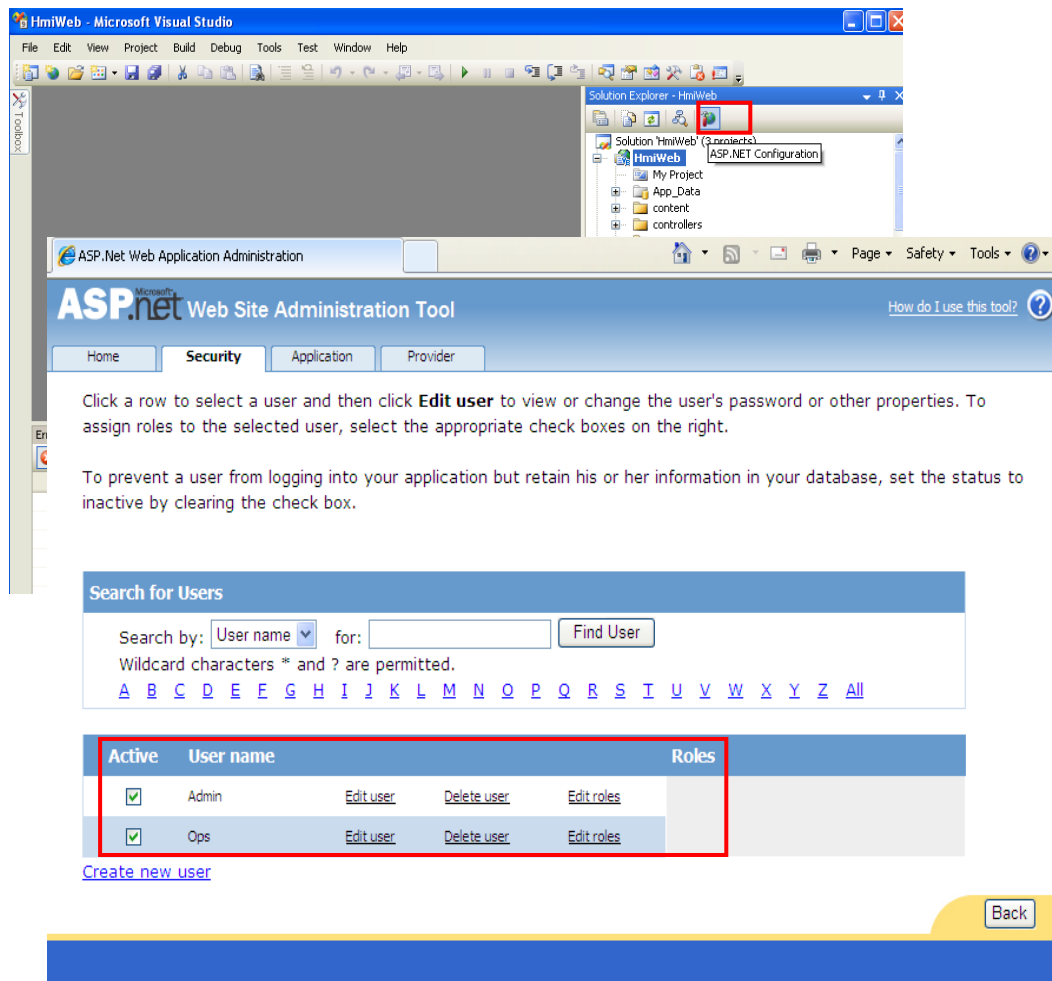


Figure 9-3: ASP.NET Web Configuration Tool for Web-HMI System Component

The approach used for evaluating this strategy is summarised in the table 9.2 where 4 major attempts are carried out to access data (HMI screens and machine operational data) using an internet explorer browser. The corresponding results obtained from this evaluation are included in the same table. Only upon providing the correct credentials to the Web-HMI system

component, a user can either access the required resource or download machine configurations and the corresponding HMI screens as shown in the figure 9.4.

Attempt Number	Username	Password	Role	Result
1	Admin	○○○○○○○	Administrator	X
2	Admin	●●●●●●●●	Administrator	√
3	Ops	○○○○○○○	Operation	X
4	Ops	●●●●●●●●	Operation	√

○○○○○○○ - Incorrect password

●●●●●●●● - Correct password

} The length and the actual password is not shown here

A number of other attempts carried out using incorrect username and correct password. The evaluation results reflect the same outcome therefore not discussed here.

Table 9.2: Web-HMI Authentication Strategy Evaluation Results

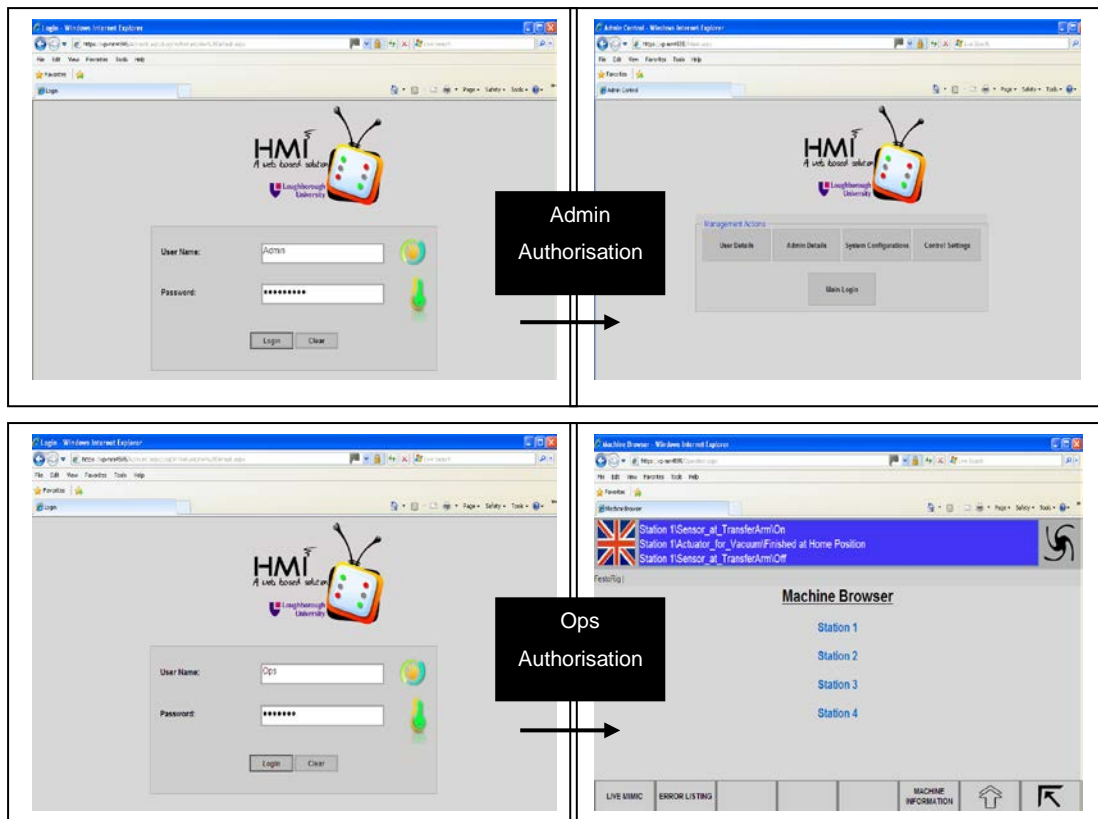


Figure 9-4: Web-HMI Authentication Strategy Evaluation

9.3.2 Data Encryption

Data transmitted between the server (i.e. Web-HMI) and the clients (i.e. HMI client browsers) is encrypted using the HTTPS implementation. It is a combination of the HTTP with SSL / TLS cryptographic protocol which provides communication encryption and secure identification of the web server component. This technique is usually used for sensitive data transmission by creating a secure channel over the insecure network (i.e. the Internet) [208]. The HMI client browser authenticates the Web-HMI through examining its certificate, transmitting requests / response over the encrypted channel implemented using algorithms found within the SSL technology. This certificate is currently self-signed using the IIS Resource Toolkit 6.0 (in contrast to purchasing from third-parties like VeriSign, Thawte, etc). The overall process through which the HMI client browsers are establishing a HTTPS session with the Web-HMI system component is described in the figure 9.5.

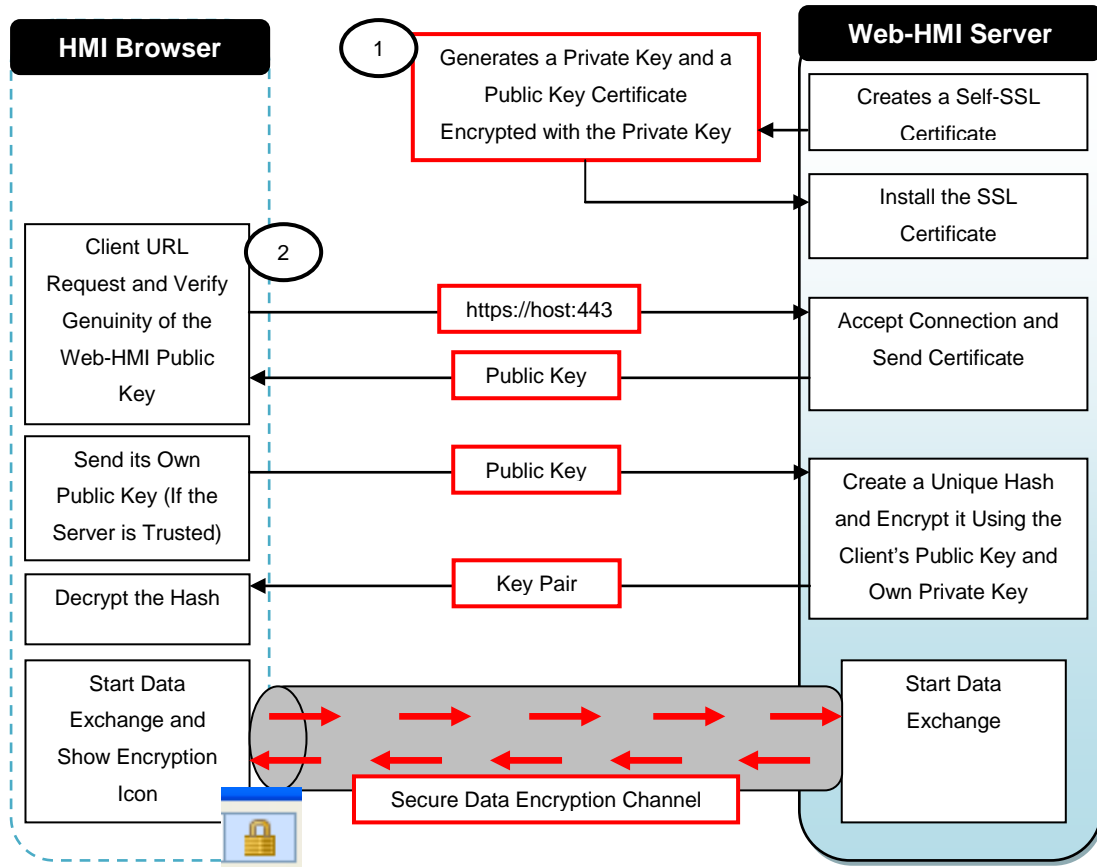


Figure 9-5: HTTPS Data Encryption Session Setup

To evaluate the above implemented data encryption solution, the Wireshark network protocol analyser has been installed that captures packets exchanged between the Web-HMI server system component and a HMI client browser. The TCP/IP packets transmitted between the server and the client contains initial session setup details, machine configurations and any data updates propagation. The aim is to sniff as a third-party onto the network to capture and analyse data travelling over the wire.

While Wireshark is a very powerful tool to analyse TCP streams, the HTTPS encryption mechanism prohibits any third-party to analyse the captured data owing to its ciphering algorithm. This avoids one to eavesdrop onto the network and / or generate any man-in-the-middle attack to the parties involved. As shown in the figure 9.6, the captured data cannot be decrypted by Wireshark packet sniffer, maintaining the integrity of the overall data exchange between Web-HMI system component and various HMI client browsers.



Figure 9-6: Web-HMI Data Encryption Evaluation Using Wireshark Protocol Analyser

9.4 Reliability

The ability of delivering messages from the machine to the operator interface system without any loss is one of the essential features of the architecture that needs evaluation. The underlying communication protocol utilised for transferring data between system components is the TCP/IP. Referring to the seven layers of the OSI model shown in the chapter 4.6.2, TCP operates on its transport layer. The transport layer is responsible for making the end to end data transmission reliable. The services provided by this layer are connection oriented; it means that the data sent by this layer must be acknowledged by the destination. These data acknowledgments are used to ensure that the data is received correctly in the required order by the destination. Owing to this, TCP ensures the reliable transmission of data using a three-way handshake mechanism, retransmission and checksum functionality [209].

The ultimate aim of reliability evaluation in this case is to identify if all machine messages are successfully collected by the Broadcaster system component and transmitted to the Web-HMI and the Marshaller within reasonable amount of time. Since the Marshaller saves all incoming data onto a database for further utilisation, monitoring the database table provides an accurate reflection of the messages being received from the Broadcaster system. By comparing the messages sent to the messages received, one can easily establish the reliability of the system components in sending and receiving data over the TCP/IP socket links.

Approximately 50 test cases have been created where in each case some random machine messages are transmitted to the Broadcaster system component. Table 9.3 shows summarised results from these test cases where machine states, machine errors, machine commands and modes have been generated and data is received by the respective system components. As shown, no message losses have been identified during their transmissions between the system components.

System Components	Message Type			
	CB Config	Machine States	Machine Errors	Machine Commands and Modes
Broadcaster	G=50	R=360, T=360	R=250, T=250	R=670, T=670
Marshaller	R=50	R=360, S=360	R=250, S=250	R=670, S=670
Web-HMI	R=50	R=360	R=250	R=670
<p>These results display only one way communication i.e. from the Broadcaster to other system components. Web-HMI serving only two HMI client browsers.</p> <p>Overall % Message Loss: None.</p> <p>G= Generated, R= Received, T= Transmitted, S= Stored</p>				

Table 9.3: Reliability Evaluation Results

9.5 Robustness

When operating the system distributed over a network, it is essential to evaluate robustness ascertaining the machine information timeliness for an operator. The goal is to always display the current state of the machine on the operator interface screens, requiring a mechanism that monitors the communication liveliness between the Broadcaster, the Marshaller and the Web-HMI. In addition, the fundamental concept of the Web-HMI system is to function properly without crashing regardless of the invalid inputs propagated by the machine.

The essential robustness criteria that need to be evaluated for the system components are:

- Monitoring communication robustness between the Broadcaster and the other system components (i.e. the Web-HMI and the Marshaller) to ensure fail safe operation of the operator interface system within industrial implementation.
- Ensuring operation effectiveness of system components despite abnormalities in the input data.

To satisfy the above criteria within the system architecture, two security strategies have been implemented and evaluated, one for each criteria. The first one is application heartbeat, and the second one is data propagation filtering.

9.5.1 Application Heartbeat

System components must be able to monitor their environment and detect any significant communication changes. Heartbeats have been utilised to detect any TCP/IP socket communication failures between the system components. This is a useful strategy to ensure that if the communication link between the Broadcaster and the other system components has been broken then the machine operator is aware and can quickly track the issue. This will ascertain that the HMI screens are always reflecting the current state of the machine owing to communication liveness between the Broadcaster and the Web-HMI.

Failing to detect abnormalities in the communication link between the system components can lead to disastrous results for example, if a communication link has failed between the Broadcaster and the Web-HMI, the machine operator may assume that the machine components are not changing their states but in reality the communication link between them is down. This is where the application heartbeat strategy implementation comes handy where it monitors TCP/IP socket link between the system components and displays a message to the operator should a link failure is detected.

This application heartbeat implementation is carried out using a timer control found within the dot Net application framework. This timer monitors the socket connection every 300 milliseconds and raises a disconnection event to notify

the system to display an appropriate message on the screens as shown in the figure 9.7. This enables the system components to be in sync with the machine broadcasted messages.

```

Public Sub SourceBDisconnectedHandler(ByVal obj As Object, ByVal e As EventArgs)
    Me.Invoke(New MethodInvoker(Function() SourceBCheckerTimer.Enabled = True))
    Me.Invoke(New MethodInvoker(AddressOf ChangeConnectionDisplayStatus))
End Sub
    
```

A

```

'connection lost... raise an event
If _isSourceBConnected Then
    _isSourceBConnected = False
    RaiseEvent SourceBDisconnected(_sourceBSocket, EventArgs.Empty)
    Settings.AppMainForm.UpdateDetailsTabText("Main thread for the Broadcaster disconnected.")
    Logger.WriteLine("Main thread for the Broadcaster disconnected.", LogType.Debug)
End If

_isSourceBConnected = False
End Try
End Sub
    
```

B

Heartbeat Strategy: Some Code Snippets
 "Raise event A (i.e. enable the timer) when the socket is disconnected (i.e. code B)"

Marshaller System Component

The disconnection is detected using a message display and the monitoring status colour changing to red. When the connection is live, this colour changes to green and an appropriate message is displayed on the data details section.

Marshaller System Component

The disconnection is detected using a message display and the monitoring status colour changing to red. When the connection is live, this colour changes to green and an appropriate message is displayed on the data details section.

Web-HMI System Component

The disconnection is detected using a message display on the top of the HMI client browser screen. This message is critical to a machine operator hence it is highlighted with a red background attracting the attention of the user.

Web-HMI System Component

The disconnection is detected using a message display on the top of the HMI client browser screen. This message is critical to a machine operator hence it is highlighted with a red background attracting the attention of the user.

Figure 9-7: System Components Heartbeat Strategy Evaluation

9.5.2 Data Filtering

Despite abnormalities in the input data, the Broadcaster system component must be able to selectively propagate only desired outputs to other system components. This is achieved through data filtering strategy where only machine states, operational commands and modes, and machine errors are propagated, ignoring any incorrect or corrupted machine messages. In short, any machine message that does not confirm to the CB configuration model is ignored by the Broadcaster system component.

To evaluate this, some corrupt messages are inputted over the network to the Broadcaster's port listening to machine events as shown in the figure 9.8. These messages are successfully captured by the Broadcaster system component, filtered as corrupt messages and stored in an external log file for further investigation. Furthermore, they are not propagated to the Marshaller and the Web-HMI system. This strategy provides the required level of robustness (in addition to safety and security) against unnecessary data propagation over the network.

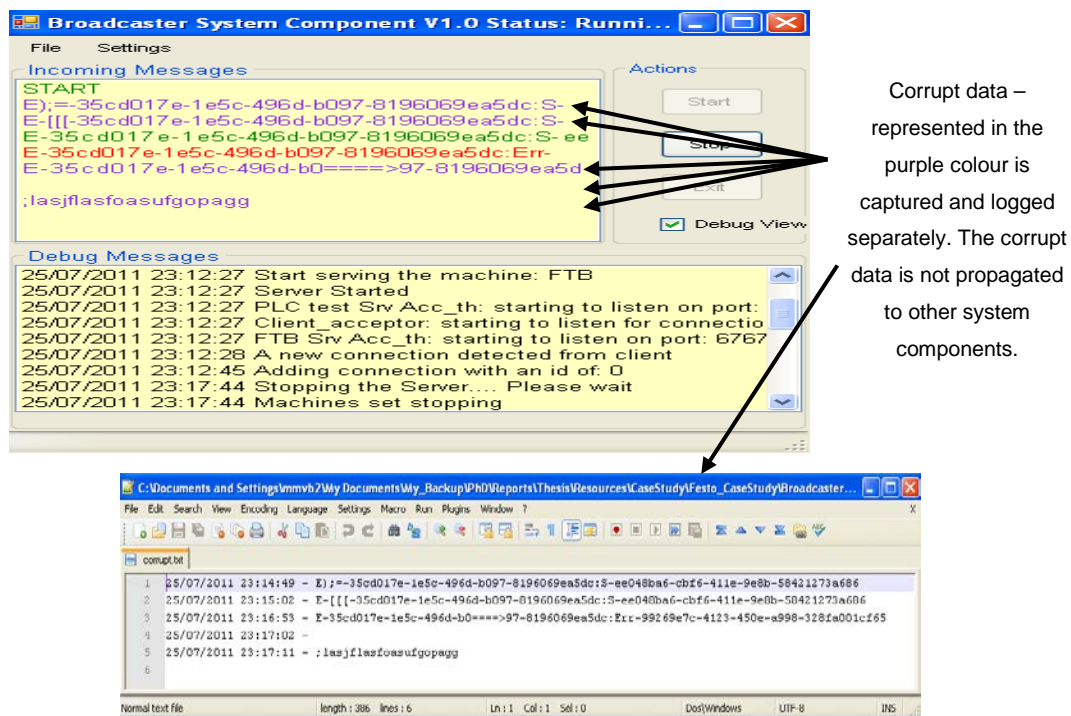


Figure 9-8: Broadcaster's Data Filtering Strategy Evaluation

9.6 Performance

The system architecture must have desired performance expectations for its industrial application. The strategy to satisfy higher performance requirements (i.e. soft real-time processing) within the system components is implemented through multithreading techniques. Multithreading improves the effectiveness of a system through task-load distribution such that the processing load within an application can be shared using number of threads [205].

Two main technical aspects that need to be considered to satisfy any system's performance expectations are its system throughput and response times. The response times can be affected by the system's processing time (i.e. the throughput) therefore its throughput is evaluated first prior to the overall response time as described in the next sections.

9.6.1 System Throughput

The main distribution mechanism within the system architecture is the Broadcaster component therefore its system throughput is evaluated. In this context, the system throughput corresponds to identifying the amount of time it takes the Broadcaster to process and propagate any given machine message. This is achieved through applying timestamps to each message as they arrive through the Broadcaster's blackboard and leave.

As shown in the chapter 6.2.2, each message passes through various processing sections of the Broadcaster (i.e. its panels). To be precise, each message passes through at least 3 of such panels. When the message arrives at the panel 1 (i.e. at the IN-LOAD), its arrival time is taken (i.e. T_{in}) and when it is propagated to any client (at the OUT-LOAD), its delivery time is taken (i.e. T_{out}). The difference between the T_{out} and T_{in} is taken as its processing time as shown in the figure 9.9. A total of 50 test cases have been created, out of which one such test case is shown in the table 9.4. As shown, on average the individual message processing time (M_{tx}) is approximately 74 milliseconds.

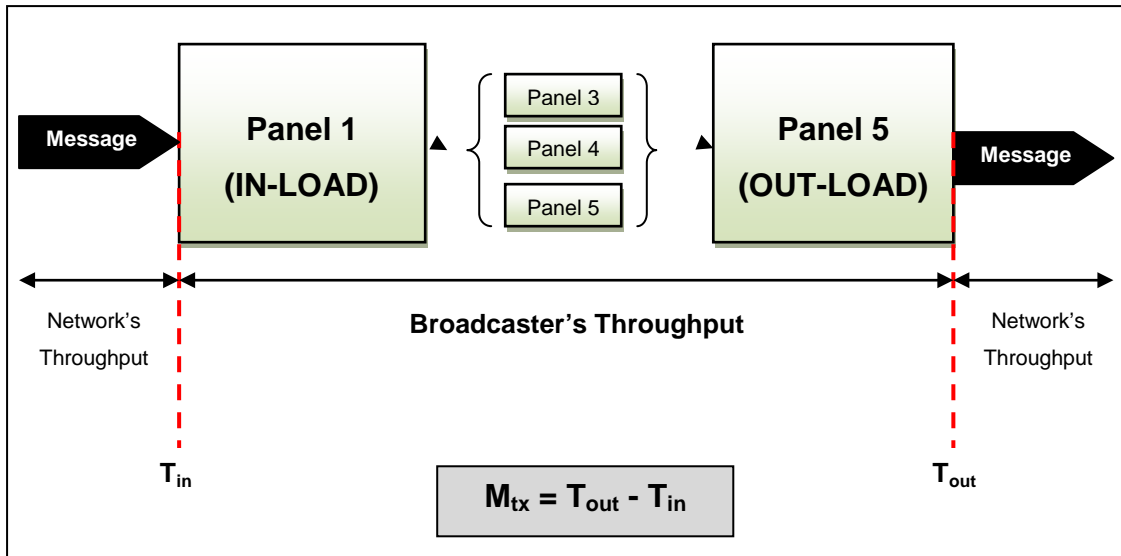


Figure 9-9: Broadcaster's System Throughput Strategy Evaluation

E-Load (Number of Messages)	T_{in} (On Entry Interface - Accumulated Time in ms)	T_{out} (On Exit Interface - Accumulated Time in ms)	$T_x (T_{out} - T_{in})$	$M_{tx} (T_x / E\text{-Load})$
500	42250	78315	36065	72.13
1000	84500	159854	75354	75.35
1500	126750	237855	111105	74.07
2000	169000	315819	146819	73.41
2500	211250	399732	188482	75.39
3000	253500	467520	214020	71.34
3500	295750	552855	257105	73.46
4000	338000	638276	300276	75.07
4500	380250	711750	331500	73.67
5000	422500	785910	363410	72.68
Average M_{tx}				73.66 \approx 74 ms

Table 9.4: Broadcaster's System Throughput Evaluation Results

9.6.2 Response Times

Two different types of response times need to be evaluated namely; HMI to machine response time and machine to HMI response time. HMI to machine response time corresponds to the total time taken from when an operator generates a request by pressing a button on the HMI client browser screen to when this command is received by a machine. Likewise when a machine changes its state, the total amount of time taken for the message to be transmitted and displayed at the HMI client browser screen is the machine to HMI response time.

It is expected that any machine state information must be propagated considerably faster than the machine takes to move. Within the powertrain assembly applications, the worst case response time must be 500 ms [80, 174]. Taking this value as a performance comparison benchmark, an evaluation study has been carried out where messages are transmitted from the HMI client to the machine (and vice versa), time stamped at their respective places as shown as in the figure 9.10. To avoid any network latency issues, all the system components have been implemented in the same PC, communicating using the loopback IP address (i.e. 127.0.0.1).

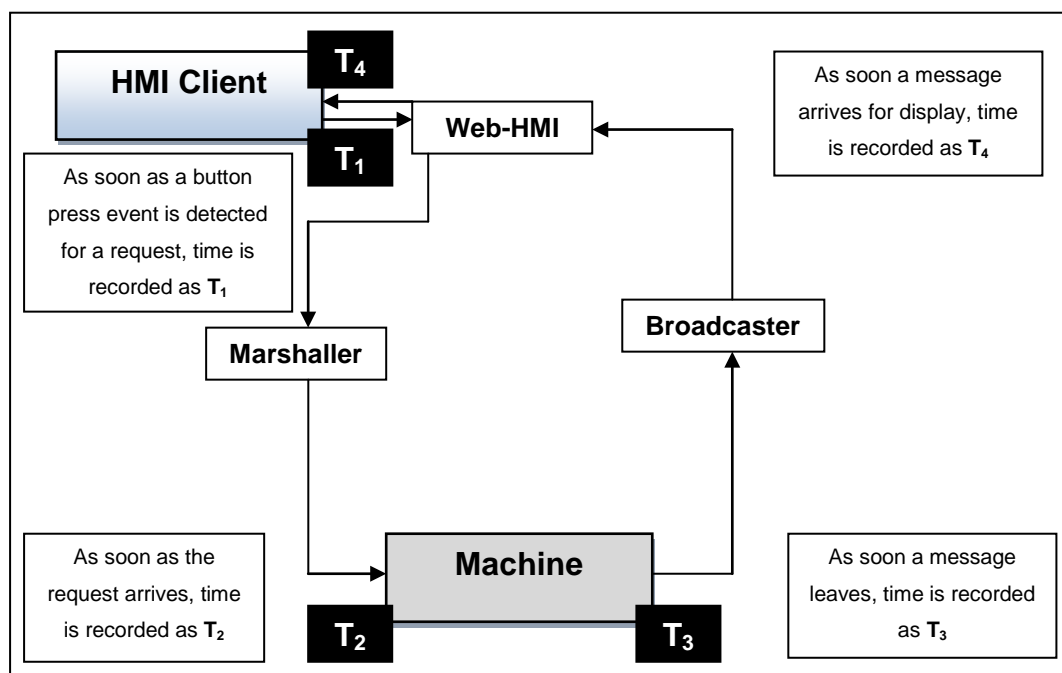


Figure 9-10: Response Time Strategy Evaluation

Throughout the evaluation study, a number of messages are exchanged between the system components and some results are illustrated in the table 9.5. To avoid any unnecessary communication overheads associated with establishing socket connections, respective TCP/IP links are left open throughout these tests. The response times for the HMI to machine is 225 ms where as the response times for the machine to HMI is 297ms, including the 100ms Profinet network performance. The HMI to the machine response time is faster comparatively owing to the fact that the Marshaller does not carry out cumbersome data processing besides just forwarding the data where as the Broadcaster does substantial processing on the received data from a machine.

Record Number	HMI to Machine Time (ms) (Only last milliseconds shown)			Machine to HMI Time (ms) (Only last milliseconds shown)		
	T ₁	T ₂	T _i (T ₂ – T ₁)	T ₃	T ₄	T _j (T ₄ – T ₃)
1	.120	.255	.135	.29	.222	.193
2	.12	.139	.127	.168	.367	.199
3	.23	.147	.124	.275	.473	.198
4	.36	.161	.125	.781	.975	.194
5	.280	.401	.121	.75	.274	.199
6	.163	.286	.123	.513	.708	.195
7	.87	.206	.119	.37	.239	.202
8	.55	.181	.126	.02	.198	.196
9	.392	.514	.122	.664	.859	.195
10	.474	.598	.124	.199	.396	.197
Average:	T _i		124.6 ≈ 125 + 100ms (Profinet) = 225	T _j		196.8 ≈ 197 + 100ms (Profinet) = 297
Worst case performance of any industrial Ethernet can be added to the overall response time, for example Profinet is 100ms.						

Table 9.5: Response Times Evaluation Results

9.7 Scalability

The Broadcaster, being the central hub supporting machine data distribution, needs to efficiently handle the continuous flow of received messages and also ensure that it can accommodate new load within reasonable amount of time. This requires one to measure its scalability to support any growth projections in when adding more manufacturing machines. To realise the practical applicability of the system components to plant architecture mapping strategy described in the chapter 8.4.2, the scalability property of the Broadcaster system component is evaluated in this section. Figure 9.11 shows the overall process used to evaluate scalability of the Broadcaster system component.

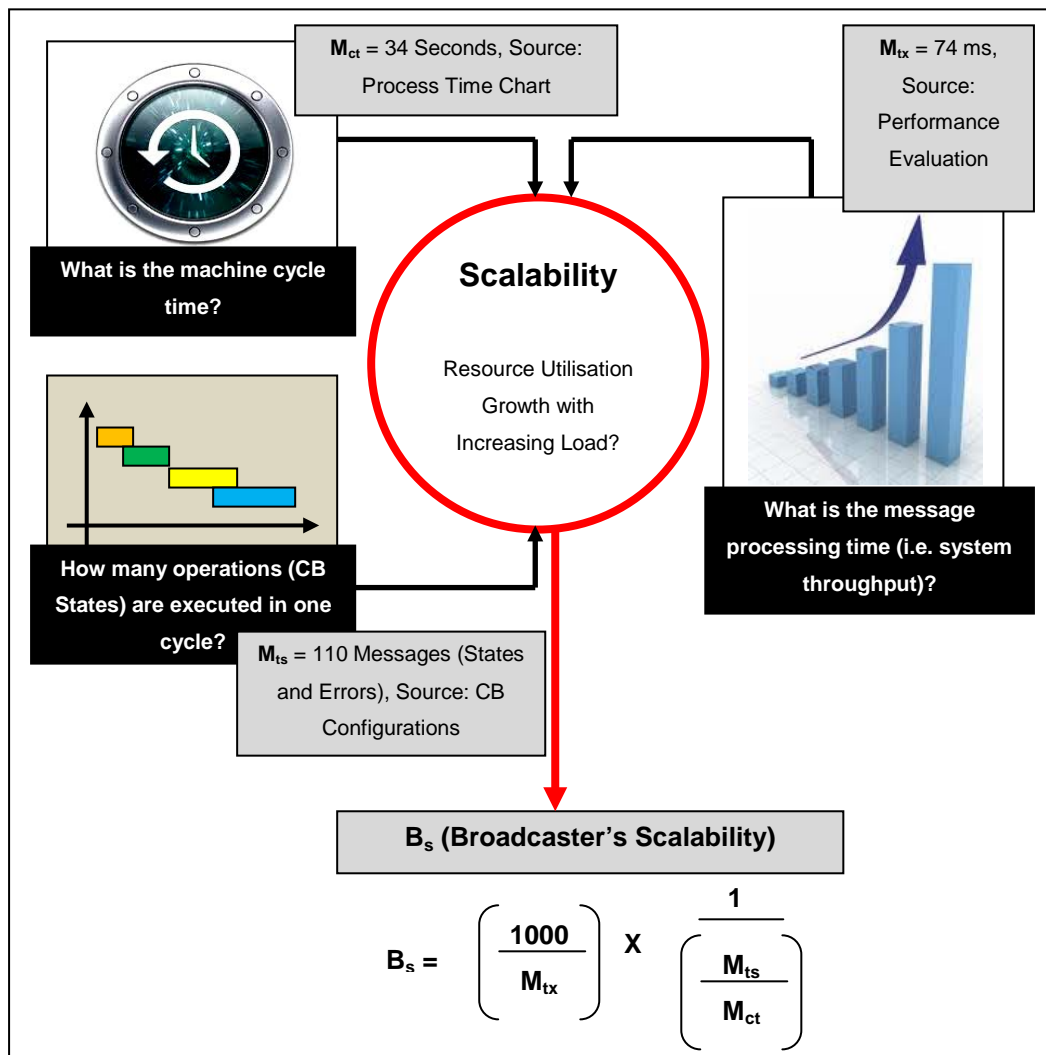


Figure 9-11: Broadcaster's Scalability Evaluation Process

With reference to the Ford-Festo test rig results shown in the above figure 9.11, the corresponding Broadcaster system component's scalability can be demonstrated using the table 9.6. From these results, it can be identified that the number of test rig machines that can be supported by the Broadcaster system component are 4. Based on this result, the author has enough information to roughly evaluate the system components to plant architecture mapping for the Fox line as described next.

Required Parameters	Corresponding Values
The cycle time of the test rig machine (M_{ts})	34 seconds
Number of states executed when operating the machine (M_{ct})	110 messages
Number of states executed per second	$(M_{ts} / M_{ct}) 110 / 34 = 3.24$ states per second
Broadcaster's system throughput per message per millisecond (M_{tx})	74 milliseconds
Broadcaster's system throughput per message per second	$(1000 / M_{tx}) 1000 / 74 = 13.51$ messages per second
Scalability Calculation (B_s):	<p>1 machine operation = 3.24 messages per second B_s machines' operation = 13.51 messages per second</p> <p style="text-align: center;">Therefore $B_s = \frac{13.51 \times 1}{3.24} = 4$ machines</p>

Table 9.6: Scalability Evaluation Results for Ford-Festo Test Rig

9.7.1 Proposed Plant Architecture Mapping Strategy Evaluation

With reference to the Ford's Fox plant architecture description (see chapter 8.4), the breakdown of the machine stations per zone, and the corresponding system components required (based on the Ford-Festo test rig's scalability calculation carried out previously) is shown in the table 9.7. As illustrated in this table, since the manual stations amount to at least 8 per zone, 2 Broadcaster system components are needed. It has to be noted that the scalability evaluation is entirely governed by inputs from the total number of machine messages generated per cycle and the corresponding cycle time. The next chapter concludes this thesis and identifies critical areas for future work needed to completely apply this research in the powertrain manufacturing automation.

Station Type	Total Stations for 10 Zones	Total Stations per Zone	Proposed Implementation Strategy	Resource Allocation
Automatic	12	1	System Components Located per Station	1 x Broadcaster 1 x Marshaller 1 x Web-HMI
Semi-Auto	20	2	System Components Located per Zone	1 x Broadcaster 1 x Marshaller 1 x Web-HMI
Manual	80	8	System Components Located per Zone	2 x Broadcaster (4 Machines per System) 1 x Marshaller 1 x Web-HMI

Table 9.7: Plant Architecture Mapping Strategy Evaluation Results

Chapter 10 : Discussion, Conclusion and Future Work

Chapter Contribution to this Thesis:

This chapter identifies the contribution to knowledge, discusses the extent to which the research requirements have been fulfilled and highlights future work to take the research forward.

10.1 Research Discussion and Conclusion

10.1.1 Contribution to Knowledge

This research has proposed a novel idea of implementing and utilising an operator interface solution for the lifecycle support of CB automation systems. The contribution to knowledge is summarised as follows:

- A detailed understanding of the emerging requirements that must be met, and current limitations that must be resolved, in order to realise next-generation operator interface systems for the lifecycle support of powertrain manufacturing systems.
- Evolution of a novel system components architecture that complements the CB approach through sharing a common machine model with supply-chain partners, establishing new ways of locally and remotely deploying and using operator interfaces throughout the lifecycle.
- Blackboard-based design models of system components describing their data sharing process with clarity, and their simplified but detailed development representation, allowing software programmers to implement fully operational HMI systems.
- Qualitative and quantitative evaluation of the results against the stated research requirements to identify the industrial readiness of the proposed solution.
- A description of how the research approach and various developed concepts aim to fulfil lifecycle requirements of CB automation systems.

- Identification of the visionary impacts of the CB operator interface solution to the powertrain manufacturing systems.

10.1.2 Fulfilling the Industrial Requirements

Industrial case studies (described in the chapter 8) and system components evaluation (described in the chapter 9) have demonstrated the feasibility of adopting operator interface system implementation approach proposed within this research, fulfilling the lifecycle usage requirements of CB automation systems summarised in the chapter 2.4. The research output fulfils the following research requirements:

Towards a Vendor-Independent (“Open”) Support Platform

The system components architecture operates using standard TCP/IP link, publishes data in a uniform XML format that can be decoded by any supply-chain partner’s engineering tool, and supports “plug-and-play” connectivity (as demonstrated in the chapter 8.2.5 using the SAP xMII application). Furthermore, the operator interface system is based on the open web technology and implemented on standard touch-screen panels (covered in the chapter 7.5), enabling the same operator interface system to be utilised with a variety of control devices such as FTB and PLC (as demonstrated in the chapter 8.2 and identified in the section 10.2.3 respectively).

Support for Reconfigurability and Reuse Requirements

Operator interface system solution complements the CB approach through implementing generic template-based screens (described in the chapter 7.5.3) that are automatically populated using machine configurations shared through the system architecture (using the Broadcaster) at runtime. This enables the same set of screens to be deployed across powertrain machine programme regardless of the underlying configurations. To demonstrate this requirement satisfaction, chapter 8.2.4 successfully illustrated a real-life process workflow reconfiguration and reuse process using the Ford-Festo rig.

Validation of Machine Logic in Virtual Environment

A 3D virtual simulation model (associated with the machine control logic, engineered using the CB tools as highlighted in the chapter 3.2.2) is integrated within the operator interface system, and the system architecture supports real, simulated and hybrid machine operation. This enables visualisation of the machine behaviour to resolve any issues prior to its build process as demonstrated using the Op 1900 model in the chapter 8.3.4.

Provision of Remote Control, Monitoring and Maintenance Support

Web-based operator interface solution incorporates 3D machine visualisation implemented within the system architecture providing the required level of remote connectivity services that support control, monitoring and maintenance activities regardless of machine's geographical location, as studied in the chapter 8.2.7. The non-functional aspects associated with remote communication (for example, safety and security of data transmission) have been evaluated in the chapter 9.

Enabling Early HMI Verification and Operator Training

Integration of the 3D machine model to the operator interface system prior to the machine build enables the HMI to be verified using the same CB configurations (shared using the system components architecture) that are going to be downloaded as the real control logic. This provides a fully-operational HMI even at the virtual commissioning stage resulting in early system verification and enabling operator training as illustrated in the chapter 8.3.5 and chapter 8.3.6 respectively.

Adhering to the Industrial Best Practices

The operator interface system conforms to industrial standards through adopting the required design guidelines such as screen layouts and navigation (as described in the chapter 5.2), and evaluating the non-functional but

essential quantitative properties of the system such as safety, security, reliability, robustness, performance and scalability as illustrated in the chapter 9.

10.1.3 Benefits to the Powertrain Manufacturing Lifecycle

In a bigger picture, this research work is anticipated to provide the following benefits to powertrain manufacturing lifecycle:

- Improvisation of the production machine design and its development process. This research solution is going to provide a mechanism for building and designing machines in new ways through the use of the CB automation approach provided by the MSI research group at Loughborough University. Adopting this practice in industries should significantly reduce lifecycle costs. For example, early validation of the machine and its operator interface system (as illustrated in the chapter 8.3.4) should significantly reduce number of issues which may arise later during the actual machine build phase, reducing the overall time of machine implementation and commissioning.
- It provides new ways of supporting manufacturing machines. Incorporating remote control and monitoring support of the key machine lifecycle phases is a promising approach towards solving the problems faced by globally distributed manufacturing activities in today's industrial era (as described in the chapter 2). Moreover, a substantial amount of revenue can be saved in terms of low machine MTTR and better customer services can be offered to the end users. For example, by remotely monitoring the machine status using the operator interface system (as illustrated in the chapter 8.2.7), it can enable maintenance engineers to identify and quickly address the actual cause of the problem at the shop-floor level, significantly reducing machine downtime.
- Operator interface system screens are rapidly but consistently auto-generated, supporting all the current industry best practice such as optimised screen layouts, navigational structure and operational icons.

This is created from instances of reusable machine interface templates stored within a system repository which when executed, transform the machine engineering and runtime data into a complete operator interface system. This leads to early HMI deployment and any machine change is dynamically reflected on the operator interface screens without further programming efforts. This also provides early operator training opportunities which ultimately lead to faster machine ramp up as described in the chapter 8.3.6.

- Stronger relationships between involved supply-chain partners can be established enabling different classes of people to efficiently interact with manufacturing machines. This is provided through an open system implementation architecture where third-party engineering resources can be easily “plugged-in” to control and monitor production machines operation, and the same operator interface system can be utilised by various stakeholders through the lifecycle.

10.2 Future Work Recommendations

Due to the finite time and resource constraints on the research, and the wide scope of this investigation, not all the research paths have been fully explored. Some potential extensions to the research work are summarised in the figure 10.1.

10.2.1 Remote Data Transmission

As identified in the chapter 4.4, large amounts of machine data may be required in their raw (i.e. unprocessed xml) form at remote locations to drive application resources such as engineering tools. A web services-based system called RemoteComm has been proposed in the figure 10.2. In this approach, datasets (used as parameters) can be marshalled over the HTTP(s) port using a web service method call from remote engineering tools. The datasets can be

serialised as XML strings over the web, and be de-serialised at the remote end within any application. The remote tools can refresh the dataset calls at specific intervals as required to obtain the current state of a machine. Dataset serialisation is the most widely accepted standard for data transmission over the web owing to the coding and data packaging flexibility, data persistence and any data source updates. This approach doesn't require any overheads such as subscriptions and license arrangements with a third-party service provider such as Citrix, WebEx, etc. The RemoteComm system is still under a conceptual design stage and is going to be addressed in the future as one potential research possibility.

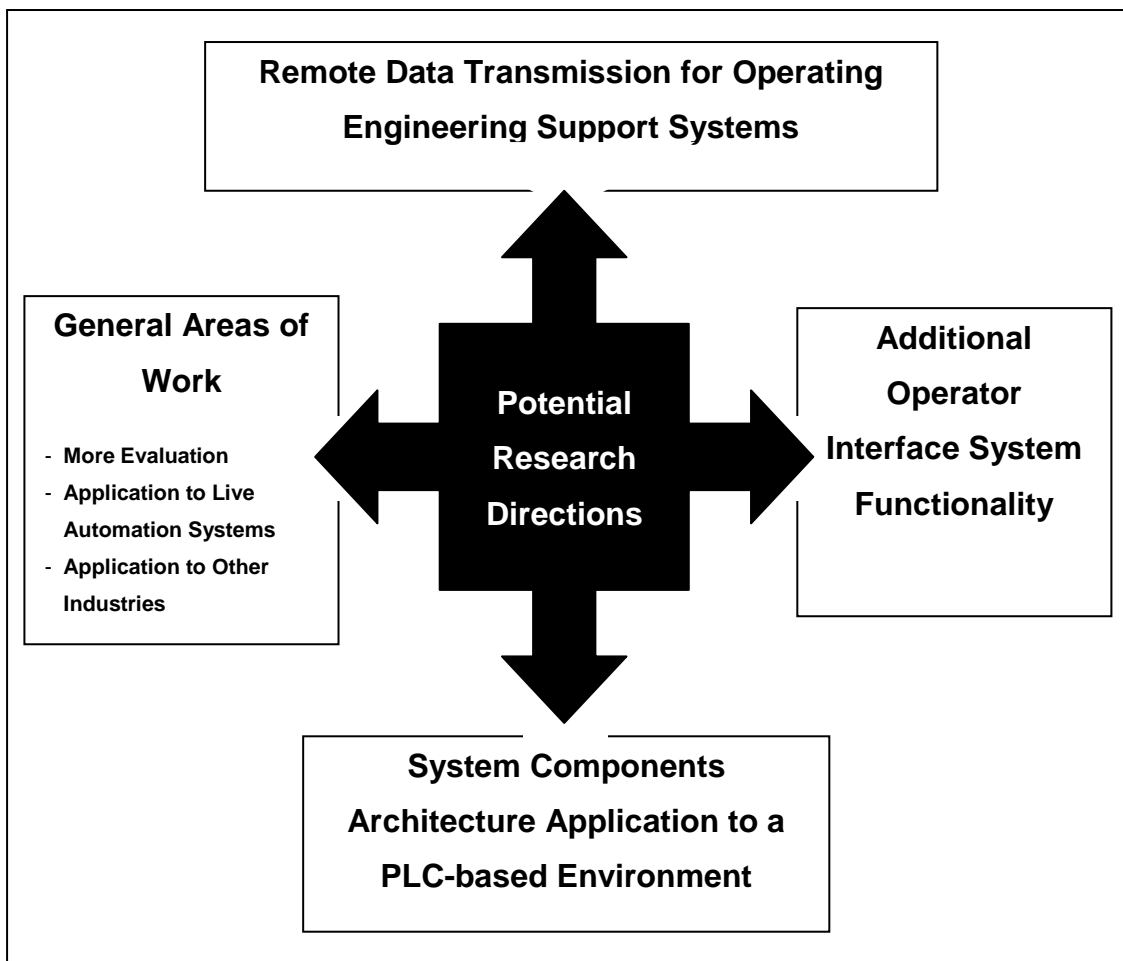


Figure 10-1: Potential Research Directions

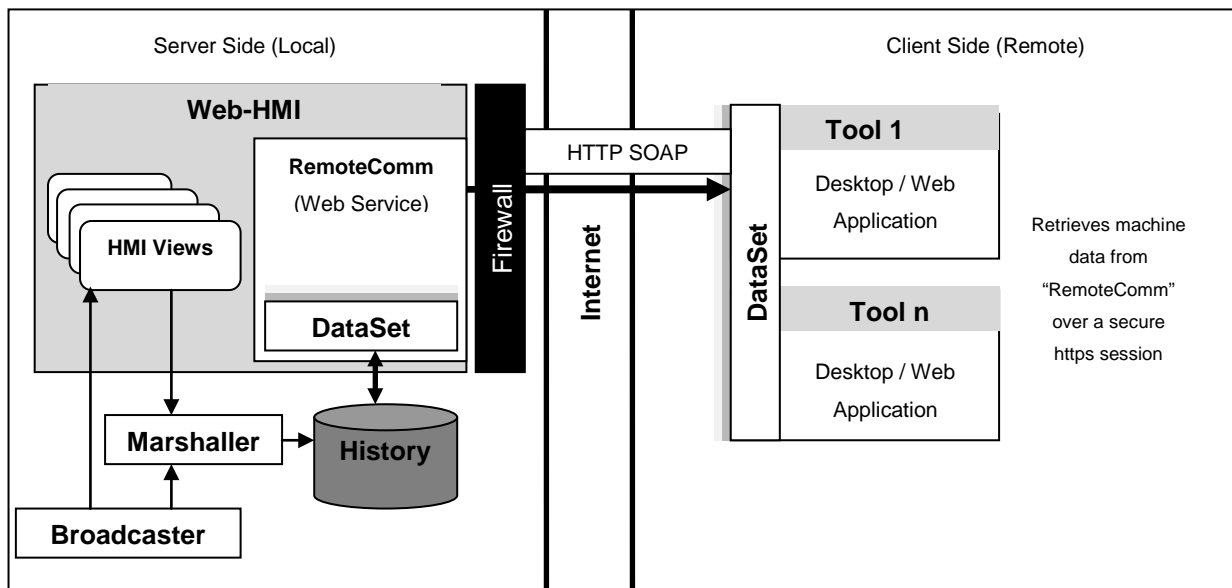


Figure 10-2: RemoteComm Conceptual Illustration

10.2.2 Operator Interface Functionality

Although the operator interface system has been designed and implemented to show the research solution in operation, some additional functionality can be accommodated in the list of future research and development opportunities.

- An operator interface system configuration tool needs to be developed which may enable rapid generation of additional screens based on user's requirements. This will provide a means of supporting various industrial domains (as identified in the section 10.2.5) and end user's screen layout preferences.
- Multilingual support is needed as currently the primary language of display on the operator interface screens is English, however operation of these interfaces in other languages like Chinese, Indian, German, Korean, Japanese etc, would be beneficial where users with different language preferences can access the system in their own language to universally make the system more acceptable.

- A number of other additional screens have to be incorporated such as RFID support screen, network monitoring screen, machine station selection screen, role profile screen and interlock monitoring screens.

10.2.3 Application to a PLC-based Control System

To successfully demonstrate the migration path of the research solution to current production machine, an OPC communication gateway needs to be implemented [76] within the system architecture. This is the de-facto standard used for communication between the PLC and the operator interface system. While OPC COM-based specifications have been widely used by the industry, due to newer technological opportunities presented especially by the cross-platform capabilities of web services and the SOA, OPC UA (Unified Architecture) is going to be adapted by major PLC vendors [210]. This provides maximum interoperability, security and standardisation across all the levels of the manufacturing operation hierarchy described in the chapter 2.3.1.

10.2.4 General Work Areas

A number of other general work areas relating to further research can be suggested as follows:

- More thorough evaluation of the system components needs to be undertaken. Evaluating the approach with real industry engineers rather than academic researchers mimicking the role of actual engineers would give invaluable feedback and information from their perspective on how the system can be utilised on day-to-day basis.
- Although the system components architecture has been successfully evaluated on a prototype machine used for proof-of-concept system, they are not yet evaluated against actual machines at the shop-floor. Application of the research approach to real life production machine

would generate new knowledge and complete the migration path to a next-generation operator interface system solution.

- The system components implementation has only been tested in the powertrain automation domain. Application of the approach to other industrial sectors would be beneficial in assisting its validity and its usefulness as identified next.

10.2.5 Application to Other Industries

From the case studies described in the chapter 9, this research has proven that the web-based operator interface system solution can be deployed and utilised for the lifecycle support of the CB automation system within the powertrain manufacturing industry. While the focus of this work has been on the automotive industry, other sectors may however benefit most from aspects of the functionalities and the system components architecture implementation offered. For example, frequent reconfigurability is considered to be a critical requirement in the powertrain sector, but if this research was applied to a petrochemical sector, remote monitoring and maintenance may have been more of a critical requirement.

Research into operator interface system's core requirements in a number of different industries for example packaging, semiconductor manufacturing, petrochemical, electronics manufacturing and textile manufacturing would grant an opportunity to comparatively analyse the suitability of the operator interface system solution across these other industrial sectors. Implementing this research approach to different industries would require further engineering activities to some aspects of the system components while other aspects would remain generic as shown in the figure 10.3. For example, the Web-HMI server component of the architecture would remain generic across all industrial sectors in addition to the Broadcaster and the Marshaller system components functionality that are core to this research approach. The template screens (describing the operator interface layouts based on the representation layout as illustrated in the chapter 7.5.3) would be particular to the individual stakeholders

in each industry. This highlights the need for an HMI configuration tool (as identified in the section 10.2.2) to rapidly support various screen layouts for different industrial sector's requirements.

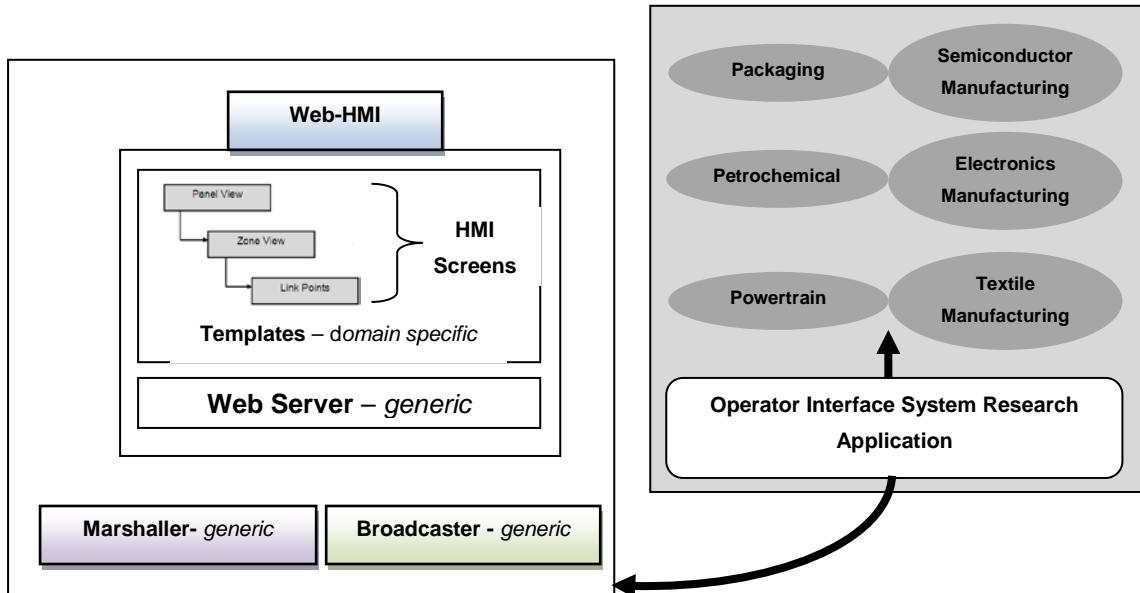


Figure 10-3: Research Application to Other Industries

References

- [1]. Harrison, R., A.A. West, and L.J. Lee. *Lifecycle Engineering of Future Automation Systems in the Automotive Powertrain Sector*. in *IEEE International Conference on Industrial Informatics 2006*.
- [2]. Sutherland, J., K. Gunter, D. Allen, D. Bauer, et al., *A global perspective on the environmental challenges facing the automotive industry: state-of-the-art and directions for the future*. *International Journal of Vehicle Design*, 2004. **34**(2): p. 86-110.
- [3]. Morel, G., H. Panetto, M. Zaremba, and F. Mayer, *Manufacturing enterprise control and management system engineering: paradigms and open issues*. *Annual reviews in control*, 2003. **27**(2): p. 199-209.
- [4]. Molina, A., C.A. Rodriguez, H. Ahuett, J.A. Cortes, et al., *Next-generation manufacturing systems: key research issues in developing and integrating reconfigurable and intelligent machines*. *International Journal of Computer Integrated Manufacturing*, 2005. **18**(7): p. 525-536.
- [5]. Department-of-Trade-and-Industry. *Environmental Impacts of Motor Manufacturing and Disposal of End of Life Vehicles*. [cited 05 Jan 2011]; Available from: <http://www.autoindustry.co.uk/docs/74289.pdf>.
- [6]. Ning, L., *Economic liberalisation for high-tech industry development? Lessons from China's response in developing the ICT manufacturing sector compared with the strategies of Korea and Taiwan*. *Journal of Development Studies*, 2007. **43**(3): p. 562-587.
- [7]. BüyüKözkan, G., T. Derel, and A. Baykaso lu, *A survey on the methods and tools of concurrent new product development and agile manufacturing*. *Journal of Intelligent Manufacturing*, 2004. **15**(6): p. 731-751.
- [8]. Ong, M.H., A.A. West, S.M. Lee, and R. Harrison, *The opportunities for multimedia supported remote maintenance provided by an implementation of a component-based system in the automotive domain*. *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, 2007. **221**(5): p. 787-798.
- [9]. Ucar, M. and R.G. Qiu, *E-maintenance in support of e-automated manufacturing systems*. *Journal of the Chinese Institute of Industrial Engineers*, 2005. **22**(1): p. 1-10.

References

- [10]. SMART. *Monitoring and control: today's market, its evolution till 2020 and the impact of ICT on these*, European Commission DG Information Society & Media. 2007 [cited 2010 15th September]; Available from: http://www.decision.eu/smart/SMART_9Oct_v2.pdf.
- [11]. Haq, I., R. Monfared, R. Harrison, L. Lee, et al., *A new vision for the automation systems engineering for automotive powertrain assembly*. International Journal of Computer Integrated Manufacturing, 2010. **23**(4): p. 308-324.
- [12]. Harrison, R., A.A. West, R.H. Weston, and R.P. Monfared, *Distributed engineering of manufacturing machines*. Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture, 2001. **215**(2): p. 217-231.
- [13]. Bi, Z.M., S.Y.T. Lang, W. Shen, and L. Wang, *Reconfigurable manufacturing systems: the state of the art*. International Journal of Production Research, 2008. **46**(4): p. 967-992.
- [14]. Boothroyd, G., *Product design for manufacture and assembly*. Computer-Aided Design, 1994. **26**(7): p. 505-520.
- [15]. Bussmann, S., N. Jennings, N.R. Jennings, and M.J. Wooldridge, *Multiagent systems for manufacturing control: a design methodology*. 2004: Springer-Verlag New York Inc.
- [16]. Wikipedia. *User Interface*. http://en.wikipedia.org/wiki/User_interface [cited 05 Jan 2011].
- [17]. Anwar, M.R., O. Anwar, S.F. Shamim, and A.A. Zahid. *Human Machine Interface Using OPC (OLE for Process Control)*. in *Student Conference On Engineering, Sciences and Technology*. 2004.
- [18]. Yao, A.W.L., *Design and implementation of Web-based diagnosis and management system for an FMS*. The International Journal of Advanced Manufacturing Technology, 2005. **26**(11): p. 1379-1387.
- [19]. Da'na, S., A. Sagahyroon, A. Elrayes, A.R. Al-Ali, et al., *Development of a monitoring and control platform for PLC-based applications*. Computer Standards & Interfaces, 2008. **30**(3): p. 157-166.

References

- [20]. Zipkin, P., *The limits of mass customization*. Harvard Business Review, 1997. **75**: p. 91-101.
- [21]. Alptekinoglu, A. and C.J. Corbett, *Mass Customization versus Mass Production: Variety and Price Competition*. 2005.
- [22]. Fisher, M.L. and C.D. Ittner, *The impact of product variety on automobile assembly operations: Empirical evidence and simulation analysis*. Management Science, 1999. **45**(6): p. 771-786.
- [23]. Anderson, D. *THE END OF LINE FOR MASS PRODUCTION : No Time for Batches & Queues*. [cited 05 Jan 2011]; Available from: <http://www.build-to-order-consulting.com/Mass%20Production.htm>.
- [24]. Deanab, P.R., D. Xueb, and Y.L. Tub, *Prediction of manufacturing resource requirements from customer demands in mass-customisation production*. International Journal of Production Research, 2009. **47**(5): p. 1245-1268.
- [25]. Da Silveira, G., D. Borenstein, and F.S. Fogliatto, *Mass customization: Literature review and research directions*. International Journal of Production Economics, 2001. **72**(1): p. 1-13.
- [26]. Maskell, B., *The age of agile manufacturing*. An International Journal of Supply Chain Management:, 2001. **6**(1): p. 5-11.
- [27]. SOCRADES. *SOCRADES ROADMAP The Future of SOA-based Factory Automation*. [cited 15th April 2012]; Available from: <http://www.socrades.eu/Documents/objects/file1274836528.84>.
- [28]. Neelamkavil, J., W. Shen, Q. Hao, and H. Xie, *Making Manufacturing Changes Less Disruptive: Agent-Driven Integration*. Information Technology For Balanced Manufacturing Systems, 2006: p. 271-280.
- [29]. Harrison, R., R.P. Monfared, and L. Lee. *Business driven engineering for powertrain industry*. in *IEEE Conference on Emerging Technologies and Factory Automation*. 2009.
- [30]. Gunasekaran, A., *Agile manufacturing: enablers and an implementation framework*. International Journal of Production Research, 1998. **36**(5): p. 1223-1247.

References

- [31]. Phaithoonbuathong, P., *Web service Control of Component-Based Agile Manufacturing Systems*, in *PhD Thesis*. 2009, Loughborough University.
- [32]. Sturgeon, T.J. and R. Florida, *Globalization, deverticalization, and employment in the motor vehicle industry*. Locating global advantage: Industry dynamics in the international economy, 2004: p. 52–81.
- [33]. Spatz, J. and P. Nunnenkamp, *Globalization of the automobile industry: traditional locations under pressure?* *The Swiss Review of International Economic Relations*, 2002. **57**(4).
- [34]. KPMG. *Globalization and manufacturing*. [cited 05th Jan 2011]; Available from: http://www.kpmg.co.uk/pubs/Global_Manu_Survey.pdf.
- [35]. Hao, Q., W. Shen, and L. Wang, *Towards a cooperative distributed manufacturing management framework*. *Journal of Computers in Industry*, 2005. **56**(1): p. 71-84.
- [36]. Ceglarek, D., W. Huang, S. Zhou, Y. Ding, et al., *Time-based competition in multistage manufacturing: stream-of-variation analysis (SOVA) methodology—review*. *International Journal of Flexible Manufacturing Systems*, 2004. **16**(1): p. 11-44.
- [37]. Koren, Y., U. Heisel, F. Jovane, T. Moriwaki, et al., *Reconfigurable Manufacturing Systems*. *Annals of the CIRP*, 1999. **48**(2): p. 527-540.
- [38]. Srivastava, S.K., *Green supply-chain management: A state-of-the-art literature review*. *International Journal of Management Reviews*, 2007. **9**(1): p. 53-80.
- [39]. Sullivan, J.L., R.L. Williams, S. Yester, E. Cobas-Flores, et al. *Life cycle inventory of a generic US family sedan overview of results USCAR AMP project*. 1998: SOC AUTOMATIVE ENGINEERS INC.
- [40]. Frosch, R.A., D.C. Bonner, J.B. Carberry, L. Carothers, et al., *Industrial Environmental Performance Metrics-Challenges and Opportunities*. Washington: National Academy of Science, 1999.
- [41]. Van Tan, V., D.S. Yoo, and M.J. Yi, *Efficient Web Service Based Data Exchange for Control and Monitoring Systems*. *International Journal of Information Technology*, 2008. **14**(1).

References

- [42]. Jammes, F. and H. Smit, *Service-oriented paradigms in industrial automation*. IEEE Transactions on Industrial Informatics, 2005. **1**(1): p. 62-70.
- [43]. Phelps, J. and B. Busby, *Service-Oriented Architecture-What Is It, and How Do We Get One?* Educause Quarterly, 2007. **30**(3): p. 56.
- [44]. Subrahmanian, E., S. Rachuri, S.J. Fenves, and S. Foufou, *Product lifecycle management support: a challenge in supporting product design and manufacturing in a networked economy*. International Journal of Product Lifecycle Management, 2005. **1**(1): p. 4-25.
- [45]. Microsoft. *Microsoft .NET*. [cited 05th Jan 2011]; Available from: <http://www.microsoft.com/net/>.
- [46]. Toncich, D., *Data Communications and Networking for Manufacturing Industries*. 2nd ed. 1994: Chrystobel Engineering.
- [47]. Sahin, C. and E.D. Bolat, *Development of remote control and monitoring of web-based distributed OPC system*. Computer Standards & Interfaces, 2009. **31**(5): p. 984-993.
- [48]. Yusuf, Y.Y., M. Sarhadi, and A. Gunasekaran, *Agile manufacturing: The drivers, concepts and attributes*. International Journal of Production Economics, 1999. **62**(1-2): p. 33-43.
- [49]. Cho, H., M. Jung, and M. Kim, *Enabling technologies of agile manufacturing and its related activities in Korea*. Journal of Computers & Industrial Engineering, 1996. **30**(3): p. 323-334.
- [50]. Elkins, D.A., N. Huang, and J.M. Alden, *Agile manufacturing systems in the automotive industry*. International Journal of Production Economics, 2004. **91**(3): p. 201-214.
- [51]. Harrison, R. and A.W. Colombo. *Collaborative automation from rigid coupling towards dynamic reconfigurable production systems*. in *16th IFAC World Congress, Prague, Czech Republic*. 2005.
- [52]. Gunasekaran, A. and E.W.T. Ngai, *Build-to-order supply chain management: a literature review and framework for development*. Journal of Operations Management, 2005. **23**(5): p. 423-451.

References

- [53]. Ishii, K., *Life-cycle engineering design*. ASME JOURNAL OF MECHANICAL DESIGN, 1995. **117**: p. 42-42.
- [54]. Harrison, R., S.M. Lee, and A.A. West. *Lifecycle engineering of modular automated machines*. in *2nd IEEE International Conference on Industrial Informatics*. 2004.
- [55]. de Souza, L., P. Spiess, D. Guinard, M. Köhler, et al., *Socrades: A web service based shop floor integration infrastructure*. The Internet of Things, 2008: p. 50-67.
- [56]. Gardoni, M., C. Frank, and F. Vernadat, *Knowledge capitalisation based on textual and graphical semi-structured and non-structured information: case study in an industrial research centre at EADS*. Computers in industry, 2005. **56**(1): p. 55-69.
- [57]. Matta, N., B. Eynard, L. Roucoules, and M. Lemerrier *Continuous capitalization of design knowledge*. [cited 15th January 2011]; Available from: <http://www-sop.inria.fr/acacia/WORKSHOPS/ECAI2002-OM/Actes/Matta.pdf>.
- [58]. Mehrabi, M.G., A.G. Ulsoy, and Y. Koren, *Reconfigurable manufacturing systems: key to future manufacturing*. Journal of Intelligent Manufacturing, 2000. **11**(4): p. 403-419.
- [59]. Harrison, R., A.W. Colombo, A.A. West, and S.M. Lee, *Reconfigurable modular automation systems for automotive power-train manufacture*. International Journal of Flexible Manufacturing Systems, 2006. **18**(3): p. 175-190.
- [60]. ElMaraghy, H.A., *Flexible and reconfigurable manufacturing systems paradigms*. International journal of flexible manufacturing systems, 2005. **17**(4): p. 261-276.
- [61]. Mehrabi, M.G., A.G. Ulsoy, Y. Koren, and P. Heytler, *Trends and perspectives in flexible and reconfigurable manufacturing systems*. Journal of Intelligent manufacturing, 2002. **13**(2): p. 135-146.
- [62]. Koren, Y. and A.G. Ulsoy, *Vision, principles and impact of reconfigurable manufacturing systems*. Journal of Powertrain International, 2002. **5**(3): p. 14-21.

References

- [63]. Moyne, J., J. Korsakas, C. Milas, T. Hobrla, et al. *A Software Infrastructure for Reconfigurable Manufacturing Systems*. in *2nd CIRP Reconfigurable Manufacturing Conference*. 2003.
- [64]. Wikipedia. *ANSI/ISA-95*. [cited 05th Jan 2011]; Available from: <http://en.wikipedia.org/wiki/ANSI/ISA-95>.
- [65]. ISA. *ISA99, Industrial Automation and Control Systems Security*. [cited 05th June 2011]; Available from: <http://www.isa.org/MSTemplate.cfm?MicrositeID=988&CommitteeID=6821>.
- [66]. Jones, A.T. and C.R. McLean, *A proposed hierarchical control model for automated manufacturing systems*. *Journal of Manufacturing Systems*, 1986. **5**(1): p. 15-25.
- [67]. Adshead, A. *Ford uses data analysis to boost productivity by 50% at Dagenham*. [cited 05 June 2011]; Available from: <http://www.computerweekly.com/Articles/2003/06/17/195308/Ford-uses-data-analysis-to-boost-productivity-by-50-at.htm>.
- [68]. Dietrich, D. and T. Sauter. *Evolution potentials for fieldbus systems*. in *IEEE Workshop on Factory Communication Systems*. 2000.
- [69]. McFarlane, D.C. and S. Bussmann, *Holonic manufacturing control: Rationales, developments and open issues*. *Agent-Based Manufacturing, Advances in the Holonic Approach*, 2003: p. 303-326.
- [70]. Tovar, E. and F. Vasques, *Real-time fieldbus communications using Profibus networks*. *IEEE transactions on Industrial Electronics*, 1999. **46**(6): p. 1241-1251.
- [71]. Leitão, P., A.W. Colombo, and F. Restivo. *A formal validation approach for holonic control system specifications*. in *IEEE Conference Proceedings on Emerging Technologies and Factory Automation*. 2003.
- [72]. DiFrank, G., *Power of automation*. *Industry Applications Magazine, IEEE*, 2008. **14**(2): p. 49-57.
- [73]. Katzel, J., *Defining(and re-defining) HMIs*. *Control Engineering*, 2004. **51**(12): p. 60-60.

References

- [74]. Lee, K.H., E.C. Tamayo, and B. Huang, *Industrial implementation of controller performance analysis technology*. Control Engineering Practice, 2010. **18**(2): p. 147-158.
- [75]. Lin, H.C., *A remote monitoring and control-based precise multilocation riveting system*. Computer Applications in Engineering Education, 2005. **13**(4): p. 316-323.
- [76]. Van Tan, V., D.S. Yoo, and M.J. Yi, *A Novel Framework for Building Distributed Data Acquisition and Monitoring Systems*. Journal of Software, 2007. **2**(4).
- [77]. Salihbegovic, A., V. Marinkovic, Z. Cico, E. Karavdic, et al., *Web based multilayered distributed SCADA/HMI system in refinery application*. Computer Standards & Interfaces, 2009. **31**(3): p. 599-612.
- [78]. Plaza, I., C. Medrano, and A. Blesa, *Analysis and implementation of the IEC 61131-3 software model under POSIX real-time operating systems*. Journal of Microprocessors and Microsystems, 2006. **30**(8): p. 497-508.
- [79]. Interview, *Ford Motor Company Control Engineers, Operators and ICT team*. 2008-2010, Dunton, Essex.
- [80]. Interview, *ThyssenKrupp Krause Commissioning and Maintenance Engineers*. 2009-2010, Bremen, Germany.
- [81]. Courses, E. and T. Surveys, *It's good to talk-THE LATEST COMPANY TO ATTACK THE MANUFACTURING IT ARENA IS ROCKWELL AUTOMATION, WITH ITS FACTORYTALK PRODUCT. MANUFACTURING ENGINEER TALKS TO ONE OF THE MEN BEHIND THAT STRATEGY*. Manufacturing Engineer, 2006. **85**(6): p. 30-35.
- [82]. Phaithoonbuathong, P., R. Harrison, A. West, R. Monfared, et al., *Web services-based automation for the control and monitoring of production systems*. International Journal of Computer Integrated Manufacturing, 2010. **23**(2): p. 126-145.
- [83]. Ong, M.H., *Evaluating the impact of adopting a component-based system within the automotive domain*, in *PhD Thesis*. 2004, Loughborough Univesity.

References

- [84]. COMPANION. *COMmon Model for PArtNers in AutomatIOn* 2005 [cited 2010 15th July]; Available from: <http://www.lboro.ac.uk/departments/mm/research/manufacturing-systems/dsg/doc/compag.htm>.
- [85]. COMPAG. *COMponent Based Paradigm for AGile Automation*. 2004 [cited 2010 15th July]; Available from: <http://www.lboro.ac.uk/departments/mm/research/manufacturing-systems/dsg/doc/compag.htm>.
- [86]. Fantuzzi, C., F. Fanfoni, C. Secchi, and M. Bonfe. *An engineering process for the mechatronic development of industrial automation systems*. in *8th IEEE International Conference on Industrial Informatics (INDIN)*. 2010.
- [87]. SOCRADES. *Service-Oriented Cross-layer infRAstructure for Distributed smart Embedded devices*. 2009 [cited 15th April 2012]; Available from: <http://www.socrades.eu/Documents/objects/file1224780946.72>.
- [88]. Luder, A., L. Hundt, and S. Biffl. *On the suitability of modeling approaches for engineering distributed control systems*. in *IEEE Conference on Industrial Informatics*. 2009. Cardiff, Wales.
- [89]. Monfared, R., I. Haq, R. Harrison, L. Lee, et al., *A new vision for the automation systems engineering for Automotive Powertrain Assembly*. 2010.
- [90]. Colombo, A.W. and R. Harrison, *Modular and collaborative automation: achieving manufacturing flexibility and reconfigurability*. *International Journal of Manufacturing Technology and Management*, 2008. **14**(3): p. 249-265.
- [91]. Takata, S., F. Kirnura, F. Van Houten, E. Westkamper, et al., *Maintenance: changing role in life cycle management*. *CIRP Annals-Manufacturing Technology*, 2004. **53**(2): p. 643-655.
- [92]. Swanson, L., *An information-processing model of maintenance management*. *International Journal of Production Economics*, 2003. **83**(1): p. 45-64.
- [93]. Titus, J.B. *Machine safety pays off. risk analysis* 2008 [cited 15th January 2011]; Available from: <http://www.ibtitus.com/Machine%20Safety%20Pays%20Off.pdf>.

References

- [94]. Yu, R., B. lung, and H. Panetto, *A multi-agents based E-maintenance system with case-based reasoning decision support*. Engineering Applications of Artificial Intelligence, 2003. **16**(4): p. 321-333.
- [95]. Moore, P.R., J. Pu, H.C. Ng, C.B. Wong, et al., *Virtual engineering: an integrated approach to agile manufacturing machinery design and control*. Mechatronics, 2003. **13**(10): p. 1105-1121.
- [96]. Lee, J., *Teleservice engineering in manufacturing: challenges and opportunities*. International Journal of Machine Tools and Manufacture, 1998. **38**(8): p. 901-910.
- [97]. Hatch, D. and T. Stauffer. *Operators on alert*. 2009 [cited 15th January 2011]; Available from: [http://www.exida.com/images/uploads/Alarm_Management_Intech_\(Sept_2009\).pdf](http://www.exida.com/images/uploads/Alarm_Management_Intech_(Sept_2009).pdf).
- [98]. Moyne, J., J. Korsakas, and D.M. Tilbury. *Reconfigurable factory testbed (RFT): A distributed testbed for reconfigurable manufacturing systems*. in *Proceedings of the Japan–USA Symposium on Flexible Automation*. 2004.
- [99]. Moyne, J.R. and D.M. Tilbury, *The emergence of industrial control networks for manufacturing control, diagnostics, and safety data*. Proceedings of the IEEE, 2007. **95**(1): p. 29-47.
- [100]. Vrba, P., P. Tichy, V. Mar i k, K.H. Hall, et al., *Rockwell Automation's Holonic and Multiagent Control Systems Compendium*. Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on, 2011. **41**(1): p. 14-30.
- [101]. Vrba, P., P. Tichy, V. Mar i k, K.H. Hall, et al., *Rockwell Automation's Holonic and Multiagent Control Systems Compendium*. IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews, 2010. **41**(1): p. 14-30.
- [102]. Garbrecht, S., *The Benefits of Component Object-Based Supervisory System Application Development versus Traditional HMI Development in Water Systems Operations Management*. Proceedings of the Water Environment Federation, 2008. **2008**(8): p. 7358-7370.
- [103]. Process-Industry-News. *Rockwell Automation Launches Web-HMI*. [cited 05 June 2011]; Available from:

References

<http://www.processindustryinformer.com/Process-Control-Drives-Automation/Rockwell-Automation-Launches-Web-HMI>.

- [104]. SIRENA. *Welcome to the ITEA SIRENA project*. [cited 01 March 2011]; Available from: <http://www.sirena-itea.org/>.
- [105]. Muto, K., *Advanced technology for manufacturing engineering development: XML technology on a system that enables user to view required information from the work shop through a web browser*. JSAE Review, 2003. **24**(3): p. 303-312.
- [106]. Shi, H.L., Y.M. Song, J.W. Xiang, W.W. Yue, et al., *The Remote Monitoring System for Fault Diagnosis Using ActiveX Control Technique*. Advanced Materials Research, 2011. **201**: p. 1993-1997.
- [107]. Kirubashankar, R., K. Krishnamurthy, and J. Indra, *Remote monitoring system for distributed control of industrial plant process*. Journal of Scientific & Industrial Research, 2009. **68**: p. 858-860.
- [108]. Campos, J., *Development in the application of ICT in condition monitoring and maintenance*. Computers in Industry, 2009. **60**(1): p. 1-20.
- [109]. Li, X., D.J. McKee, T. Horberry, and M.S. Powell, *The control room operator: The forgotten element in mineral process control*. Minerals Engineering, 2011. **24**(8).
- [110]. Pantförder, D., B. Vogel-Heuser, and K. Schweizer, *Benefit and Evaluation of Interactive 3D Process Data Visualization for the Presentation of Complex Problems*. Human-Computer Interaction. Novel Interaction Methods and Techniques, 2009: p. 869-878.
- [111]. Agrusa, R., V.G. Mazza, and R. Penso. *Advanced 3D visualization for manufacturing and facility controls*. in *IEEE HSI '09. 2nd Conference on Human System Interactions, 2009*. . 2009.
- [112]. BDA. *Business Driven Automation*. 2011 [cited 2010 15th July]; Available from: <http://www.lboro.ac.uk/eng/research/imrc/brochure/engineering-change.html>.

References

- [113]. Lee, S.C. and A.I. Shirani, *A component based methodology for Web application development*. Journal of systems and software, 2004. **71**(1-2): p. 177-187.
- [114]. Kopetz, H., *Component-based design of large distributed real-time systems*. Control Engineering Practice, 1998. **6**(1): p. 53-60.
- [115]. Hill, J.H., J.R. Edmondson, A. Gokhale, and D.C. Schmidt, *Agile Development of Component-based Distributed Real-time and Embedded Systems via Model-Driven Engineering Techniques*. 2009.
- [116]. Sommerville, I., *Software Engineering*. 9th ed. 2011, Reading, Massachusetts: Addison-Wesley Publishing Company.
- [117]. Bouyssounouse, B. and J. Sifakis, *Embedded Systems Design: The ARTIST Roadmap for Research and Development*. 2005: Springer Verlag.
- [118]. Ong, M.H., A.A. West, S.M. Lee, and R. Harrison, *A structured approach to evaluating the impact of implementing a component-based system in the automotive engine manufacturing domain*. International Journal of Production Research, 2006. **44**(13): p. 2645-2670.
- [119]. Harrison, R. and A.A. West, *Component based paradigm for the design and implementation of control systems in electronics manufacturing machinery*. Journal of Electronics Manufacturing, 2000. **10**(1): p. 1-17.
- [120]. Lee, S.M., R. Harrison, and A.A. West, *A component-based control system for agile manufacturing*. Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture, 2005. **219**(1): p. 123-135.
- [121]. Lee, S.M., R. Harrison, A.A. West, and M.H. Ong, *A component-based approach to the design and implementation of assembly automation system*. Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture, 2007. **221**(5): p. 763-773.
- [122]. Raza, M.B., T. Kirkham, R. Harrison, R.P. Monfared, et al. *Evolving knowledge based product lifecycle management from a digital ecosystem to support automated manufacturing*. 2009: ACM.

References

- [123]. Stephanidis, C., *User interfaces for all: New perspectives into human-computer interaction*. User Interfaces for All—concepts, methods and tools. 2001, Mahwah, New Jersey: Lawrence Erlbaum Associates. 3-17.
- [124]. Degani, A. and M. Heymann, *Formal verification of human-automation interaction*. International Journal of Human Factors, 2002. **44**(1): p. 28-43.
- [125]. Luyten, K., T. Clerckx, K. Coninx, and J. Vanderdonckt, *Derivation of a dialog model from a task model by activity chain extraction*. Interactive Systems. Design, Specification, and Verification, 2003: p. 83-83.
- [126]. Szekely, P., P. Sukaviriya, P. Castells, J. Muthukumarasamy, et al., *Declarative interface models for user interface construction tools: the MASTERMIND approach*. Engineering for Human-Computer Interaction, 1996: p. 120-150.
- [127]. Pinheiro da Silva, P., *User interface declarative models and development environments: A survey*. Interactive Systems Design, Specification, and Verification, 2001. **1946/2001**: p. 207-226.
- [128]. Puerta, A.R., *A model-based interface development environment*. IEEE Journal of Software, 2002. **14**(4): p. 40-47.
- [129]. Mori, G., F. Paternò, and C. Santoro, *CTTE: support for developing and analyzing task models for interactive system design*. IEEE Transactions on software engineering, 2002. **28**(8): p. 797-813.
- [130]. Vanderdonckt, J.M. and F. Bodart. *Encapsulating knowledge for intelligent automatic interaction objects selection*. 1993: ACM Press.
- [131]. Birnbaum, L., R. Bareiss, T. Hinrichs, and C. Johnson. *Interface design based on standardized task models*. 1998: ACM.
- [132]. Sinnig, D., P. Chalin, and F. Khendek, *Consistency between task models and use cases*. Engineering Interactive Systems, 2008. **4940/2008**: p. 71-88.
- [133]. Booch, G., R. Maksimchuk, M. Engle, B. Young, et al., *Object-oriented analysis and design with applications*. 2007: Addison-Wesley Professional.

References

- [134]. Pineda, L., I. Meza, and L. Salinas, *Dialogue model specification and interpretation for intelligent multimodal HCI*. Advances in Artificial Intelligence–IBERAMIA 2010: p. 20-29.
- [135]. Jacko, J.A., *Human-computer Interaction: Design Issues, Solutions, and Applications*. 2009: CRC.
- [136]. Fischer, G., *User modeling in human–computer interaction*. User modeling and user-adapted interaction, 2001. **11**(1): p. 65-86.
- [137]. Lozano, M.D., F. Montero, and P. González. *A Usability and Accessibility Oriented Development Process*. in *8th ERCIM Workshop on “User Interfaces For All”(UI4ALL’04)*. Viena, Austria. Junio. 2004.
- [138]. Moreno, L., P. Martínez, and B. Ruiz-Mezcua, *Integrating HCI in a Web Accessibility engineering approach*. Universal Access in Human-Computer Interaction. Applications and Services, 2009. **5616/2009**: p. 745-754.
- [139]. Ambler, S.W., *The object primer: Agile model-driven development with UML 2.0*. 2004: Cambridge University Press.
- [140]. Koch, N., H. Baumeister, R. Hennicker, and L. Mandel. *Extending UML to Model Navigation and Presentation in Web Applications*. in *Workshop on the UML and Modelling Web Applications, UML’2000*. 2000.
- [141]. Lank, E., J.S. Thorley, and S.J.S. Chen. *An interactive system for recognizing hand drawn UML diagrams*. in *Proceedings of the 2000 conference of the Centre for Advanced Studies on Collaborative research*. 2000.
- [142]. John, B.E., L. Bass, R. Kazman, and E. Chen. *Identifying gaps between HCI, software engineering, and design, and boundary objects to bridge them*. in *CHI '04 extended abstracts on Human factors in computing systems 2004*.
- [143]. Wahid, S., D.S. McCrickard, J. DeGol, N. Elias, et al. *Don’t drop it! Pick it up and storyboard*. in *CHI 2011*. 2011. Vancouver, Canada.
- [144]. Stephanidis, C. and A. Savidis, *Universal access in the information society: methods, tools, and interaction technologies*. Universal Access in the Information Society, 2001. **1**(1): p. 40-55.

References

- [145]. Mayhew, D.J. *The usability engineering lifecycle*. in *CHI '99 Extended abstracts on Human Factors in Computing Systems*. 1999.
- [146]. Constantine, L.L. and L.A.D. Lockwood, *Software for use: a practical guide to the models and methods of usage-centered design*. 1999: ACM Press/Addison-Wesley Publishing Co. New York, NY, USA.
- [147]. Schneiderman, B., *Designing the user interface: strategies for effective human - computer interaction*. . Fifth Edition ed. 2010: Addison-Wesley.
- [148]. Amditis, A., L. Andreone, K. Pagle, G. Markkula, et al., *Towards the Automotive HMI of the Future: Overview of the AIDE-Integrated Project Results*. IEEE Transactions on Intelligent Transportation Systems, 2010. **11**(3): p. 567-578.
- [149]. Framinan, J.M. and R. Ruiz, *Architecture of manufacturing scheduling systems: Literature review and an integrated proposal*. European Journal of Operational Research, 2010. **205**(2): p. 237-246.
- [150]. Coury, B.G. and C.M. Pietras, *Alphanumeric and graphic displays for dynamic process monitoring and control*. Ergonomics, 1989. **32**(11): p. 1373-1389.
- [151]. Marcus, A., *Designing the User Interface: Strategies for Effective Human-Computer Interaction*. Information Design Journal, 2009. **17**(2): p. 157-158.
- [152]. Sanderson, S. and D. Sanderson, *Pro Asp. net Mvc V2 Framework*. 2010, New York, NY: Apress.
- [153]. Azam, M.A. and K. Khan. *Design of the Ethernet based process data extraction algorithm and storage technique for industrial HMI systems*. in *2nd International Conference on Computer and Automation Engineering (ICCAE)*. 2010: IEEE.
- [154]. SchneiderElectric. *Vijeo Designer - HMI, SCADA and Historian software*. [cited 15th February 2011]; Available from: <http://www.schneider-electric.co.uk/sites/uk/en/products-services/automation-control/products-offer/software-tools/hmi-and-scada-software-tools/vijeo-designer.page>.

References

- [155]. Siemens. *SCADA System SIMATIC WinCC*. [cited 15th February 2011]; Available from: <http://www.automation.siemens.com/mcms/human-machine-interface/en/visualization-software/scada/Pages/Default.aspx>.
- [156]. Sauer, O., *Production Monitoring Linked to Object Identification and Tracking a Step Towards Real Time Manufacturing In Automotive Plants*. Digital Enterprise Technology, 2007: p. 149-156.
- [157]. Hohpe, G., B. Woolf, and K. Brown, *Enterprise integration patterns*. 2004: Citeseer.
- [158]. Wikipedia. *Desktop Sharing*. [cited 15th February 2011]; Available from: http://en.wikipedia.org/wiki/Desktop_sharing.
- [159]. SonicWALL. *SSL VPN Secure Remote Access*. [cited 05th Jan 2011]; Available from: http://www.sonicwall.com/us/products/Secure_Remote_Access.html.
- [160]. No-1-Reviews. *Remote PC Access Reviews*. [cited 05th Jan 2011]; Available from: <http://remote-pc-access.no1reviews.com/>.
- [161]. Juniper-Networks. *SA Series - Secure Access VPN Appliances*. [cited 05th Jan 2011]; Available from: <http://www.juniper.net/us/en/products-services/security/sa-series/>.
- [162]. Cisco. *Cisco Easy VPN*. [cited 05th Jan 2011]; Available from: <http://www.cisco.biz/en/US/products/sw/secursw/ps5299/index.html>.
- [163]. R*HUB. *Web Conferencing Comparisons - RHUB, WebEx and Citrix*. [cited 05th Jan 2011]; Available from: <http://www.rhubcom.com/front/comparison.htm>.
- [164]. Encyclopedia, e.-T.W.a.A. *Is WebEx PCNow secure?* [cited 05th Jan 2011]; Available from: <http://www.experts123.com/q/is-webex-pcnow-secure.html>.
- [165]. WebEx. *WebEx MediaTone Technology Series White Paper*. [cited 05th Jan 2011]; Available from: http://www.webex.com/pdf/wp_mediatone.pdf.
- [166]. FDS. *Fully Distributed Systems*. [cited 05th Jan 2011]; Available from: <http://www.fullydistributedsystems.com/>.

References

- [167]. SAP. *SAP Business Suite*. [cited 15th February 2011]; Available from: <http://www.sap.com/solutions/business-suite/index.epx>.
- [168]. Siemens. *Scalance S Security Modules*. [cited 15th February 2011]; Available from: <http://www.automation.siemens.com/mcms/industrial-communication/en/ie/industrial-security/scalance-s/Pages/scalance-s.aspx>.
- [169]. Stallings, W., *Cryptography and network security: principles and practice*. 2011, New York: Prentice Hall.
- [170]. Neumann, P., *Communication in industrial automation--What is going on?* Control Engineering Practice, 2007. **15**(11): p. 1332-1347.
- [171]. Stallings, W., *Cryptography and network security*. 2003: Prentice Hall Upper Saddle River, NJ.
- [172]. Funderburk, J.E., S. Malaika, and B. Reinwald, *XML programming with SQL/XML and XQuery*. IBM Systems Journal, 2002. **41**(4): p. 642-665.
- [173]. Lee, K.C. and S. Lee, *Performance evaluation of switched Ethernet for real-time industrial communications*. Journal of Computer Standards & Interfaces, 2002. **24**(5): p. 411-423.
- [174]. LEE, I.J., *A next generation manufacturing control system*, in *PhD Thesis*. 2003, Loughborough University.
- [175]. Metz, C., *IP anycast point-to-(any) point communication*. Internet Computing, IEEE, 2002. **6**(2): p. 94-98.
- [176]. Bass, L., P. Clements, and R. Kazman, *Software architecture in practice*. Second Edition ed. 2003, Boston, MA: Pearson Education, Inc.
- [177]. Trowbridge, D., *Integration Patterns*. 2004: Microsoft Press.
- [178]. Ossher, H., W. Harrison, and P. Tarr. *Software engineering tools and environments: a roadmap*. in *Proceedings of the Conference on The Future of Software Engineering*. 2000.
- [179]. Bosch, J., *Software architecture: The next step*, in *Software architecture*. 2004, Springer Berlin / Heidelberg. p. 194-199.

References

- [180]. Gamma, E., R. Helm, R. Johnson, and J. Vlissides, *Design patterns*. Vol. 1. 2002: Addison-Wesley Reading, MA.
- [181]. Wikipedia. *Model-View-Controller*. [cited 25th February 2011]; Available from: <http://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller>.
- [182]. Stallings, W., *Network Security Essentials: Applications and Standards*. 2007: Prentice Hall.
- [183]. Al-Ameed, H., *Architecture of reliable Web applications software*. 2007, London, United Kingdom: Idea Group Publishing.
- [184]. Hexatec. *What makes a successful Operator Screen?* [cited 15th April 2011]; Available from: http://www.hexatec.co.uk/Consultancy/hmi_display_design_guidelines.aspx.
- [185]. Scott, C., *The Industrial Ethernet Book*. Don't let colours hide the alarms. 2007.
- [186]. ISA-SP101. *ISA Forms Human-Machine Interface Standards Committee*. 2005 [cited 2012 15th April]; Available from: <http://www.ihc.com/news/2005/isa-human-machine-interface-standard.htm>.
- [187]. Siemens. *Transline HMI PRO*. [cited 15th April 2011]; Available from: <http://www.automation.siemens.com/mcms/industrial-controls/en/industrial-communication/as-interface/diagnostics/transline/Pages/default.aspx>.
- [188]. Ponsa, P. and M. Díaz, *Creation of an ergonomic guideline for supervisory control interface design*. *Engineering Psychology and Cognitive Ergonomics*, 2007. **4562/2007**: p. 137-146.
- [189]. Booch, G., J. Rumbaugh, and I. Jacobson, *The unified modeling language user guide*. 1999: Addison Wesley Longman Publishing Co., Inc. Redwood City, CA, USA.

References

- [190]. Karampelas, P., I. Basdekis, and C. Stephanidis, *Web user interface design strategy: Designing for device independence*. Universal Access in Human-Computer Interaction. Addressing Diversity, 2009: p. 515-524.
- [191]. Landay, J.A. and B.A. Myers. *Sketching storyboards to illustrate interface behaviors*. 1996: ACM.
- [192]. Corkill, D.D., *Blackboard systems*. AI expert, 1991. **6**(9): p. 40-47.
- [193]. Hunt, J. and H. Park. *Blackboard Architectures*. 2002 [cited 2011 15th February]; Available from: http://www.agent.ai/doc/upload/200402/hunt02_1.pdf.
- [194]. Shaw, M. and D. Garlan, *Software architecture: perspectives on an emerging discipline*. 1996: Prentice-Hall, Inc. Upper Saddle River, NJ, USA.
- [195]. Corkill, D.D. *Collaborating software: Blackboard and multi-agent systems & the future*. in *Proceedings of the International Lisp Conference*. 2003.
- [196]. Lau, T.L., H.Y.K. Lau, and A. Ko. *A Distributed Blackboard-based Control System for Modular Self-Reconfigurable Robots*. in *The University of Hong Kong. Department of Industrial and Manufacturing Systems Engineering*. 2003.
- [197]. Dong, J., S. Chen, and J.J. Jeng. *Event-Based Blackboard Architecture for Multi-Agent Systems*. in *Proceedings of the International Conference on Information Technology: Coding and Computing*. 2005. Las Vegas, Nevada.
- [198]. Craig, I.D., *Blackboard systems*. Artificial Intelligence Review, 1988. **2**(2): p. 103-118.
- [199]. Abbod, M.F., D.A. Linkens, A. Browne, and N. Cade, *A blackboard software architecture for integrated intelligent control systems*. Kybernetes, 2000. **29**(7/8): p. 999-1015.
- [200]. Hughes, C. and T. Hughes, *Parallel and distributed programming using C++*. 2003: Prentice Hall Professional Technical Reference.

References

- [201]. Craig, I.D., *The Cassandra architecture: distributed control in a blackboard system*, in *Ellis Horwood Series In Applied Science and Industrial Techn* 1989.
- [202]. Philip, G.C., *Software design guidelines for event-driven programming*. *The Journal of Systems & Software*, 1998. **41**(2): p. 79-91.
- [203]. Stevens, W.R. and G.R. Wright, *TCP/IP illustrated: the implementation*. Vol. 2. 1995: addison-Wesley.
- [204]. Booch, G., R. Maksimchuk, M. Engle, B. Young, et al., *Object-oriented analysis and design with applications*. 2007.
- [205]. Holzner, S., *Visual Basic. net Programming Black Book*. 2004: The Coriolis Group.
- [206]. Paulson, L.D., *Building rich web applications with Ajax*. *Computer*, 2005. **38**(10): p. 14-17.
- [207]. SAP. *SAP MANUFACTURING INTEGRATION AND INTELLIGENCE*. [cited 20 June 2011]; Available from: <http://www.sap.com/solutions/manufacturing/manufacturing-intelligence-software/index.epx>.
- [208]. Wikipedia. *HTTP Secure*. [cited 15th July 2011]; Available from: http://en.wikipedia.org/wiki/HTTP_Secure.
- [209]. Wikipedia. *Transmission Control Protocol*. [cited 31st July 2011]; Available from: http://en.wikipedia.org/wiki/Transmission_Control_Protocol.
- [210]. Hannelius, T., M. Salmenpera, and S. Kuikka. *Roadmap to adopting OPC UA*. in *6th IEEE International Conference on Industrial Informatics*. 2008. Daejeon.
- [211]. Schneider-Electric. *OPC Factory Server Software*. [cited 15th September 2011]; Available from: <http://products.schneider-electric.us/products-services/products/scada-mes-and-hmi-software/opc-factory-server-software/>.

Appendices

Appendix A: List of Abbreviations

Acronym <i>(Alphabetical Order)</i>	<i>Description</i>
3D	<i>Three Dimensional</i>
AJAX	<i>Asynchronous JavaScript and XML</i>
API	<i>Application Programming Interface</i>
ASP	<i>Active Server Page</i>
BDA	<i>Business Driven Automation</i>
CB	<i>Component-Based</i>
CO₂	<i>Carbon dioxide</i>
COMPAG	<i>Component Based Paradigm for Agile Automation</i>
COTS	<i>Commercial Off-The-Shelf</i>
DPWS	<i>Devices Profile for Web Services</i>
EPSRC	<i>Engineering and Physical Research Council</i>
ERP	<i>Enterprise Resource Planning</i>
FIFO	<i>First In, First Out</i>
FTB	<i>Field Terminal Block</i>
Gbps	<i>Gigabits per second</i>
GUI	<i>Graphical User Interface</i>
HMI	<i>Human Machine Interface</i>
HTML	<i>HyperText Markup Language</i>

HTTP	<i>HyperText Transfer Protocol</i>
HTTPS	<i>HyperText Transfer Protocol Secure</i>
ICT	<i>Information and Communication Technology</i>
IEC	<i>International Electrotechnical Commission</i>
IMCRC	<i>Innovative Manufacturing and Construction Research Centre</i>
IMS	<i>Issue Management System</i>
I/O	<i>Input/Output</i>
IP	<i>Internet Protocol</i>
ISP	<i>Internet Service Provider</i>
KS	<i>Knowledge Source (s)</i>
LIFO	<i>Last In, First Out</i>
Mbps	<i>Megabits per Second</i>
MII	<i>Manufacturing Integration and Intelligence</i>
MSI	<i>Manufacturing System Integration</i>
MVC	<i>Model View Controller</i>
OEM	<i>Original Equipment Manufacturer</i>
OOP	<i>Object-Oriented Programming</i>
OSI	<i>Open System Interconnection</i>
PC	<i>Personal Computer</i>
PDF	<i>Portable Document Format</i>
PLC	<i>Programmable Logic Controller</i>
RDBMS	<i>Relational Database Management System</i>

RFID	<i>Radio Frequency Identification</i>
RMS	<i>Reconfigurable Manufacturing System</i>
SCADA	<i>Supervisory Control and Data Acquisition</i>
SOA	<i>Service Oriented Architecture</i>
SOAP	<i>Simple Object Access Protocol</i>
SOCRADES	<i>Service Oriented Cross-layer Infrastructure for Distributed Smart Embedded Devices</i>
SQL	<i>Structured Query Language</i>
SSL	<i>Secure Sockets Layer</i>
TCP	<i>Transmission Control Protocol</i>
TLS	<i>Transport Layer Security</i>
UML	<i>Unified Modelling Language</i>
URL	<i>Uniform Resource Locator</i>
VRML	<i>Virtual Reality Modelling Language</i>
WAN	<i>Wide Area Network</i>
WS	<i>Web Services</i>
WWW	<i>World Wide Web</i>
W.Y.S.I.W.Y.G	<i>What You See Is What You Get</i>
XML	<i>Extensible Markup Language</i>

Appendix B: Documentation on Remote Desktop Service Providers

Numerous products exist in the market targeting remote desktop sharing functionality. The following table shows comparison of major solution providers with their associated product features.

Product Features	Major Solution Providers (e.g. Product)					
	<u>Cisco WebEx</u> (e.g. PCNow)	<u>Citrix</u> (e.g. GoToMyPC Pro)	<u>TeamViewer</u>	<u>Laplink</u> (e.g. Laplink Everywhere)	<u>Symantec</u> (e.g. PC Anywhere)	<u>LogMeIn</u> (e.g. LogMeIn Pro)
Application Sharing	√	√	√	√	√	√
Auto Reconnection	√	√	√	√	√	√
<u>Costs</u>	10PC's - \$59.95 /month (Additional PCs - \$9.95 PC/month) Or 45cents per minute per user	10PC's - \$169.50 /month (More than 20 PCs - \$14.00 PC/month). One user: \$19.95/month	Business License - \$699 Unlimited clients – one supporter (Lifetime). Free for private users	3PC's - \$99.95 /year (More than 5 Pc's – additional deals)	1 PC + 1 remote - \$199.99 (Additional remotes \$99.00 each) (Lifetime)	10PC's - \$99.50 /month (Additional PCs - \$7.95 PC/month). Free limited version available
<u>Data Encryption Standard</u>	End-to-End 128 bit SSL encryption	End-to-End 128 bit AES encryption	RSA+256 bit AES encryption	End-to-End 128 bit SSL encryption	RC4+AES 128/192/256 – bit encryption	End-to-End 128/256 bit SSL encryption
File Transfer	√	√	√	√	√	√
Firewall Friendly	√	√	√	√	Requires modifications (Port 5631/2)	√
Free Trial / Version	√	√	√	√	None	√
Hardware Implementation	None	None	None	None	None	None
Implementation Time	Within 24 hours	Within 24 hours	Within 24 hours	Within 24 hours	Within 24 hours	Within 24 hours
<u>Integration API for remote session</u>	Open API	?	None	None	None	?
International (or Multiple) Language Support	√	√	√	√	√	√
Licensing	Per PC	Per PC	Per client	Per PC	Per PC	Per PC
<u>Machine State Transmission (Remote Monitoring)</u>	√	√	√	√	√	√

Appendices

Product Features	Major Solution Providers (e.g. Product)					
	<u>Cisco WebEx</u> (e.g. PCNow)	<u>Citrix</u> (e.g. GoToMyPC Pro)	<u>TeamViewer</u>	<u>Laplink</u> (e.g. Laplink Everywhere)	<u>Symantec</u> (e.g. PC Anywhere)	<u>LogMeIn</u> (e.g. LogMeIn Pro)
Mobile Device Support (as a client)	<i>iPhone and other windows mobile devices. No desktop access, only files and email.</i>	<i>Windows mobile, smartphones only. Full desktop access.</i>	<i>PDA support</i>	<i>Mobile devices, smartphones and Nintendo Wii (Almost any device)</i>	<i>Major mobile devices supported.</i>	-
<u>Performance (Bandwidth)</u>	<i>Better than in-house VPN. Modest requirements</i>	<i>Optimal but modest requirements</i>	<i>Varies (based on license type)</i>	<i>Efficient management</i>	<i>Configurable</i>	<i>Optimal but modest requirements</i>
Remote Control	√	√	√	√	√	√
Remote Control Speed	?	?	?	?	?	?
<u>Remote Desktop (Screen Copy / Desktop Sharing)</u>	√	√	√	√	√	√
Remote Login Process	<i>Username and password; access key on remote PC</i>	<i>Username and password; access key on remote PC</i>	<i>Username and Password</i>	<i>Email and Password; password on remote PC</i>	<i>Active Directory Integration. Username and Password</i>	<i>Username and password</i>
<u>Scalability</u>	<i>Upgrade necessary (based on license)</i>	<i>Upgrade necessary (based on license)</i>	<i>Upgrade necessary (based on license)</i>	<i>Upgrade necessary (based on license)</i>	<i>Upgrade necessary (based on license)</i>	<i>Upgrade necessary (based on license)</i>
<u>Security Mechanism</u>	<i>2 level authentication and SSL + Proprietary</i>	<i>Dual authentication + SSL + AES</i>	<i>RSA public/private key exchange + AES</i>	<i>2 level authentication and SSL + Proprietary</i>	<i>Multiple authentication and AES with RC4</i>	<i>SSL + Proprietary</i>
Software Requirements / Installations	<i>Agent software download, internet browser. Supports Windows and Mac hosts.</i>	<i>Agent software download, internet browser. Supports Windows hosts.</i>	<i>No installation needed. Supports Windows and Mac hosts.</i>	<i>Agent software download, internet browser. Supports Windows hosts.</i>	<i>Agent software installation on both host and client required. Supports Windows, Mac and Linux hosts.</i>	<i>Agent software download, internet browser. Supports Windows hosts.</i>
State Playback (Record)	√	√	√	?	√	√
Subscriptions	<i>Monthly / Yearly</i>	<i>Monthly / Yearly</i>	<i>Lifetime</i>	<i>Yearly</i>	<i>Lifetime</i>	<i>Monthly / Yearly</i>

Appendices

Product Features	Major Solution Providers (e.g. Product)					
	<u>Cisco WebEx</u> (e.g. PCNow)	<u>Citrix</u> (e.g. GoToMyPC Pro)	<u>TeamViewer</u>	<u>Laplink</u> (e.g. Laplink Everywhere)	<u>Symantec</u> (e.g. PC Anywhere)	<u>LogMeIn</u> (e.g. LogMeIn Pro)
Supported Remote Access Types by Vendor	<i>SSL VPN, RDA</i>	<i>RDA (Remote Desktop Access)</i>	<i>RDA and VPN</i>	<i>RDA</i>	<i>VPN</i>	<i>RDA</i>
Technical Support	<i>24*7 Live (no extra charge)</i>	<i>24*7 Live (no extra charge)</i>	<i>24*7 Live (no extra charge)</i>	<i>24*7 Live (no extra charge)</i>	<i>24*7 Live (no extra charge)</i>	<i>24*7 Live (no extra charge)</i>
<u>Text Chat</u>	√	√	√	√	√	?
Training Complexity	<i>Minimal or None</i>	<i>Minimal or None</i>	<i>Some</i>	<i>Minimal or None</i>	<i>Some</i>	<i>Minimal or None</i>
Video Integration / Transmission	<i>Possible</i>	<i>Possible</i>	<i>Possible</i>	<i>Possible</i>	<i>Possible</i>	<i>Possible</i>
<u>VOIP</u>	√	√	√	<i>None</i>	√	√
Web Conference	√	√	√	?	-	√
Whiteboard (Discussion Board)	√	√	√	√	√	√
NOTES:		<i>Needs an always "ON" internet connection to the host PC</i>		<i>Modem-to-Modem link available to call host remotely to activate internet connection</i>	<i>A very expensive and complex to use solution.</i>	

- Underlined features are the most important and directly related to Powertrain systems remote support requirements
- Costs vary according to the type of service, additional features requested and scalability requirements. Solution provider's charges are based on the number of computers and users. Discounts may be obtained with special arrangements like paying annually in advance, guaranteed contracts, direct debits, etc.
- Firewall Friendliness corresponds to the provision of transmitting data through the firewall without opening any additional ports. Some vendor products require additional ports depending on the type of the product service requested.
- Implementation time varies based on the type of product and number of users.
- Performance of remote solution depends on the Internet bandwidth available to the end user network.

Appendices

Product Features	Major Solution Providers (e.g. Product)					
	<u>Cisco WebEx</u> (e.g. PCNow)	<u>Citrix</u> (e.g. GoToMyPC Pro)	<u>TeamViewer</u>	<u>Laplink</u> (e.g. Laplink Everywhere)	<u>Symantec</u> (e.g. PC Anywhere)	<u>LogMeIn</u> (e.g. LogMeIn Pro)
- Security Mechanism can be configured based on the requirements and the product. Some additional features like call-back service; device locking and session management can also be requested.						

Appendix C: List of Publications

Barot, V., R. Harrison, and S. McLeod, *Distribution of Machine Information Using Blackboard Designed Component for Remote Monitoring of Reconfigurable Manufacturing Systems*, in *24th IEEE Conference on Advanced Information Networking and Applications Workshops*. 2010: Perth, Australia. P.145-151.

Barot, V., R. Harrison, S. McLeod, and A. West, "*Broadcaster*": *An architectural description of a prototype supporting real-time remote data propagation in distributed manufacturing*, in *7th IEEE Conference on Industrial Informatics* 2009: Cardiff, United Kingdom.

Barot, V., S. McLeod, R. Harrison, and A. West, *Efficient real-time remote data propagation mechanism for a Component-Based approach to distributed manufacturing*. *International Journal of Mechanical Systems Science and Engineering*, 2009. 1(3).