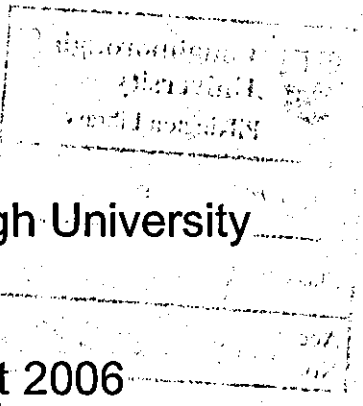
A COMPONENT BASED APPROACH TO HUMAN MACHINE INTERFACE SYSTEMS THAT SUPPORT AGILE MANUFACTURING

By

Edward William Mellor

Submitted to Department Manufacturing and Mechanical Engineering, Loughborough University

in partial fulfilment of the requirements for the degree of Doctor of Philosophy



Loughborough University

August 2006

A Component Based Approach to Human Machine Interface Systems that Support Agile Manufacturing

Edward William Mellor

Loughborough University 2006

Under the direct supervision of Dr. Robert Harrison and Dr. Andrew A. West

The development of next generation manufacturing systems is currently an active area of research worldwide. Globalisation is placing new demands on the manufacturing industry with products having shorter lifecycles and being required in more variants. Manufacturing systems must therefore be agile to support frequent manufacturing system reconfiguration involving globally distributed engineering partners. The research described in this thesis addresses one aspect within this research area, the Human Machine Interface (HMI) system that support the personnel involved in the monitoring, diagnostics and reconfiguration of automated manufacturing production machinery.

Current HMI systems are monolithic in their design, generally offer poor connectivity to other manufacturing systems and require highly skilled personnel to develop and maintain them.

The new approach established in the research and presented in this thesis provides a specification capture technique (using a novel storyboarding modelling notation) that enables the end users HMI functionality to be specified and rapidly developed into fully functional End User HMI's via automated generation tools.

A novel feature in this HMI system architecture that all machine information is stored in a common unified machine data model which ensures consistent accurate machine data is available to all machine lifecycle engineering tools including the HMI.

The system's run-time architecture enables remote monitoring and diagnostics capabilities to be available to geographically distributed engineering partners using standard internet technologies.

The implementation of this novel HMI approach has been prototyped and evaluated using the industrial collaborators full scale demonstrator machines within cylinder head machining and engine assembly applications.

Keywords

Human Machine Interface, Component-Based Development, User Interfaces, Agile Manufacturing, Automotive Industry, Manufacturing Systems.

Dedication

To my encouraging parents Cynthia, John and late grandfather Kenneth whose engineering ingenuity inspired me from an early age.

Acknowledgements

The road to the accomplishment of this research study has been a long, winding and challenging one. The lessons learned and the experiences gained during the entire process have been invaluable. However, this would not have been possible without the encouragement of my family and friends, the guidance and direct supervision of my supervisors and the practical expertise of my professional associates.

I would like to thank my major mentors and thesis advisors Dr Robert Harrison and Dr Andrew West. There patience, encouragement, continual reassurance and belief in my abilities and potential made this thesis work a success.

I would also like to thank all the members of the MSI group who I have learnt so much from during my 5 years of working and studying. Especially I would like to thank David Thomas, Daniel Vera, C.Stuart McLeod, Szer Ming Lee and Radmehr P. Monfared.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

PREFACE

ABBREVIATIONS

ASP	Active Server Pages
C-B	Component Based
GIF	Graphics Interchange Format
GUI	General User Interface
HMI	Human Machine Interface
HTML	Hyper Text Mark up Language
HTTP	Hyper Text Transfer Protocol
IT	Information Technology.
I/O	Input/Output
MMI	Man Machine Interface
OEM's	Original Equipment Manufacturers
PDF	Portable Document Format
PLC	Programmable Logic Controller
PC	Personal Computer
SCADA	Supervisory Control And Data Acquisition.
UI	User Interface
VRML	Virtual Reality Mark up Language
WYSIWYG	What You See Is What You Get
WWW	World Wide Web

1. Introduction

Globalisation has created new demands for manufacturers to produce a wide range of products of higher quality and at lower prices, which in turn demands shorter product manufacturing lead times. Agile manufacturing is a paradigm which enables enterprises to respond quickly to changing market demands and organisations must globally distribute manufacturing plants to remain competitive in the global marketplace through the formation of flexible and dynamic manufacturing systems.

There is a market pull for manufacturing systems where new technologies, tools and system architectures give automated manufacturing machinery more flexibility and are easier to use by providing more intuitive human machine interaction and recovery from failures at a lower investment [1]. The goal of automated manufacturing machinery is not to replace personnel (directly) by industrial systems. One of the primary goals in the automation of assembly tasks is an increase in process repeatability and, consequently, product quality. In today's manufacturing environment, production process user information is in demand more than ever. Real-time information is required to drive supply chain execution systems, quality assurance, maintenance, and scheduling activities [8].

The most agile components within any industrial system are the human personnel involved with its operation, maintenance and re-configuration. It is essential that the interface between the human and the machine is designed to meet the demands of an agile manufacturing environment. A Human Machine Interface (HMI) system is where the users interact with the automated manufacturing machinery through a series of screens that display machine information. Three aspects of HMI performance that are considered in this research to be critical to enable automation systems to better meet the demands of agile manufacturing are; 1) HMI systems that support a faster manufacturing systems ramp up time, 2) the maximised operating effectiveness of manufacturing systems and 3) increasing the manufacturing systems up time.

1.1 Scope of Research

The focus of the research outlined in this thesis is to identify the problems with current best industry practice and the adopted technology used in HMI system's to support the operation of automated manufacturing production machinery. To address properly these issues the unique processes that exist when designing HMI systems for manufacturing machines and the implementation methods currently used need to be understood. In this manner the system can be designed to fully meet the requirements and then accurately benchmarked against the existing solutions. The drivers that industry faces from agile manufacturing paradigms are outlined to provide the requirements for this research study. The research activities then focus towards proving the concepts of a system that satisfies this need. The principal research activities have included;

- Determination of current best practice within the industry, which involves understanding; 1) the roles of all stakeholders and interaction involved in the lifecycle of manufacturing production machines and 2) the technical limitations of the architectures and technologies adopted.
- Understanding of the external drivers that the manufacturing industry faces from "Agile Manufacturing" and the ability of current HMI systems to meet these requirements.
- A review of current state of the art tools, architectures and approaches within the research community that can be used with the next generation of HMI system's and their selection to facilitate the design of a HMI system to support the needs of agile manufacturing.
- Implementation of the HMI system on industrial collaborator's machines.
- An evaluation and comparison between best practices in current HMI systems and the new HMI paradigm described in this thesis.

1.2 Thesis Outline

The focus of all remaining chapters in this thesis is briefly explained as follows.

In chapter 2 the emerging demands the manufacturing community faces from agile paradigms are described. Current best industry practices for HMI systems are described together with their associated limitations. Then the state of the art technologies and approaches within the research community that are applied to general user interface systems are reviewed and characterised together with system architectures and development approaches. The research focus and the contributions generated are detailed together with the research method adopted are described in chapter 3. Chapter 4 describes how the user's functional requirements are specified and modelled using a novel storyboarding technique to create consistent well defined HMI system specifications. Chapter 5 describes the development of a HMI system architecture that fully supports the requirements of agile manufacturing. The construction of HMI systems and the roles of the stakeholders are described in chapter 6. Chapter 7 describes three case studies where the HMI approach presented in this thesis that have been implemented and core attributes of the system are tested. Chapter 8 evaluates the knowledge and results from the case studies and benchmarks them against current industry best practice. Chapter 9 concludes the research by highlighting the capabilities of this new approach to HMI system's and the limitations. Areas for further work are also detailed.

2 Literature Review

2.1 Introduction

The aim of this literature review is to firstly describe the need for this research study from the demands and challenges that manufacturing enterprises face in engineering manufacturing production machinery. Enterprises are facing tremendous pressures to be faster to market and achieve quicker machine builds. Manufacturing production systems must address issues surrounding mass customisation, quick response times and demand activated manufacturing [29]. This has motivated new paradigms within the manufacturing research community, such as reconfigurable manufacturing systems [43-45, 47, 48, 143-148 and 150]. Their highly modular characteristics enable manufacturing machinery to be quickly built and reconfigured to meet new product requirements. It is the HMI aspect of these machines which is the focus of this research study and how they how they can better address the needs of the emerging global manufacturing demands.

This chapter has the following structure. Terminology and referenced definitions applicable to the context of the research are stated. The demands and challenges that manufacturing enterprises face in engineering manufacturing production machinery is then described. Reconfigurable manufacturing systems are introduced and reviewed. The state of the art within user interface modelling and systems implementation within the research community is then described. This provides a knowledge base of user interface engineering which is expanded in this research study.

2.2 Preliminaries

The context of this research is user interface systems that are used by machine operators, machine diagnostic engineers and machine maintenance engineers to interact with manufacturing production machinery. The terminology used to describe this type of system is a Human Machine Interface (HMI).

There are many HMI definitions; two definitions considered by the author to be most appropriate for the context of this research study are;

A HMI is a software application that presents information to the operator about the state of a process, and to accept and implement the operators control instructions. It may also interpret the plant information and guide the interaction of the operator within the system. It is also known as Man Machine Interface (MMI) [108].

The user interface of a mechanical system, plant machinery or an industrial installation is often referred to as the Human Machine Interface (HMI) [107].

A critical aspect of all production machinery is the machine control system and how the operators interface using a HMI [114, 115]. A HMI traditionally would functionally consist of one or more screens executed on an industrial computer located in the immediate vicinity of the production machine [114]. The screens would provide the targeted users (typically these would be machine operators, machine diagnostic engineers and machine maintenance engineers) [22] will information relating to the machines status [115]. The users control varies aspects of the production machine using a series of buttons and controls on the screens [116].

2.3 Challenges in engineering Manufacturing Machinery

The globalisation of product design and manufacture is radically affecting the way companies do business [2]. As a consequence of globalisation, large organisations involved in the manufacturing process are geographically distributed with production often occurring in a different country or continent to design, which in turn is remote from the products final assembly [2]. With enterprises facing tremendous pressures to be faster to market and achieve quicker machine build, manufacturing production systems must address issues surrounding mass customisation, quick response times and demand activated manufacturing [29]. Shorter product life cycles are achieved through applying simultaneous engineering programmes to enable collaborative work between geographically dispersed product development team members [87]. Within this emerging environment, products are moving towards greater product variety with shorter lifecycles which is affecting design and manufacturing supply chain relationships. It has been recognised [109, 110, and 111] that this is creating new demands for manufacturing enterprises where they must respond quickly to changing market demands to remain competitive. This is accomplished through the formation of a flexible and dynamic manufacturing environment that can utilise a network of manufacturers to produce "customised" products [43,112, 113].

Figure 2.1 illustrates how shorter product lifecycles are demanding faster design, build and ramp up of manufacturing machinery. Figure 2.1a illustrates a traditional sequential manufacturing approach [112] where the initial development of a product, is followed by the design and build of the manufacturing machinery. The manufacturing machinery is then ramped up to its full production capacity where the product is then produced [112]. However, current global market pressures demand the rapid introduction of new, upgradeable, customised products and enterprises with long lead times can no longer remain competitive [145]. Machinery must be quickly ramped up to full production capacity and readily reconfigured to meet the manufacturing requirements of new products (see figure 2.1b).

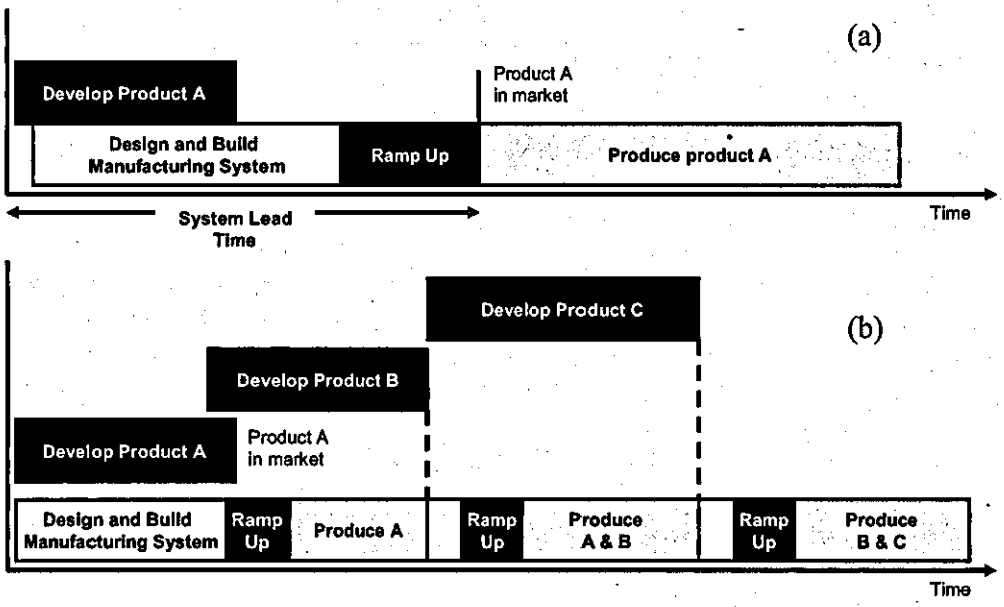


Figure 2.1 Shorter product lifecycles demand faster design and build, ramp up of manufacturing machinery [112]

2.4 Reconfigurable Manufacturing Systems

Reconfigurable manufacturing systems have been conceptualised and developed [43-45, 47, 48, 143-148 and 150] in response to the emerging market oriented manufacturing environment. Reconfigurable manufacturing systems have a modular structure (both software and hardware) that allows ease of reconfiguration as a strategy to adapt to changing market demands [146, 148]. Modular machines and open-architecture controllers [116] are the key enabling technologies for reconfigurable manufacturing systems and have the ability to integrate, remove new software / hardware modules without affecting the rest of the system [148]. This enables reconfigurable manufacturing systems to be quickly adapted to the production requirements of new products and the exact capacity requirements as the market changes [29].

A machining system can be created by incorporating basic process modules, both hardware and software that can be rearranged or replaced quickly and reliably [147]. Reconfiguration allows adding, removing or modifying specific process capabilities, controls, software, or machine structure to adjust production capacity in response to changing market demands or technologies [147, 148]. This type of system provides customised flexibility for a particular part-family and will be open-ended, so that it can be improved, upgraded, and reconfigured, rather than replaced [145, 146 and 148]. A reconfigurable manufacturing system provides the exact functionality that is required in the product manufacturing process and achieves [145], 1) reduction of lead time for launching new systems and reconfiguring existing systems, and 2) the rapid modification and quick integration of new technology and new functions into existing systems [148].

Reconfigurable manufacturing systems have six major characteristics, which are [143, 144, 145, 146, 147 and 148]:

1. Modularity: All the major components, for example mechanical structural elements, controls systems, software and tooling within a reconfigurable manufacturing systems are modular [143, 144, 145, 146, 147 and 148].

2. Integrability: The machine system is engineered from a pre-designed set of modules using established system configuration and integration rules [143, 144, 145 and 148].
3. Customisation: Reconfigurable machine systems are configured to meet the requirements of a whole part family by utilising the concepts of customised flexibility and customised control. Customised flexibility means that the dominant features of the part family being manufactured will determine the overall machine configuration and system configuration. Customised control is achieved by integrating control modules with the aid of open-architecture technology, providing the exact control functions needed [143, 144 and 146].
4. Convertibility: Reconfigurable machine systems allow easy conversion between parts, as well as sensing and controls methods that enable quick calibration of the machines after reconfiguration [143, 144, 146 and 148].
5. Scalability: Reconfigurable machine systems are highly scalable, for example a machine may require addition of spindles to increase the productivity or adding machines to increase overall system capacity [144, 145, 146 and 148].
6. Diagnosability: Information relating to the machines status and manufacturing process is disseminated using web based technologies in the form of web based HMI's to ensure the organisations personnel can detect unacceptable part quality. This is critical in reducing ramp-up time for reconfigurable manufacturing systems [144, 145 and 148]. This aspect of the reconfigurable machine system is the primary focus of the research described in this thesis.

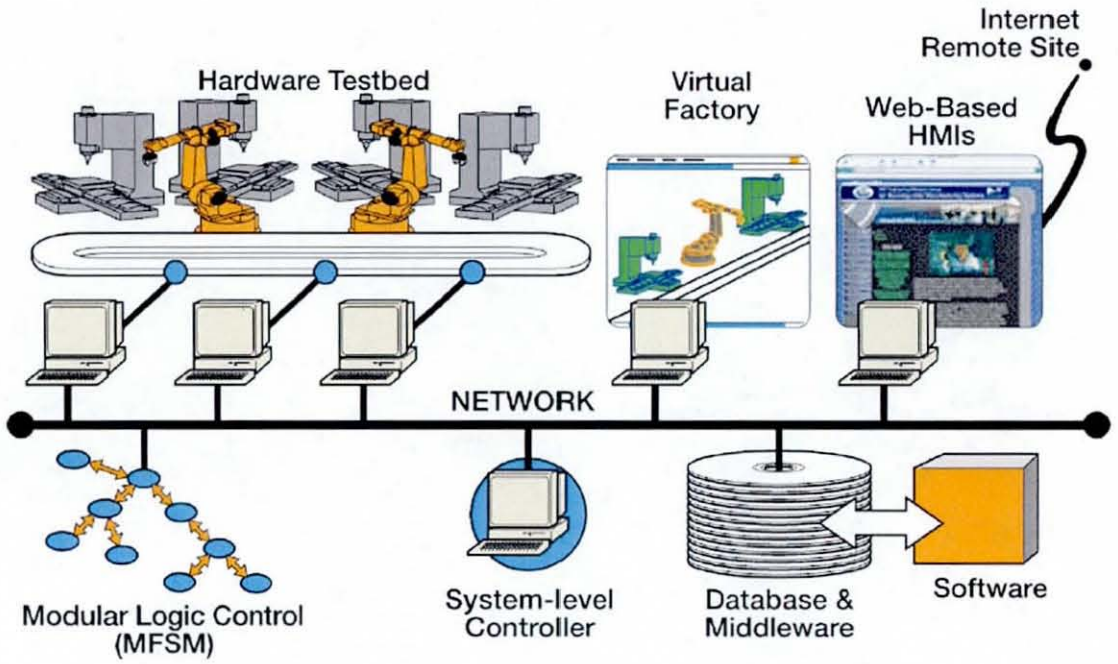


Figure 2.2 Primary components implemented within a Reconfigurable machine system [148]

The concept of a reconfigurable machine system has been implemented by Engineering Research Centre at the University of Michigan. Their implementation consists of both real and virtual machines that are controlled over a communication network and coordinated through the systems unified software architecture [148]. The primary components that are implemented are illustrated in figure 2.2 and described below.

1. Hardware Testbed: A machining system as been implemented that has two manufacturing machines. Machines operate in parallel and parts are transferred too and from them via a conveyor.
2. Virtual Factory: Factory simulation software simulates both the real machine and virtual parts that do not exist in hardware. The virtual factory is controlled in the same manner as the real hardware.
3. Database and Middleware: The data-centric software architecture [147,148] connects all the aspects of the reconfigurable machine system to facilitate the rapid prototyping, integration, complementary utilisation and transfer of newly-developed software solutions [143, 144, 145 and 148].

4. Remote Viewing and Collaboration Tools: These tools are implemented in the form of web based HMI's that provide users with detailed information about the machine and its status. This enables operation from any remote location with an internet connection.

The HMI within a reconfigurable manufacturing system consists of a web based user interface that is considered [144, 145 and 146] to be key in supporting effective machine diagnostics, which is one of the six major characteristics of a reconfigurable manufacturing system [144, 145 and 146]. The web-based distribution of manufacturing machines information is rapidly becoming a key requirement [144 and 147] to enabling capabilities such as remote machine control and diagnostics [18]. An example of a reconfigurable manufacturing systems remote HMI capability is shown in Figure 2.3. The reconfigurable manufacturing systems framework contains an open source architecture which supports a secure web environment for external partner's user management and user interface software to be integrated [148].

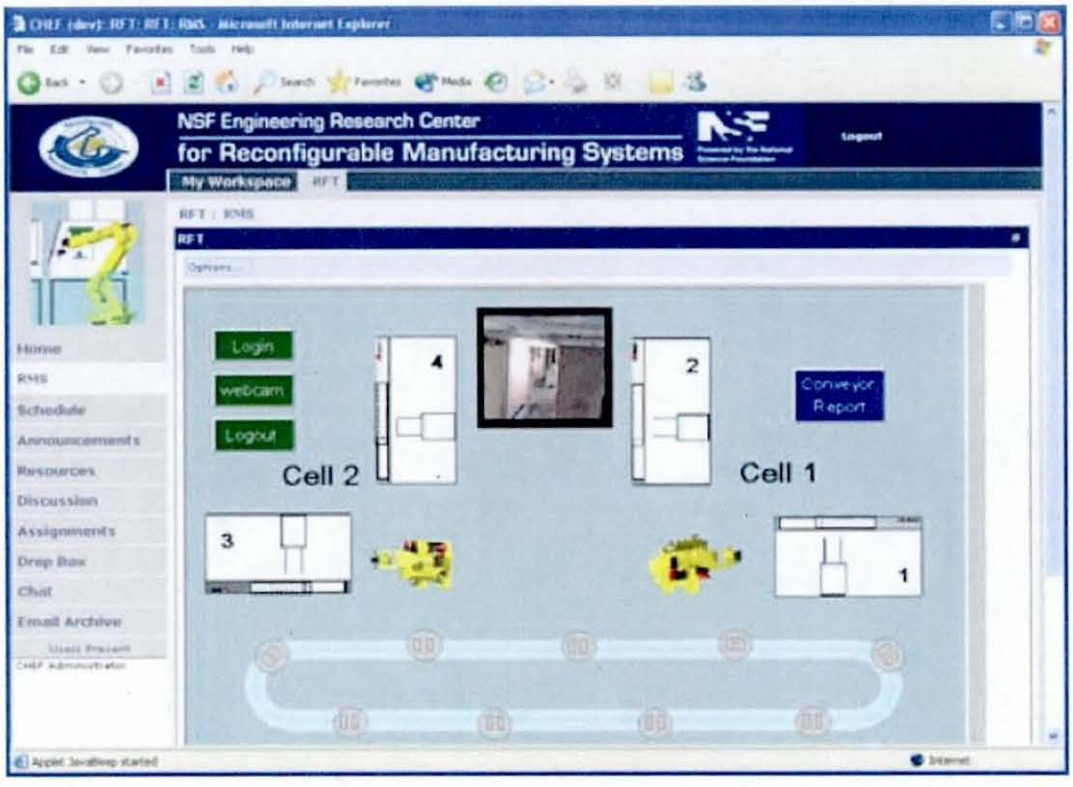


Figure 2.3 RFT Remote Web-based User Interface Capability [148]

The major research activities related to the HMI aspect of reconfigurable manufacturing systems [150] that have been defined and conducted by the University of Michigan are;

i) Mechanism for factory-wide distributed HMI infrastructure,

ii) Mechanism for multi-software component dashboard solution (removing the one-to-one correspondence between a machine and the HMI),

iii) Mechanisms using remote monitoring for machine control,

iv) Reconfigurable manufacturing system remote web-based HMI capabilities.

The focus of the research activities [150] is on the machines run-time data and the communications infrastructure to support getting machine information disseminated from the machine to the users via standard HMI's. Furthermore it is stated that [148] *"an open source framework solution provides a secure web environment for external partners user interface software to be integrated within the reconfigurable manufacturing systems"*. This implies that an external partner user interface tools are adopted within the system and the research is limited to the integration capabilities using an open source framework solution provides a secure web environment. This suggests that limited research has been conducted in the modelling, subsequent development and engineering of the HMI for particular users. The literature survey now focuses on user interface modelling and implementation technology which is not dedicated to manufacturing machine applications.

2.5 User Interface Modelling

It is stated [121] that four major types of declarative models (presented in table 2.1) are required to describe the aspects of a user interface. Each aspect partly affects the process of engineering a user interface. In this section, each model is described at a superficial level and their associated attributes are briefly discussed. The objective of modelling the user interface is to provide a notation that describes the end users functionality and details the user interaction process [122]. User machine interaction is dependant on the machines behaviour, the operational goals (what the user must do), the users model of the machine (what they expect to happen, their assumption, knowledge of how it should operate), the user interface (the system which supports the user interaction with the machine) [120]. A well documented concise specification using various modelling notations is therefore required to ensure consistency in the end users requirements being meet in the design and implementation of the user interface.

Table 2.1 Components in User Interface Modelling [120, 127]

User Interface Guideline	Guideline attributes
Task model	A task model describes how users do their tasks in a certain application. The process of determining the task model is known as Task Analysis [124]. It contains the task structure and the order of interactions between user and system [123].
Dialogue model	Dialogue models contain such information as to which objects exist in the user interface and what are their possible states. The actions that the user may initiate through the user interface, as well as the reactions that the application may execute via those elements are represented [123]. State transition diagrams are commonly used in to represent dialogue models [126]. Some models combine tasks and dialogues into one model, which is usually called a task-dialogue model [125].
User model	A user model defines the attributes and roles of users and it can be used to provide a way to model user interface preferences for specific users or roles [123]. User models are described vaguely in literature and are present in very few user interface implementations [122].
Presentation model	Presentation models represent the visual, navigational elements provided to the user by the user interface. The presentation model is basically a static collection of elements [123], but attaching stylistic properties, such as colours and font size, to the user interface is also considered to be a part of this model.

2.5.1 Task Models

Norman and Draper define the practice of designing products to meet their user's needs as user centred design [117]. Producing usable user interfaces, for example undertaking user-centred design, requires a thorough understanding of the underlying goals of the users [118]. Within the Human-Computer Interaction (HCI) community, task analysis is considered to make an important contribution to the design of interactive applications [124]. This is due to the fact that it fundamentally is about designing user interfaces by first understanding the user's tasks between domain experts and UI developers [127].

Describing user interfaces has traditionally started by describing the static visual interfaces with certain structures and controls, often called widgets [128, 129].

However, it has widely been suggested that this is an ineffective starting point. Designers must think in terms of functionality [117]. Starting the design by specifically thinking about

the user's task and especially the steps (subtasks) that the user must take in order to get the task completed is considered to better support the idea of user-centred design [117]. If the flow of the task and its subtasks is clear, it is then easier to choose the right layout and widgets to give the user concrete tools to complete the task [130]. Providing highly task specific interactive applications that allow people to focus on the actual task domain, rather than having to map that domain to the domain of user interface technology, is the underlying idea of task-based user interface design [118].

Use cases have commonly been used for describing the requirements to satisfy the needs of the user. Yet, the definition for a use case [133] is commonly considered vague.

Use cases are a generic modelling notation concerned with the interaction between the user and the user interface [132]. Use cases are represented as ellipses, and users (actors) are represented as icons connected with solid lines to the use cases they interact with. Such a relation is called a users relation and is represented by a directed dashed line. One use case can call upon the services of another use case achieving a hierarchical order to the system design. The direction of a uses relation does not imply any order of execution. As an example to illustrate the process figure 2.4 shows a web site use for project administration used within an organisation [132]. The use case contains two users, a web user and an employee. The web user tasks and interaction with the system is to visit projects. The employee tasks and interaction with the web site is add projects, remove projects, edit projects and also visit projects. This simple example illustrates that the users tasks and interaction that is performed in a user interface system can be defined using a use case notation. The use case is the output of the task analysis but is does not necessarily capture the user's task flow [131].

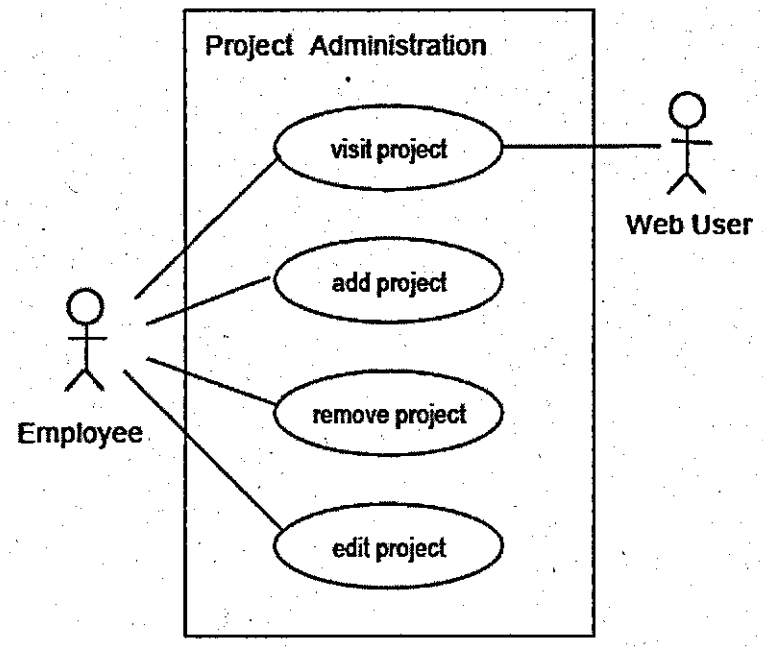


Figure 2.4 Use Case that describes a user interface for a project management application [132].

2.5.2 Dialogue Model

A dialogue model describes the particular objects within a user interface and their possible states. The actions that the user may initiate through the user interface, as well as the reactions that the application may execute via those elements are represented [123].

State diagrams are commonly used capture the states of either a complete or particular aspect of a user interface system [125]. A major limitation with state diagrams is that they suffer from a combinatorial explosion in the number of states and transitions as system complexity increases. Jacob [16] solved part of this problem by allowing co-routines to represent a number of state diagrams for a particular aspect of the interface being modelled. Many forms of state diagrams have been created with special notations to fit the needs of specific applications areas such as e-commerce, and word processing. This technique only provides a model of the action (user input) and the result (interface states) and does not provide a information relating to the visualisation or layout of the user interface.

2.5.3 User models

The user model describes the characteristics of the different types of users. The purpose of this model is to support the creation of individual and personalised user interfaces at design time [121, 127]. Firstly, it defines, for each kind of user, the set of tasks he/she can perform. Secondly, for each kind of user in relation with a concrete task, a projection on the actions within the task that he/she can perform is established [121].

Finally, depending on the user's particular characteristics (user skill level, experience), the information and the interaction established by the dialog model to show the information contained in the domain model is adapted to the user [121].

2.5.4 Presentation Models

Research into current state of the art found that although user interfaces represent an essential part of software systems, software modelling techniques such as UML seems to have been developed with little specific attention given to user interface issues [132, 133]. User interface stakeholders and their requirements, user interface software components and their states can be modelled using the UML notation together with how the system reacts to user interface commands but there is no support within the UML notation to specify the layout of user interfaces [134].

Researchers at Ludwig-Maximilians-University in Munich [97] have defined extensions to UML notation to model the navigation and presentation for user interface that is targeted for web based applications. They propose to use a UML collaboration diagram which consists of one or more rectangles that are objects within the modelled system, and in the context of user interfaces they are screens. The rectangles are connected by lines that indicate the allowable links between the objects which are navigational paths within the context of user interfaces.

Firstly in the process of designing a web application, the web designer proposes a sketch of each main user interface view. These are rough drawings of a couple of relevant elements of each navigation node. This sketching technique is frequently used by web designers, but without having a precise notation for it as described by Sano [99, 97].

Storyboard models [135, 136, 137, 138 and 139] only define the structural organisation of the presentation given by interface objects. Interface objects are typically texts, images, forms and menus, and not the layout characteristics, in terms of fonts, colours, special formats which are defined at the implementation stage of the user interface development. The storyboarding user interface model still may provide hints on the position and the size of the interface objects relative to each other.

After designing the different user interface views they are combined using navigational links in the storyboarding model to show sequences of the user interface in the order in which a user can navigate from one view to another [99, 100]. It is claimed [97] that this aids in visualising the organisation of the web application structure in a more intuitive manner than using standard UML notation. It is also claimed that the storyboarding models are a useful means for the communication between the end users and user interface web designers, enabling them to be validated with the use cases identified during an analysis phase of the user interface development. An example storyboard model is illustrated in figure 2.5.

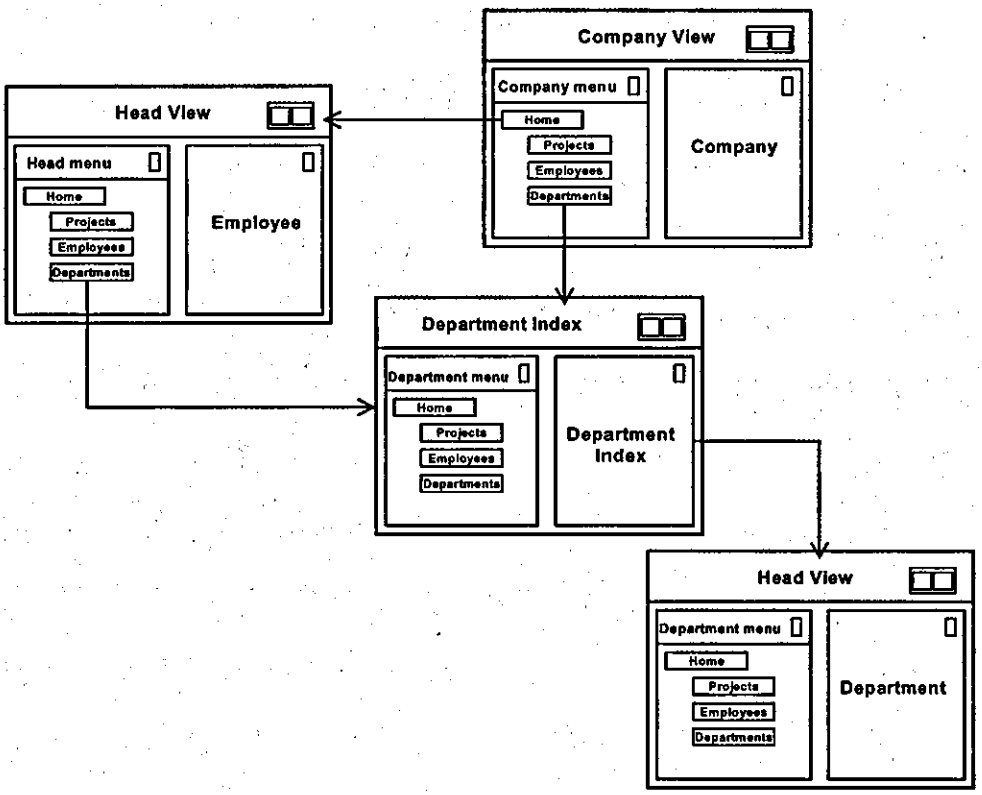


Figure 2.5 Example storyboard for a head of department [97]

2.6 UI Systems Engineering - Combining the Model Components

Some proposals have arisen trying to contribute new ideas in this field. Mayhew proposes a Usability Engineering Lifecycle [121] in which the lifecycle structured in three stages: 1) requirements analysis, 2) design/testing/development and 3) installation [140]. This development process follows the waterfall lifecycle and therefore it cannot be considered as an iterative approach. The sequential manor of the approach, starting with requirements analysis, then the second stage of design, testing, development and the final process with the installation does not return to requirements analysis [140]. This methodology is focused on user interface software development and not user centred designed [121].

Constantine and Lockwood [141] propose a user centred design approach can be considered as a method that describes the techniques to use more than a development process as defined in software engineering practices [121].

User centred design consists of a set of usability-oriented coordinated activities [121, 117, 122, 125]. They include task analysis and user modelling (see section 2.5.1). The user centred design process [121] consists of three main processes, 1) user requirements, 2) analysis, 3) design and implementation and 4) user interface evaluation. The result of this process is the user interface final system. The fundamental concept of applying user centred design to user interface engineering is that it is an iterative process where the task 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. Shneiderman [96] states that, "usability engineering" is an iterative design method which is a fundamental basis in reaching a high quality user interface designs. The method allows feedback to be performed on each development process by introducing the information collected in the tests performed by the users in the user interface usability testing [121]. This way, the usability of the designed interfaces improves notably [121, 96]. Figure 2.6 illustrates this iterative user interface design approach.

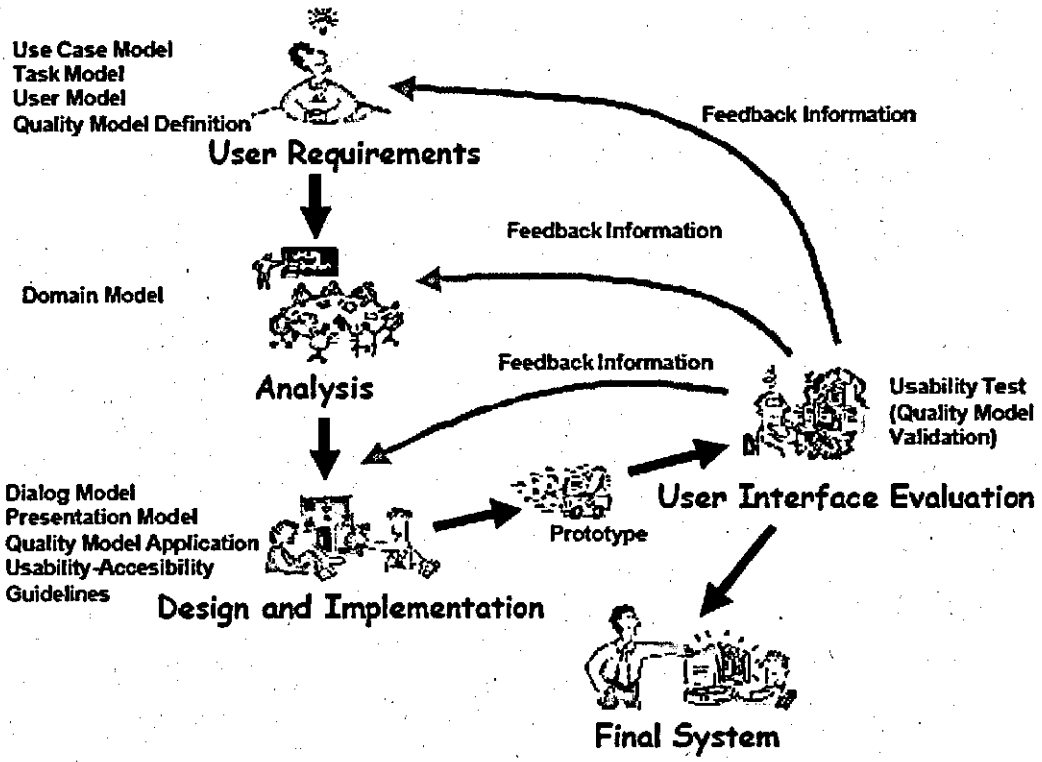


Figure 2.6 Iterative development process which is fundamental to user centred design approaches [121]

2.7 User Interface System Implementation

In this section characteristics of the different tools used to implement user interfaces are described. There are many different implementation tools that can be used to meet the desired functional requirements user interface.

Much effort within the research community is made towards the development of improved tools to support the user interface design and implementation, maintenance and evaluation. Collectively these tools have come to be known as user interface implementation tools.

The user interface implementation tools are a set of tools that are used to support managing, administering user interface development. The two main categories of user interface implementation tools are; 1) model based user interface development and 2) user interface design tools which can be divided into two further categories, user interface application tools or user interface toolkits.

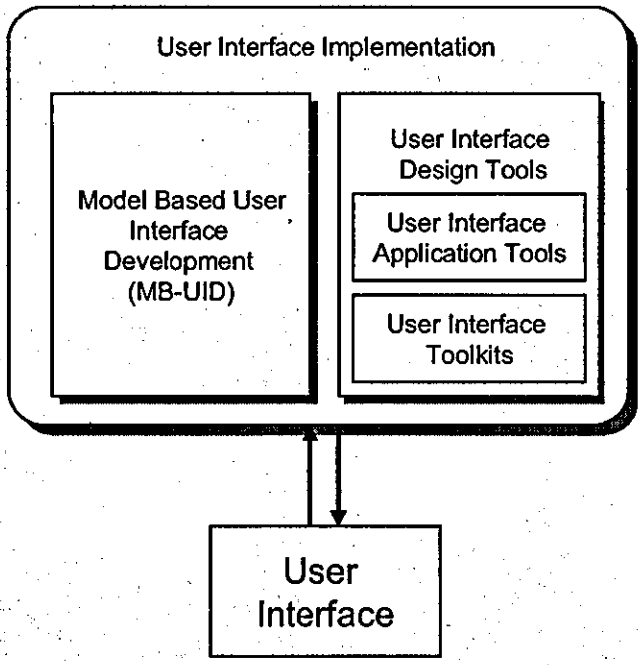


Figure 2.7 User Interface Implementation Context – A set of tools that supports the design, implementation, maintenance and evaluation of the user interface [96].

2.7.1 Model Based User Interface Development Approaches

A promising approach to the construction of HMI systems during the 1990's has been Model Based User Interface Development (MB-UID) [122, 123 and 149]. Using this approach the user interface is automatically generated from a series of user interface models. This approach aims to support all relevant aspects of the user interface in formalised models that represent the different requirements of each context of use of the HMI system.

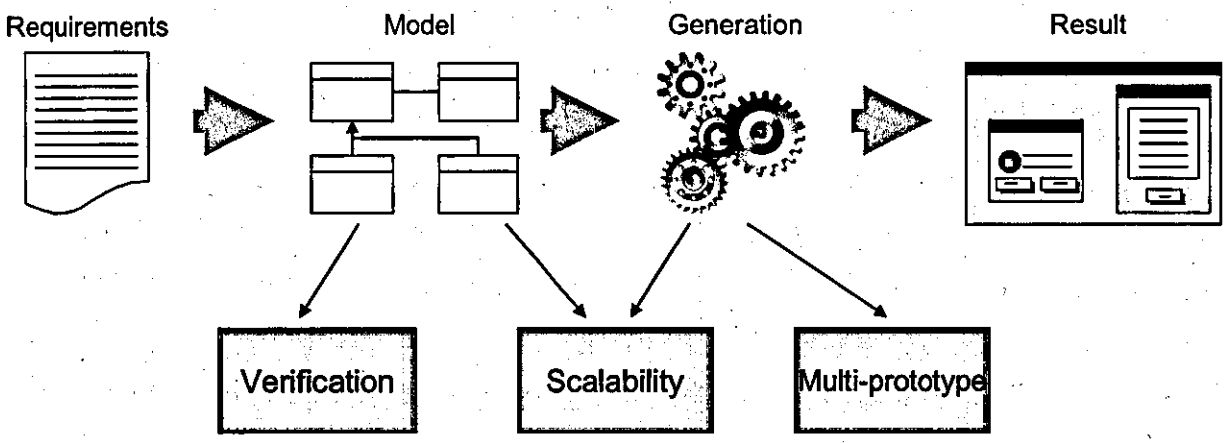


Figure 2.8 Generalised Model-based User Interface development approach [96]

The process of developing a user interface within a MB-UID is an iterative process of developing and refining a set of declarative models using graphical editor tools or high-level specification languages. Once developed, these models are transformed according to ergonomic rules and/or style guidelines into an interface specification. This specification is subsequently linked with the underlying application code to generate a running application [11].

The following attributes are associated with the MB-UID approach [76, 122, and 123];

1. Higher level of abstraction for user interface development allowing the designer to concentrate on the user interface design issues and not the lower level system code generation [76, 136].
2. Support a well structured development method that the user interface designer must follow. This enforces best practices that the designer must follow, for example

applying ergonomic design rules and using formal notation to specify the interface ensuring there is not ambiguously is the requirements [137, 139].

3. Quicker to develop user interface systems due to the lower level implementation being automated by the systems infrastructure. This allows the designer to develop user interfaces without the need for software engineers [76, 122].
4. The user 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 [76].

Shortcomings that are associated with these types of systems are;

1. With user interfaces increasing in functionality the complexity of the models and their notation required is considered as a major limitation of these systems [76].
2. Difficulty to model relationships between models (mapping problems) [122].
3. Mostly only support a very simple user interface system, typically these are form based and these are one reason why MB-UID has not been adopted within the commercial sector [76].
4. The scope of the user interface is often very limited using these systems and does not support a broad enough scope to be applied successfully in commercial areas [76].

2.7.2 User Interface Design Tools

User interface design tools are defined by Shneiderman [96] as a *"software development suite that facilitates the implementation of a user interface by programming which maybe graphical or textual based"*.

Developer centred environments which are typical of most commercial user interface systems give support to using and managing software elements and organising, arranging layouts and testing prototype interfaces but fall short of answering key questions such as how user interface components can be used to accomplish a particular end user task. The designer is

forced to rely on their own experience to answer such questions or to distil a solution from loosely connected documentation.

User Interface Design Tools can be divided into two different types; 1) User Interface Toolkits and 2) User Interface Application Tools. The characteristics of these two different types of design tools are detailed in table 2.2.

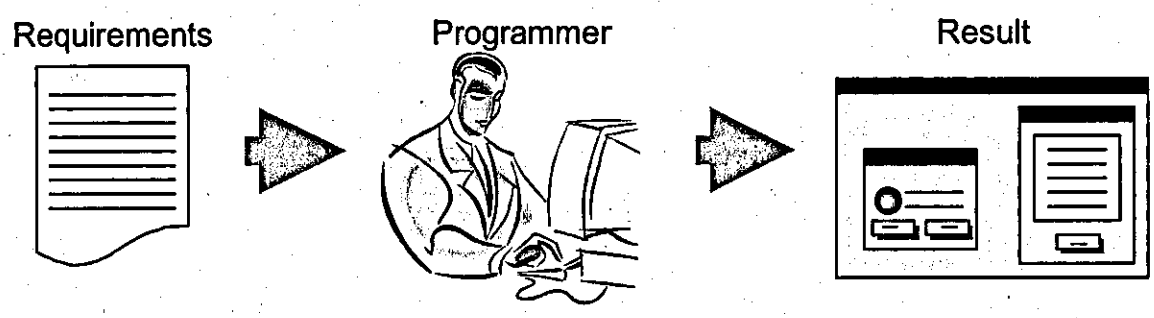


Figure 2.9 User interface development process adopting User Interface Design Tools [96]

Table 2.2 Different types of User Interface Design Tools [96, 127 and 134]

Type	Description
User Interface Toolkits	+ Greater flexibility in user interface design + Wide range of end applications supported highly skilled programmers - Contain software libraries with widget controls
User Interface Application Tools	+ Low skill designers + Faster development + Targeted at particular end applications e.g. web pages - Less control of user interface application

User Interface Toolkits

User Interface Design Toolkits use software libraries that contain common widgets such a windows, scroll bars, pull down menus, data entry fields, buttons and dialog boxes [96]. Programming languages with these common widgets can provide great flexibility in the user interface design. However familiar experienced programmers are required to develop user interfaces that can become very complex and require months of learning for programmers to gain proficiency. The advantage is that programmers have extensive control and great flexibility in creating the user interfaces [96].

User Interface Application Tools

User Interface Application Design Tools provide more constraints than toolkits [96]. Using these design tools user interfaces are developed for a particular application area which maybe for example; 1) web browser HTML user interface page or 2) a user interface is designed for a 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 development and very little or no programming is required which supports lower skilled designers and promotes to faster development time [96].

2.8 Literature Analysis

In this chapter, the major challenges and demands faced by the manufacturing community have been derived from examining current literature. To summarise, the manufacturing community faces demands to achieve faster design, build and ramp up of manufacturing machinery [29]. The onset on globalisation is affecting the way companies do business, this global marketplace is enabling greater product variety which in term in creating shorter product lifecycles. The time to market for new products is a critical metric in companies remaining competitive. The manufacturing community faces demands to achieve faster design, build and ramp up of manufacturing machinery [29].

Reconfigurable manufacturing systems are a [145, 146 and 147] response to these emerging demands and challenges in engineering manufacturing machinery. Reconfigurable manufacturing systems are highly modular and can be rapidly reconfigured to suit frequent product manufacturing process changes. Reconfigurable manufacturing systems can be remotely monitored to provide more efficient machine diagnostics. Although reconfigurable manufacturing systems specify that web based HMI's are key enablers, they only support the integration aspect of existing HMI systems through their open source web based framework. No published literature within the domain of reconfigurable manufacturing systems suggests that modelling and subsequent engineering of HMI's that address the needs of particular users has not been undertaken. Furthermore with manufacturing systems machinery being more frequency reconfigured, the associated HMIs have a requirement to be rapidly created and deployed.

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

Modelling the user interface provides a notation that describes the end users functionality and details the user interaction process [122]. It is considered that there are four major components in modelling a user interface [127], 1) task model, 2) dialogue model, 3) user

model and 4) presentation model. The goal of task models is to define the user's task and the steps (subtasks) that the user must complete in achieving their task. Use cases are commonly adopted as the notation in building these models. The second major component in user interface modelling is dialogue models. These contain information that describes the objects or widgets that exist within the user interface are their possible states. State transition diagrams are predominately adopted as the notation for this model. User models define the tasks that the user performs dependant of their characteristics such as experience and skills. The presentation model describes the screen layouts and navigational structure of the user interface. Although informal methods are used in this model such as sketching, storyboarding is a technique where a series of screens and the user interface widgets are created and the navigation between the screens is depicted by arrows. This is considered to provide users with a "walkthrough" of the user interface system.

New systems engineering methods are emerging that provide an iterative process to the design, development and implementation of user interfaces. The usability engineering lifecycle [121] has three stages: 1) requirements analysis, 2) design/testing/development and 3) installation [140] which are performed in a sequential manner and therefore follow the engineering workflow of waterfall methodologies [121]. The user centred design approach consists of a more iterative process where the user interface is evaluated and fed back into the user requirements, analysis and design and implementation stages. Shneiderman [96] suggests that this notably increases the usability of the developed user interfaces.

From the user interface implementation aspect, model based user interface development [122, 123 and 149] have emerged from well established research groups worldwide, in order to overcome the limitations of their own former development techniques [123]. Model based user interface development provides user interfaces that are directly created for one of more user interface models that specify the functionality. The model based user interface development approach automates the highly skilled lower level implementation aspects of the

development process. This allows the designer to develop an HMI without the need for specialist software engineers.

3 Research Design and Focus

3.1 Introduction

In this chapter the research design and focus into a new approach to engineering HMI systems that better support issues associated with global manufacturing is described. The HMI system requirements are derived from the global manufacturing needs together with the limitations of current industry best practice. This provides the demands for a new approach. For example the ability to have HMI systems that can be rapidly developed to support more frequent re-configuration of the manufacturing machine. State of the art from the research community in terms of user interfaces modelling and development approaches are applied to the domain of HMI systems to establish a new approach.

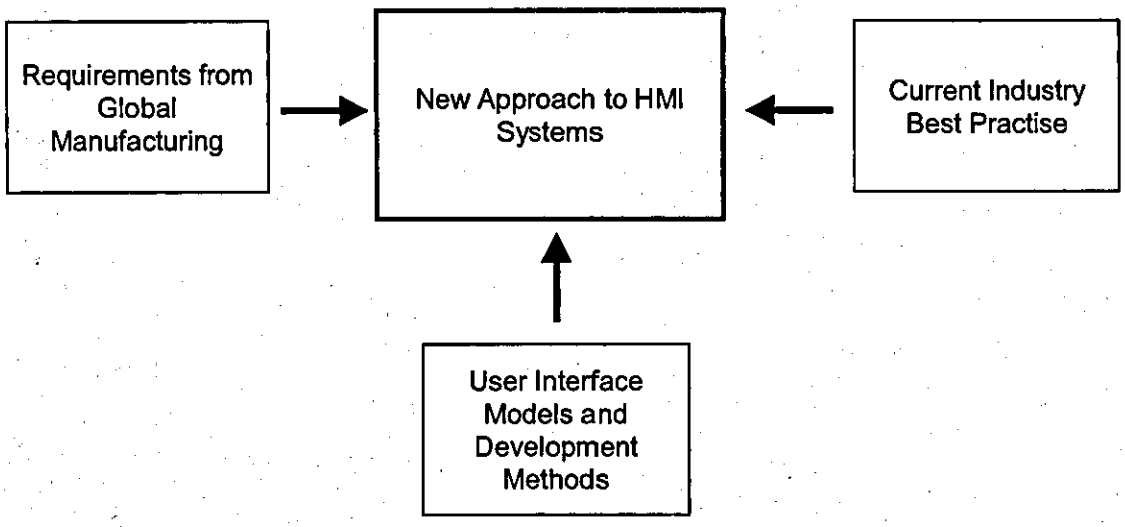


Figure 3.1 Problems areas that are appropriate for studies into a new approach to HMI systems within automated manufacturing production machinery.

3.2 Current Industrial HMI Practice

The lifecycle of manufacturing automation machines from a HMI systems perspective is illustrated in figure 3.2 [8, 88, 89 and 90]. The diagram has been determined from interviews with a number of industrial collaborators within research conducted at Loughborough University [88, 89 and 90]. Engineering large scale production machines consist of many different processes in each phase of a machine's lifecycle. Seven phases have been identified in the entire lifecycle of a machine with each phase creating unique requirements for HMI systems [8].

At the initial concept phase of the machine's lifecycle, end users have a requirement for a new manufacturing automation machine to address a particular set of business needs for example; 1) Increased production capacity, 2) New product manufacture and 3) Product manufacturing process changes [8, 106]. At this stage no consideration is given to the human interface aspect of the new production machinery [22]. During the next lifecycle phase the machine's specification is formalised which involves detailing the behaviour and logic of the machine together with the number of HMI stations[1] required and their locations. The stakeholders who are involved in producing this specification are end users and machine builders [86]. The end users HMI functional requirements are defined to meet their process objectives and machine builders provide information regarding the practical constraints of the machines and technologies available for the implementation based on knowledge acquired through previous machine applications. The HMI specification is defined by manually drawing a schematic of each screen and detailing each button or control of every HMI screen in the system [22, 86]. This process cannot begin until the machine has been designed as the buttons relate specifically to the particular machine's configuration.

When the requirements specification has been agreed and described it is approved by both end users and machine builders and then the next phase in the lifecycle, HMI Design and Development may begin. Due to the inherent characteristics of traditional approaches to HMI

1 HMI Station is a term used within manufacturing industry to describe the system that the machine operator interacts with to control a manufacturing machine. In this example the HMI Station is a PC based machine that has a colour touch screen panel and interfacing PC cards to the controlled machine.

systems implementation, the 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 Programmable Logic Controller (PLC[2]) system [22]. The traditional sequential manner in which these tasks are performed contributes significantly to the machine's development time – a critical metric in today's global manufacturing environment [8, 109, 110]. Due to the translation that is required between the process engineers, who define the requirements, and the HMI programmers, who implement the systems, problems can often occur for example:

1. Installed systems cannot easily be changed or modified on an incremental basis by process engineers at a later date without involving a highly specialist skilled HMI engineers who implement the HMI software [22].
2. The usability and suitability of the operation of the new HMI systems remains largely unknown until the physical machine has been built, the controls wired up, the HMI software implemented and the system finally tested [22, 86].
3. HMI system development tools are not integrated with production machines process and control logic (program within the machines PLC), therefore often the same information is input many times into different systems [8, 22, 86].

It is not until the machine has been installed and commissioned at the end users plant that the machine operators can be trained on the HMI systems usage. 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 [88].

The manner in which these machines are engineered is a tradition top down approach that does not support simultaneous engineering programmes, is not reversible and often errors cannot be found until the machine is completely built.

2 A PLC is a microprocessor based device with either modular or integral input/output circuitry that monitors the status of field connected "sensor" inputs and controls the attached output "actuators" (motor starters, solenoids, pilot lights/displays, speed drives, valves, etc.) according to a user-created logic program stored in the microprocessor's memory.

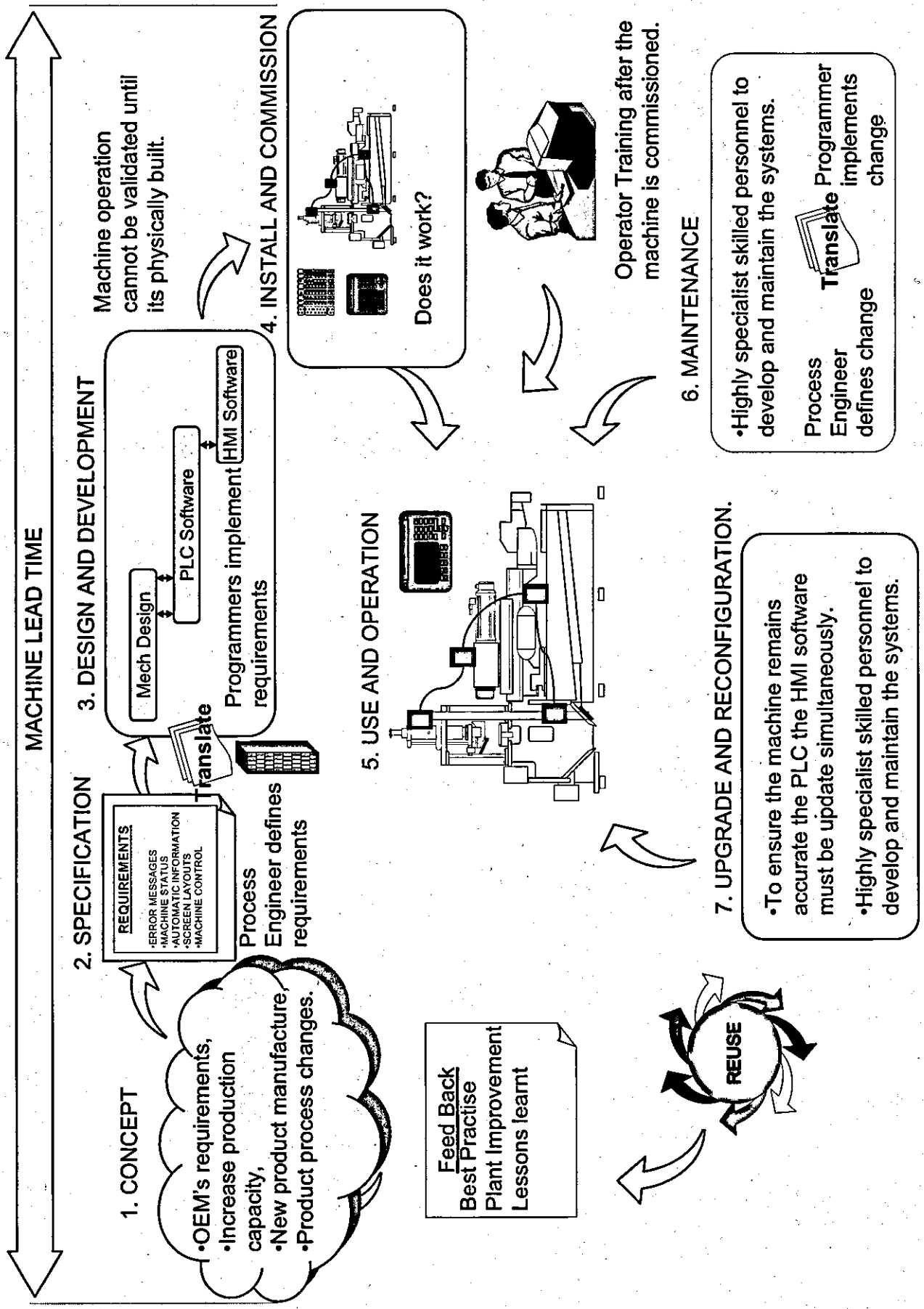


Figure 3.2 Lifecycle of HMI systems within production machine applications [8]

3.3 Requirements and Emerging Trends in HMI Systems

The previous section describes industry current best practice and section 2.3 (chapter 2) describes the challenges in engineering manufacturing machinery. The key enabler from global manufacturing is that the manufacturing system can be designed / built and re-configured in the fastest time to support shorter product lifecycles [43, 109, 110, and 111]. Early verification of the HMI system is a key attribute to enable rapid machine design and ramp up to full production capacity [8, 22, 86, 88, 89 and 90]. The limitation with current industry practice is that the HMI system cannot be verified until the machine is fully built and therefore operators cannot be trained on the system until the final stages of the machine development (see table 3.1 no. 2) [86]. The HMI system needs to be validated and fully functional in the early stages of the machine design to allow operators to be trained on the machines operation and any errors / inconsistencies and change requirements identified in the lifecycle [22, 88].

Currently HMI systems provide limited support for remote diagnostics of machine errors. With engineering partners becoming global distributed it is increasingly inefficient for machine engineers to visit and diagnose the machine error at the machines site [88]. Future HMI systems (see table 3.1 no. 3) must enable remote monitoring and error diagnosis to increase machine up time.

HMI systems currently offer poor change capabilities (see table 3.1 no. 4) due to the lack of integration between the different systems within the machine control system (see table 3.1 no. 6). For example, the machine controller which typically would be a PLC device, must have its software program updated consistently with any changes to the HMI program, otherwise there is potential for inconsistencies between the two software programs, leading to inaccurate machine information being displayed to the machine user [22, 86]. A machine interlock change for example would require a change to the HMI software to display the new machine interlock messages together with the machine controller. The current approach where machine data is fragmented across multiple systems within the machine system leads

to longer machine re-configuration times [22]. Machine changes applied to this fragmented system are increasingly susceptible to errors leading to inaccurate HMI systems and further errors / downtime.

There are currently no standardised approaches for the end user specification of the HMI system (see table 3.1 no. 5) [9, 12, 22]. A well defined method of specifying the user interface functionality is critical so the HMI has the right information for the targeted user [22, 86].

Currently HMI systems are developed using proprietary user interface toolkits that cannot be readily transferred to other vendor's machine control systems [9, 10 12]. Industry pressure to use the best machine software products from many different vendors [22, 86] create a requirement where the machine integration platform must allow different vendors machine components to be readily integrated. The current development process relies heavily on the expertise (see table 3.1 no. 7) of those who have participated in previous similar projects [22, 86]. There is a drive for HMI Systems to encapsulate best industry practice so machines become standardised throughout different global plants [17, 18].

Table 3.1 Current Limitations and the Requirements from Global Manufacturing

No.	Current Limitation	Description	Requirement
1.	Late HMI system verification	The HMI system cannot be tested until the machine is fully built.	HMI systems must be validated early before the machine is built to achieve faster machine ramp up time.
2.	Late operator training support	Machine operators cannot be trained on the operation of the HMI until the machine is fully commissioned on the end user site.	Machine operator must be trained in early stages or able to be transferred directly from other machine with common HMI.
3.	Lack of remote diagnostics and support	Engineers must visit the machine plant to diagnose errors and support the machine.	Globally distributed engineering partner must be able to increase machine up time by remotely diagnosing and supporting the machine.
4.	Poor system change capabilities	Fragmented systems mean that large efforts are required to support changes to the machine.	Manufacturing production machines must be quickly re-configured to meet the process requirements of new products.

Table 3.1 continued

5.	Lack of common standards for specifying HMI system	Specification of the HMI system is not formally described.	A well defined method of specifying the user interface functionality so the HMI has the right information for the user.
6.	Lack of integration between associated machine systems	HMI Systems are developed using proprietary user interface toolkits that cannot be readily transferred to other vendor's machine control systems.	A system platform that allows the integration of different vendors machine components.
7.	Experience based design	The development relies heavily on expertises of those who have participated in previous similar projects.	Encapsulating best industry practice in the system so machines are standardised throughout different global plants.

3.4 Component Based Paradigm

Component based software development is a relatively new approach within the software community [101]. The main idea is to reuse pre-developed and tested components instead of developing everything from basic principles each time [102]. Use of component based development claims to bring many advantages such as faster development, lower cost of the development, better usability and encapsulation of best practise [101, 102]. Component based development is however still not a mature process and there still exist many problems, for example granularity of components, standardised interfaces between interconnecting components and configuration processes and management of components.

The motivation behind the use of components with the context of industrial automation systems was initially to reduce the cost of development, but it later became more important to reduce the time to market, to meet rapidly emerging consumer demands [24]. When new components are introduced in a system, new issues must be dealt with which may include for example dynamic configurations supporting component variants and providing scalability [38].

Component based software development provides methods, models and guidelines for the developers of component based systems [74]. Component based development denotes the development of systems making considerable use of components [74]. Although very promising, component based software engineering is a new discipline and there are many associated problems which remain unsolved. Many solutions can be arrived at by using principles and methods from other engineering disciplines.

The basic idea is that new systems are composed from components that are already have been developed, tested and validated. Components are developed with specific functionality and are designed to have a wider applicability. This discipline of component based software engineering is applied here to the development of HMI systems for manufacturing automation systems with the aim of meeting the requirements industry faces from global manufacturing demands.

3.5 Application of Component Based Paradigm to Manufacturing Automation Systems

The goal of the component based automation paradigm is to fully support the lifecycle requirements of automated manufacturing machines. The development and implementation of component based machines has been undertaken over a three year period by the MSI Research Institute, Loughborough University through research projects *"Assessment and Implementation of a Component Based Paradigm for Agile Automation"* (COMPAG) and *"Common Model for Partners in Automation"* (COMPANION) and *"Global Engineering of Manufacturing Machinery"* (GEMM) funded by the Engineering and Physical Sciences Research Council (EPSRC). These research projects developed and prototyped a novel method for engineering manufacturing machine's using a C-B approach in response to industry's needs to meet the demands of rapidly changing markets. A C-B Manufacturing Machine is inherently flexible such that it may be cost effectively converted, quickly and reliably as market demands change.

The fundamental concept of this paradigm is to compose complete automated machines from modular machine components.

The major elements of component based automation are; 1) an Integrated Engineering Software Environment to support the machines lifecycle, 2) a common machine data model which contains all machine information and 3) machine components that can be either real or simulated.

An integrated engineering environment consists of an extendable set of engineering tools that can be used by a set of globally distributed engineering stakeholders at all phases of the machines lifecycle. These tools include a Process Definition Environment (PDE) which supports the process planning, machine design and sequencing, interlocking logic and provides simulation tools to test the machine [88, 89, 90, 93].

A common machine data model is a key concept of the component based automation system. All machine information relating to the machine is stored in this common machine data

repository. This ensures that all information is consistent and not fragmented across different systems (machine controller, HMI systems) in the automation system [88, 89, 90].

The machine components can be either real or modelled. Modelled machine components are simulated using a three dimensional solid models. Real machine components are physical modular parts (actuator, sensors, servo drives) of the machine which contain embedded machine interlocking and sequencing logic [87, 88, 89, 90].

The research described in this thesis addresses one particular aspect within this research area; Human Machine Interface (HMI) system's that facilitate monitoring and diagnostics of manufacturing automation machines.

The research presented in this thesis was conducted within the COMPAG [88], GEMM [89] and COMPANION [90] research projects.

3.6 Research Focus and Contributions

The primary focus of the research presented within this thesis is *"The Design, Development, Prototyping and Evaluation of a new approach to HMI Systems that supports Run-time Monitoring, Diagnostic, Servicing and Control capabilities for Component Based Automated Manufacturing Production Machinery within a Global Manufacturing Environment"*.

To accomplish this primary research goal the contributions that have been established during the course of the research are:

- Studies of current industry best practice within machine HMI system's to determine limitations and inadequacies in supporting the requirements that are demanded from Global Manufacturing.
- A modelling technique that formally and unambiguously describes the HMI system functionality and layout that is easily understood by end system users and provides sufficient information to allow software programmers to implement fully functional HMI systems.
- A System Architecture that supports the activities for the design and implementation of a HMI system that meets the requirements of global manufacturing.
- Description of the HMI system construction process for implementation of this novel HMI approach to the automotive manufacturing industry.
- Evaluation of the new approach developed in this research in an attempt to describe the benefits, characteristics and its significant improvements over current practices Comparative study can be drawn based on example scenarios between current approaches and the researcher's C-B approaches to composing HMI systems.
- Description of how approaches and concepts developed in this research aim to support the complete lifecycle requirements of C-B automated manufacturing machines.

3.7 Research Plan and Methodology

The general methodology for the research conducted and described in this thesis is illustrated in figure 3.2. The research is split into 5 phases. The initial phase involves observing and reviewing current state of the art literature within user interface models, principles and development approaches. In addition, a field study to understand current industry best practice in HMI engineering is undertaken. This exploration research enables the problem to be formulated. Phase 2 involves hypothesizing solutions to the problem formulated in the previous phase. Theories and architectures for a HMI are conceptualised, considering all aspects incorporated into the system. Phase 3, instantiates the conceptual theory and architecture through the implementation of a proof of concept demonstration system. Phase 4 is a qualitative and quantitative evaluation to understand the effectiveness of the system. The evaluation method involves applying different user scenarios to test particular attributes of the system. Phase 5 formulates knowledge and provides general conclusions of the research.

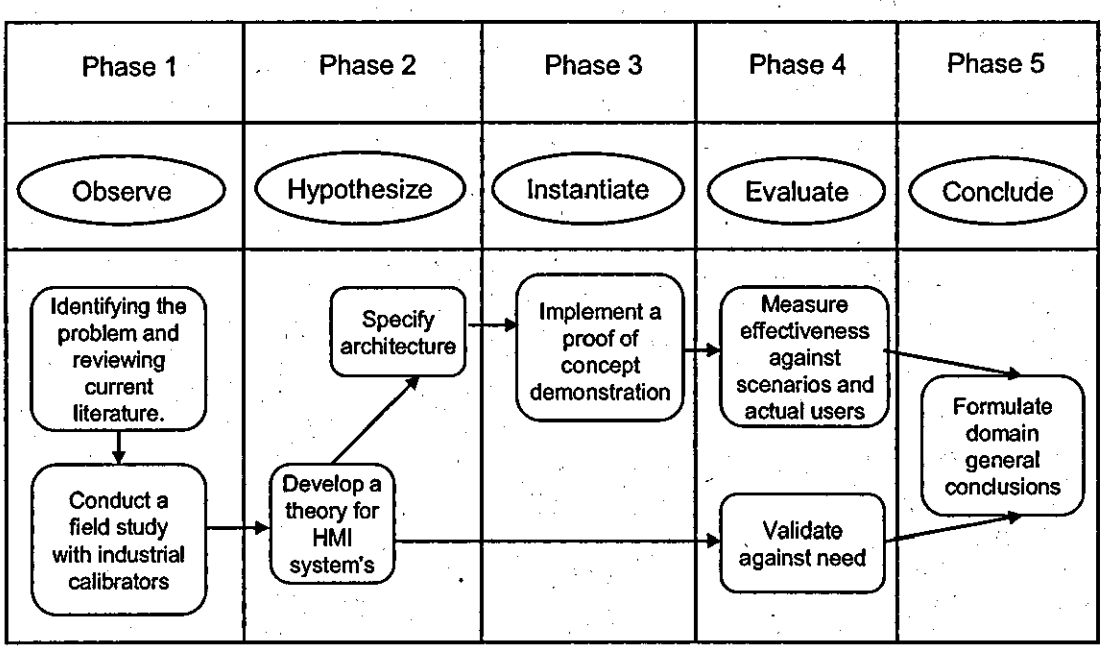


Figure 3.2 Research methodology phases

Figure 3.3 illustrates the major activities within the research project and identifies the areas of the author's contributions.

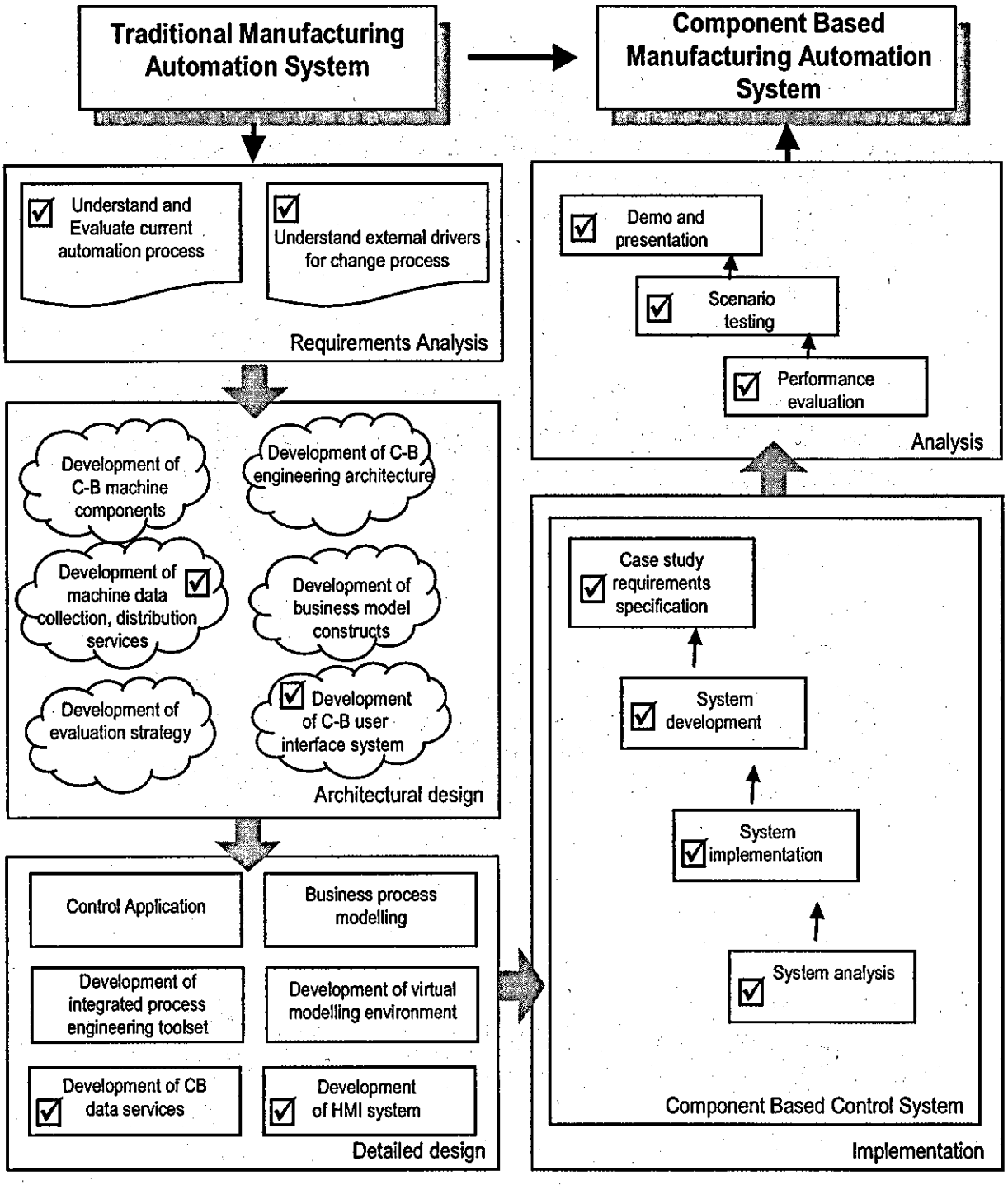


Figure 3.3 Authors involvement in the development of next generation C-B manufacturing automation system developed during the COMPAG/GEMM / COMPANION research projects.

3.8 HMI Systems Engineering Method

The structure for the development of the HMI system in this research which is described in this thesis is illustrated in figure 3.4. The structure adopted is based on the fact that all large scale complex systems (e.g. a HMI system in this instance) can be represented by four different views [103]. Therefore to design the HMI system comprehensively the following four system views must be considered; 1) the functional view described in chapter 4 of this thesis, 2) the process view described in chapter 5 of this thesis, 3) Physical view which is also described in chapter 5 of this thesis and 4) Development view described in chapter 6 of this thesis. HMI System scenarios in this thesis are formed from case studies in chapter 7 of this thesis and applied to the four different system views to test and evaluate (described in chapter 8) the systems design. The four views to describe a system are:

Functional View

The Functional Viewpoint also termed as the *Logical Viewpoint* represents the functional requirements that are required by the HMI systems end users. The functional viewpoint in the context of HMI systems is derived from the end user requirements to enable them to perform their associated enterprise role. The architecture adopted for specifying the HMI systems functionality is described in chapter 4. The major concept behind the functional architecture is that the HMI system end user functionality can be decomposed into a set of HMI Tasks. Using this approach the End User HMI functionality is represented using a storyboarding technique (see chapter 4 of this thesis). This technique comprehensively describes the screen navigation structure, screen layout and content.

Process View

The process view describes the executable process of the HMI System. The HMI System is formed from a logical structured set of interconnecting software components. The executable process describes system elements, the element processes (i.e. what the elements do) and their communication with other system elements. The HMI System architecture described in chapter 5 of this thesis details the process view of the HMI System.

Physical View

The physical view describes the physical resource and the software deployment of the system. This view may be considered as the physical configuration of the system. The system topology describes specific links through which different physical elements of the system communicate with each other. The physical deployment is organized in a hierarchy of layers, each layer providing a narrow and well-defined interface to the layers above it. This physical view of the system is described in chapter 5 of this thesis, HMI System architecture.

Development Viewpoint

The development view describes how the software that is used to build complete systems is organised and the overall system construction methodology. In the approach proposed in this thesis the software is packaged in modular HMI Tasks components and stored in system libraries. This is described in chapter 6 of this thesis, "HMI System construction".

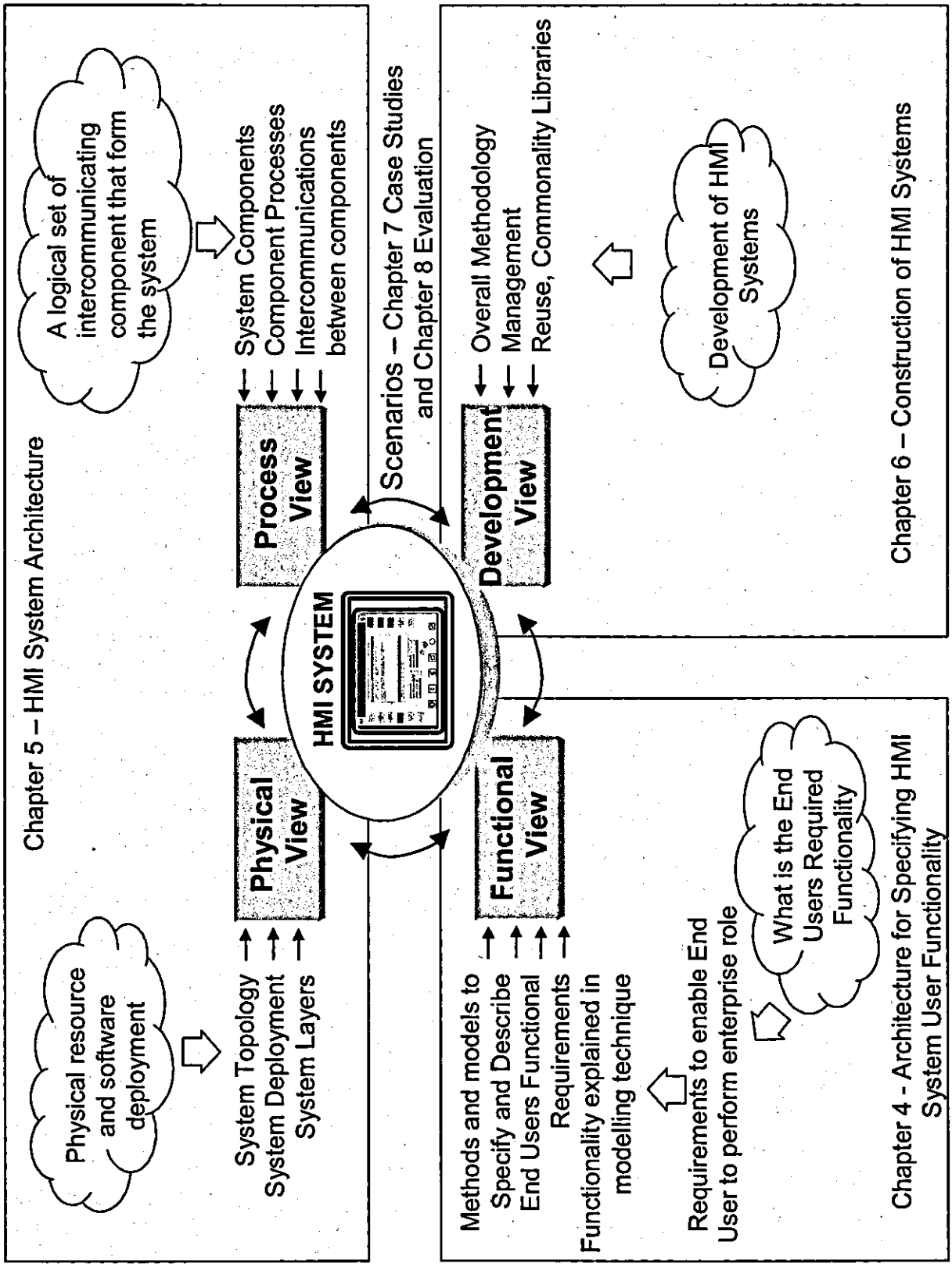


Figure 3.4 Four views [103] for the design and development of the HMI System in this research.

4 End User HMI Functionality

4.1 Introduction

This chapter describes a structure and techniques that has been developed in this research for modelling the end user's functionality in the context of automotive manufacturing automation systems. The HMI end users functionality is defined as the information and the interaction that is required by the HMI end user, for example a machine operator HMI end user may require certain types of machine error information, current machine state and interaction with the mode control of the machine. The type of machine information and interaction required by the end user forms the end user functional specification for the HMI System. Current approaches to describing the end users HMI functionality for manufacturing automation machines involve describing the required machine information and interaction using screen shots and textual description (see section 3.2 for more details on the current HMI process). As the competitiveness of global manufacturing increases, enterprises must have HMI systems that are reconfigurable and scalable to accommodate frequent production manufacturing machine changes. A novel approach to describing and modelling the HMI's functionality has been established in this research that involves using a well defined hierarchal structure of the different levels of functionality to support the scalability for different end user functional requirements. Within this structure the different functionality levels are modelled using a storyboarding technique to comprehensively describe the HMI screens navigation structure, screen layout and content. This approach to specifying the HMI functionality provides sufficient information for HMI programmers and or automated generation tools (see section 6.5 where a composition tool is described that facilitates direct execution of a HMI storyboarding specification model) to create executable HMI systems. The major benefits of this approach are that end users can; 1) understand the storyboarding technique and hence 2) compose fully functional HMI systems in a faster time that meet their HMI functional requirements.

The End User HMI functionality described in this chapter only specifies the type of information and interaction end users require with the HMI system and does not specify or address any issues relating to the systems architecture that are concerned with how the HMI system processes machine information to the end users and its performance in its delivery. The requirements for the HMI systems architecture, for example how the HMI can provide accurate information, the delivery of real-time information and interfacing to a real or virtual machine will be described in chapter 5 (HMI Systems Architecture).

The remainder of this chapter describes the decomposition and corresponding structure required for the end user HMI functionality. This structure is based on the concept that end users HMI functionality can be decomposed into different aspects of functionality which are the different HMI Tasks that the end users must perform to fulfil a particular enterprise role. The structure of the end user functionality consists of three levels that relate to different levels of granularity; 1) End User HMI's which are composed from one or more HMI Tasks, 2) HMI Tasks which support a particular aspect of the HMI functionality required and 3) HMI Widgets that provide the functionality to directly interact with an aspect of the controlled machine.

4.2 Requirements for Specifying End User Functionality

To adequately specify the end users HMI functionality within the context of automotive manufacturing machines five requirements (detailed in table 4.1) have been derived in this research from the global manufacturing marketplace (see section 2.2).

HMI end users functional specification and requirements must be comprehensively represented in an unambiguous consistent manner to help facilitate a HMI system design that meets the needs of the targeted end user. It is a critical requirement to the efficient use of HMI system functionality is what the end user requires to achieve their enterprise role [88]. To achieve the design, a HMI System that meets the end users requirements it is essential that the modelling techniques adopted can be commonly understood by end users and system implementers.

Any structure and modelling technique that is adopted to specify the end users HMI functionality must have sufficient agility and scalability to support the requirements from many different end users, for example a machine operator would require different HMI functionality to a machine diagnostic engineer due to the different enterprise role that they have. A well designed structure that identifies and partitions the different levels of end user functionality is a requirement for the architecture achieving this scalability. The HMI elements must logically map onto an HMI software system but remain independent of any implementation technologies.

Table 4.1 Requirements for architecture to specify the End User Functionality

	Requirements for Specifying End User Functionality
1.	HMI end users functional specification and requirements must be comprehensively represented in an unambiguous consistent manner.
2.	Commonly understood by non technical end users and provides sufficient detail to facilitate either; 1) software engineers developing an HMI and or 2) automated tools to generate an HMI.
3.	Have sufficient agility and scalability to support the requirements of many different types of end users HMI functionality.
4.	A well designed structure that identifies and partitions the different levels of end user functionality.
5.	Elements must map onto an HMI software system but remain independent of implementation technologies.

4.3 Establishing HMI System Functional Requirements

To determine the end users HMI functional requirements both the machine and user perspectives must be considered.

The end user has a responsibility to perform one or more tasks on the machine to fulfil their enterprise role. The individual end users characteristics such as their computer experience, their cognitive ability and locality to the machine build a profile of the user which determines the machine tasks that they require to fulfil their enterprise role. This is known as a task model [121, 127]. The HMI must display different types of machine information (for example machine diagnostic, error information) to the end user and support interaction in the form of user actions (for example requests to move the machines components or change the operating mode). From the machines perspective, the tasks the user performs on the machine are characterised by the complexity of the machine, the type of machine and the end machine application. The HMI system exchanges machine data and or control with end users to fully support the machines continuous run-time operation throughout its lifecycle.

For example in a simplified case if the end user's enterprise role is to recover the machine from an error state they would require machine error information and control of the machine mode.

There are different end users of the HMI system (machine operators and engineers, diagnostic engineers) all having different users profiles. The proposed architecture for specifying the HMI end users functionality must be scalable to effectively support the different end users functionality.

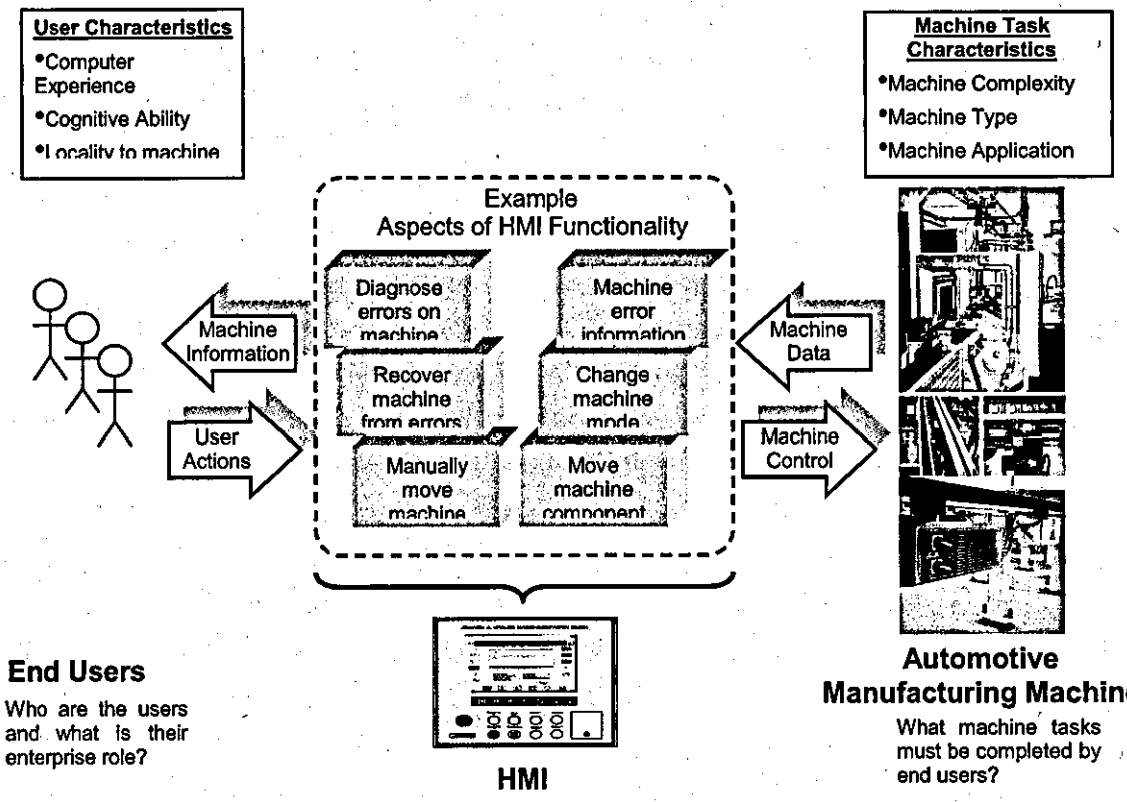


Figure 4.1 Establishing HMI Functional Requirements

4.4 End User HMI Functional Structure

A structure for modelling the end users HMI functionality is described in this research. The objective of this structure is to group the end users functionality of the HMI system into different levels. Identifying and partitioning a consistent structure for a system it is an enabler in supporting the reusability of the system's elements by having well defined system structure. The scalability of the system is also increased when composing a complete system from well defined systems elements.

The end users HMI functional structure is based on the concept that the end users require one or more different aspects of functionality for an HMI system to perform their enterprise role. These different aspects of end user functionality are modularised and defined as HMI Tasks in the HMI functional structure. A HMI Task consists of one or more direct interactions with individual components of the machine. This direct interaction is the lowest level of functionality in the HMI system and typically would be implemented in the form of buttons or machine textual information. This direct interaction is defined in the functional hierarchy as HMI Widgets. An End User HMI is the highest level of functionality in the structure composed from a set of HMI Tasks. This three tier hierarchical structure provides a logical decomposition of the HMI functional requirements.

Figure 4.3 illustrates this structure which consists of three levels; 1) End User HMI's which are composed one or more HMI Tasks, 2) HMI Tasks support a particular aspect of the HMI functionality required and 3) HMI widgets that provide the functionality to directly interact with an aspect of the controlled machine.

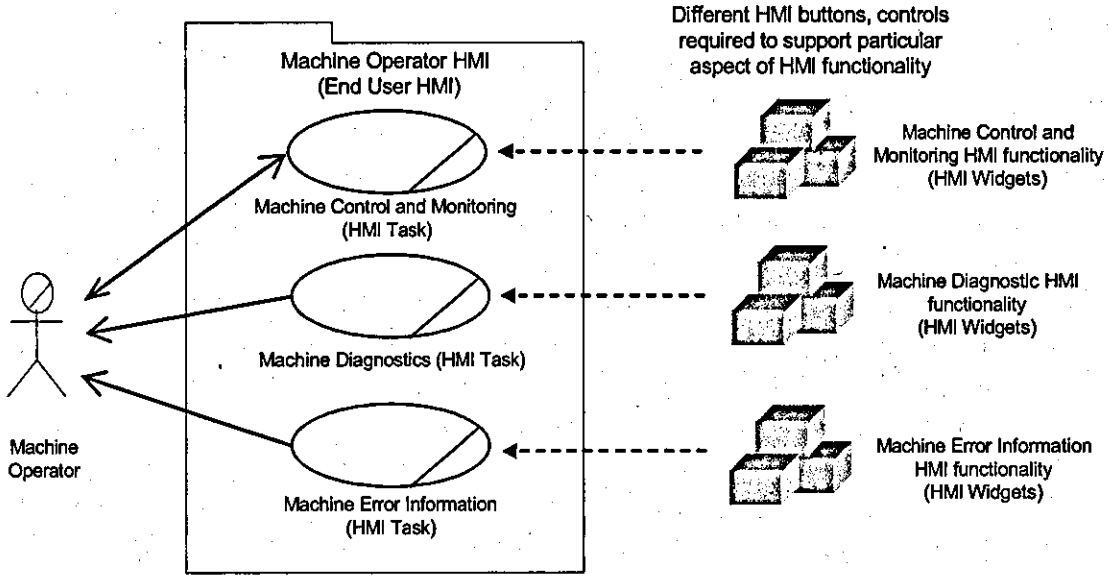


Figure 4.2 Use Case illustrating the elements in the HMI functional structure

Figure 4.2 illustrates an example where a machine operator requires three different types of functionality (modelled as HMI Task's) which are; 1) Machine Control and Monitoring, 2) Machine Diagnostics and 3) Machine Error Information. The HMI Task's are typical of machinery within the automotive manufacturing machine domain [106].

Each HMI Task is composed from many HMI buttons and controls (HMI Widgets) to support the particular aspect of HMI functionality.

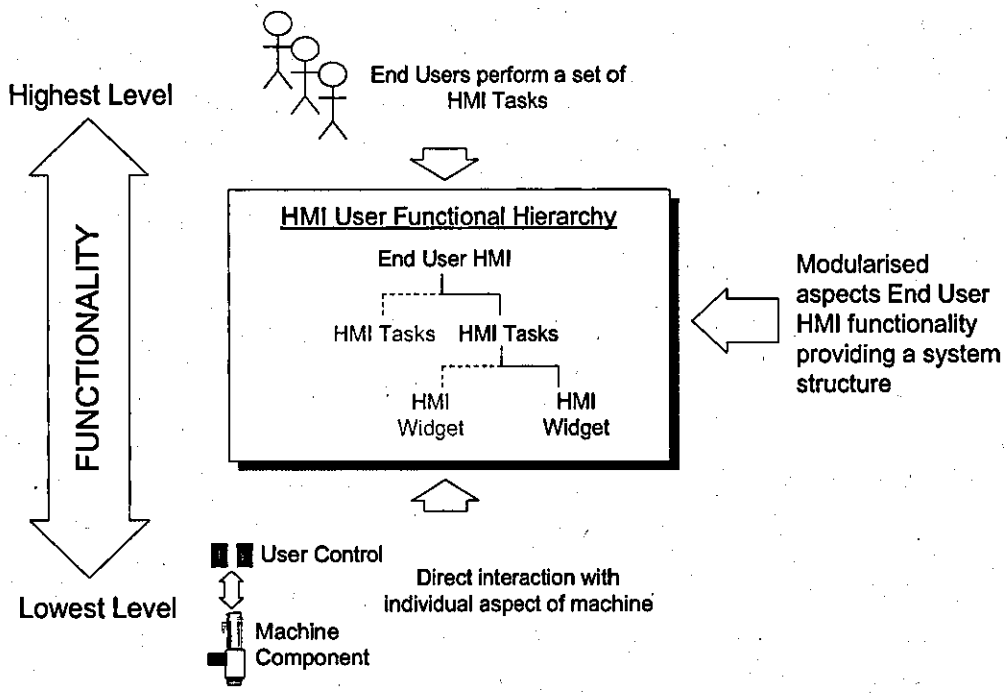


Figure 4.3 Structure of the End User HMI Functionality

The End User HMI describes the navigational structure and the screen layout. Many different End User HMIs may exist to support their associated end users, for example a Diagnostic Engineers HMI and Machine operators HMI may exist (see illustration in figure 4.4). End Users require different HMI Tasks to fully support their role, a Diagnostic Engineer for example will only require the Machine Diagnostics Task but a Machine operator will require Machine Error Information and Machine Component and Monitoring Task. The HMI Tasks are composed from one or more Interaction Elements, for example a Machine Diagnostic Task will contain Component Interlocking Configuration, Component service Information and Three Dimensional Machine Solid Model interaction elements. A Machine Error Information Task will require a Three Dimensional Machine Solid Model, Machine Sub System Error Information and Real-time Sub System Information. The Interaction Elements provides the machine components interaction sequence.

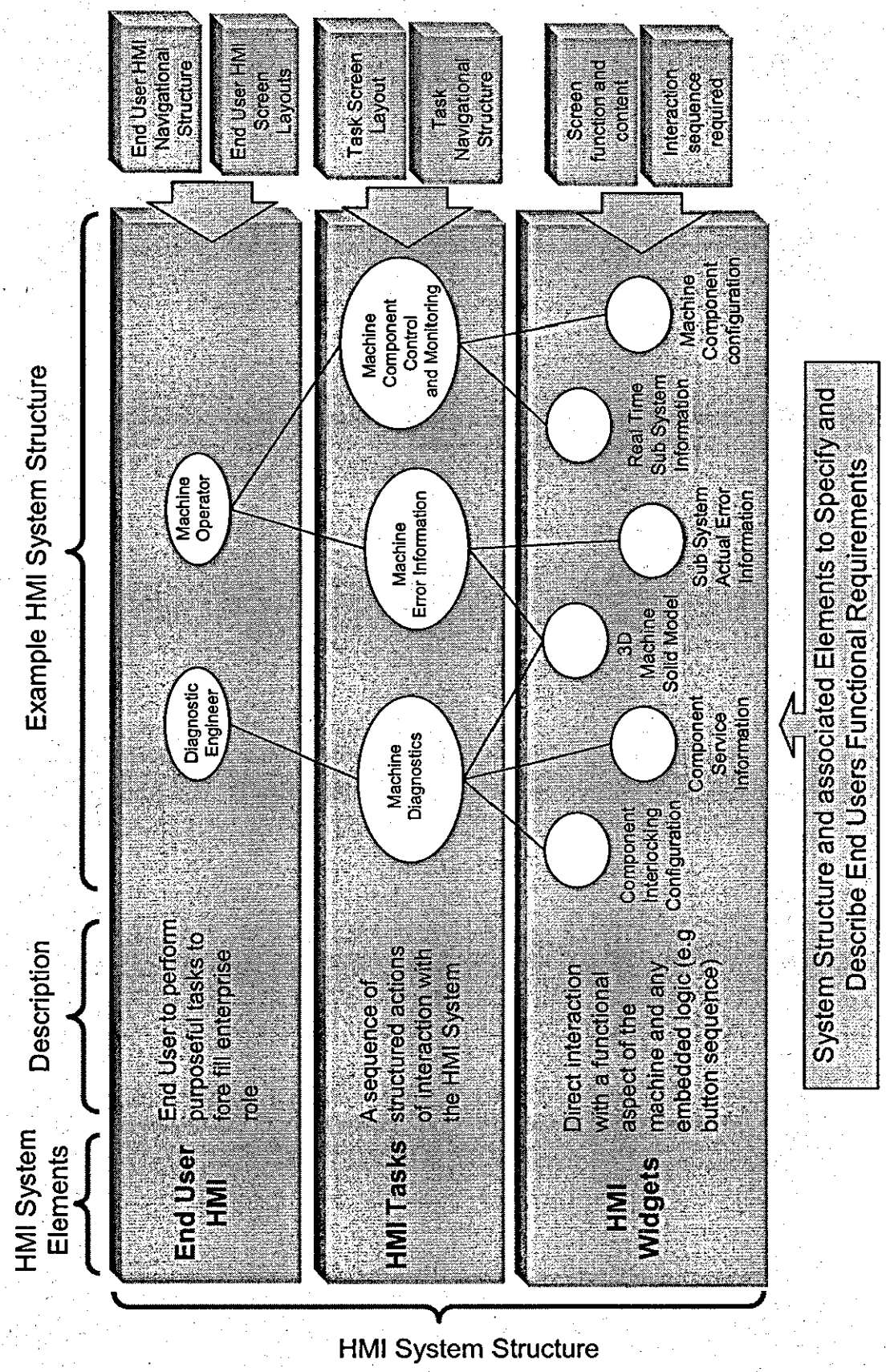


Figure 4.4 Elements and Their Structure to Represent Functional Viewpoint of HMI System

Figure 4.5 expands on the structure previously described that identifies the elements and their associated functionality by describing the connectivity of components and the mapping of functionality into models and their associated software components.

The End User HMI maps to a complete end user HMI which consists of one or more end user HMI screens. These screens display the different aspects of HMI functionality required by the end user and are laid out dividing the screens into a number of compartments. The screen content is defined by populating the screen compartments with; 1) an HMI Task which represents a particular aspect of the end users functionality or 2) navigation between different End User HMI screens.

The next hierarchal element in the functional architectures structure, the HMI Task is composed from a series of HMI Task screens; these screens being a screen compartment in the end user HMI. The HMI Task screens contain screen compartments which are populated with; 1) HMI widgets or 2) HMI Task navigation. The storyboarding technique (see section 4.5) is used as the mechanism to describe the representation of the screen layout, content and navigation for the End User HMI and HMI Tasks in this architecture. When screens are combined and ordered they can comprehensively describe the end users functional requirements of a HMI system. One or more HMI widgets can exist from a single machine component. HMI widgets are typically HMI buttons and machine information. These HMI Widgets can be represented using state diagrams (see section 4.8) which for example may show the states of a button which is linked to controlling a machine actuator component.

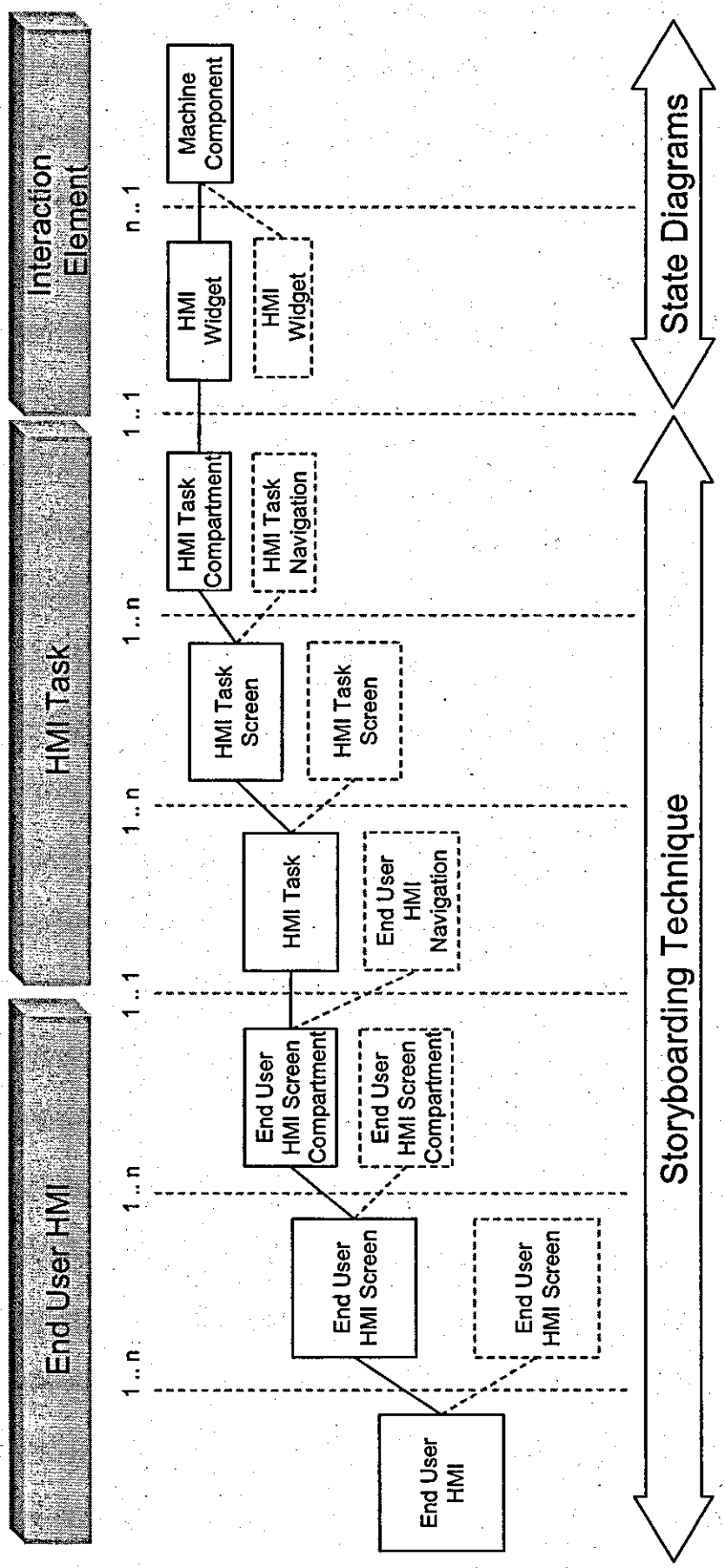


Figure 4.5 Schematic of the End User HMI Functionality Structure and Modelling Techniques

4.5 Storyboarding Technique

A novel approach developed in this research to the specification of the requirement of the end users HMI functionality involves using a storyboarding technique [135, 136, 137, 138 and 139] to describe the End User HMI and the HMI Tasks. This storyboarding technique comprehensively describes the HMI functionality in terms of screens navigation structure, screen layout and content. It is suggested that this approach to end users HMI functional specification provides sufficient information for HMI programmers and or automated generation tools (see section 6.5 where a composition tool is described that facilitates direct execution of a HMI storyboarding specification model) to create executable HMI systems.

A storyboard is a way to tell a story through the use of discrete static pictures. In its application to a HMI System, the screens are the individual boards or sub boards that when strung together tell a story of the HMI component. This technique; 1) describes screen layouts, 2) specifies the navigation between the screens and 3) describes the functionality of the screens. This diagrammatic representation on which the storyboarding technique is based allows non technical HMI End Users to understand the functionality and the technique is comprehensive enough for HMI software designers to develop complete HMI systems.

The storyboarding technique involves outlining the screens required and populating the screens with a set of compartments which define the screen layout. An architecturally important issue of the HMI functionality is the navigational path. The diagrams used in this technique express the structure of the HMI screens with their potential navigational pathways, providing a roadmap of the HMI screens. An important characteristic of a storyboarding diagram is that it expresses all the legal and expected paths through the HMI. Expected navigational paths are detailed using arrows with a textual description of the action or method that associates the link between the screens. The screen compartments are linked either using; 1) application specific functionality or 2) generic functionality. The application specific functionality in the HMI system are the HMI components (HMI Tasks and HMI Widgets) that are required to build a unique screen, for example in a HMI Task screen a

series of machine mode control HMI Widgets (see section 7.5.5) or end user HMI screen may require a HMI machine error task (see section 7.5.5).

The navigation between different HMI screens must always exist and is a generic screen required for all HMI's with more than one screen. The screen compartment which contains the navigation is represented by arrows indicating the navigation direction between the screens. This storyboarding technique provides, from these basic constructs, sufficient information to provide a representation of complex end user HMI systems functionality.

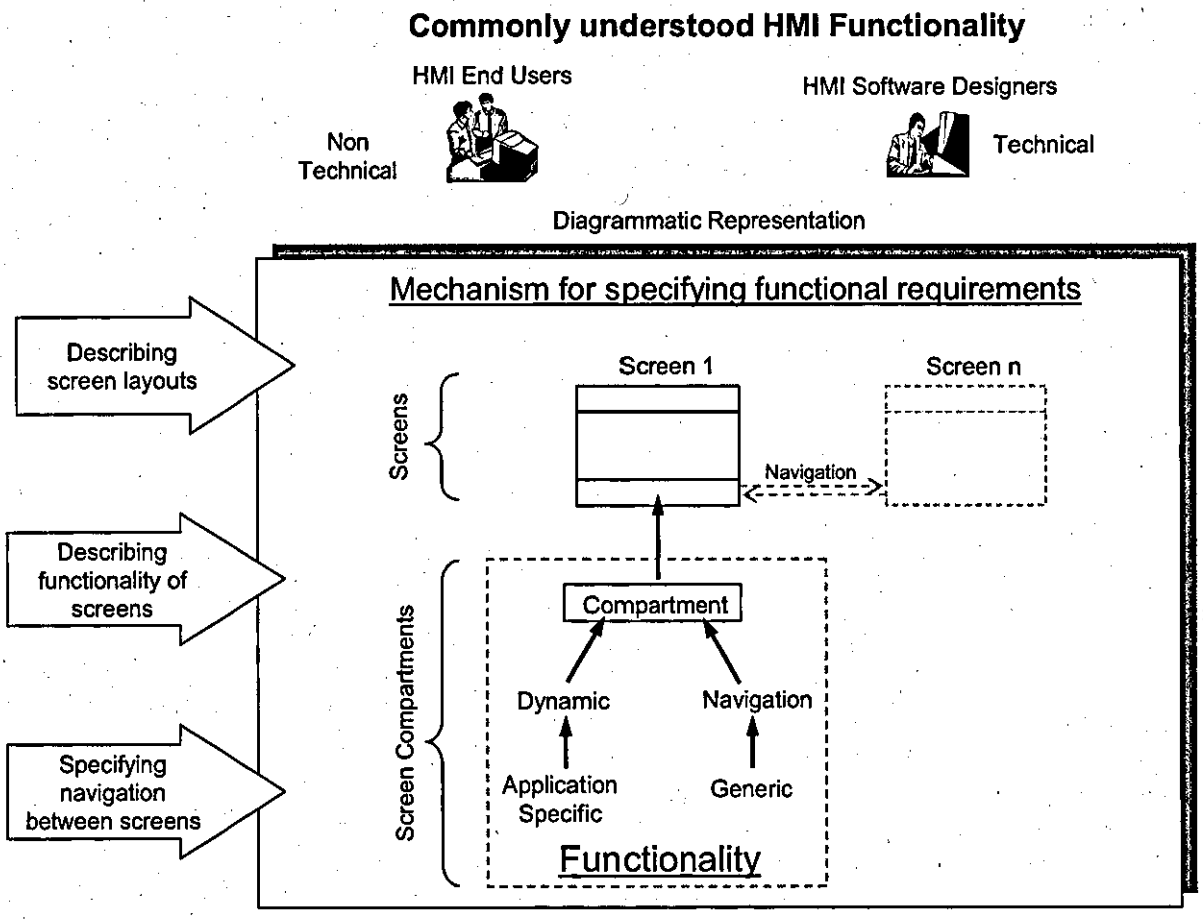


Figure 4.6 Introducing the storyboarding technique developed for HMI Systems

4.5.1 Storyboarding Method Semantics

The two constructs that are adopted in the storyboarding model are; 1) Screen and 2) Screen Compartments.

The screen storyboarding model constructs represent a complete screen or a screen compartment of a higher hierarchical storyboarding model. A screen may contain one or more screen compartments and its function is to represent the layout of these screen compartments. The screen does not represent any intercommunication between the screen compartments. The graphical representation for screen is a rectangular box which with the screen name written below.

The screen compartments are self contained functional elements that are contained within a screen. The screen compartments contain a HMI software component (either a HMI task or a HMI Widget) that support a particular aspect of HMI functionality. Communication between the screen compartments defines the navigational links between different screen compartments. Arrows represent the direction of the navigation links and a textual description is used and may denote a software function call or simple description to instigate the navigational link. Screen compartments are graphically represented by a rectangular box and inside side the box is the name of the contained software component.

Figure 4.7 illustrates that both the end user HMI and the HMI Tasks are modelled using the storyboarding technique. The screen compartments within the end user model reference a HMI Task. The HMI Task is also modelled using the storyboarding technique. This creates a two level hierarchical of storyboarding models.

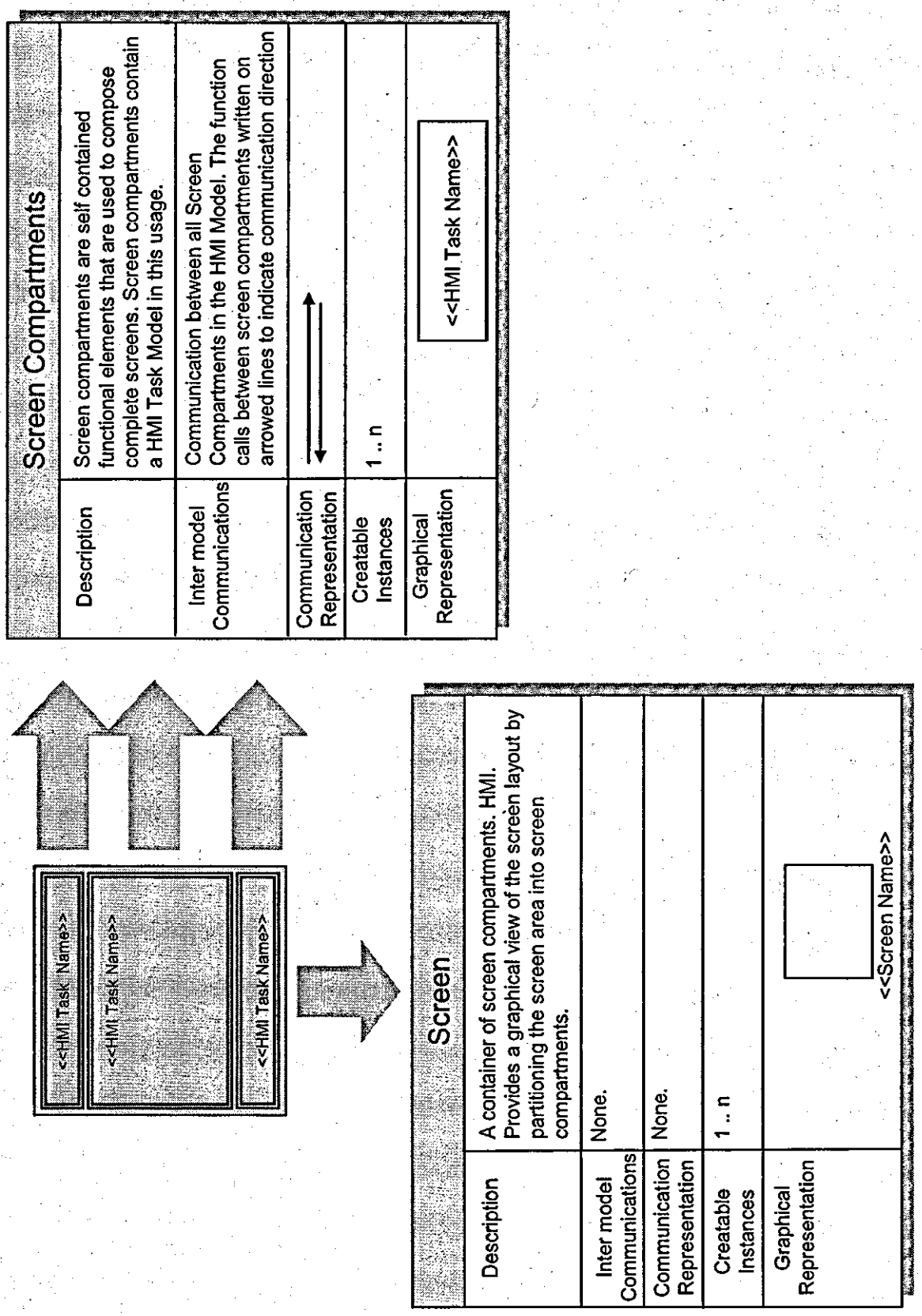


Figure 4.7 Storyboarding Method Semantics

4.6 End User HMI

Figure 4.8 illustrates a how the storyboarding technique is used to describe a series of individual screens and their associated functionality for an End User HMI

The End User HMI is represented using the storyboarding technique which describes the HMI screen layouts and navigational structure. The End User HMI screen compartments contain HMI Tasks that support different aspects of the end users HMI functionality. The generic functionality that is required in all HMI's with one or more screen is the screen to screen navigation. These screen compartments represent the screen layout and the HMI Tasks describe the functionality of the End User HMI Screens.

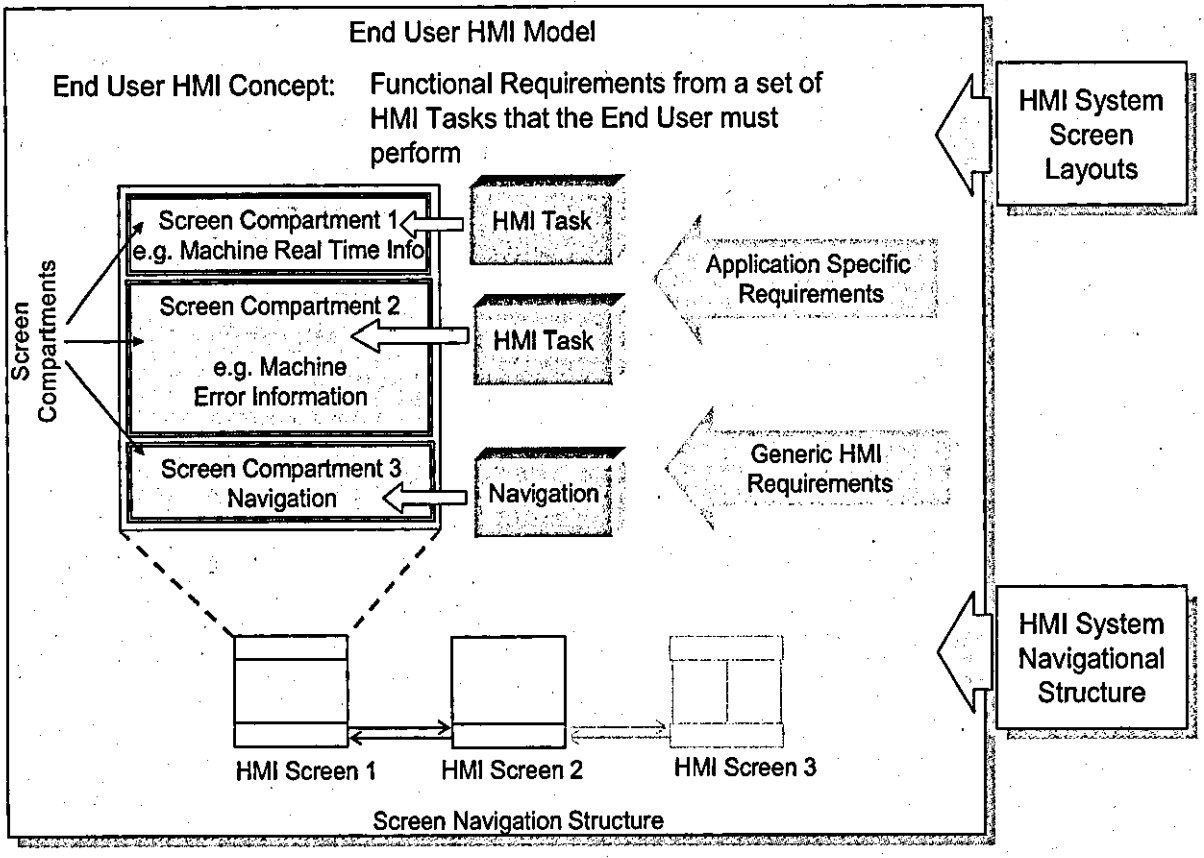


Figure 4.8 Storyboarding Technique Used To Describe a Series of Individual Screens and their Associated Functionality for an End User HMI

An example End User HMI storyboard is illustrated in figure 4.9. In this example the Machine Operators HMI consists of two screens with each screen containing two screen compartments. The generic HMI Task for both screens is the navigation. The description of the navigational HMI Task action to navigate to the target screen is written on the arrow.

Screen 1 contains the Machine Component Control and Monitoring HMI Task. The second screen contains the Machine Error Information and is navigated to from screen one by the Machine Error navigation action.

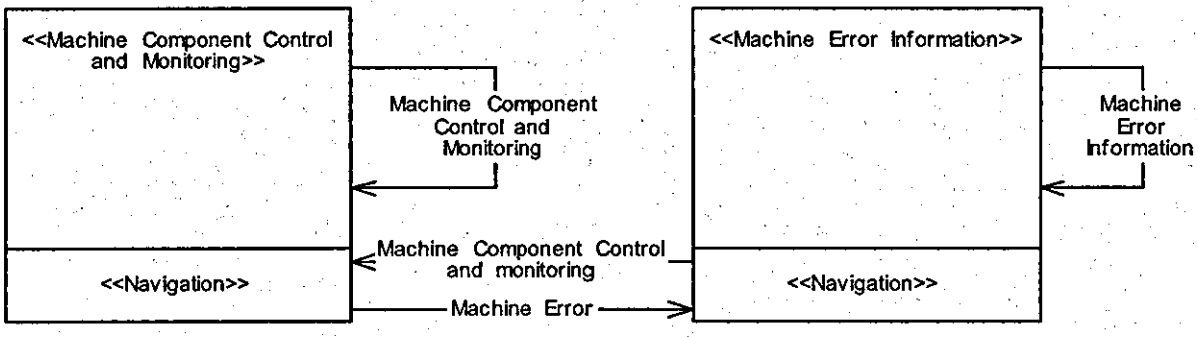


Figure 4.9 Functional Description of Example Machine Operator End User HMI System

4.7 HMI Tasks

Figure 4.10 illustrates the storyboarding technique that is used to describe a series of individual screens and their associated functionality for a HMI task.

The HMI tasks are represented using the storyboarding technique which describes the HMI task screen layout and navigational links between the screens. In this architecture the HMI tasks are contained within screen compartments in the end user HMI. The HMI task screen compartments contain HMI Widgets that are required to support the particular aspect of HMI functionality required. For example a HMI error information task may would consist of screens that have different HMI errors information widgets to provide this required machine error information HMI functionality. The HMI task screen compartments represent the layout and the required HMI Widgets to support the functionality of the HMI Task Screen

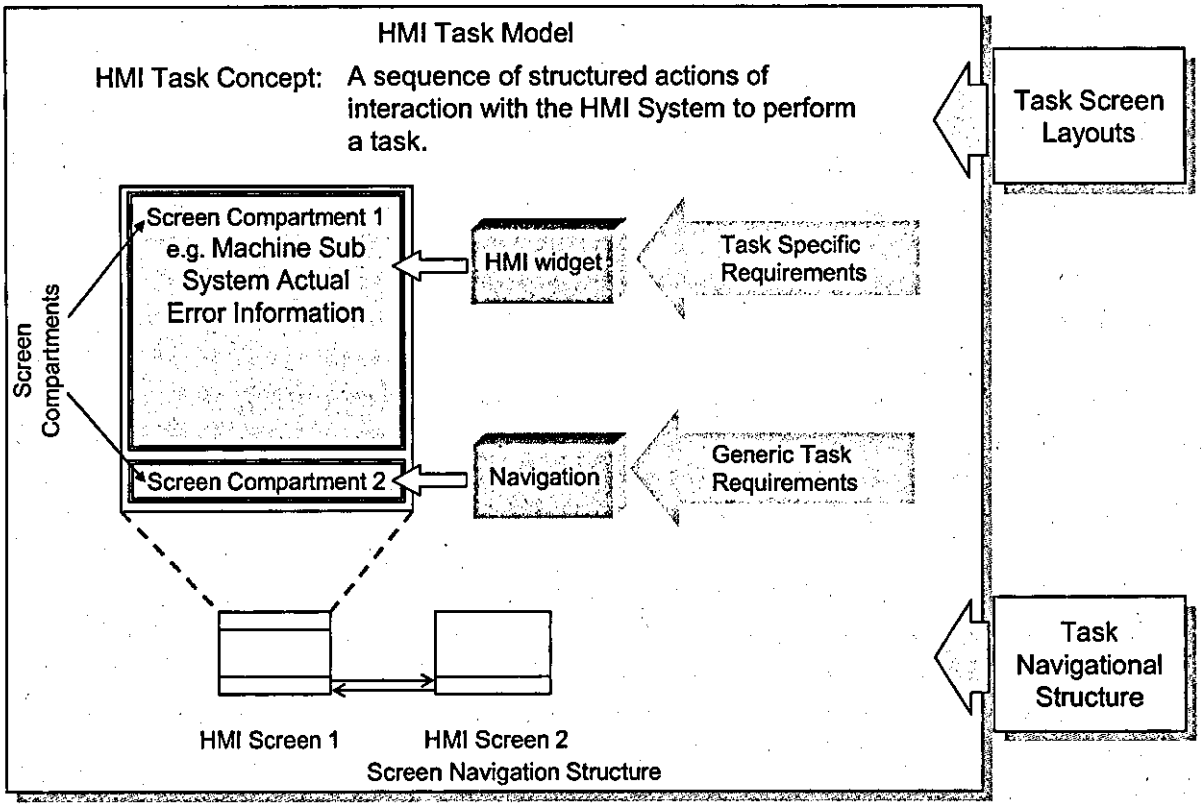


Figure 4.10 Storyboarding Technique Used To Describe a Series of Individual Screens and their Associated Functionality for HMI Tasks

An example HMI Task model is illustrated in figure 4.11. In this example storyboard the Machine Error Information HMI Task is illustrated that consists of two screens, each containing two screen compartments. The generic compartment for both screen s is the

navigation between the screens. A description of the navigational HMI Task action to navigate between the screens is written on the arrow. Screen 1 contains the Machine Sub System Error Information HMI Widget. The second screen contains the Machine Three Dimensional Solid Model HMI Widget and is navigated too from screen one by the Machine 3D Model navigation action.

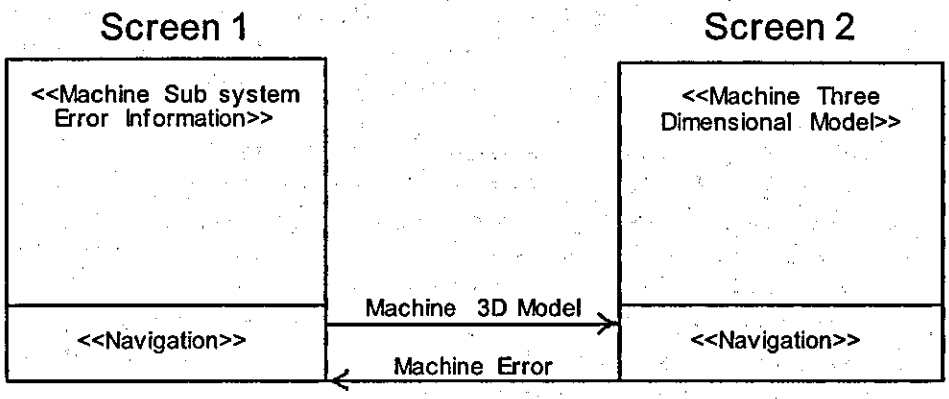


Figure 4.11 Functional Description of Example Machine Error Information HMI Task

4.8 HMI Widgets

The concept of the HMI Widgets is to provide the functionality to directly interact with an aspect of the controlled machine. The HMI Widgets are the lowest level of functionality in the three level structure developed in this architecture. An HMI Widget for example would be a button that the end user has to control a machine component or a text field displaying some type machine information. The HMI widgets are modelled using state transition diagrams [126]. The state transition diagram is used to describe the relation between the HMI widget state and the automation machine state. For example if the HMI Widget button controls the movement of a machine actuator, then the state transition diagram would have the machine state as the transition to the corresponding HMI Widget state which is illustrated in figure 4.12. The objective of the state transition diagram is to describe the behaviour of the HMI widgets is terms of a number of states. The machine status that represents the HMI widget state is the transition to the state in the state transition diagram.

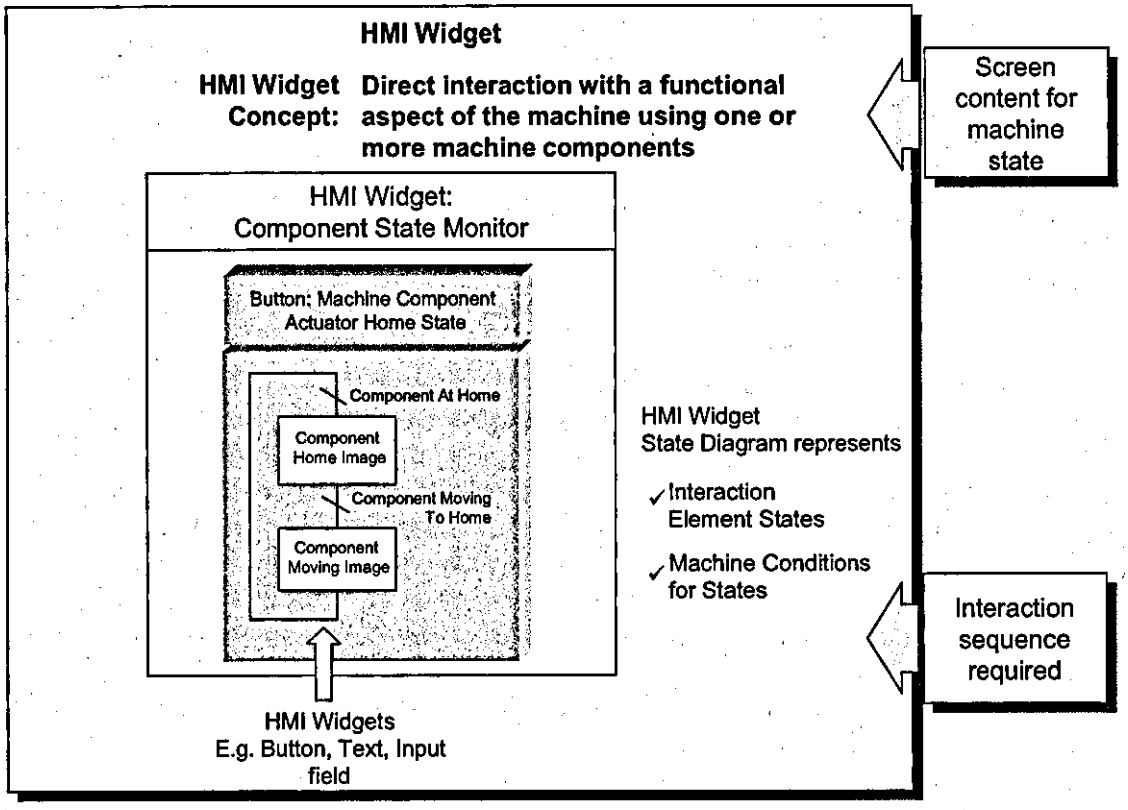


Figure 4.12 State transition diagram is used to describe a HMI Widgets and its corresponding machine states.

4.9 Summary

In this chapter a structure has been described that supports the logical decomposition of the end user HMI functionality into well defined elements. An HMI modelling technique, known as storyboarding, is introduced that comprehensively represents in an unambiguous and consistent manner the functionality of the HMI. The storyboarding technique can readily understood by end users enabling them to specify their functional requirements and providing sufficient detail to support the implementation of a HMI by either; 1) software engineers or 2) automated HMI software generation. This is achieved through the storyboarding graphical notation that clearly specifies the end user functionality. The structure of the end user functionality identifies and partitions the end user functionality into three different levels; 1) End User HMI's which are composed one or more HMI Tasks, 2) HMI Tasks support a particular aspect of the HMI functionality required and 3) HMI Widgets that provide the functionality to directly interact with an aspect of the controlled machine.

In chapter 5 the system architecture will be described and this is concerned with how the HMI system can support the end user functionality from a system perspective, for example how system executable processes support the HMI in providing accurate information, the delivery of real-time information and interfacing to a real or virtual machine (see chapter 5, HMI Systems Architecture).

5 HMI Systems Architecture

5.1 Introduction

Chapter 4 has comprehensively described a suitable structure and corresponding modelling technique for the end user HMI functionality requirements. This has specified a structure for the types of information and interactions end users have with the HMI system and has not specified and or addressed any issues relating to the systems architecture, i.e. how the HMI system executes and delivers machine information.

The HMI system architecture described in this chapter is concerned with the run-time executable process of how a HMI system provides a set of end user interactions that supports different types of end user HMI functionality (see chapter 4) from which many different End User HMI's can be constructed (see chapter 6). The system architecture does not specify the requirements for individual end user HMI's, this is described in chapter 4. Instead the systems architecture has been designed to support generic end user / machine interactions which can be composed in a structure to provide complete end user HMI's. The HMI systems architecture described in this chapter has been designed for automotive manufacturing automation systems. The architecture describes the arrangement, interaction, and executable processes that together are used to form the HMI system. Its purpose is to ensure that the HMI system satisfies a specified set of requirements. These requirements can be categorised into two distinct types. Firstly the functional requirements, i.e., the individual user / machine interactions which are described in section 5.3.1. These are; 1) request machine commands, 2) display real-time machine events, 3) displaying machine configuration information and 4) requesting real-time machine information. Secondly the system architecture is designed to meet non functional requirements which are providing accurate machine information, performance of the systems interactions (adequate real-time performance), scalability of the architecture across many different machine applications to applications and the agility of the system in terms of its responsiveness to changes in the machine configuration or end user

functional requirements. The HMI system architecture has been designed to support machines within the context of automotive manufacturing automation systems.

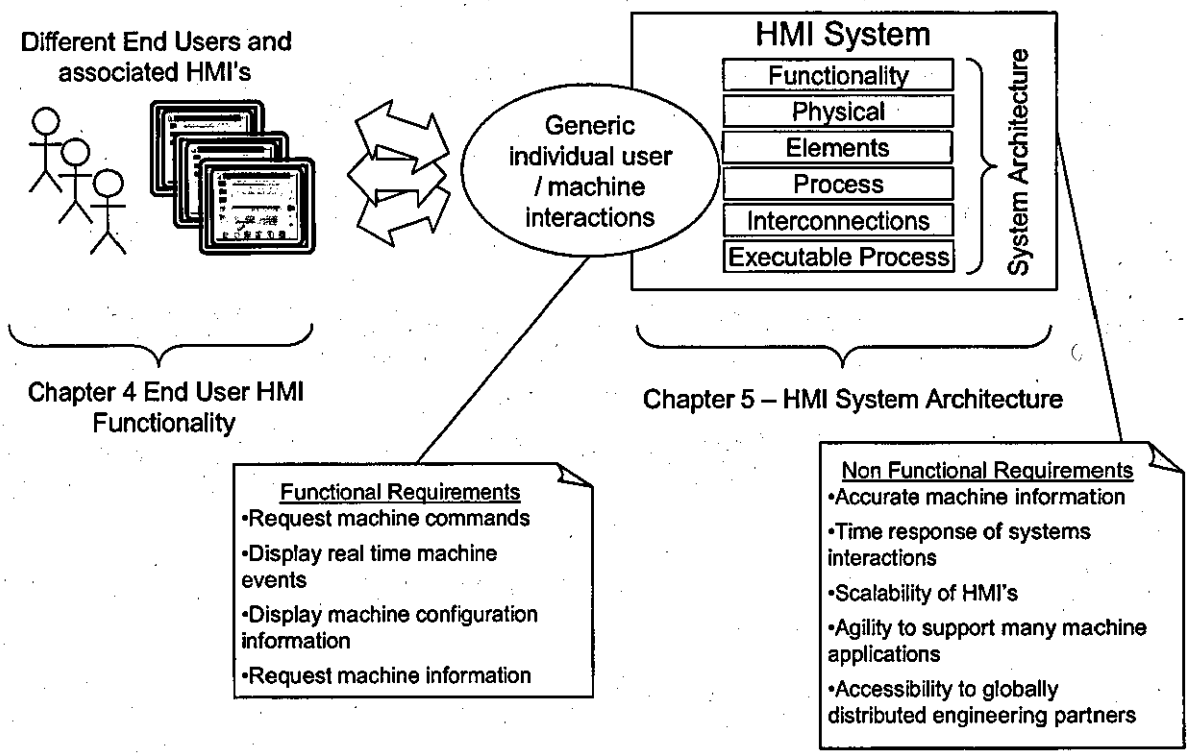


Figure 5.1 HMI System architecture must be designed to fully support functional and non functional requirements.

5.2 Method for the Design of the HMI System Architecture

The method that has been applied to the design of the HMI Systems architecture is illustrated in figure 5.2. The system architecture is described from five viewpoints, 1) Requirements of the system, 2) System Functionality, 3) System Structure, 4) Sequence of the system and 5) System Implementation [152].

The requirements of the system are captured in Use Case descriptions. The Use Case descriptions describe the high level interaction between the HMI system and the stakeholders. Use Case diagrams are based on standard UML notation 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. The scenarios provide a walk through of what the system is expected to do and how it responds for a single use case. Scenarios are developed for all use cases.

From sets of scenarios the functionality of the system can be derived to provide the functional requirements. The functionality of the system is captured in a package diagram. The package diagram partitions the functionality into logical modules.

The systems architecture considers how the modules defined in the package diagram are linked to devices used in the real system.

The functionality of the system modules in the package diagram are used to derive the architectural elements of the system. The architectural elements and their relationships that provide the system structure is represented using standard UML based class diagrams.

The system sequence viewpoint describes the operation of each architectural element in the system. This includes describing; 1) interfaces to other system elements and 2) their processes.

The flow of messages between the architectural elements of the system is described using standard UML sequence diagrams. These diagrams represent complete flow of messages between system elements for a particular operation. These diagrams are used to validate that the system has the correct elements and interactions between them to perform an operation.

It's often an iterative process between designing the systems structure and then validating the functionality by checking the sequence of the system to complete the desired function [134, 152].

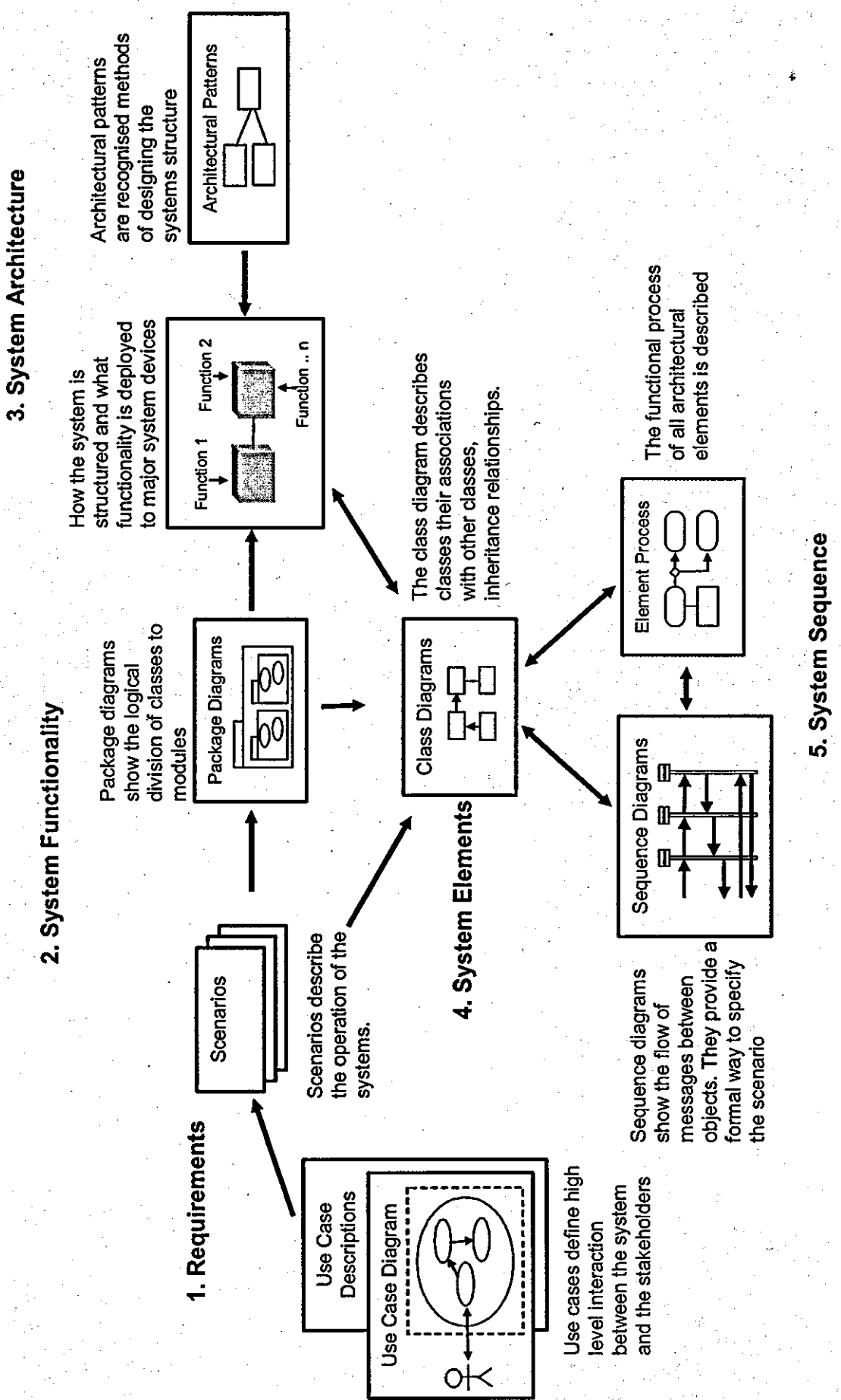


Figure 5.2 Roadmap for HMI Systems architecture designed in this chapter [134, 152]

5.3 HMI System Requirements

The process of developing the HMI System architecture begins with a definition of the requirements. These requirements can be categorised into two distinct types.

The functional requirements are the individual user / machine interactions and are described in use cases. Many of these individual user / machine interactions are typically used to compose a complete end user HMI system (chapter 4 describes a structure and modelling technique to comprehensibly describe the complete end user HMI). The context of the four user / machine interactions that have been identified through observations and interviewing of industrial collaborators [88] is detailed in table 5.1.

Table 5.1 Context of the four user / machine interactions

User / Machine interaction	Context
1. Request machine commands	The end user needs to request machine commands using the HMI system. These commands can control the machine's position or mode of operation using a button of the end user HMI.
2. Display real-time machine events	All machine events need to be displayed to the end user so they can determine the state of the machine. These machine events are time critical and must be delivered in real-time so the HMI reflects the state of the machine.
3. Displaying machine configuration information	Configuration information relating to the machine must be displayed to end users. This machine configuration maybe the sequence and interlocking of the machine so that the end user can diagnose or detect any malfunction, fault with the machine.
4. Requesting real-time machine information.	The end user may need to request real-time machine information, for example current error information of the machine.

The non functional requirements of the system are providing accurate machine information, the performance of the systems interactions, the scalability of the architecture across many different machine end user applications and the accessibility and agility of the system in terms of its responsiveness to changes in the machine configuration or end user functional requirements. These non functional requirements of the HMI system architecture are detailed in table 5.2.

Table 5.2 Non Functional Requirements of the HMI System Architecture

Non Functional Requirements	Context
1. Accurate machine information	Integrated machine automation systems where a common repository provides all machine information not fragmented across multiple systems with poor integration.
2. Time response of systems interactions	Real-time machine events must be propagated to the user in a deterministic manner.
3. Scalability	The architecture must be inherently scalable to support different automotive machine automation applications.
4. Agility	The HMI system architecture must support frequent machine re-configuration with minimum effort required to the HMI.
5. Accessibility	Globally distributed engineering partners must be able to increase machine up time by remotely diagnosing and supporting the machine.

5.3.1 Use Cases

The use case illustrates the individual user / machine interactions. As defined in system engineering texts, a use case captures a goal-oriented set of interactions between external actors and the system under consideration. Actors are parties outside the system, in this case HMI End Users.

Use cases capture who (actor) does what (interaction) with the system and for what purpose (goal) without dealing with the internal working of the system itself. A complete set of use cases specifies all the different ways to use the system, and therefore defines all behaviour required of the system, bounding its scope.

Only one actor is shown in the use case diagram illustrated in figure 5.2. The purpose of the uses case diagram in the context of the HMI systems architectural design is to outline all interactions that users may have with the HMI system to enable a HMI system architecture to be designed that can meet these requirements. It is not the purpose to design each individual user's requirements and interactions with the system, this is captured in the end users HMI functionality explained in chapter 4.

It has been identified in this research in collaboration with industry partners [88] that their are four types of basic interactions end users have with a HMI System which are; 1) requesting machine commands, 2), request machine information, 3) display machine configuration information and 4) display real-time machine events. It is any combination of one or more of these interactions that is required by each individual HMI end user.

The machine which is either simulated or real (see section 3.5 for an overview of C-B automation systems which has a simulation and or a real machine) transfers real-time information and machine configuration information to the HMI System were it must be displayed to the end user. The end user requests machine commands via the HMI systems which are propagated to either the real or simulated machine. The HMI systems architecture must support these basic interactions between the end users and the machine which are the low level building blocks from which many different end users HMI's and classes of automotive production machine applications can be supported (see case studies in chapter 7).

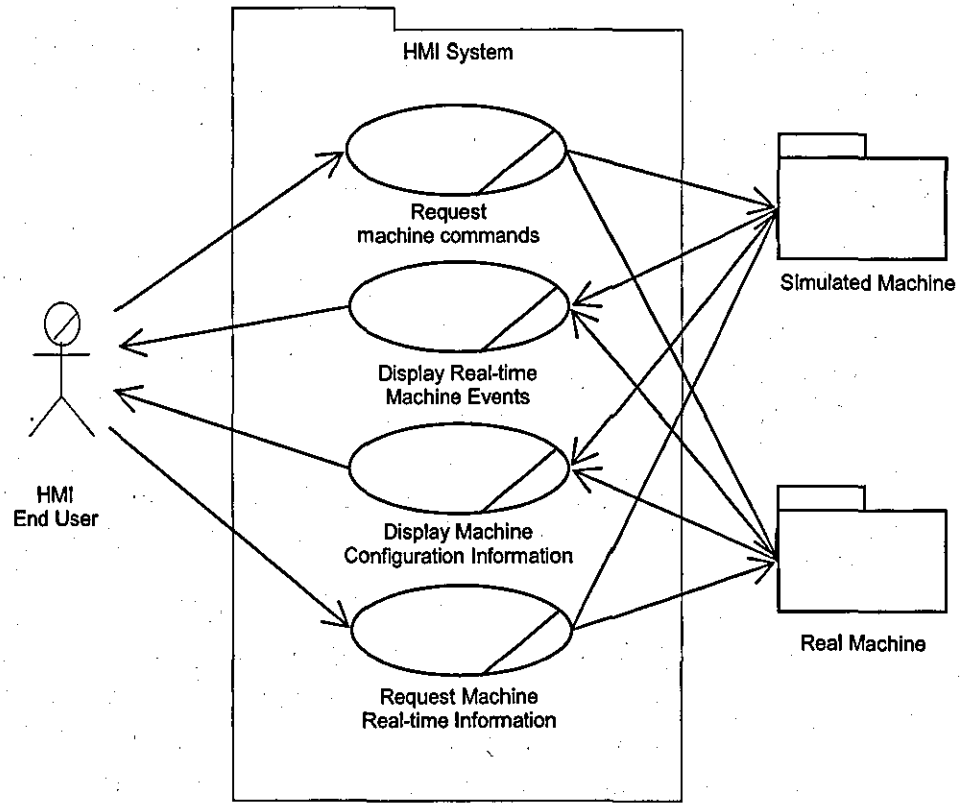


Figure 5.2 Use Cases required in supporting any HMI end users.

5.3.2 Scenarios

Scenarios are applied to each of the use case system interactions formulated in the previous section. The scenarios provide details about the process which the HMI system must conduct. Functionality required in the HMI system can then be extracted from these scenarios to support more detailed design of the architecture.

As this stage in the HMI systems architectural design some very general system elements are assumed to exist so the scenarios can explain an initial abstract executable process. The general elements are; 1) a machine server that contains applications to process logic associated with the user / machine interaction and 2) a end user HMI which the end user interacts with.

Table 5.3 details a scenario where a HMI end user requests a machine command. The scenario is only described at a generalised level, at a more particular level the scenario maybe that the HMI end user requests a machine component movement or machine operating mode is changed.

This scenario involves an end user requesting a machine command using the end user HMI which is then transferred to a machine server where the command is processed. The processing of the command involves checking if it is valid. The validity of the move is based on the machine's current configuration and this information must be stored on the machine server. It must be determined from the machine's configuration information if the request is directed to the real machine or a simulated solid model.

Table 5.3 Scenario involving HMI End User requesting a machine command

Scenario – HMI End User Request Machine Commands

1. Stakeholder requests a machine command using the End User HMI.
2. The End User HMI transfers request to machine server.
3. The Machine Server applications check if the requested command is valid using machine configuration information.
4. The Machine Server applications determine from machine configuration the candidate machine (real or simulated) to execute command.
5. Machine command is requested
6. Response is returned to the End User HMI to confirm command request.

Table 5.4 details a scenario based on the HMI System displaying a real-time machine event. The scenario is described at a generalised level, an example scenario maybe a machine component state change and an event is generated on the end user HMI.

When a machine event is generated it is propagated to a HMI application on the machine server that stores the real-time machine events. The machine event may be generated from either a simulated or real machine. The machine event is then broadcasted from the machine server to all end user HMI using a real-time communication protocol. The HMI(s) receive the machine event and display it to the end user

Table 5.4 Scenario involving HMI Displaying Real-time Machine Events

Scenario – Display Real-time Machine events

1. Machine generates event.
2. The event is propagated to a machine data application that stores real-time machine events on the machine server.
3. Machine application broadcasts the machine event information in real-time to all End User HMI's.
4. End User HMI receives and then displays the information to end user.

Table 5.5 details a scenario based on the HMI system displaying machine configuration information. An end user HMI requests all machine component names for a particular machine sub system as an example case of this scenario.

An End User HMI makes a request to the machine server for a particular type of machine configuration information. The machine server then invokes a HMI application that has the functional services to retrieve information from the machines configuration database. The request is then returned to the HMI application where it is then transferred to the End User HMI. The End User HMI then displays the requested machine configuration information the stakeholder.

Table 5.5 Scenario based on the HMI Displaying Real-time Machine Events

Scenario – Display Machine Configuration Information

1. End User HMI requests from the machine server a particular type of machine configuration to be displayed.
2. The request is processed by a HMI application that retrieved the requested information form the machine configuration database.
3. The requested machine configuration information is returned to the HMI application.
4. The machine configuration is transferred to the End User HMI where it is displayed to the stakeholder.

Table 5.6 details a scenario based where the end user HMI requests real-time machine information which for example maybe requesting the actual error events that have occurred while the machine has been operating.

An end user HMI requests real-time machine information from the machine's server for example actual errors that have occurred. The machine server then invokes a HMI application that retrieves the requested information from the machines real-time information. The requested information is then returned to the HMI application where it is then transferred to the end user HMI. The end user HMI displays the machine real-time information the stakeholder.

Table 5.6 Scenario based on requesting machine real-time information

Scenario – Request Machine Real-time Information

1. End User HMI requests from the machine server for a particular type of machine configuration to be displayed.
2. The request is processed by a HMI application that retrieved the requested information form the machine configuration database.
3. The requested machine configuration information is returned to the HMI application.
4. The machine configuration is transferred to the End User HMI where it is displayed to the stakeholder.

5.4 HMI System Functionality

A package diagram used in this method for designing the HMI systems architecture illustrates the high level logical grouping of the HMI systems required functionality. The functional requirements in the package diagram have been derived directly from the scenarios in the tables 5.3-5.6. The individual steps in the scenarios are composed into a set of high level logical groups, each of which define the complete required end functionality.

Defining high level logical groups and their associated functionality allows the architectural elements within each package to be specified and designed to meet package requirements.

5.4.1 HMI System Package Diagram

The package diagram consists of a hierarchy of packages, implemented as one or more software components in the real system. The top level package is the HMI System functionality which contains six sub packages; 1) end user HMI 2) machine data 3) machine interface 4) web server, 5) HMI applications and 6) the machine (Real or Simulated). These six HMI System packages are identified in this research as major functional aspects required in the HMI System.

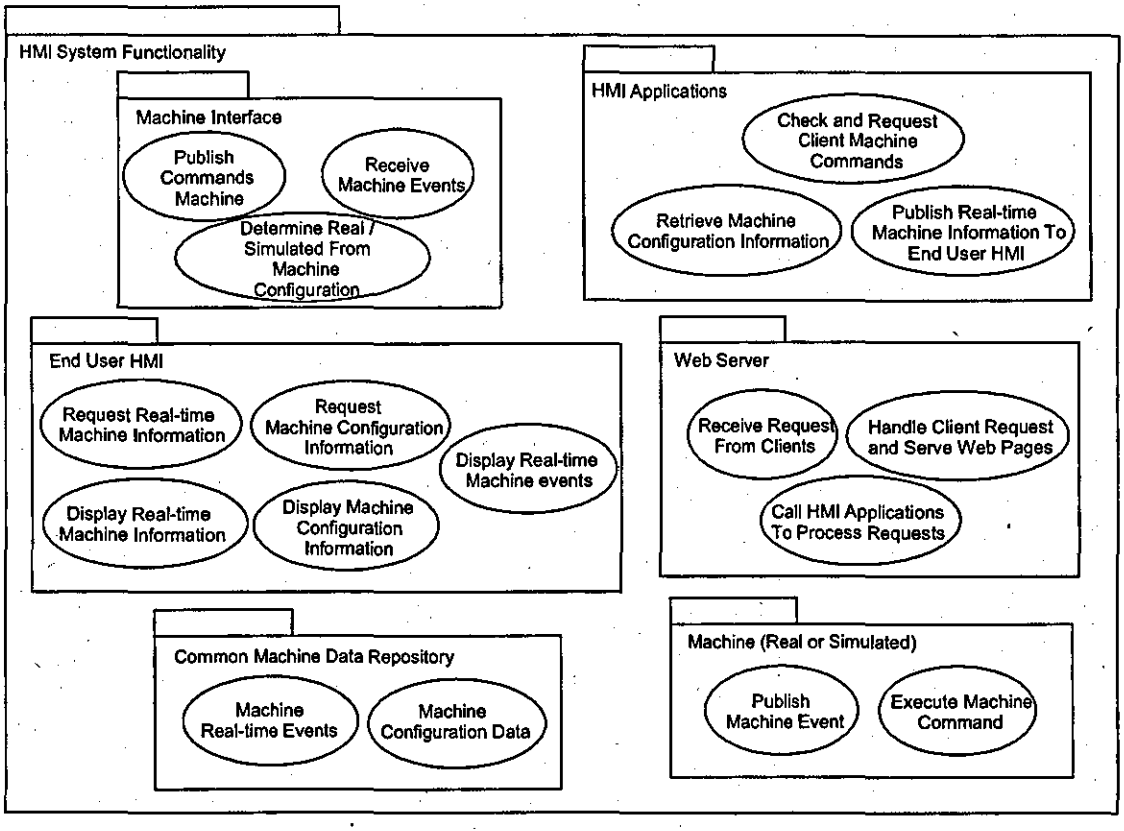


Figure 5.3 HMI System Package Diagram

End User HMI

The functionality defined in the end user HMI package must support all interactions the stakeholder (HMI end user) has with the HMI System. The HMI systems architecture is only concerned with the delivery of the machine information and handling end user HMI interactions. The particular end users HMI functionality will be implemented in the form of a HMI client (consisting of a set of HMI screens, the structure and modelling technique is described in chapter 4) that the stakeholder interacts with. The end user HMI must display all types of machine information to the stakeholder which may for example include, displaying machine configuration information, real-time machine information and real-time machine events. The end user HMI must also support stakeholders performing actions on the real or simulated machines which are for example requesting machine configuration information and real-time machine information.

HMI Applications

The HMI applications package was formed from the functionality in the scenarios that require application logic. This application logic for example may be checking that the machine commands are valid against the machine configuration, retrieving machine configuration information or publishing real-time machine information.

Machine Interface

The Machine Interface function is to provide the interconnection between the HMI System and the external machine. The external machine maybe either simulated or a real machine (see section 3.5 for an overview of C-B automation systems which has a simulation and or a real machine). The machine interface directs the machine bound information from the HMI System to either the real or simulated machine which is determined from the machine configuration information. The Machine Interface receives machine commands from the machine (real or simulated) and propagates the information to the required elements of the HMI System.

Web Server

The HMI System is web based and is derived from the HMI System requirement (see table 5.2, no.5) that globally distributed engineering partners must be able to increase machine up time by remotely diagnosing and supporting the machine. The HMI System must facilitate remote connectivity and many end users that may simultaneously use the system. All web based systems require a web server, which provides the principal access point for clients to retrieve web pages and execute web applications. The web server receives the requests for real-time and configuration types of machine information. The web server must call the appropriate HMI Applications to process the requests and then return the requested information to the client. The web server must also serve web pages to clients.

Common Machine Data Repository

The Common Machine Data Repository supports the storage of all machine data. There are two categories of machine data, 1) Real-time Machine Events and 2) Machine Configuration Data both of which have different characteristics.

The Real-time Machine Event is continually updated with machine events that are generated while the machine is operating. It is the within the machine data package that these events are stored and managed.

The Machine configuration information is all information that describes the configuration of the machine. This for example would include what are the machine components, their interlocking and sequencing logic, if they are real or simulated, machine request commands and modes of the machines operation. Accurate machine information is a non functional requirement of this systems architecture (see table 5.2 no. 1). The common machine data repository provides all machine information and is not fragmented across multiple systems with poor integration therefore information in the repository is always accurate. This is a key to fulfilling the requirements of global manufacturing where changes to the machines configuration frequently occur and must remain accurate across all sub systems associated with the machine system (machine controllers, HMI Systems).

Machine (Real or Simulated)

The HMI System controls or monitors either a real or simulated machine or a combination of the two (hybrid machine). Both types of machines have the same interfaces and functionality (they publish machine events and execute commands) from the HMI Systems viewpoint. The simulated machine would be a three dimensional solid machine model. The real machine defines the machine plant. The type of machine the HMI System controls or monitors is transparent because the interfaces are identical.

5.5 HMI System Architectural Design

The HMI Systems architectural design captures the design information and specifies how the functionality is combined into a real system that can be implemented. To fully describe the system architectural design we must consider; 1) the functionality captured in the design 2) the grouping and structuring of the functionality, 3) the interconnections required in the system. Very few systems are designed totally from scratch – generally systems are designed using a single or a collection of architectural patterns. Architectural patterns help designers build on the collective experience. They capture existing, well proven experience in software development and help to promote good design practice. Architectural patterns are used to construct software architectures with specific properties. A high level architectural pattern must be chosen and a high level mapping of the requirements onto the architectural pattern can be produced. The architectural pattern defines the structure of the system.

A requirement for the HMI System is for it to provide an end user HMI to many users who are often remote to the machine which is the requirement specified in table 5.2 no. 5. Using client server web architectures allows many clients using standard web browsers to request web pages that form an end user HMI from a web server (also fulfilling the requirement specified in table 5.2 no. 5 in supporting the access of the HMI to remote users).

The client server architecture is often a generic umbrella term for any application architecture that divides processing among two or more processes, often on two or more machines. The different variants of this architectural pattern are discussed and reviewed in chapter 2. There is no architectural pattern that will meet all the unique requirements of the HMI System which are detailed in table 5.7.

The HMI System is based on Client Server Web Application architecture. The HMI requirements and attributes of this architecture in meeting these mandatory requirements are detailed in table 5.7 and a generalised schematic of the Generic Client Server Web Applications Architecture is illustrated in figure 5.4.

Anytime, anywhere accessibility to the HMI System's is a supported feature that is inherent to the client server web applications architectural pattern.

Table 5.7 Suitability of the Client Server Web Application Architectural Pattern meeting the unique HMI Requirements

HMI Requirements	Solution using Client Server Web Applications Architectural Pattern
Display Real-time machine information.	Client Objects remote object communications with web applications.
Many remote users.	Many clients using standard web browser may simultaneously request web pages from principal access point web server.
Display machine configuration information to users.	Machine configuration information is stored in data repository on server and displayed to clients.
Request and process machine commands.	Clients request commands from server and web applications process commands.
Execute machine commands.	Not Supported in this architecture – Machine interface component is required
Receive machine events	Not Supported in this architecture – Machine interface component is required

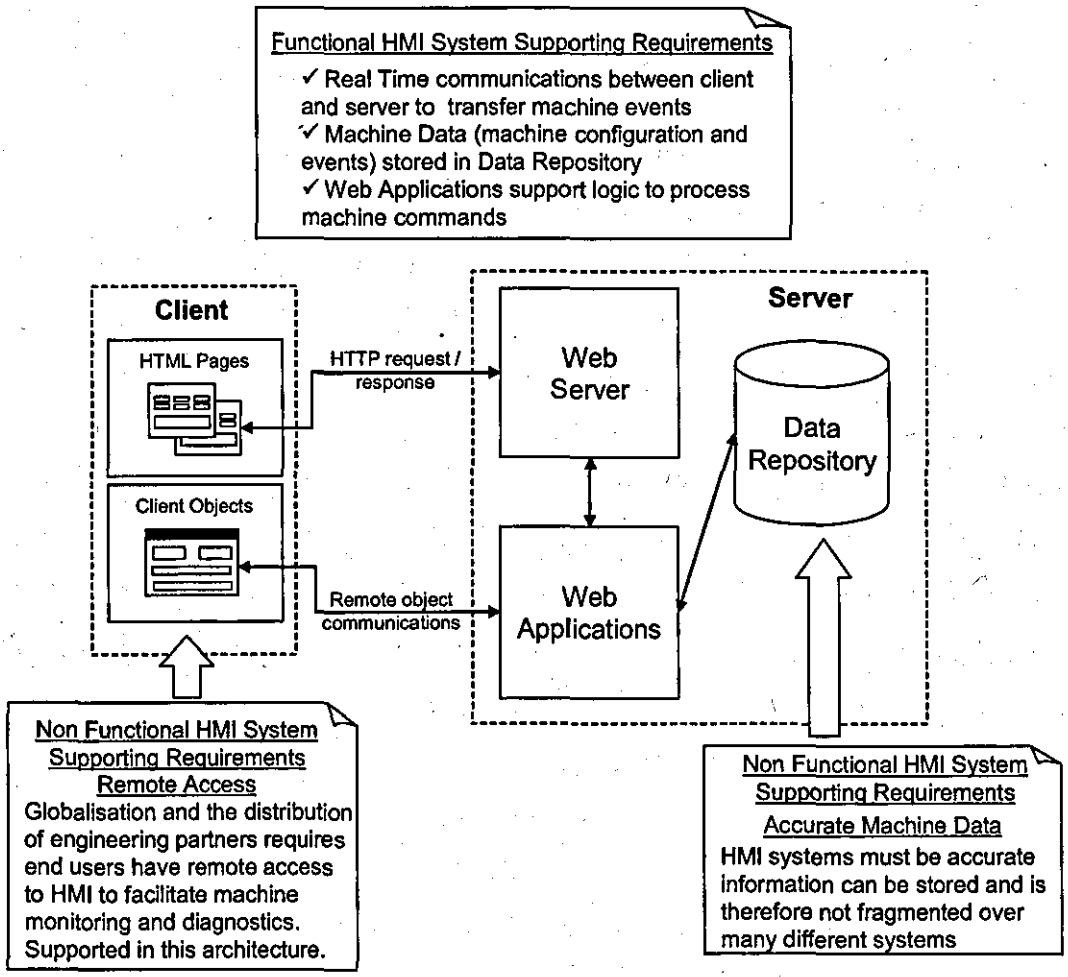


Figure 5.4 Generic Client Server Web Applications Architectural Schematic

The generic client server web application architectural pattern consists of a web server, web applications and a data repository that is on one or distributed on many servers. Clients use a web browser to perform a HTTP requests and retrieve web pages which provide the end user HMI from the web server. The client's web pages can contain objects (stand alone applications within the web browser) that have direct remote object communications with web applications. This architectural characteristic is a mandatory requirement for HMI System to enable real-time machine events to be displayed to clients. Without this communication channel information could not be displayed in real-time by the client.

The data repository in this architecture allows information to be retrieved using web applications and then displayed in the client using standard web pages requested from the web server or directly to client objects. All machine data is stored in a data repository on the server and then can be displayed to clients using either of these methods depending on the characteristics of the machine information - real-time machine data (machine events) is displayed using client objects and non real-time machine data (machine configuration) is displayed using standard HTML web pages.

Machine commands are requested by the client and processed by the HMI Applications on the server. However this generic architecture does not support any communications with external machines to execute the commands. This generic architecture has been enhanced to support the unique requirements of the HMI system with a machine interface software component which is illustrated in figure 5.5. This additional component in the architecture allows the HMI system to communicate with the machine where machine commands can be transferred and executed. This also allows the HMI System to receive machine events where they can be stored in a data repository and processed by web applications.

Figure 5.5 also illustrates the deployment of the functional packages (from the package diagram in figure 5.3) onto the HMI System architectural implementation that has been described.

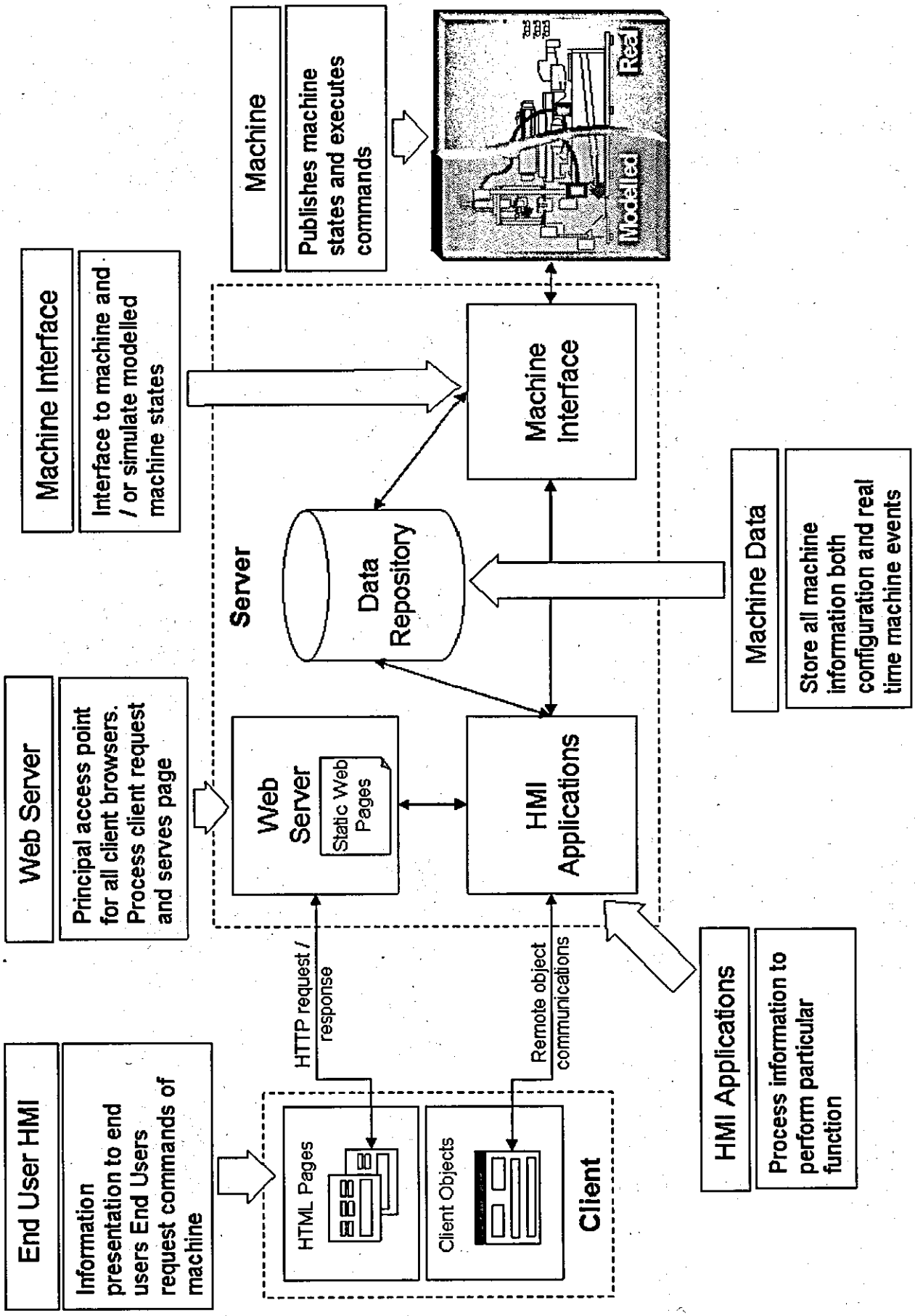


Figure 5.5 HMI System Architectural Schematic based on Client Server Web Application architectural pattern

It is an essential requirement of the HMI System to be an accurate representative of the controlled automation production machine. As the frequency of new products is increasing manufacturing machines are more frequently reconfigured. It is therefore imperative that the HMI systems architecture should ensure that the HMI system provides accurate and consistent information about its associated production machine. The HMI systems architecture proposed in figure 5.5 supports this requirement by having all machine information stored in a common machine data repository.

5.6 Principal HMI System Elements

The principal architectural elements in the HMI system are derived from the functionality in section 5.4 and the relationships between these elements are established in section 5.5 which describes the systems architectural structure. The functionality of each of the six packages is met by one or more architectural elements.

Figure 5.6 illustrates the architectural elements in the HMI System and shows the relationships between them. The elements that are deployed to the client are on the left hand side and the machine and server elements are on the right hand side of the figure.

Each rectangle is a architectural element which represents a logical unit of functionality. The line between elements represents an interconnection and the details of (type of information, direction) are described in the architectural elements description.

An arrow between any two elements shows a relationship between them, for example, "element A" is dependant on "element B". The arrowhead describes the direction of dependency, and the name attached to the arrow describes the relationship.

Client browsers request web pages (which form the end user HMI) from a web server using standard HTTP internet technologies. The web server can serve static web pages from its file system or create them dynamically using scripts. The dynamic scripts are executed using a scripting engine which retrieves and or writes information to the machine configuration database. The machine configuration database stores all information regarding the machines configuration.

A client browser renders or marks up a webpage into a screen which HMI end user interacts with. A web page contains one or more frames. Frames provide a mechanism for dividing a web page into a number of screen compartments. Each frame contains one or more HTML pages which contain one or more different types of HMTL elements. The HMTL elements support different functional aspects of the client HMI. To enable users submit information to the HMI system HTML forms are required. HTML forms submit information to the web server using HTTP requests. The client HMI supports the presentation of information to users

through HMTL table, images and text. The client HMI implements HTML links to support the user in navigating between different HTML pages. Logic and control of these HTML elements can be executed using client side scripts. Objects in the client HMI are software applications within the HTML page that have capabilities to support any type user interaction.

The client objects communicates directly using a Remote Object Transfer Protocol (ROTP) with a broadcaster architectural element which is on the machine server. This communication mechanism (ROTP) is used to transfer real-time information between the server and client. An instance of a broadcaster is created for each connected client.

The broadcaster receives all machine event information from the machine blackboard. The machine blackboard stores all real-time information from the machine interface. The machine interface provide the communication interface between the server and the machine which maybe real or simulated. The machine interface receives all events and transfers them to the machine blackboard. When machine commands are requested the machine interface must determine if they must be executed on the real or simulated machine using information in the machine configuration database. The servlet engine handles all machine command requests from the web server and determines the correctness of the command and transfers then command to the machine interface.

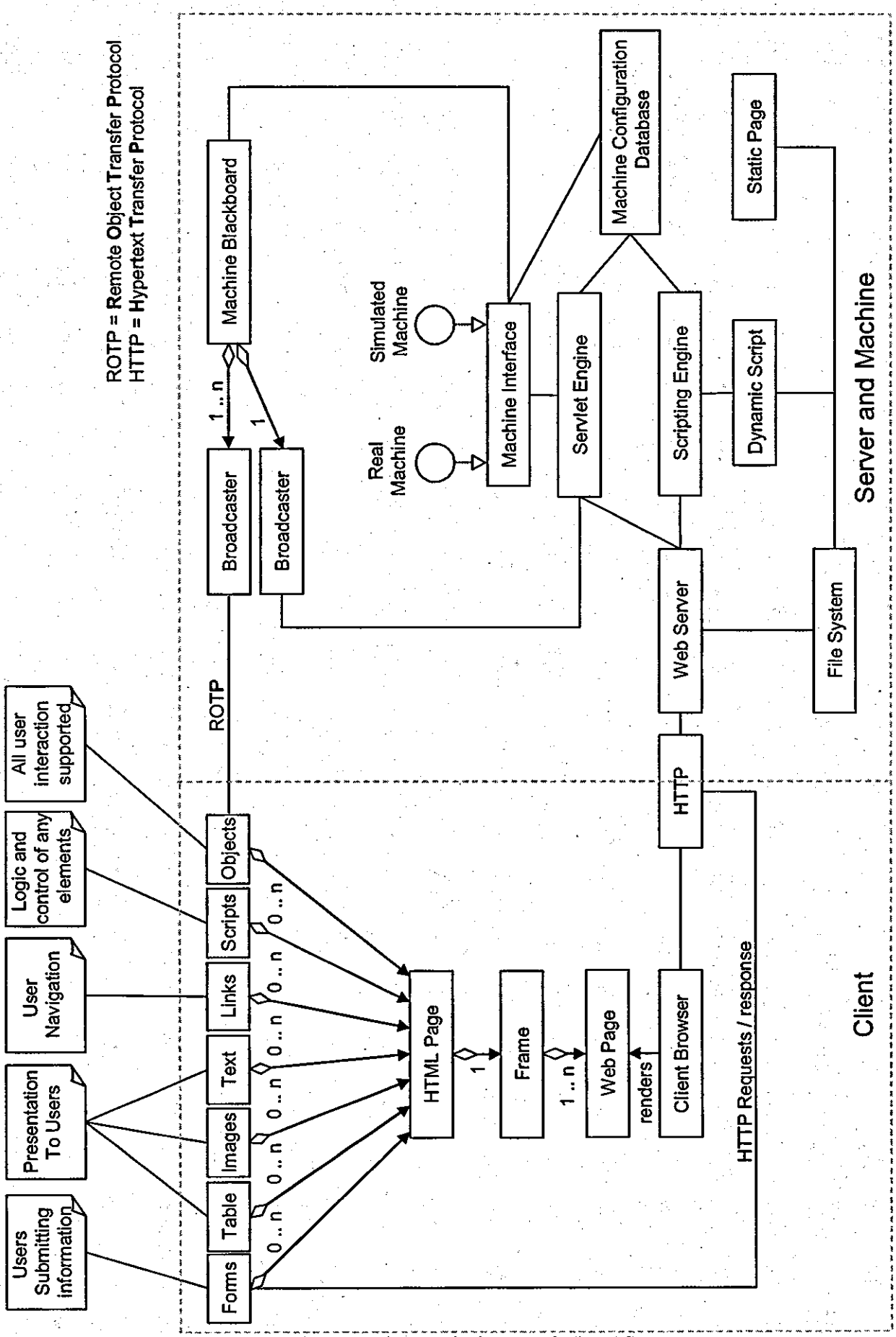


Figure 5.6 Principal Architectural Elements in HMI System

5.7 HMI System Architectural Elements Design

The HMI System architectural elements are detailed in this section. Each of the architectural element function and it operating process is described. The implementation that must be adopted is standardised web based technologies.

5.7.1 Web Page

Figure 5.7 illustrates the end user HMI process. The end user HMI is implemented as set of web pages. The initial step in the operation of the end user HMI is for the web browser to request a URL of the required HMI. This URL (figure 5.7, ref 1) specifies the global address of the web server and the top level web page. The web server returns the requested web page that contains one or more frames (figure 5.7 ref 2). Each frame contains a HTML page which contains several different types of one or more HTML elements all of which are downloaded to the End User HMI from the web server. The HTML Link elements request (figure 5.7 ref 3) different HTML pages which can be returned to any of the frames. This supports the navigational functionality required in the End User HMI.

HTML Forms support the End User in requesting information. HTML Forms specify 1) Machine Error Information, 2) Machine interlocking Configuration or 3) Machine Commands that are required in the End User HMI. These requests are sent to the web server where they are distributed and executed and a response in returned to the HTML Form.

Client objects (applications within the HTML page) are used to transfer real-time machine events information between the server and client. The client objects request a connection and subsequently communicate directly using a Remote Object Transfer Protocol with a broadcaster architectural element which is on the machine server. The broadcaster transfers all real-time machine information to connected clients where they display the information to the user. An instance of a broadcaster is created for each connected client.

The end user HMI has intercommunication between all HTML elements. This supports for example real-time information in the client object updating images, tables, text and the HTML page.

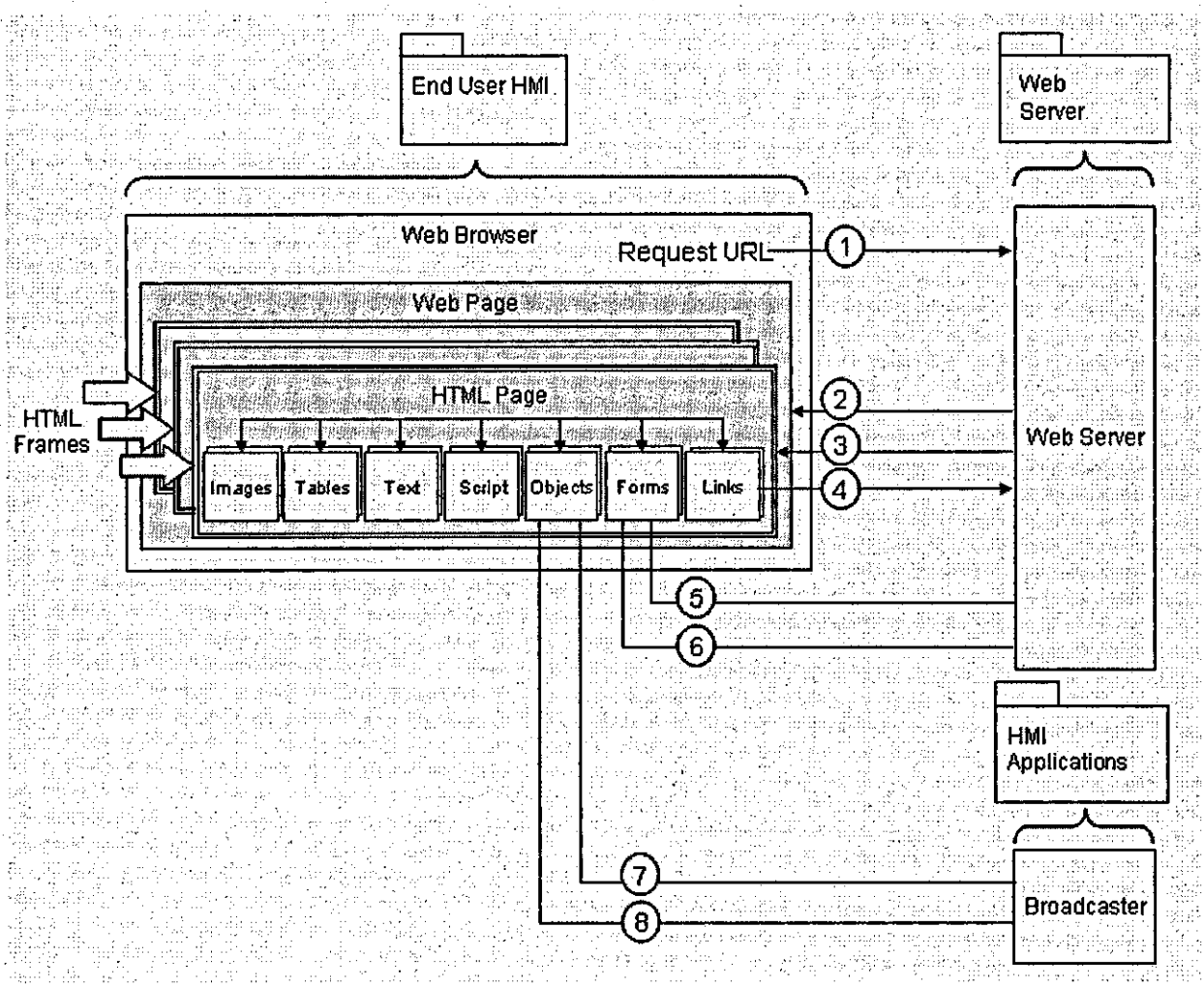


Figure 5.7 Operation process of the End User HMI

5.7.2 Machine Broadcaster

The principal functionality of this architectural element is to broadcast machine events using real-time data communication mechanisms to client objects in the end user HMI. The broadcaster supports the requirement (figure 5.3 package diagram) in the HMI Applications to publish real-time machine information to End User HMI's.

End User HMI contains objects that request a connection to the broadcaster (figure 5.8 ref 1). The broadcaster has a socket permanently open that is listening for connection requests. When a connection request is received by the broadcaster a new broadcaster thread is created and requests to subscribe to the machine events information on the machine blackboard. When a subscription is requested for the machine events the blackboard returns all current

machine events and all subsequent events. The machine events information is transferred to the client object through a new socket instance that is created by the listening socket when the client requests a connection. This new socket instance is dedicated to the client object and transfers all real-time machine event information to the end user HMI.

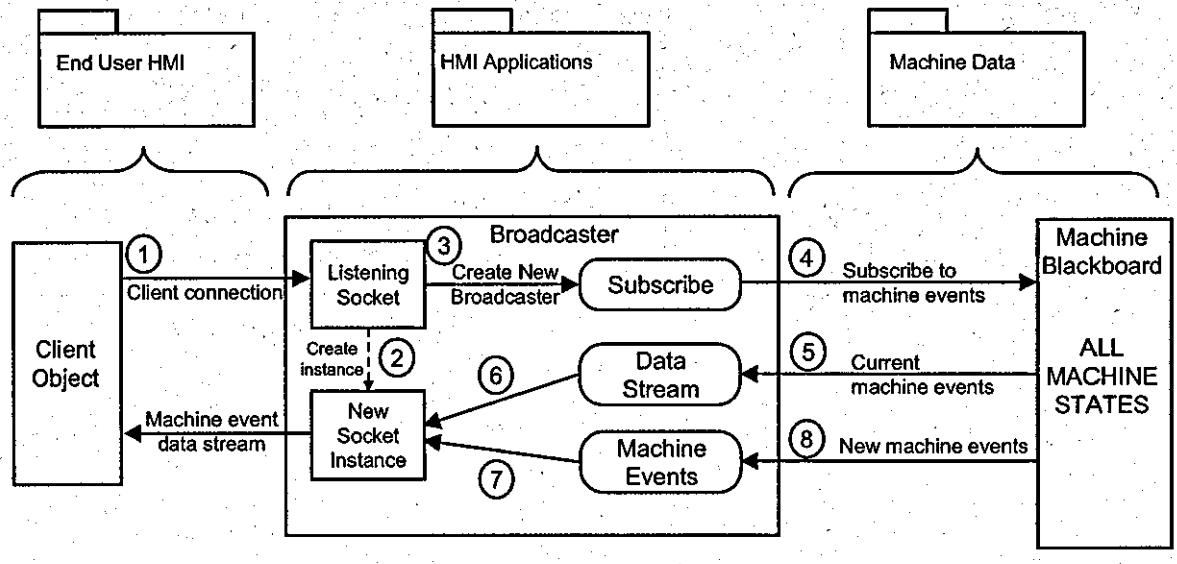


Figure 5.8 Machine Broadcaster operating process

5.7.3 Machine Blackboard

The blackboard is an architectural element that supports the machine data functionality (figure 5.3 package diagram) by providing a data service to capture and distribute the real-time machine information. The blackboard receives and stores all machine events to provide HMI applications with machine information.

The machine blackboard consists of a connection manager and machine event storage.

The machine events storage is implemented as an area of memory used to store a continuous stream of data received from the machine interface. A circular buffer mechanism increments to a the next memory location when new machine event is received, starting again at the beginning of the buffer after reaching the end.

There are separate read and write pointers to access this circular buffer. There is only a single pointer that writes to the circler buffer which is connected to the machine interface that handles all machine events from both the real or simulated machine. Read pointers are created for each broadcaster in the system. The connection manager handles all housekeeping

and management of the circular buffer. It creates new read pointers and a reference to the broadcaster. When a broadcaster subscribes to the machine events service the connection manager publishes all current machine events. A read pointer then publishes subsequent machine events to the broadcaster.

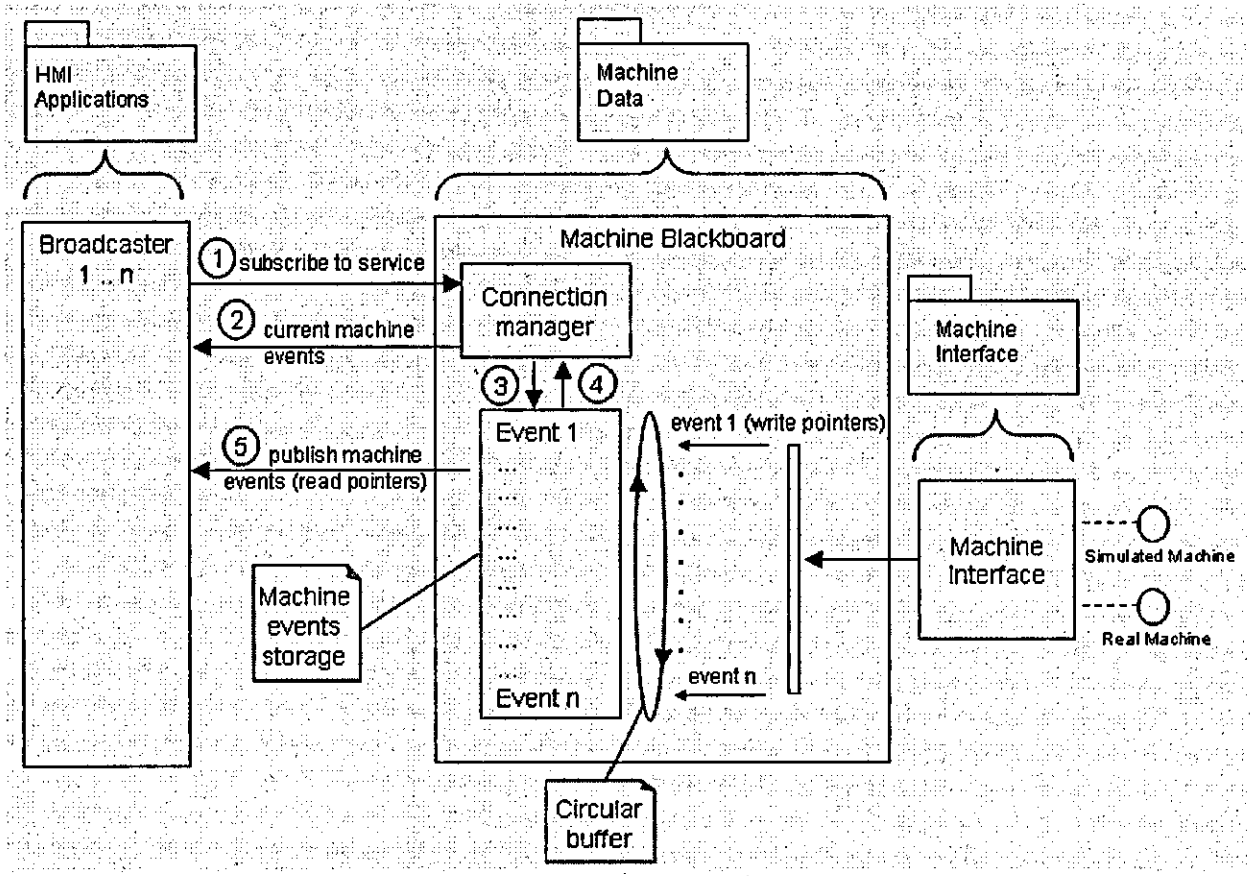


Figure 5.9 Machine Blackboard operating process

5.7.4 Web Server

The web server must support all requests from one or more end User HMI clients which can be HTML over HTTP, or in case of rich end user HMI clients, the client can act as a web service entity and use direct HTTP interactions with server applications.

The clients request web pages from the web server which may be 1) static or 2) dynamic. Static web pages remain constant and are stored on the web server file system. Dynamic web pages store a script on the web server file system which is executed on a scripting engine. The web server must invoke this scripting engine when the dynamic scripts are requested by clients.

Rich End User HMI requirements support interaction with the server beyond downloading of the web page. This is for example an End User HMI requesting machine commands from a web page. The command is send to the web server using a HTML request mechanism. The web server receives the request and they must invoke a HMI application to process the command and return a response. This response is then returned to the requested web page where it can be processed.

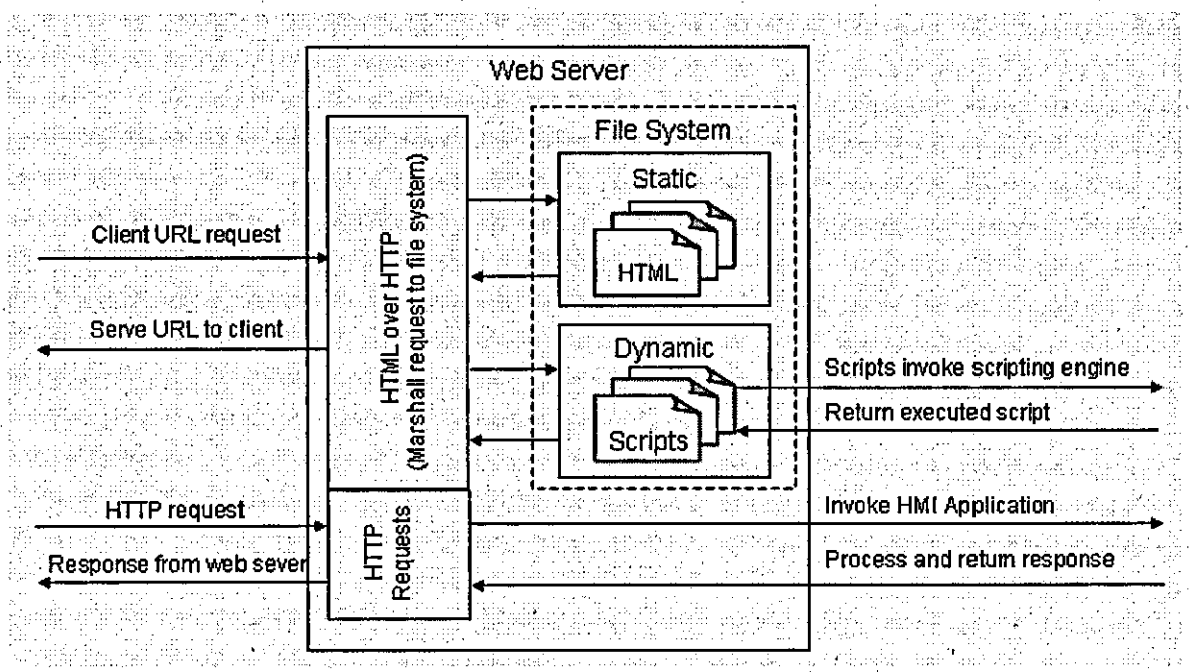


Figure 5.10 Web Server operation

5.7.5 Scripting Engine

The scripting engine architectural element supports the web server functionality in processing client requests to serve web pages that require information from the machine's configuration information.

Dynamic scripts located on the web server and each HTML page may contain a number of script expressions. Each script expression specifies a piece of code generating HTML code and/or controlling the generation of HTML code on the page.

The scripting language can request machine configuration information and then generates HTML code for the End User HMI. The major advantage of creating a web page dynamically is that machine configuration information can be requested and dynamically creates web a

page ensuring there are no consistency issues between the webpage information and the machine configuration because the web page is created on every HTTP client request. This is a very important benefit to HMI Systems. The system architecture ensures that other applications that modify the machines configuration during its lifecycle do not introduce consistency errors due to the machine configuration always generating the HMI web pages.

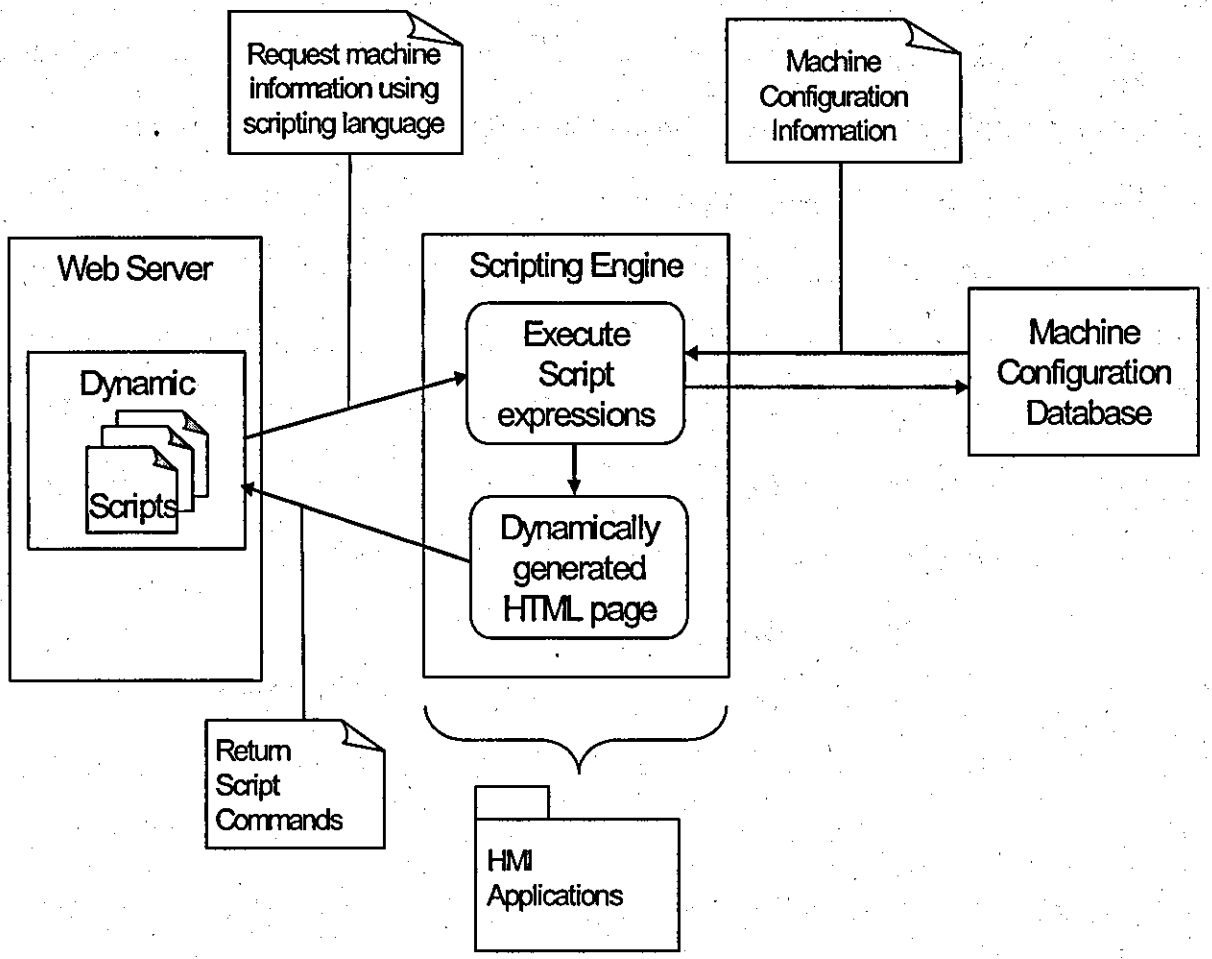


Figure 5.11 Scripting Engine Operation

5.7.6 Servlet Engine

The Servlet Engine is a core part of the HMI applications functionality. The servlet Engine provides the application logic to; 1) check and request client machine commands 2) retrieve machine configuration information and 3) request run-time machine errors. These requests are made from the web server.

The Servlet Engine receives all machine events and stores them in a machine events array. When the web server makes a request for machine error information the machine error events

logic in the Servlet Engine determines the errors from is machine events array. This error information is then formatted and returned to the web server. Different types of machine errors may be requested by the web server dependant on the machine's application.

The web server can request machine commands using the Machine Control Command logic that is in the servlet engine. This application logic checks if the command is valid in terms of 1) it is a recognised command and 2) the machine is in the correct state to execute this command. The machine control command logic communicates with the machine configuration database and the machine events array to execute this application logic. If the application logic conditions are satisfied then the machine command is sent to the machine interface where it is executed. Machine configuration information is request from the web server. This information maybe for example machine components interlocking configuration information. This machine configuration information is requested and retrieved from the machine configuration database and then formatted and returned to the web server.

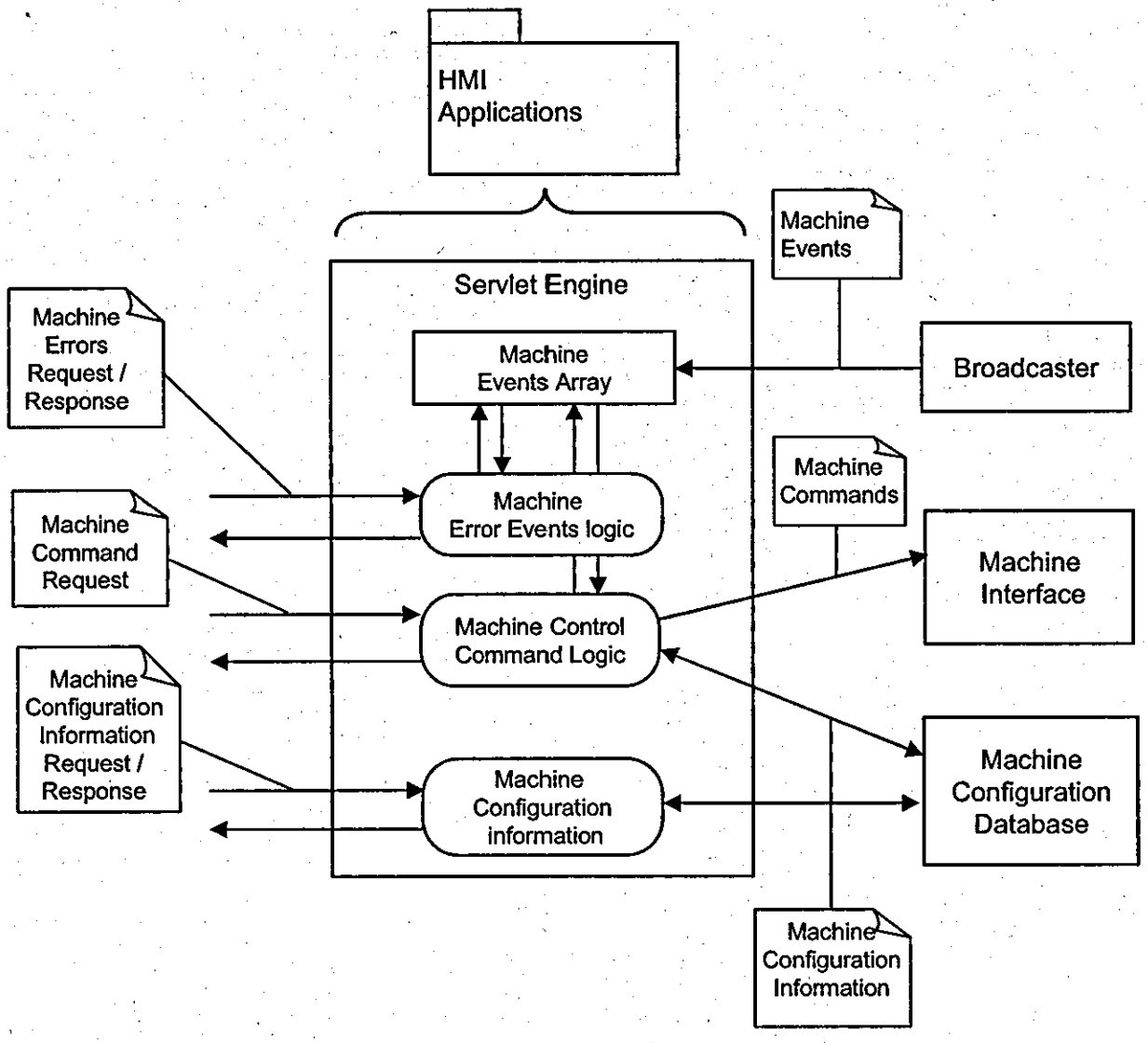


Figure 5.12 Servlet Engine Operation

5.7.7 Machine Interface

The machine interface function is to transfer information bi-directionally between the machine server and the machine. The machine maybe real or simulated and the application logic in the machine interface must determine the correct type of machine to route commands.

Machine commands are received by the machine interface from the servlet engine. The machine interface application logic communicates with the machine configuration database where the machine type of all machine components is specified. The application logic can then route the command to the relevant machine type (real or simulated) via an interface. These commands can now be executed by the machine. The real or simulated machine

generates machine events that are received by the appropriate interface and then are propagated to the machine blackboard which mangers all machine states.

The communication link between the real machine interface and the real machine would be a control network. The Simulated machine would want to be viewed in tools that support the engineering design of the machine or remotely using World Wide Web technologies. The communications link between the simulated machine interface and the simulated machine is TCP/IP based.

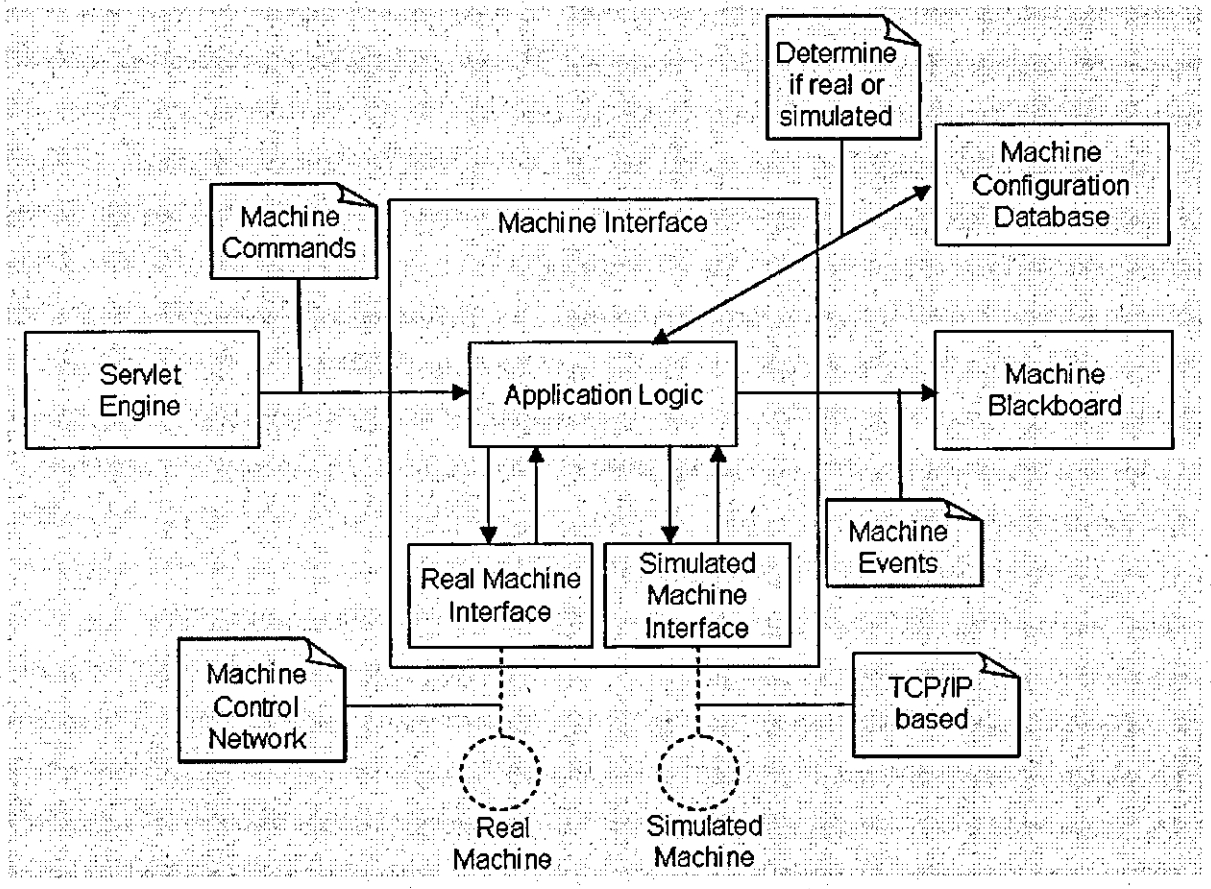


Figure 5.13 Machine Interface

5.8 HMI System Sequence

The sequence in which the HMI System executes the use cases interactions illustrated in figure 5.2 is described using sequence diagrams. These sequence diagrams describe how messages are propagated through the architectural elements in the HMI system to perform a

particular functional requirement. This validates the design of the architecture by testing that is meets it's required functionally.

5.8.1 Presenting Machine Configuration Information

The dynamic sequence of the system illustrated in figure 5.14 describes how the HMI system architecture handles a scenario where the End User HMI is presenting machine configuration information. The first stage in the system sequence is for the client browser to marshal the request via HTTP. HTTP then requests the URL on the web server. The web server resolves the URL and processes the filename request, the dynamic script. The dynamic script then executes the scripting expressions which invokes the scripting engine. The scripting engine retrieves machine configuration data from the database. This machine configuration information is returned to the dynamic script which marshals the response via the HTTP the client browser.

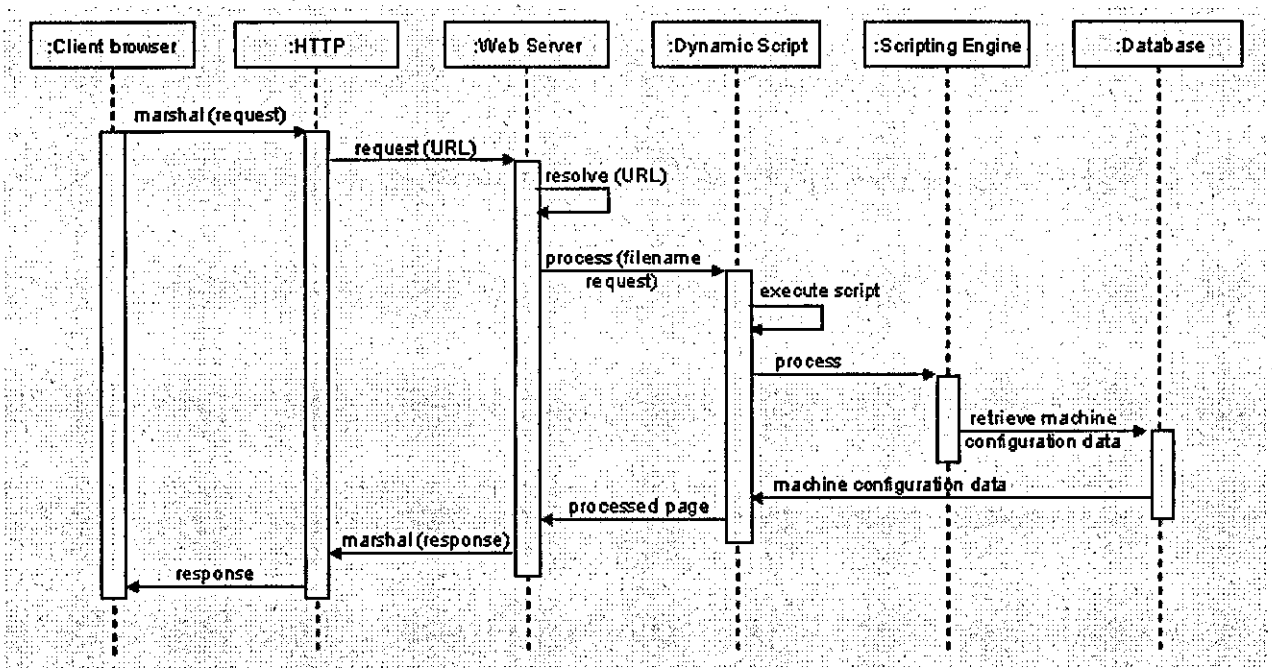


Figure 5.14 End User HMI presenting machine configuration information execution scenario

5.8.2 Requesting Machine Commands

The system sequence for End User HMI requesting the execution of machine commands is illustrated in figure 5.15. HTML forms in the client browser generate a HTTP request. This request is marshalled to the web server using HTTP. The web server resolves the URL and invokes the Servlet Engine that executes command logic. This logic requires information

from the machine configuration database to evaluate the logic and then call the machine interface to execute the machine command. The machine interface then requests data from the database to determine if the command is sent to the real or simulated machine. When the Servlet Engine has executed the machine command a return response is propagated back through the web server, HTTP, client browser to the HTML form. This provides a confirmation to the HTML Form that the request has been received.

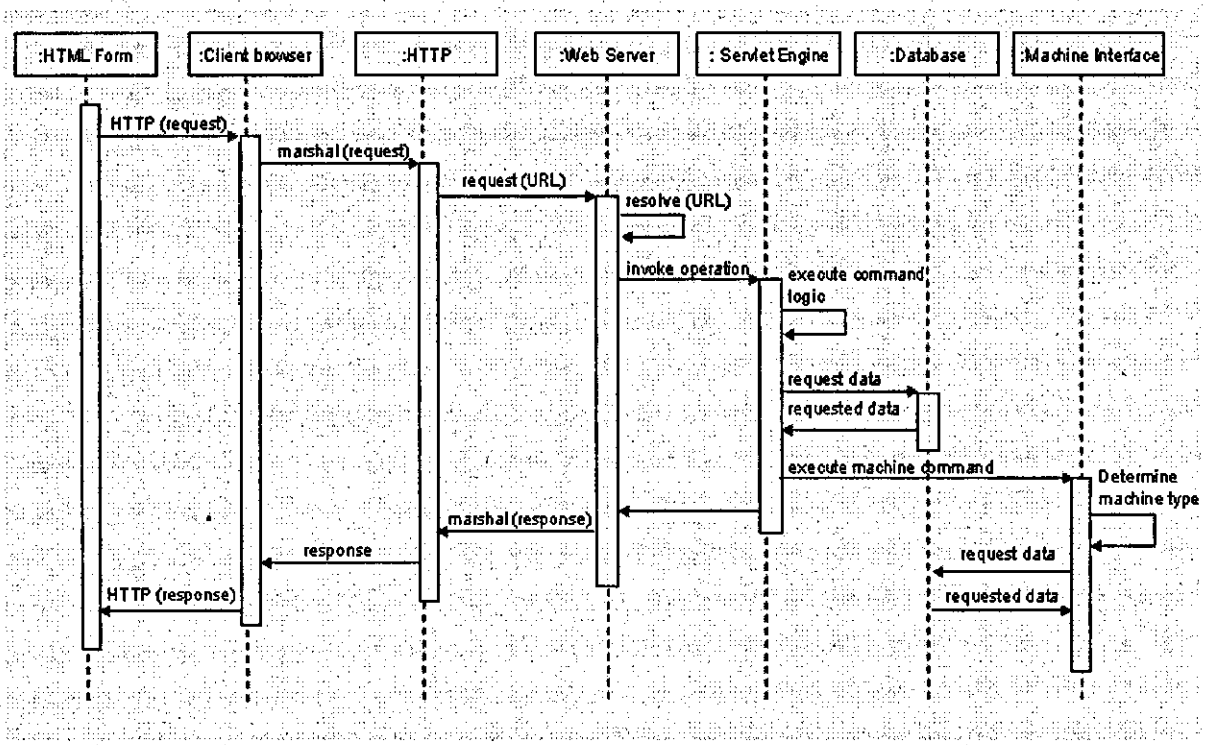


Figure 5.15 HMI System sequence for requesting machine commands

5.8.3 End User HMI Displaying Real-time Machine Events

The end user HMI displays the real-time machine events. The machine generates events during its run-time operation and publishes them on the machine control network. The events are received by the machine interface and then propagated to the machine blackboard. The machine blackboard stores all the machine events and then calls the broadcaster threads. Each broadcaster marshals the information to a ROTP (Remote Object Transfer Protocol) which transfers the machine events data the client objects. These client objects display the machine event information in the end user HMI.

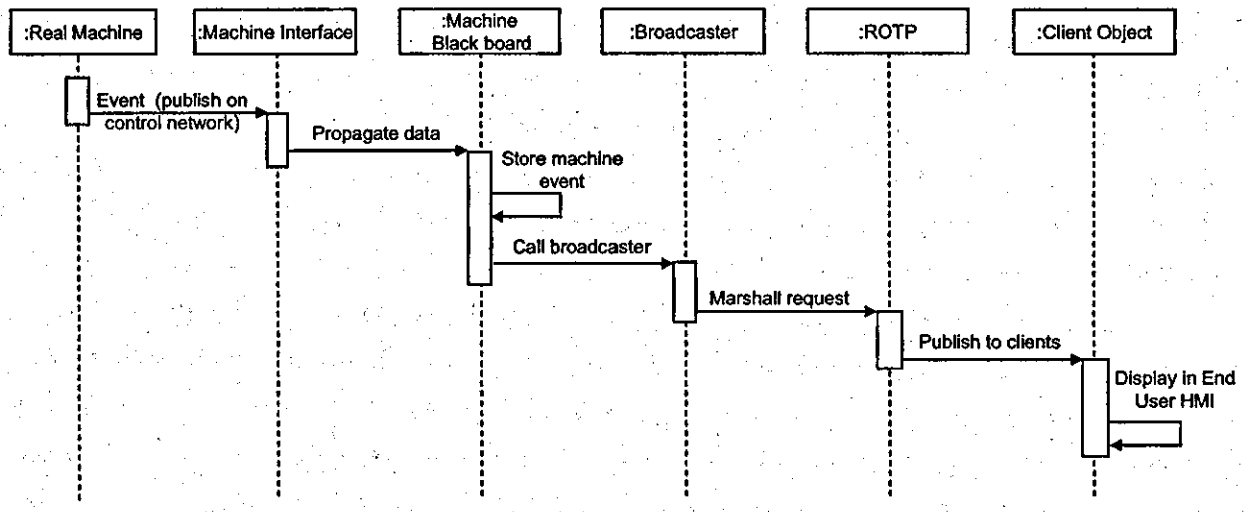


Figure 5.16 System sequence for End User HMI Displaying Real-time Machine Events

5.8.4 Requesting Real-time Machine Information

Figure 5.17 illustrates the system sequence for requesting real-time machine information. HTML forms in the client browser generate a HTTP request for the type of real-time machine information. This request is marshalled to the web server using HTTP. The web server resolves the URL and invokes the Servlet Engine that executes logic to get the real-time machine information from the machine events array. This information is then returned to the web server where the response is marshalled to the client browser to provide a HTTP response.

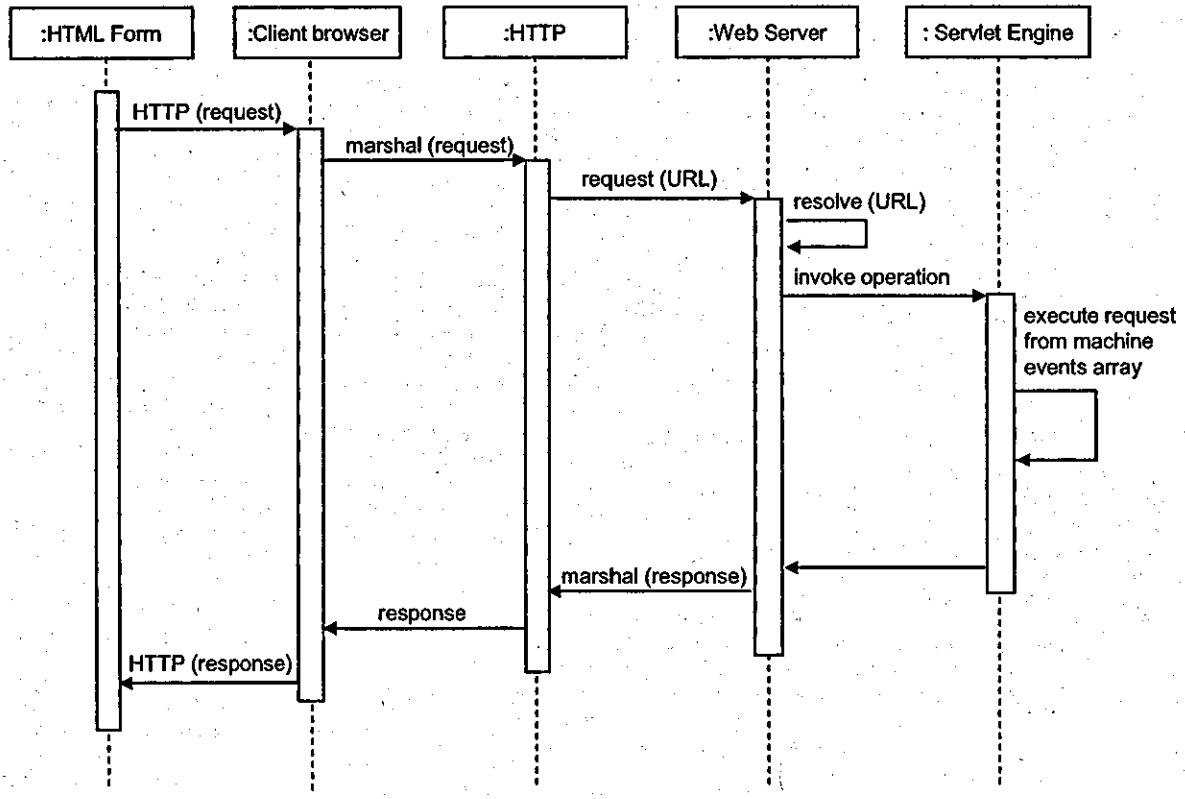


Figure 5.17 System sequence to request Real-time Machine Events Information

5.9 Summary

The HMI system architecture that has been described in this chapter has been designed to support the emerging requirements of automated manufacturing systems.

The HMI architecture meets the unique characteristics of HMI systems by further developing a recognised client server web applications architectural pattern. The architectural elements executable process and interfaces are described and mapped to software components. The HMI system architecture is validated using sequence diagrams that prove the design is meeting the functional requirements.

The structure and partitioning and the system elements supports scalability in applying the HMI system architecture to different types of automotive production machine applications.

A common system data repository stores all machine information. A common data source for all machine information ensures the HMI always has machine accurate information and eliminates problems of inaccurate machine information being fragmented across different systems. Using the client server architecture the HMI is accessible to end users via an internet enabled PC using a standard web browser. This supports the globally distributed engineering partners who are involved in the engineering of the automotive production machine. It is suggested that the remote monitoring and diagnostic with increase the efficiency of the machines.

In chapter 6 the process of constructing an HMI using the architecture described in this chapter will be explained from the perspectives of the stakeholders involved.

6 HMI System Construction

6.1 Introduction

In this chapter the processes involved in constructing an end user HMI will be described. The systems architecture (chapter 5 of this thesis) describes the executable process of the HMI system. The HMI system architecture has been designed to support machines within the context of automotive manufacturing automation systems. This chapter describes process, tools and resources that the stakeholders require to implement an end user HMI for a particular machine application.

To construct the end user HMI software components must be developed. Three different types of software components are required to implement end user HMI which correspond to the end user HMI functional structure described in chapter 4 of this thesis. The highest level element defined in this defined structure is implemented as the end users HMI client which is a fully functional HMI that supports the end users in fulfilling their enterprise role. A HMI configuration tool developed in this research and described in this chapter supports directly inputting of a storyboarding model in order to automatically generate a complete end user HMI. The development process for lower level components that are required to facilitate the composition of an end user HMI client, (HMI tasks and HMI widget components), are described from the stakeholders role(s) and the implementation technologies perspectives.

The processes described in this chapter to construct a end user HMI are subsequently implemented using the case studies to be described in chapter 7 of this thesis.

6.2 Context of HMI within C-B Automation System

The goal of the C-B automation paradigm is to fully support the complete lifecycle requirements of automated machines. The fundamental concept of this paradigm is to compose complete automated machines from modular machine components.

The major elements of a component based automation system are; 1) an integrated engineering environment to support the machine throughout all lifecycle phases, 2) a common machine data repository which contains all machine information and 3) machine components that can be either real or simulated.

An integrated engineering environment consists of an extendable set of engineering tools that can be used by a set of globally distributed engineering stakeholders at all phases of the machine's lifecycle. These tools include a Process Definition Environment (PDE) which supports the process planning, machine design and sequencing, interlocking logic and provides simulation tools to test the machine.

The lifecycle of a machine can be split into two major phases, 1) machine design and installation and 2) the machine operating and reconfiguration phase [88]. The second lifecycle phase encompasses all machine reconfigurations, from minor machine interlocking changes to major machine recomposition.

The initial stage in the machine design and installation phase involves determining the specification for the new machine. The process engineers plan the operation of the machine from the manufacturing process requirements that the new machine must support. Machine engineers then select the machine components that meet the specification of the new machines requirements. This is followed by control engineers inputting the sequencing and interlocking logic of the machine. The PDE tools allow this information (sequencing and interlocking logic) to be inputted using a high level representation (state based diagramming) that defines the machine process. The control logic can be validated using the machine simulation tools. These tools provide a three dimensional solid model of the machine enabling the machine logic to be tested before the physical machine is built. The end user

HMI's are configured using a tool within the integrated engineering environment and would typically be done by the process engineer. This tool supports the selection of an exiting end user HMI or the design of new end user HMIs from a set of HMI tasks that the intended end user must perform. The HMI can be tested using the machine simulation in the PDE.

The second phase in the machine's lifecycle is the operating and reconfiguration phase. During this phase the HMI is used to support diagnostics, maintenance and operating functions. The machine can be reconfigured to meet new product requirements. Again the integrated engineering environment fully supports this lifecycle stage. During the operating lifecycle stage the end user HMI is displayed using a web browser to provide the accessibility to stakeholders in performing diagnostics, maintenance or operating roles.

As illustrated in figure 6.1, a single common machine data repository is used throughout the machine lifecycle to store all information related to the machine. All the data is structured in accordance with the C-B architecture [23]. The essential idea is that the information is defined once but used many times by all stakeholders throughout the machines lifecycle.

The machine components can be either real or modelled. Modelled machine components are simulated using three dimensional solid models. Real machine components are physical modular parts (actuator, sensors, servo drives) of the machine which contain embedded machine interlocking and sequencing logic.

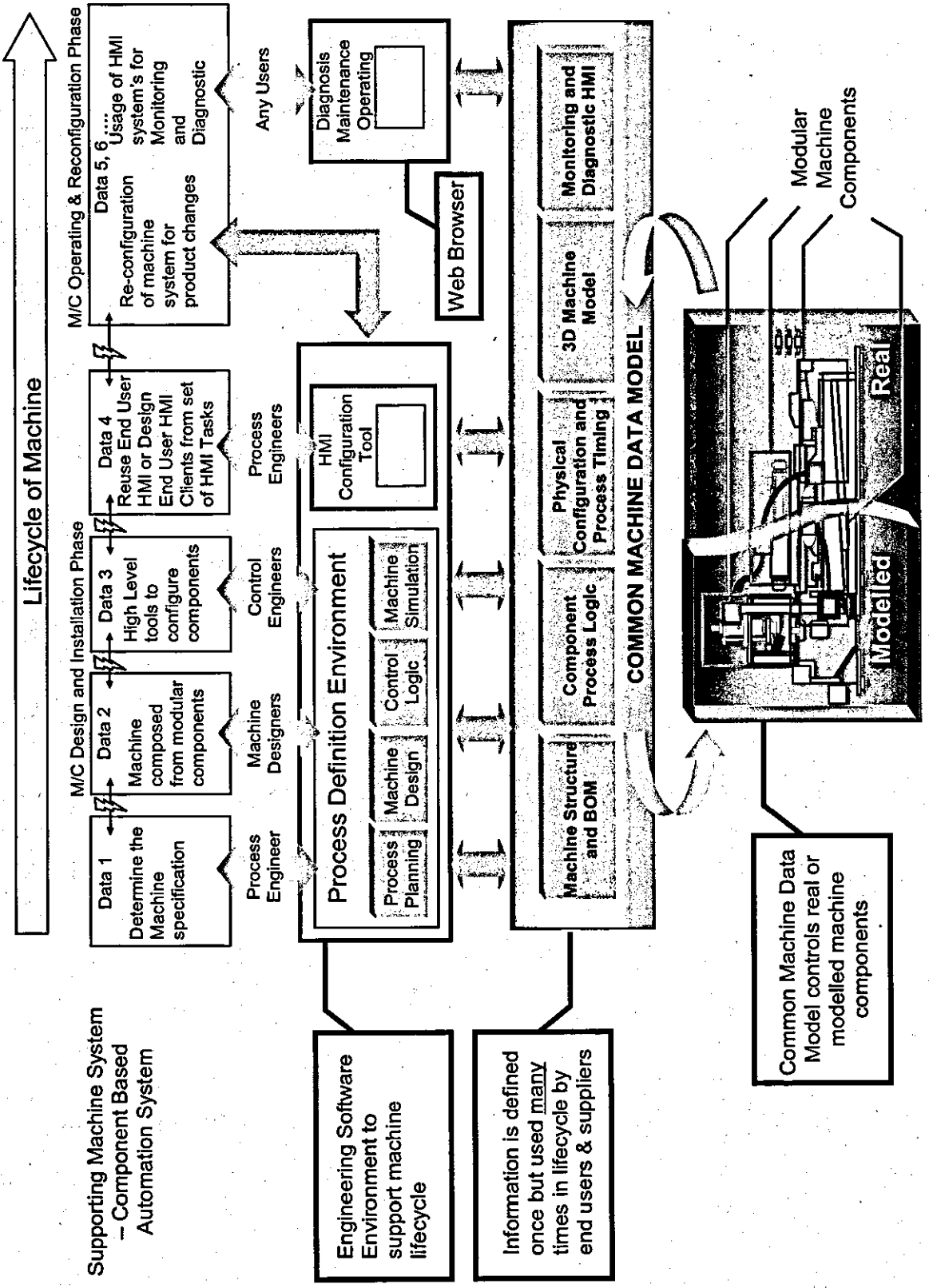


Figure 6.1 Component Based Automation Supporting Machine System

6.3 End User HMI Implementation

The HMI end users functionality is described using a well defined structure and associated storyboarding modelling technique in chapter 4 of this thesis. The structure of the end user's functionality consists of three levels that relate to different levels of granularity; 1) End User HMI's which are composed from one or more HMI Tasks, 2) HMI Tasks which support a particular aspect of the HMI functionality required and 3) HMI Widgets that provide the functionality to directly interact with an aspect of the controlled machine. The elements within this structure are implemented in the form of software components that are used to compose a complete fully functional end user HMI client. The software components implemented in the HMI system directly maps onto a corresponding element within the HMI end users functional structure.

Figure 6.2 illustrates the three types of software components used to construct end user HMI's and their attributes in development process.

The role of the HMI Widgets is to provide the functionality that facilitates direct interaction with one or more machine components. HMI Widgets support the HMI system in providing component information to the end user. Machine component design engineers develop these components that allow specialist knowledge to be embedded in the machine component and displayed to the end users. These HMI widgets provide a standard way to support machine builder's interaction with machine components. One or more HMI Widgets are combined to develop a HMI task component. Machine process engineers specify the machine tasks that can be performed and machine HMI engineers work from these specifications to develop the HMI tasks software components. One or more HMI tasks components are combined to compose a complete end user HMI client. Enterprise management and or the HMI end users can specify what machine tasks they must perform on the machine and select the relevant HMI machine task components. End user HMI's are composed using the HMI configuration tool in the integrated engineering environment that support direct inputting of the end user

HMI for a storyboarding model that specifies the functionality of the end user HMI. This enables users with no programming skills to compose HMI system and hence they can be rapidly configured.

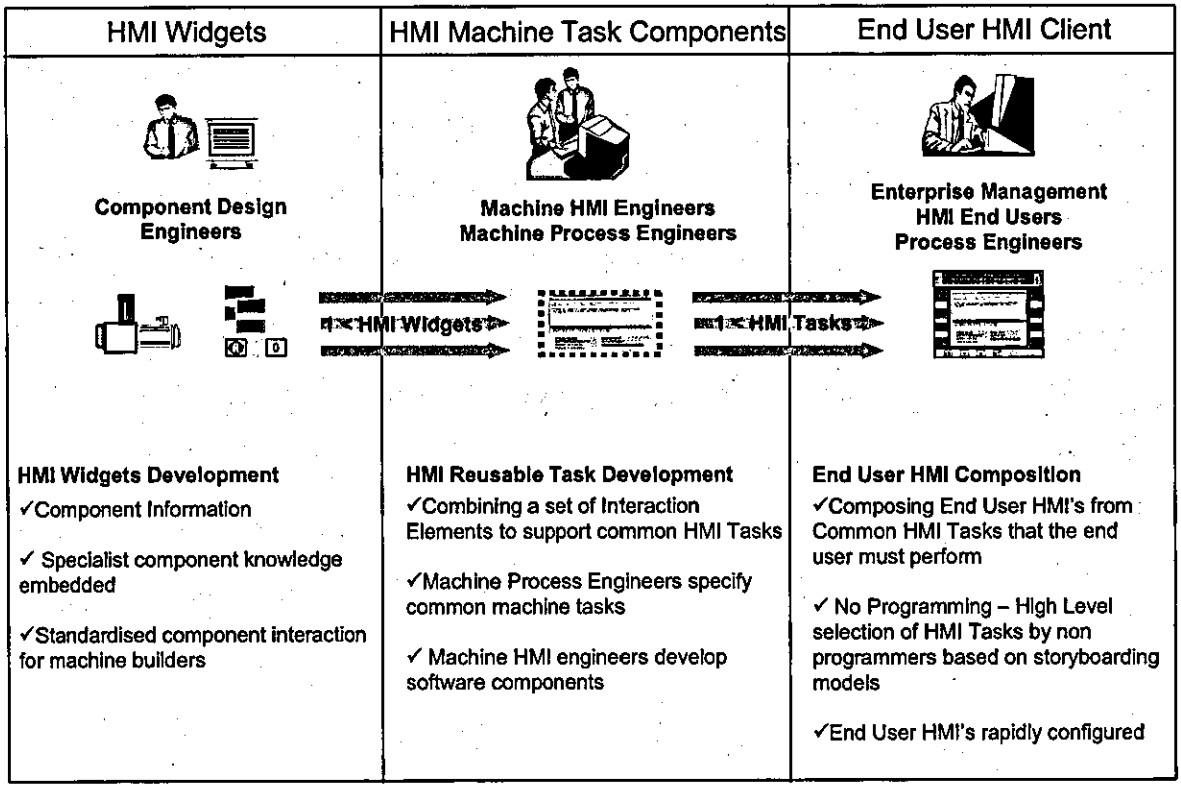


Figure 6.2 Generalised attributes of development method devised in this research.

6.3.1 Technologies Implemented in HMI Architecture

Figure 6.3 illustrates the different web technologies implemented for the three different types of software components in the end user HMI.

The end user HMI consists of one or more HMI screens. These screens are divided into screen compartments using HTML Frames. HTML Frames is a recognised technology that divides a web page into a series of sub pages or screen compartments [95]. The layout within each screen compartment is defined using HTML. The HTML may be static, where it's in a file and not changeable, or dynamic where it is generated at run-time by the web server (when the web browser requests the web page) according to some conditions or information, for example the number of components the machine consists of. The HTML can be generated from scripting engines either on the web server or the web browser client. Client side

scripting implemented technologies are either JavaScript or VB Script. Standard Server side scripting web technologies that are implemented in this HMI system are PHP or ASP and these are executed on the server where it can interface with all the HMI applications (HMI applications are illustrated in figure 5.5). The HMI system can support any 3rd party web browser plug in technologies. Vendor's design plug-ins technologies to provide a unique aspect of functionality for the web browser to perform. 3rd party plug in technologies that are typically implemented in the HMI system are; 1) Portable Document Format (PDF) that support machine documentation, 2) Virtual Reality Mark-up Language (VRML) that support a animated three dimension solid model of the machine system, 3) Macromedia Flash[1] that supports high quality animation and graphics in the HMI, 4) Java Applets technologies that support applications being executed within a web page. It is a combination of these technologies that enable complete end user HMI systems to be constructed.

[1]Macromedia Flash (SWF) is a file format, as it is officially called by Macromedia, has become the de facto standard for vector graphics in web pages.

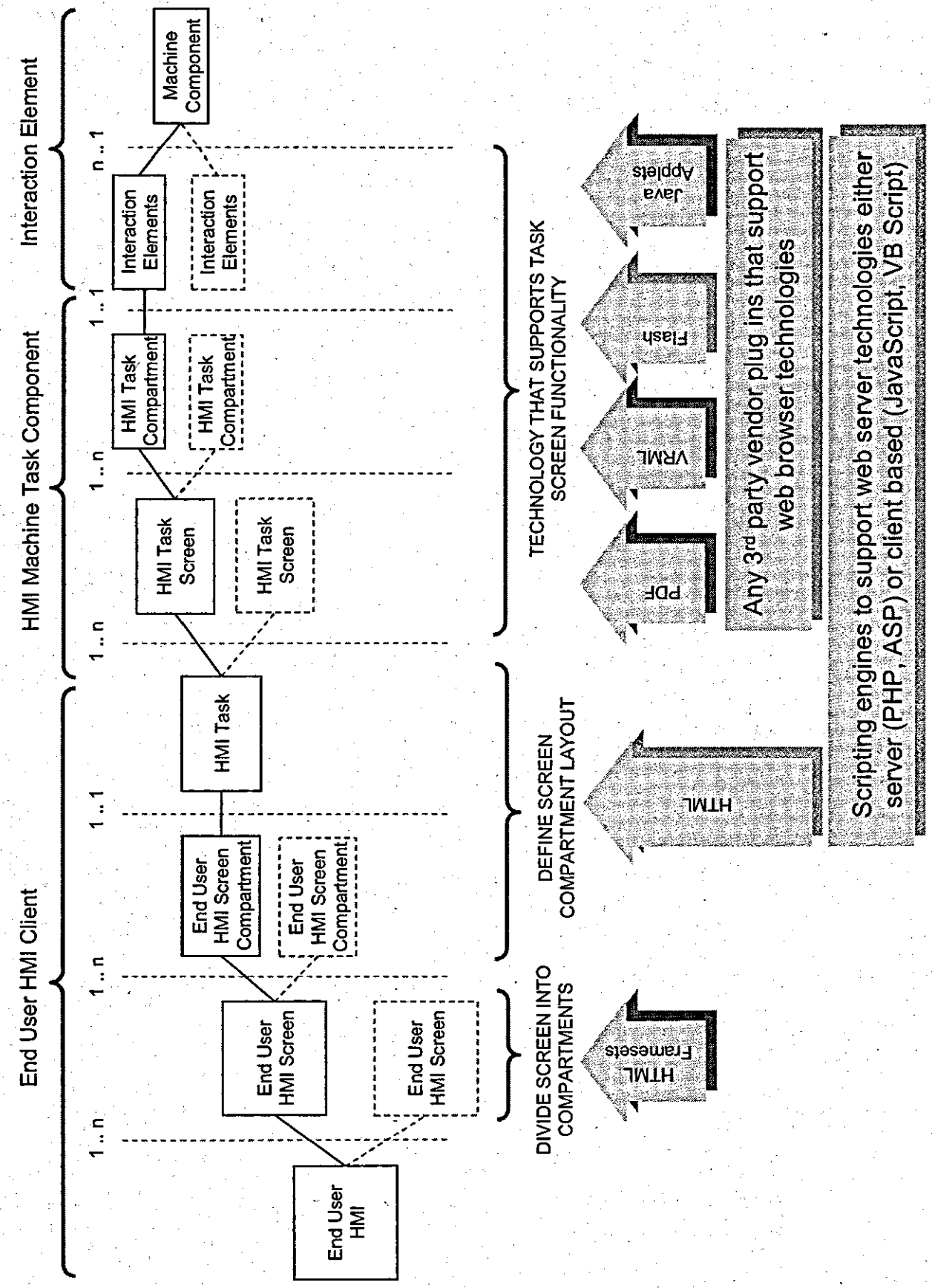


Figure 6.3 Technologies implemented in the end user HMI

6.4 Process for creating a new End User HMI Client

This section describes the process for the creation of a new end user HMI client. The end user HMI client is designed to meet the requirements of a certain type of end user. For an existing type of end user this HMI approach supports the reuse of the existing end user HMI clients that have been developed. If an end user HMI client is required for a new type of end user then a new End user HMI client must be constructed and stored for reuse.

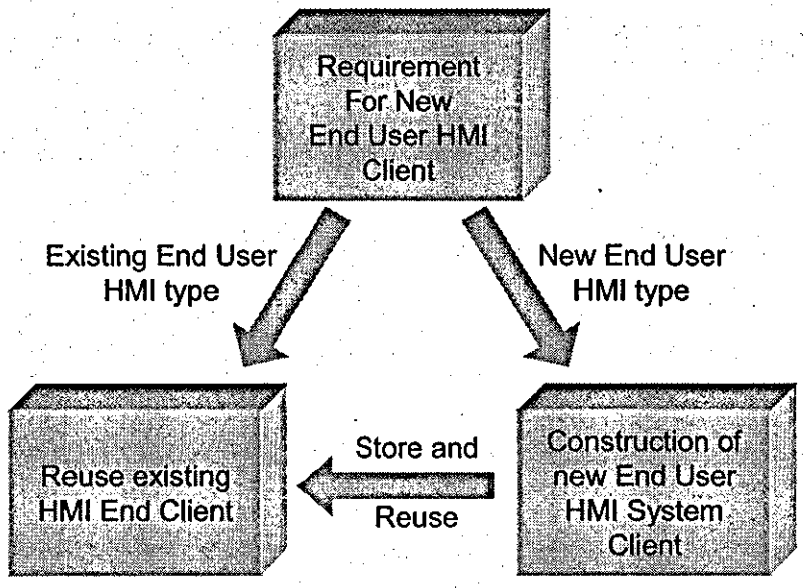


Figure 6.4 Construction of new End User HMI

6.4.1 Reuse of existing end user HMI client

Figure 6.5 illustrates the process for the reuse of an exiting end user HMI client. Process engineers, enterprise management and HMI end users are the stakeholders that would use the HMI configuration tool to view all end user HMI clients stored the system library. The type of HMI end user must be determined and then the appropriate end user HMI Client maybe selected. The end user HMI is then deployed to the candidate machine server using the HMI configuration tool. The machine server contains the machine web server, HMI applications, and an interface to the machine and the machine common data repository. These are all the system elements that are required to support the run-time operation of the end user HMI client. When the end user HMI client has been deployed to the machine server the HMI system is fully functional. The HMI end users can now be fully trained on the operation of

the new HMI. With the HMI being web based this training can be either local or remote to the machine.

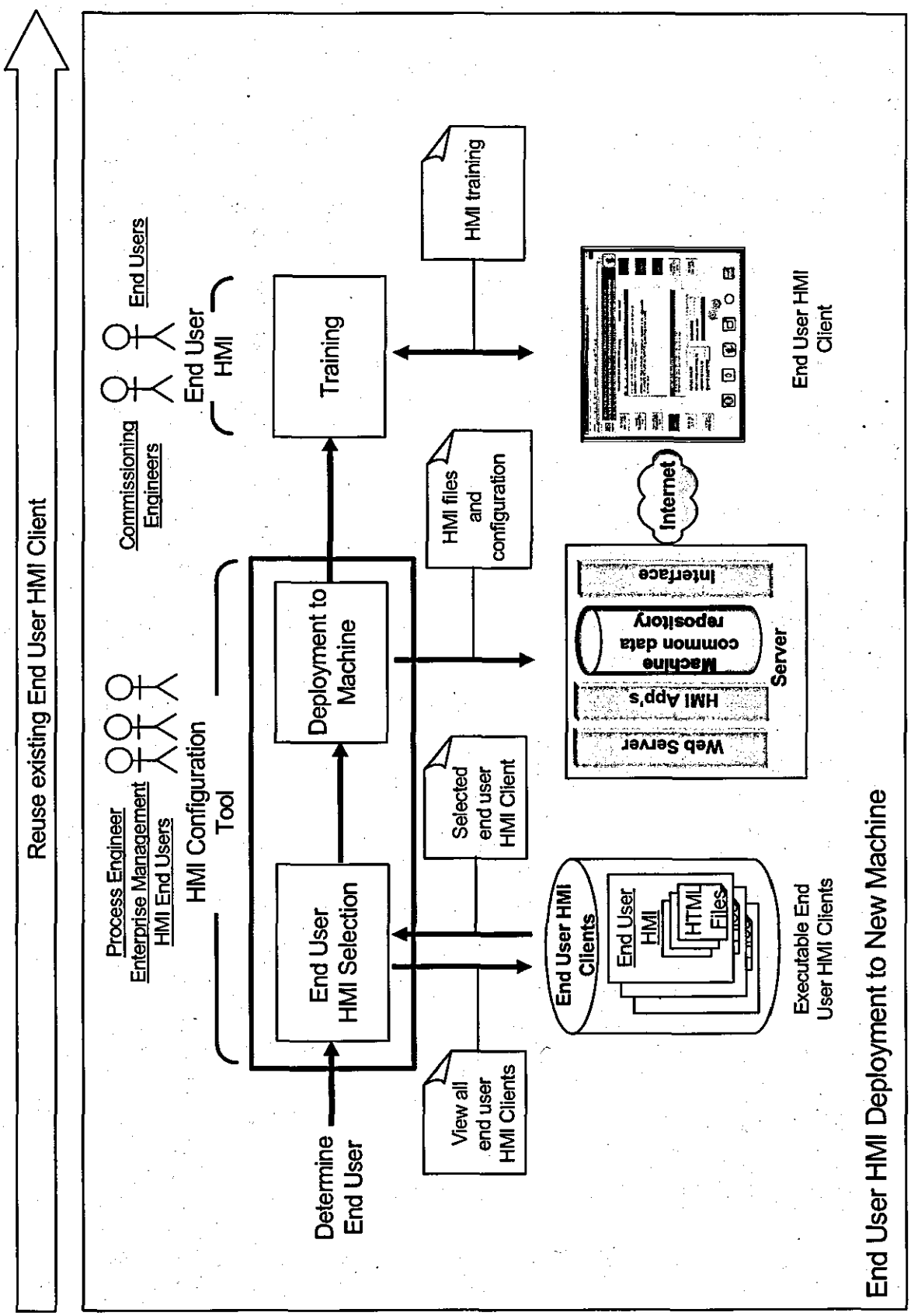


Figure 6.5 Process for Reuse of existing End User HMI Client

6.4.2 Construction of new End User HMI client

The process for the construction of the new end user HMI system is illustrated in figure 6.6. The construction process of a complete end user HMI client involves two other sub processes that support the development of the two types of software components in the HMI structure; 1) HMI task components and 2) HMI widget components.

This initial stage in the process for the composition of a new end user HMI client is to determine the HMI end users requirements. These HMI requirements are specified by one or more of the following; 1) the enterprise manager who determines the role of the HMI end user, 2) the HMI end user and or 3) the process engineer. From these HMI requirements the HMI tasks can be specified that enable the end user to perform a specified task for a particular machine type.

The HMI task component factory describes the process for the reuse / development of HMI task components. The HMI task requirements are used to either; 1) select and reuse a HMI task component stored in the HMI task component repository or 2) design and develop a HMI task component and store in the HMI tasks component repository for future reuse. The HMI tasks components are designed by HMI process engineers who fully understand the operation of the machine and what information and layouts are most appropriate to fulfil the particular aspect of HMI functionality. The HMI software engineers select the machine HMI widgets components from a system repository where they are then combined into a structured series to perform the HMI task. The HMI machine task component is then stored in a HMI Task component repository where they can be reused. The HMI widgets are designed and developed by component vendors and stored in system repositories for reuse.

The HMI task components are imported into the HMI configuration tool where they are composed into an end user HMI client. The developed end user HMI client is now comprehensively tested by a commissioning engineer. The expected benefit of this process is that it will be significantly faster than traditional HMI validation [106] due to the high degree

of reuse of existing pre tested software components in the development process. The end user HMI client is then stored in a system end user HMI system repository for reuse.

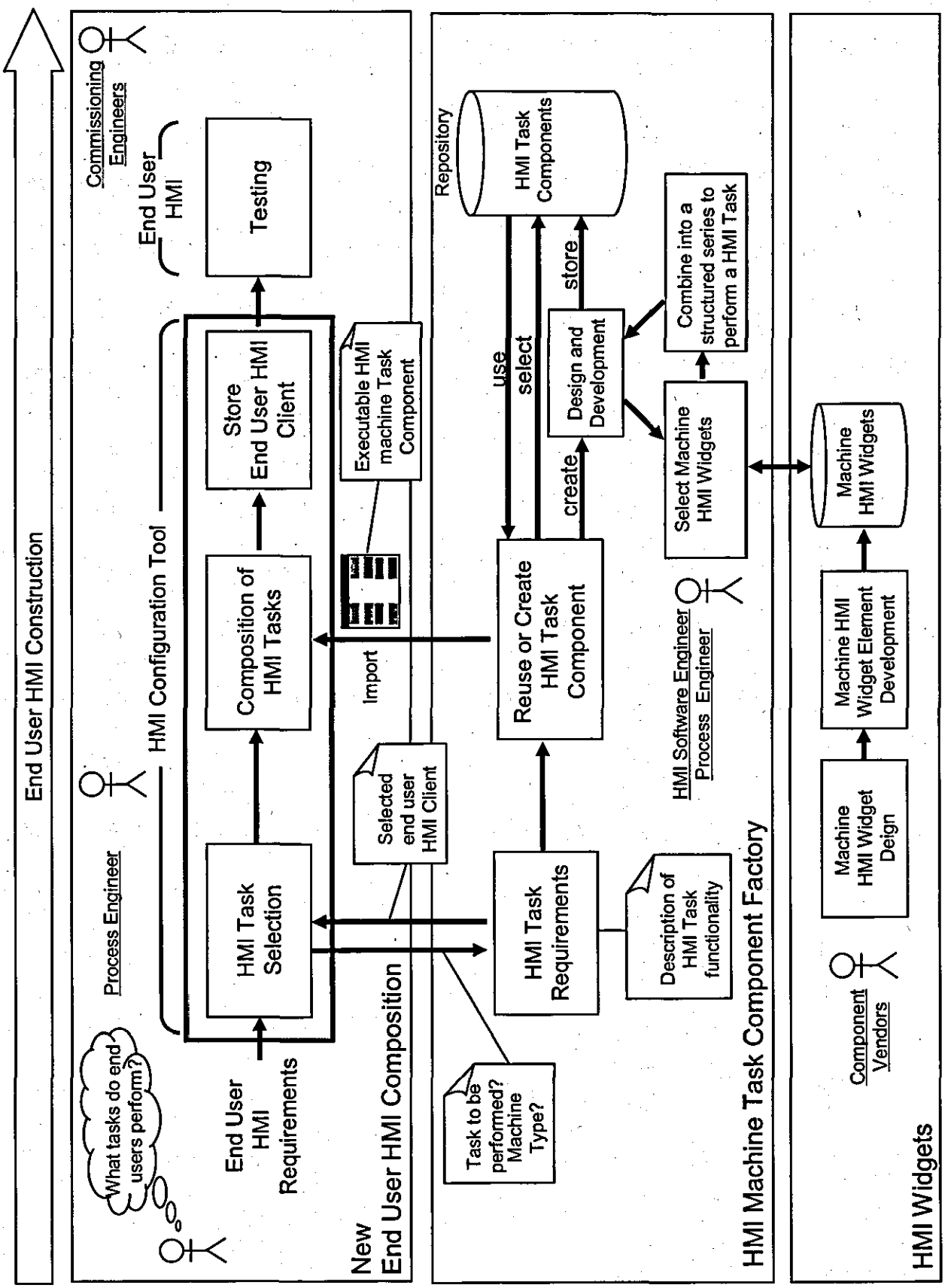


Figure 6.6 Process for the construction of End User HMI

6.5 End User HMI Configuration Tool

The end user HMI configuration tools functionality is derived from the requirements of the process of reuse and construction of an end user HMI client described in sections 6.4.1-2. The requirements of these processes are; 1) HMI task selection, 2) composition of HMI Tasks, 3) test and store end user HMI and 4) end user HMI selection and deployment to machine.

To support end users and process engineers in selecting HMI tasks the HMI configuration tool must be able to view and select any HMI tasks stored in the systems HMI task repository.

The composition of the HMI tasks involves the process engineer specifying the screen layout of the end user HMI, were the HMI tasks will be placed and the navigation between them is specified. The storyboarding technique that is used to model HMI systems (described in chapter 4 of this thesis) is directly inputted into the HMI configuration tool.

The composed end user HMI must be tested and then stored in a system library for reuse. The testing of the end user HMI client only involves validating the navigation and the layout of the HMI tasks on the screen. The HMI tasks that are used in the end user HMI client are pre-tested and do not require any further testing.

The end user configuration tool must not only support the construction of a new end user HMI but it must also support the selection and deployment of exiting end user HMI clients to new machine applications. The HMI configuration tool allows users to select end user HMI clients from the system library and configure them for deployment to the candidate machine.

The key attributes of the HMI configuration tool for composing end user HMI clients are;

1) The configuration tool supports an "executable end user HMI specification". The composition process directly supports the storyboarding technique which is used to specify the end user HMI requirements.
2) Low skill set required to construct fully functional end user HMI's with no programming required. The high level HMI configuration tool supports a graphical interface for the composition of HMI tasks into complete end user HMI clients.

3) End User HMI clients can be rapidly developed because of the high degree of reusing exiting pre developed and pre tested common HMI machine tasks.

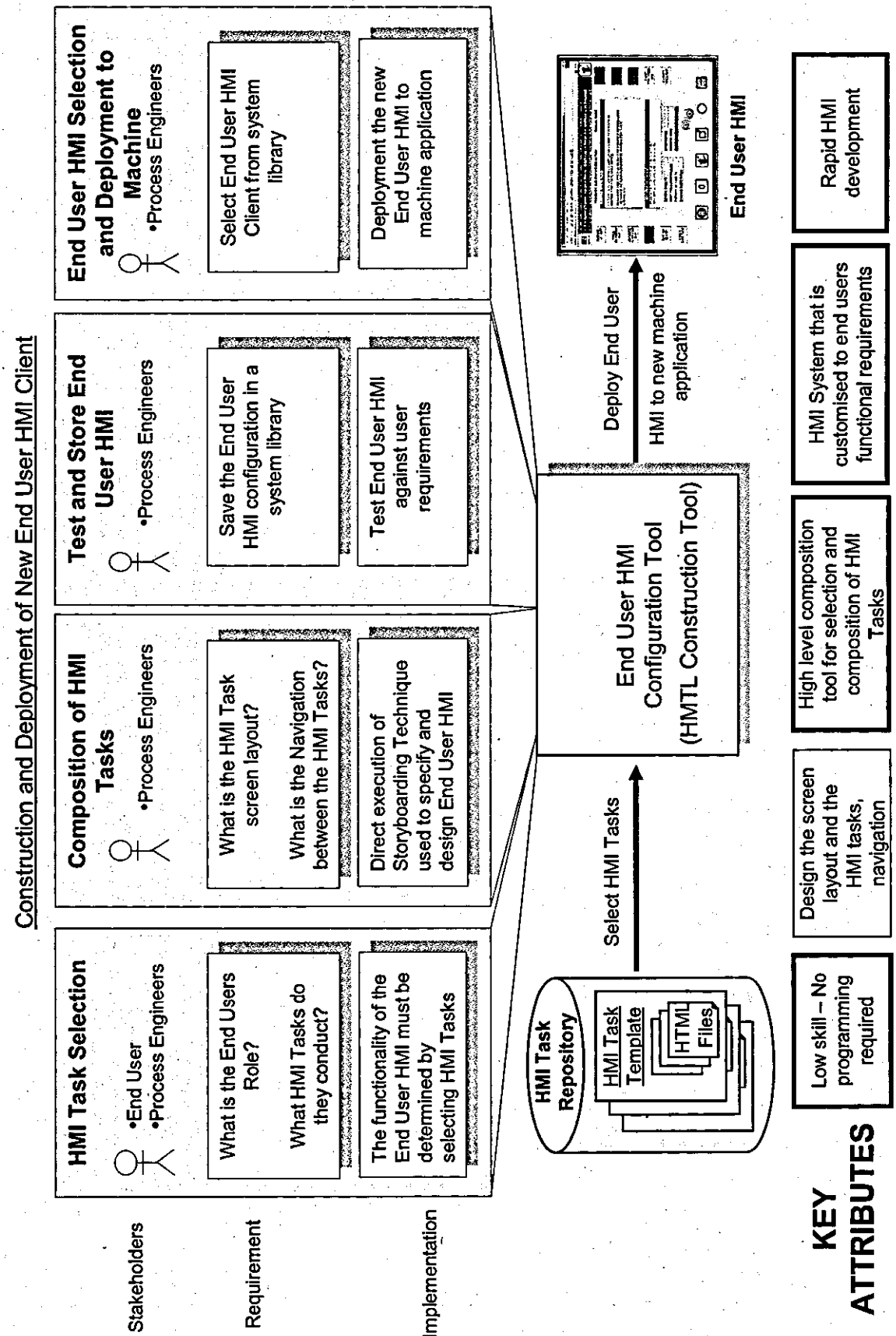


Figure 6.7 Process for the Construction and deployment of New End User HMI

6.5.1 Attributes of different types of HTML Construction Tool

The HMI configuration tool can be categorised as a type of HTML construction tool. A HTML construction tool is defined in this thesis as a software tool or toolsets that facilitate users in creating HTML code. The HMI configuration tool creates HTML code for screen compartments (see figure 6.4) and navigation links.

Table 6.1 details the requirements for the end user HMI configuration tool together with the attributes of two candidate type of HTML construction toolset that are considered for the implementation which are; 1) standard web development tools and 2) customised HTML generator.

A general requirement for the end user HMI configuration tool is that it is contained within the integrated engineering environment that supports the machine throughout its lifecycle. Using a standard web development tool, for example Macromedia Dreamweaver™ or Microsoft Frontpage™ would not support this requirement. A customised HTML generator would provide the flexibility for integration within such a software environment. The approach for standard web development tools is a very general method which supports a wide range of application needs. In contrast a customised HTML generator only provides the exact functionality that is required for the application and in this instance it must directly support the creation of HTML code for a storyboarding model. The file management of generated HTML involves manually selecting file locations on servers using standard web development tools. To support the requirement that the HMI configuration tool requires lower skilled users this process can be automated if a customised HTML generator is used. The HTML generation functionality for the HTML configuration tool must only support the generation of HTML frameset code and navigation links. Standard web development tools support extensive HTML syntax which is not required and will only increase the skill level required to use the tool. Due to the customised HTML generator being developed for this unique application of composing HMI systems the skill level is lower than a more general purpose

standard web development tool that supports more complex functionality. A customised HTML generator is implemented for the end user HMI composition tool.

Table 6.1 Attributes of two types of HTML construction tool

Attributes	End User HMI Composition Tool Requirements	Types of HTML Construction Tool	
		Standard Web Development Tools	Customised HTML Generator
General	Tool part of integrated engineering environment	Single, one off Ad Hoc implementations	✓ Can be customised to run in any environment
Approach	Directly support storyboarding method	Generalised method for creating HTML code	✓ Provide exact functionality
File Management	User requires low skill approach	Select files manually and save to server	✓ Deploy directly to server and automate processes
Functionality	Support HTML frameset code generation	✓ Supports extensive HTML generation	✓ Only supports required HTML generation
Skill Level	Low as possible	Higher skills required due to complexity of tool	✓ Only required functionality is supported.
Characteristics	Support HMI system generation	✓ Horizontal Product – All functionality to all domains. e.g. web site, help systems	✓ Vertical Product – Limited functionality customised to domain.
Example		✓ Macromedia Dreamweaver™ Microsoft Frontpage™	✓ Bespoke Tools for End User HMI composition.

6.5.2 Using the HMI Configuration Tool to compose an End User HMI

The process for the implemented HMI configuration tool to compose a new end user HMI is illustrated in figure 6.8. The end user HMI configuration tool is run within the integrated engineering environment. The application supports the design of the end user HMI screen layout using a tool that supports dividing the screen into compartments and specifying the size and name of compartments. When the screen layout is designed the screen is placed on

the HMI composition application work space. The HMI composition application workspace provides a direct resemblance to the storyboarding technique that is used to specify and design the end user HMI. The HMI composition application allows HMI tasks to be assigned to screen compartments. The navigation controls of the end user HMI are created by adding buttons and the corresponding navigational link using a individual tool design for this function. The composed end user HMI is then saved to the end user HMI library for reuse. The concept is that the HMI configuration tool is used to compose an end user HMI using the storyboarding technique.

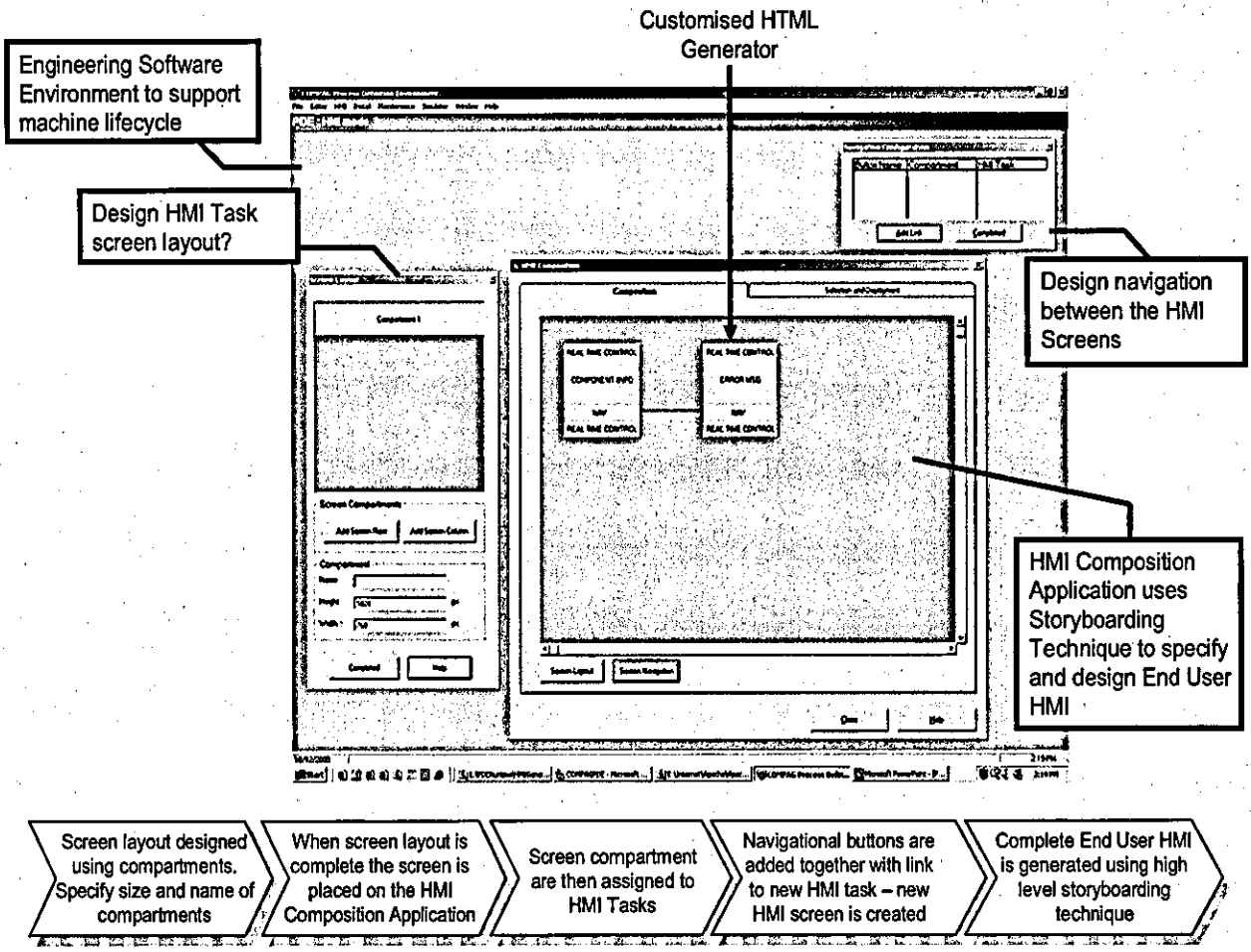


Figure 6.8 Implemented End User HMI Configuration Tool - Composition of HMI Tasks

6.5.3 Using the HMI Configuration Tool to Select and Deploy End User HMI

A screen shot of the implemented end user HMI configuration tool used to select and deploy an end user HMI is illustrated in figure 6.9. The tool supports the selection of the end user HMI and its configuration. The selected end user HMI is then deployed to the candidate machine server where it is fully operational.

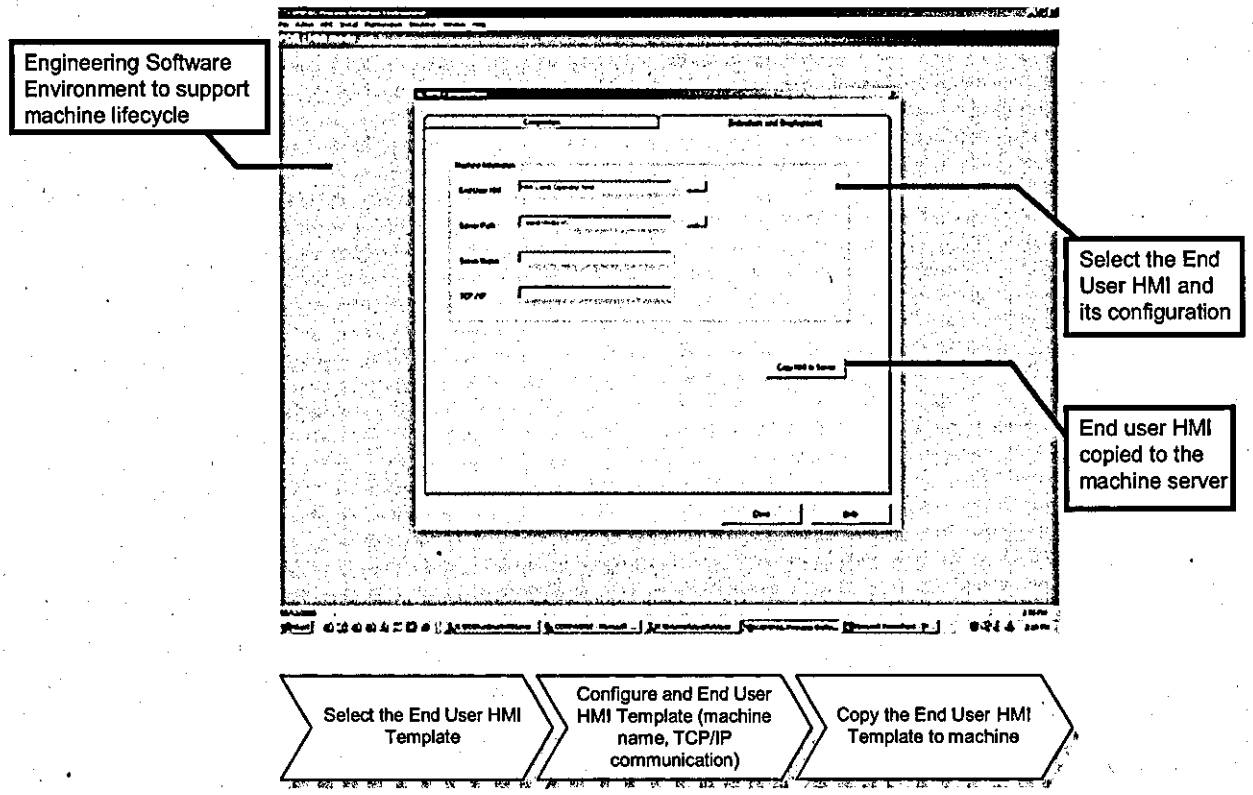


Figure 6.9 Implemented End User HMI Configuration Tool – Select and deploy End User HMI

6.6 Supporting tools for HMI Task Component Construction

The tools used to develop HMI tasks components are standard web development tools. The process engineers and HMI end users identify aspects of functionality that users require from the HMI system. The HMI tasks requirements are specified using the storyboarding technique (see section 4.7 of this thesis). The specification of the HMI task component in the form of a storyboarding model provides sufficient information to enable a HMI software engineer to develop the HMI task software component. HMI widget components are selected and combined into a structure to enable the HMI task to support a particular aspect of HMI functionality. Standard web development tools are used by the HMI software engineers to develop the HMI task components. The HMI task components are tested by machine process engineers to validate that HMI end users will be able to perform the intended machine function. The HMI tasks components are stored in a system library when they can be selected and reused. Figure 6.10 illustrates this process that the storyboarding technique and standard web development tools must support.

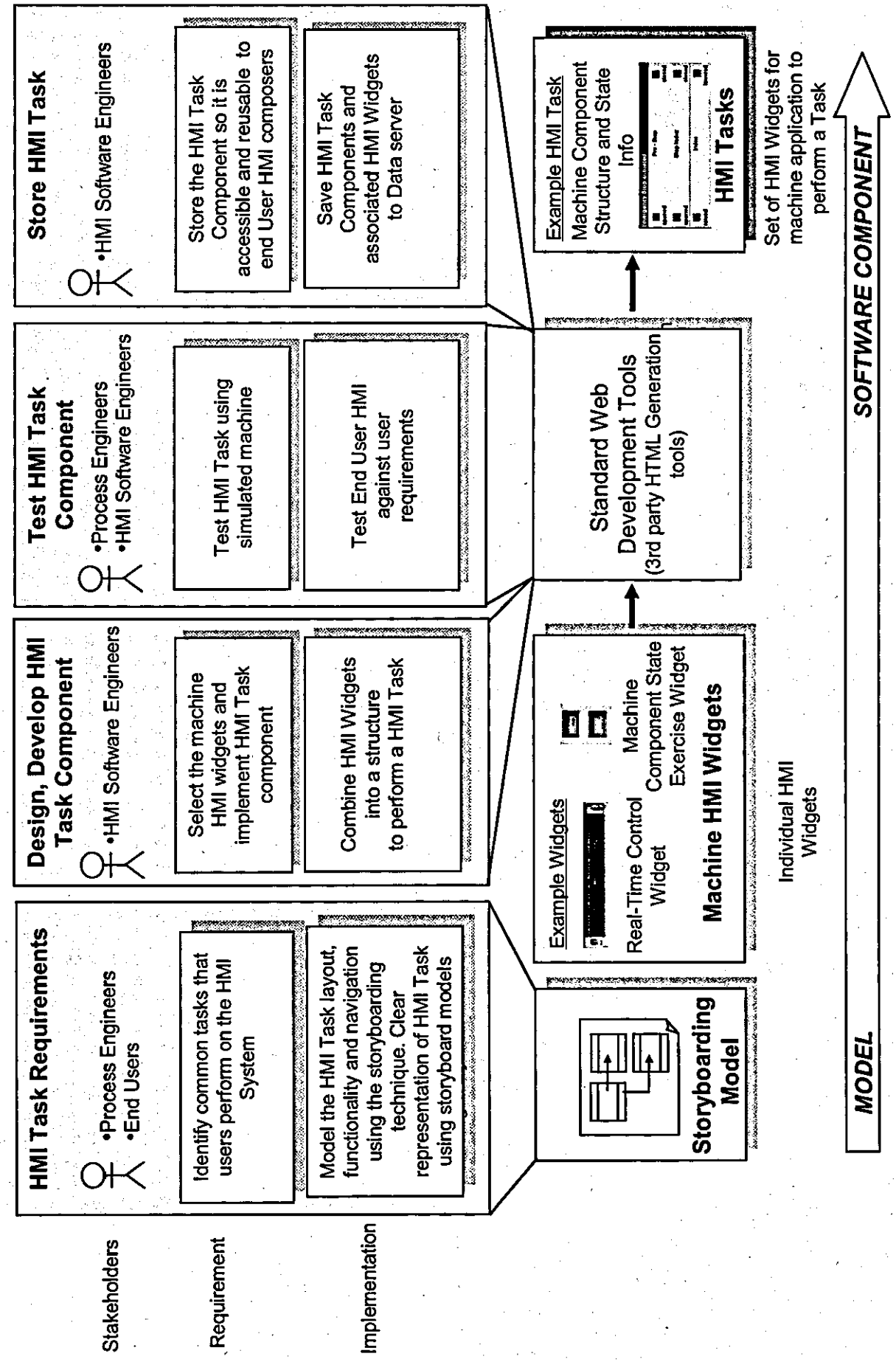


Figure 6.10 Implementation process for HMI Tasks

6.6.1 HMI Task Component Implementation

A HMI task component implements HTML code which structures the screen layout and displays one or more HMI widgets that are required to support a particular aspect of the HMI functionality. The HTML code is developed by HMI software engineers using standard web development tools and can be either static or dynamic.

Static HTML code is when the code is always constant and will not change with the variability of being deployed to different machines. Static HTML code may contain syntax for constant layouts of a screen for example.

Dynamic HTML is used when the HTML code required is dependant on machines configuration information and or logic within the HMI. There are two types of dynamic HTML; 1) server side and 2) client side. Server Side is used when machine information is required and scripts are executed on the server that retrieves information from the machine common data repository. Client side scripting is used to dynamically create HTML in the web browser from logic within the web browser for example navigational links or button sequences.

An example scenario of server side dynamic HTML usage when there is a requirement to create HMI widgets for machine component, information that must be retrieved from the machine common data repository. The server side dynamic HTML would request and retrieve the component information from the machine data model and then create the HTML to support an HMI widget for every machine component retrieved from the machine data model. The benefit of using server side scripting to create the HTML in the HMI task is that the HMI is always consistent with the machine data model for which is created from. The benefit of using dynamic HTML to create HMI tasks components is that it provides only the logic of what information should be displayed and the actual information is always retrieved from the machines common data repository. This enables the HMI task components to be reused across all machine applications and hence increasing the reuse of the HMI Task

software components. The implementation of HMI task components is illustrated in figure 6.11.

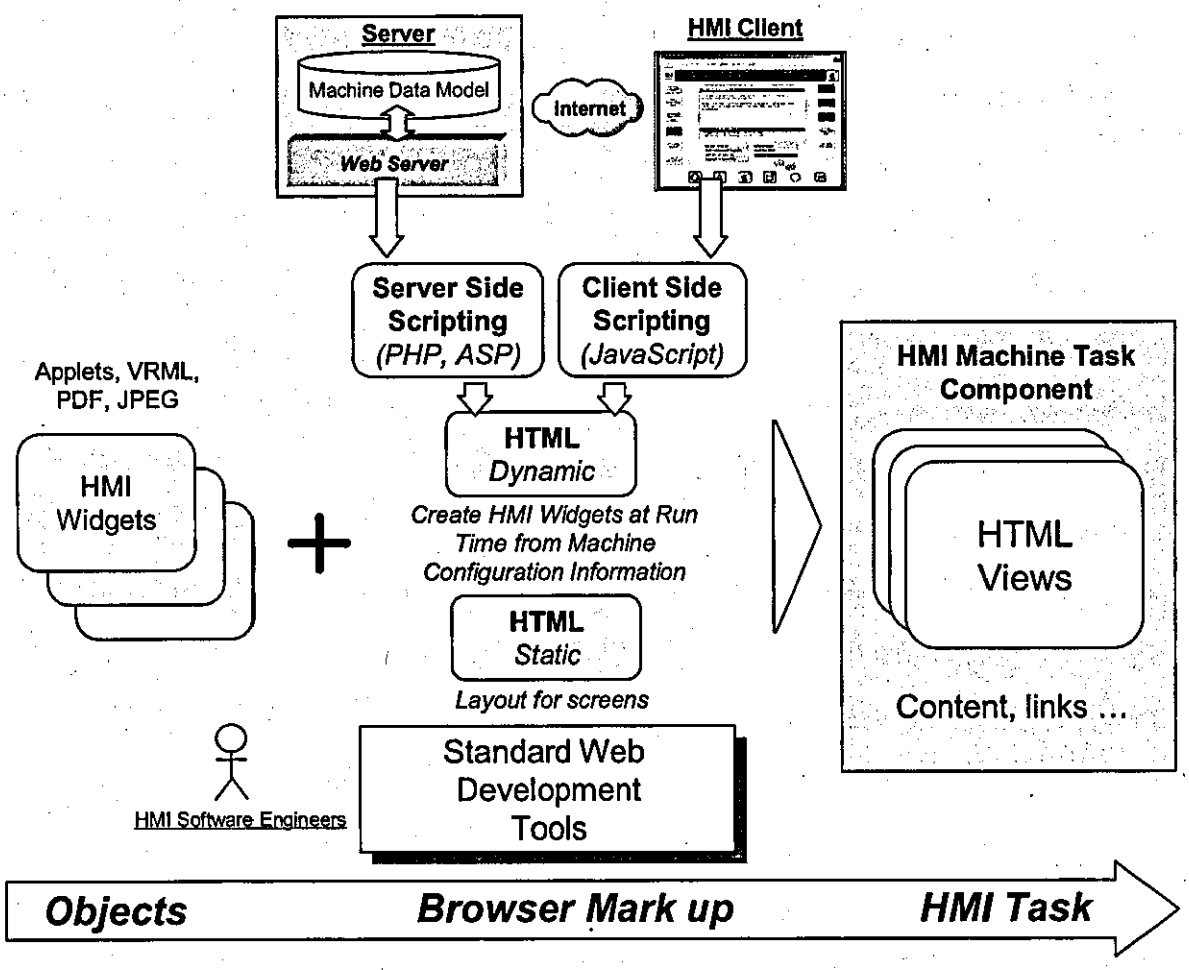


Figure 6.11 HMI task Component Implementation

6.7 HMI Widget Construction

The concept of the HMI widget is that they are supplied to the machine builders with machine components that support the human component interaction.

The HMI widgets are designed by component engineers who specify functionality the end user requires. State transition diagrams are used to model the behaviour of the HMI widgets (see section 4.8 of this thesis). An example could be a button that controls a machine component and consists of two states. The state transition conditions would detail the state of the machine component represented by the button. The HMI widgets are constructed using software development environments that support the HMI widgets technologies. Both component engineers and software engineers are involved in the development of the HMI widgets. Due to the component vendors designing the HMI widgets they dictate what information is displayed and how the machine component can be controlled by end users. This enables best practice and specialist component vendor's knowledge is encapsulated in the machine components HMI widget. The HMI widgets are pre-tested with the machine component to aid the end user in faster integration of individual HMI widgets into complete HMI systems.

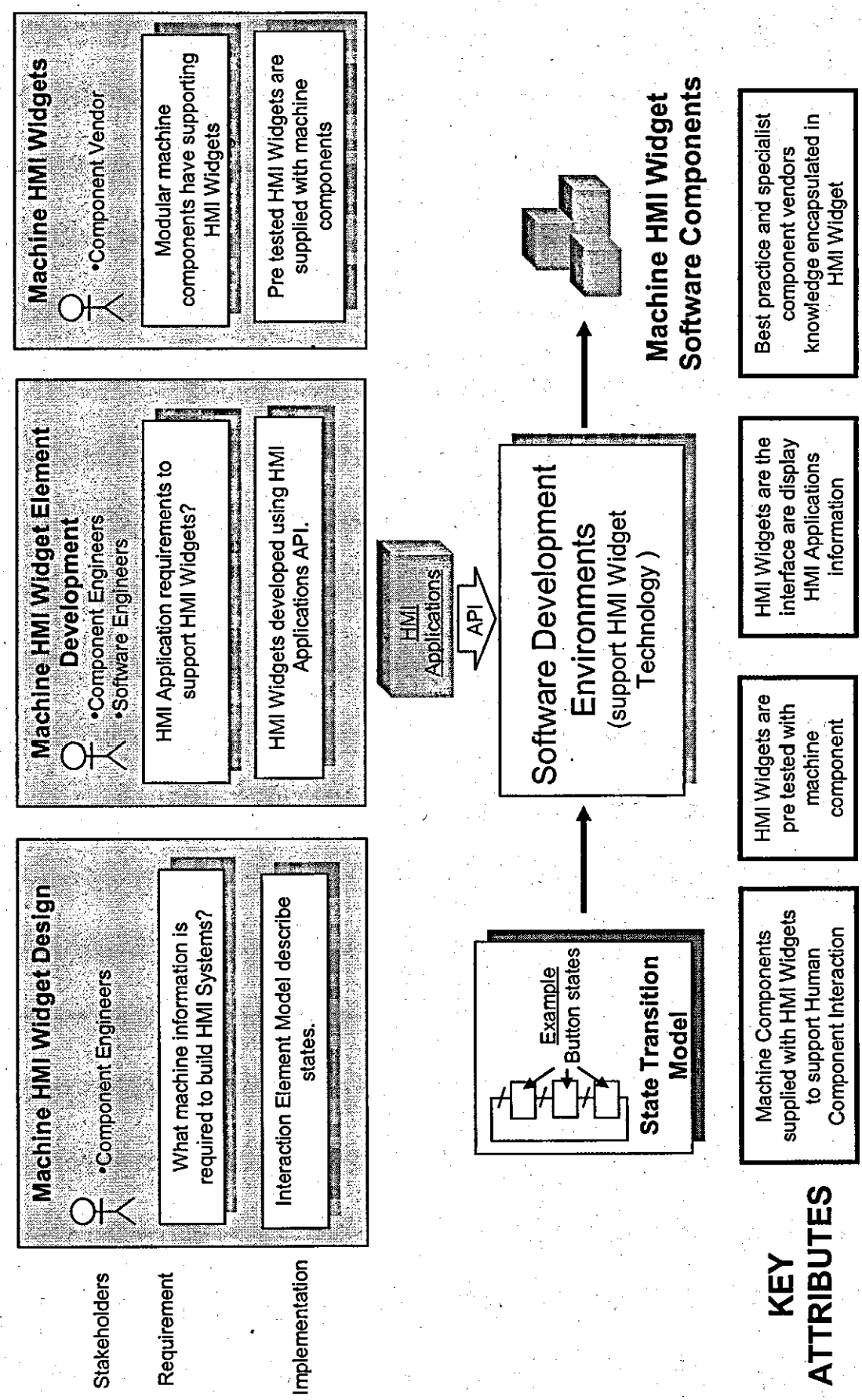


Figure 6.12 Process for implementation of machine interaction elements

Figure 6.13 illustrates the integration of HMI widgets into the HMI systems run-time architecture. An Application Programmers Interface (API) supports the integration to HMI application that are executed on the machines server. The API describes the interface to the HMI applications that component software engineers use in the HMI widget. HMI widgets can be categorised into two different types; 1) Thick Client or 2) Thin Clients.

Thick clients are applications that are downloaded form the server and executed within the HMI client. Example candidate technologies for thick client widgets are Java Applets[2] or Microsoft ActiveX[3] controls. Once downloaded to the client these applications can then communicate with HMI Applications to display machine information to end users providing they have been developed in conformance with the API.

Thin client HMI widgets only use the standard web browser functionality. HTTP requests are the mechanism that web browser clients use to request information from a web server. The API will specify the commands and the expected response from the HMI applications on the machine server. These request and responses are built into the logic of the HMI widget.

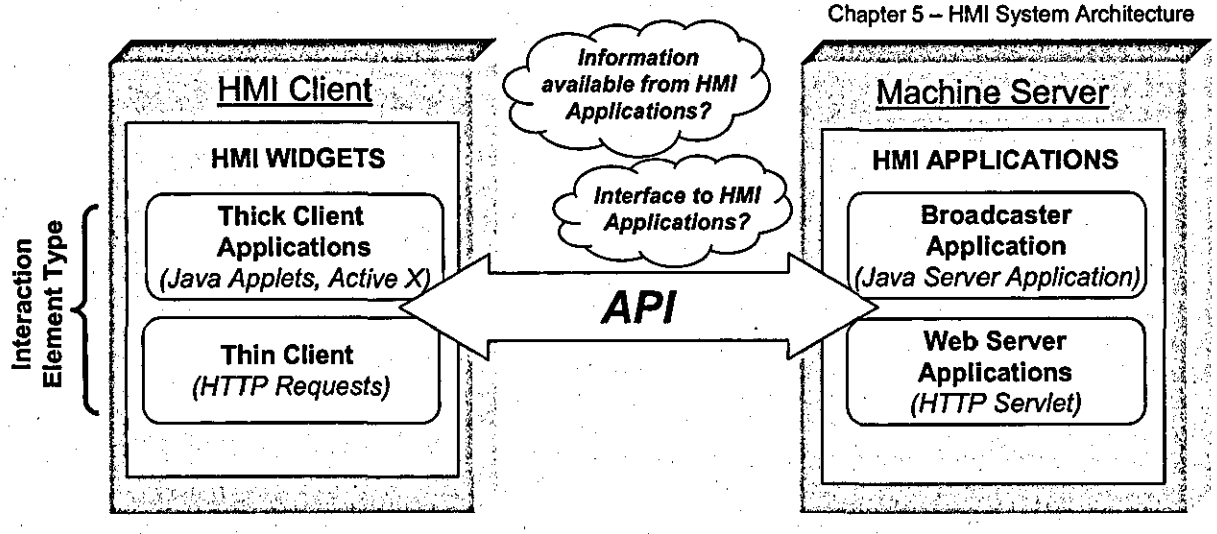


Figure 6.13 Integration of HMI Widget into HMI System Architecture using HMI Applications.

2 A technology developed by Sun Corporation that enables programs to run inside a web browser. Java applets are developed using the java language.

3 Microsoft ActiveX controls are the interactive objects in a web page that provide interactive and user-controllable functions.

6.8 Summary

In this chapter a structured process has been defined for constructing new end user HMI's. The systems architecture (chapter 5 of this thesis) describes the executable process of the HMI system. The HMI system architecture has been designed to support machines within the context of automotive manufacturing automation systems. This chapter has described the process, tools and resources that the stakeholders require to implement an end user HMI for a particular machine application.

To realise the construction of end user HMI's appropriate implementation technologies for the software components must be determined and selected taking into account the role of each of the stakeholders. The end user HMI is composed using a HMI configuration tool which;

1. Supports an "executable end user HMI specification". The composition process directly supports the storyboarding technique which is used to specify the end user HMI requirements.
2. Eliminates the need for specialist programming skills, i.e., enabling the creation of fully functional end user HMI's with no programming required. The high level HMI configuration tool supports a graphical interface for the composition of HMI Task components into complete end user HMI clients.
3. Enables end user HMI clients to be rapidly developed with a high degree of reuse of exiting pre developed and pre tested common HMI Task components.

The benefits of these features are quantitatively measured and evaluated in chapter 8 of this thesis.

The process and stakeholders involved in the development of the HMI Task components and HMI widgets which are the lower level software components used in the end user HMI client are described.

The application of the C-B HMI approach to the implementation of three case studies will be described in chapter 7.

7 Industrial Case Studies

7.1 Introduction

The systems concepts, architecture, models and construction process for a C-B HMI approach have been introduced in chapters 4, 5 and 6 of this thesis. Application and implementation details of this C-B HMI approach to three case studies are described in this chapter (see figure 7.1). The development and implementation of C-B machines and their associated HMI's were undertaken over a three year period by the MSI Research Institute through research projects "Assessment and Implementation of a Component Based Paradigm for Agile Automation" (COMPAG), "Global Engineering of Manufacturing Machinery" (GEMM) and "Common Model for Partners in Automation" (COMPANION) funded by the Engineering and Physical Sciences Research Council (EPSRC). The industrial demonstrator machines which used for the implementation, testing and evaluation of the C-B HMI Paradigm are described in the following three case studies;

1. A full size automotive engine assembly machine in Johann A. Krause Maschinenfakrik GmbH in Bremen, Germany,
2. A full sized transfer line demonstrator machine in Lamb Technicon Machining Systems (Lamb) in Mildenhall, UK,
3. A test rig developed in conjunction with Ford Motor Company at the MSI Research Institute, Loughborough University.

In this chapter the industrial users are introduced and their core organisation roles within the automation manufacturing domain are described. The key expected benefits with the C-B HMI approach are detailed. The application of the C-B HMI to the case studies involves describing the HMI system's implementation and the associated context of each respective machine.

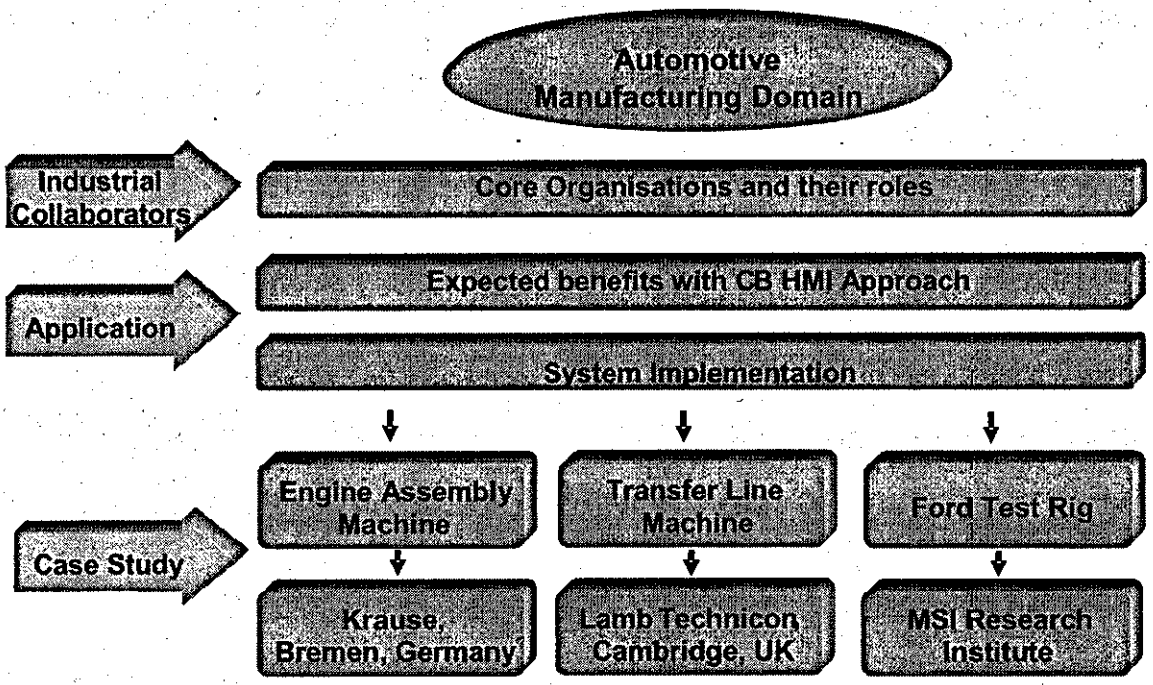


Figure 7.1 Structure of the case studies outlined in this chapter

7.2 Automotive Case Study Organisations and their Roles

Figure 7.2 illustrates a general overview of the automotive supply chain and the project collaborators roles. The supply chain within the automotive sector consists of three levels, 1) End Users who collaborated on this project; Ford, Jaguar and Mazda, 2) 1st Tier suppliers; Lamb Technicon and J.A Krause who are machine builders and provided the industrial test beds for these case studies and 3) 2nd tier machine component vendors being Parker Hannifin, Indramat and Bosch Rexroth who supported this research with the supply of required machine components.

Numerous industrial applications (e.g. machining, parts transportation and assembly) are required by End Users to enable the product manufacturing process. The implementation of such systems in the automotive domain is predominately based on automated production machinery. It is the End Users who specify to machine builders (1st tier suppliers) the requirements for new automated production machinery. They require the machine builders to design, build, test and maintain systems that they can operate.

Adopting the C-B approach to machine system's development the machine builder must; mechanically design the machine, configure the control behaviour, then either reuse or compose a new End User HMI Client, validate the machine using simulation models and then install and commission the machine in the end users plant [82]. The machine builder's engineers design the machine from selecting modular units from the component suppliers.

A machine component is a modular unit that consists of the physical device which may be a machine actuator, servo drive or sensing devices that contains: 1) embedded control capabilities, 2) an associated three dimensional solid model to aid machine design engineers and support a solid model of the complete machine and 3) monitoring and diagnostic capabilities in the which is the aspect that is the focus of this research. A component vendor's role is to supply and maintain their machine components (see chapter 6.2 of this thesis for more details of C-B Automation).

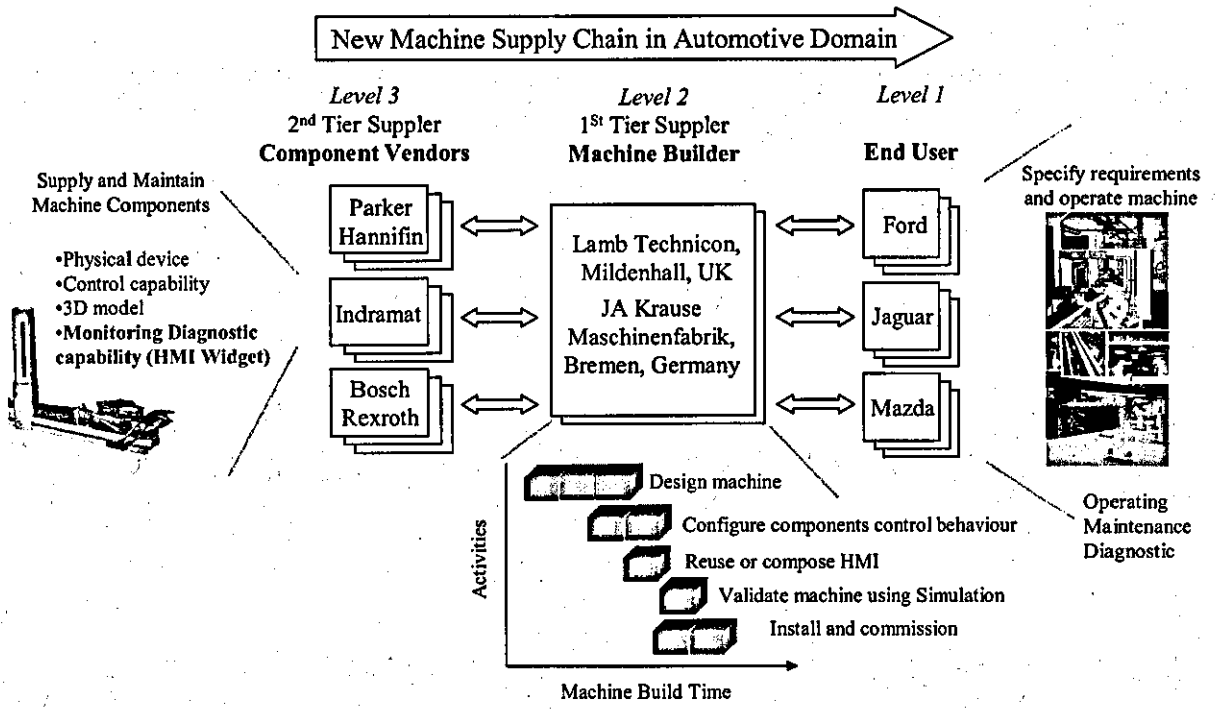


Figure 7.2 COMPAG, GEMM and COMPANION research project Industrial Collaborators and their roles in the Automotive Supply Chain

7.2.1 Benefits illustrated in case studies

The benefits inherent to the C-B HMI approach described in chapters 4, 5 and 6 of this thesis are fully supported in all three case studies described in this chapter however for clarity each case study is used to identify a particular set of core benefits of the C-B HMI approach.

The Automotive Assembly Machine case study (figure 7.3) at J. A. Krause is used to demonstrate rapid HMI composition by using the End User HMI Configuration Tool (see section 6.5 of this thesis) that enables direct implementation from a end user HMI model (a fully functional Machine Engineer HMI client). The rapid configuration capability is also illustrated through the reuse of existing machine HMI Task software components. Furthermore this case study is also used to demonstrate another core feature of the C-B HMI approach where the machines component structure is changed and the HMI is automatically updated to reflect this change.

The Automotive Transfer Line Machine case study is used to illustrate how machine HMI messages are generated from the machine common database (see figure 7.3). This ensures

that the machine and its associated HMI are always consistent. Industry best practice is shown in this example by demonstrating industry standard types of error messages that are embedded in HMI Task software components so they can be reused in many applications.

The benefits of remote monitoring and diagnostics through the use of internet technologies that the HMI system supports are illustrated using the Ford Test Rig case study (see figure 7.3). The three dimensional solid model is described and how this supports early HMI validation and operator training.

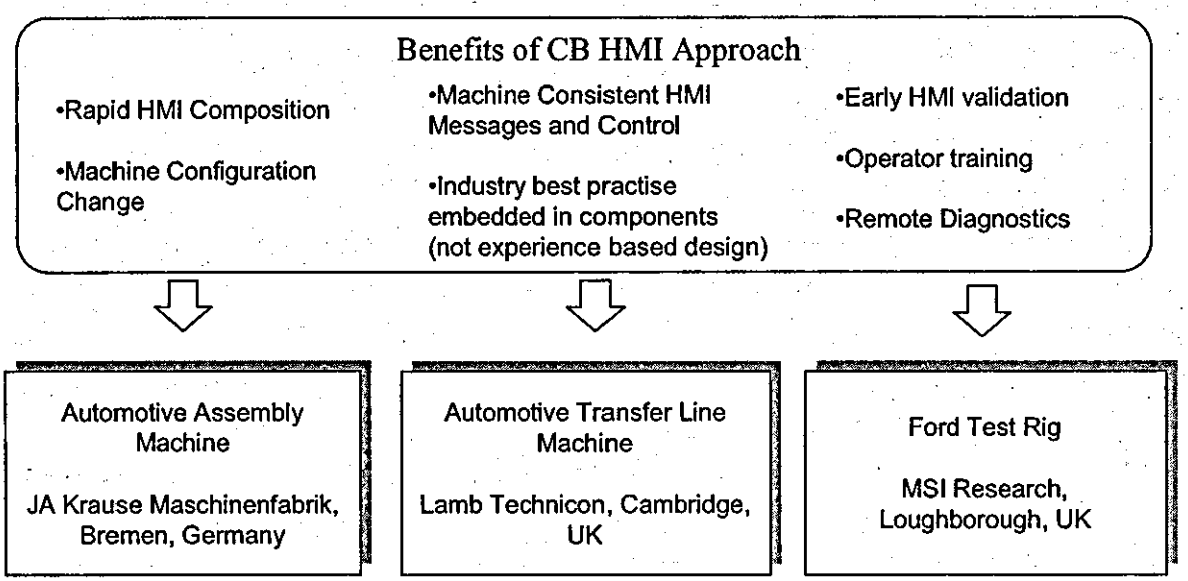


Figure 7.3 Benefits of C-B HMI focussed on in each case study

7.3 C-B HMI Implementation Issues

In this section issues associated with the implementation of the C-B HMI System are introduced. The different users involved during the lifecycle of automotive manufacturing machines have been identified and their required HMI functionality is described which has been determined through the collaboration with industrial partners [88, 89]. The HMI Client Software Components (HMI Tasks and HMI Widgets) that have been implemented in the case studies described in this chapter are detailed. HMI Task software components support a particular functional aspect of the required HMI functionality. One or more HMI Widgets are combined in a HMI Task component (the functional architectural structure for a HMI system in this research is described in chapter 4 of this thesis). End User HMI Clients are composed from one or more HMI Task components that support the end users required functionality. An End User HMI Configuration Tool (see section 6.5 of this thesis) supports the direct input of a storyboarding model (see chapter 4 of this thesis) to generate a complete End User HMI Client.

The HMI system's architectural elements (see chapter 5 of this thesis) which have been developed in these case studies are; 1) the *Common Machine Data Repository* where all machine information is stored, 2) *Web Applications* that provide the mechanism to support HMI client request / respond commands and real-time machine information and 3) the *web server* is where the HMI Client files are requested and distributed.

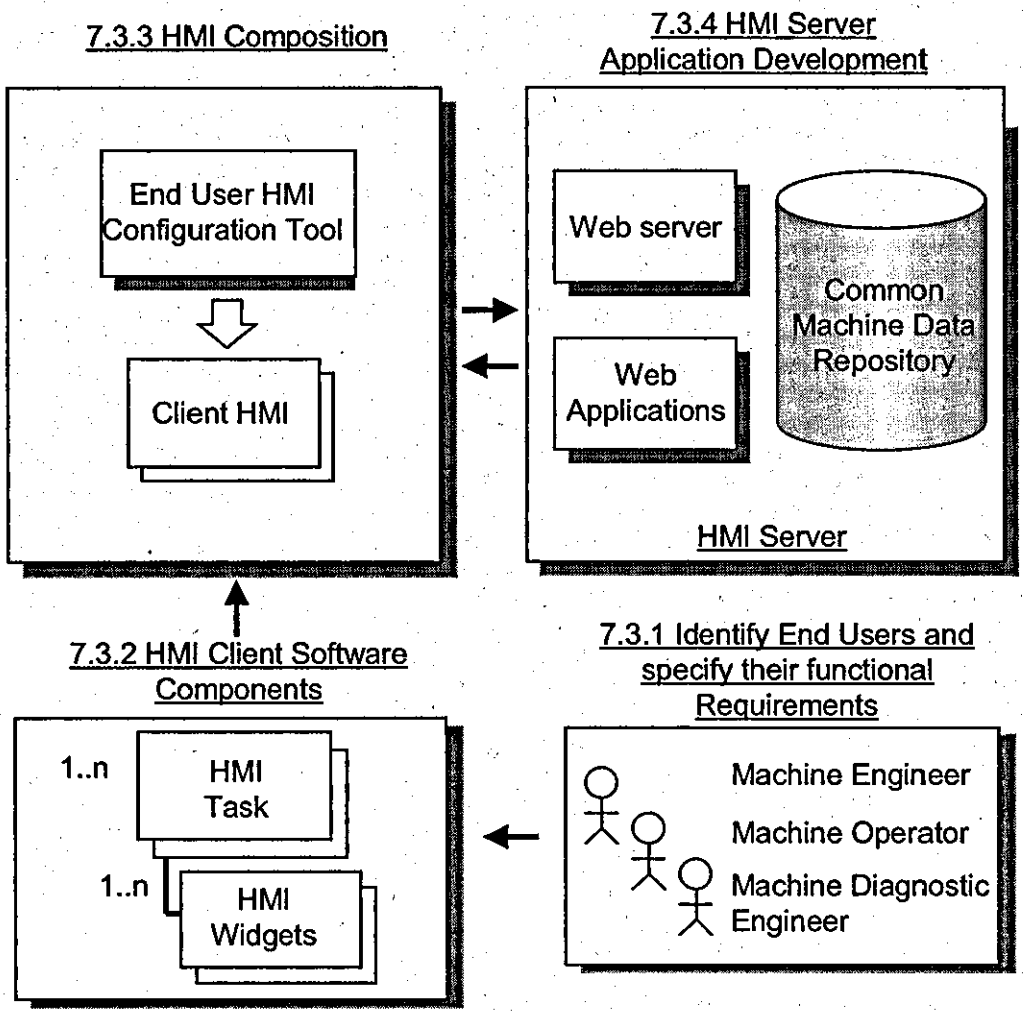


Figure 7.4 Core aspects of the C-B HMI System Implementation

7.3.1 Identify End Users and specify their functional Requirements

Within the automotive domain three main types of HMI end users have been identified to support the operation of the machine and each type of user requires a different combination of machine task components in their HMI to perform their enterprise roles which are detailed in table 7.1. For example the operation and support of the machine involves the following users: 1) Machine Operators who monitor the operation of the machine and report any machine errors or machine malfunctions to Machine Engineers, 2) Machine Engineers who would have the appropriate skill level to diagnose and recover the machine from errors and 3) Machine Diagnostic Engineers who have the knowledge to diagnose complex machine errors.

To enable Machine Operators to monitor the operation of the machine they require the following HMI functionality; 1) presentation of real-time sub system information which enables them to see the event messages from the machine, the mode status of the machine and if any errors are present, 2) determination of sub system errors from which the operator can drill down to and report on detailing information about machine errors which are displayed in industry standard formats, 3) the state of the machine components and 4) the visualisation of the machine simulated model so they can determine the machine state from 3D machine model on the HMI screen.

The Machine Engineer requires additional HMI functionality over and above those required by the Machine Operator to support the recovery from rectification of faults on the machine. The additional HMI functionality is sub system model control which allows the machine engineer to change the mode of the machine and manually drive components to safe states, for example to recover work piece jams. The second additional machine task required by the machine engineer is sub system component diagnostics. This allows the engineer to view electrical schematics, service information regarding the machine components to ensure that the correct solution to the problem, i.e. the investigation of electrical connectivity of a selected component is determined.

The machine diagnostic engineers HMI is essentially identical to the machine builders HMI but this type of operator is generally offsite (remote from the machine) and therefore would not be allowed to have control of the machine due to safety aspects.

Table 7.1 HMI End Users in Automotive Domain all their associated machine tasks

End User	Role	HMI Functionality
Machine Operator	To monitor the operation of the machine and report errors to Machine Engineers	Real-time Sub System Information
		Sub System Error Information
		Component and Element Monitoring and Control
		Machine Simulation Model
Machine Engineer	To recover the machine from errors involving diagnosing / rectifying faults on the machine.	Real-time Sub System Information
		Sub System Control
		Sub System Error Information
		Component and Element Monitoring and Control
		Machine Simulation Model
		Sub System Component Diagnostics
Machine Diagnostic Engineer	To diagnose faults at a component level on the machine generally from a remote location.	Real-time Sub System Information
		Sub System Error Information
		Component and Element Monitoring and Control
		Machine Simulation Model
		Sub System Component Diagnostics

Figure 7.5 illustrates the HMI system functional architecture (HMI functional architecture that is described in chapter 4 of this thesis) for the three different End User's (and their HMI are implemented as End User HMI Clients). Each End User HMI Client consists of one or more HMI Task that supports a particular aspect of the required HMI's functionality. Each HMI Task contains one or more HMI Widgets that support the direct interaction (for example control, monitoring) with a machine component.

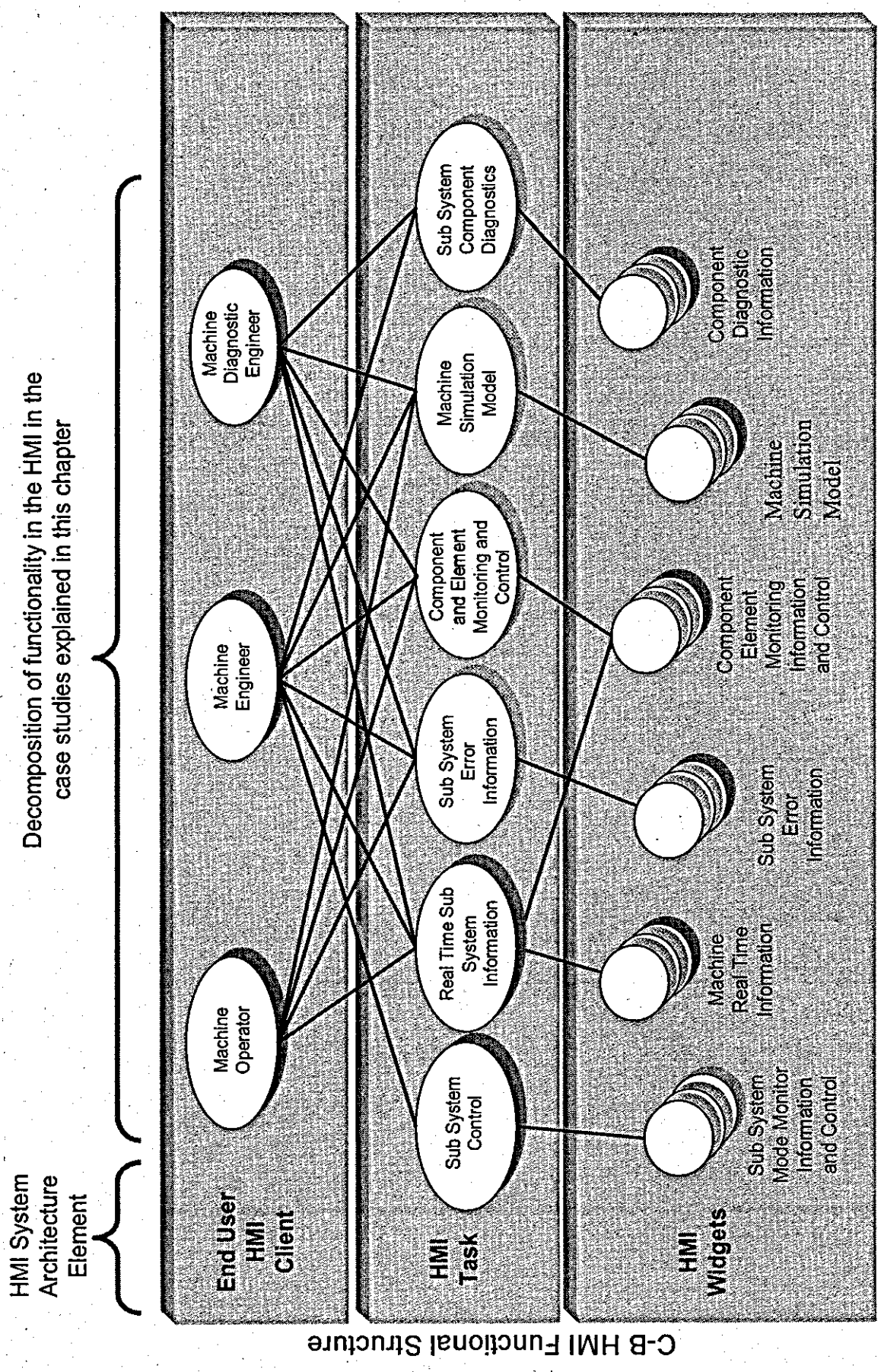


Figure 7.5 Functional Structure of HMI System for each case study described in this chapter

7.3.2 HMI Client Software Components

The role of the HMI Widgets is to provide the functionality to facilitate direct interaction with one or more machine components. HMI Widgets are the mechanism for the HMI system to provide machine information to the end user.

Machine information that is required by the End User in these case studies can be divided into two categories; 1) Sub System level and 2) Machine Component level. Six different HMI Widgets have been implemented to provide the required HMI information required in these case studies. Sub System level HMI Widgets provide information relating to the Kernel / Management of machine components, for example Error Information, Real-Time Information and the Mode Control are implemented in these case studies. Component level HMI Widgets support particular functional aspects that are unique to the machine components for example Machine Simulation, Component Diagnostics and Component Element Control and Monitoring are implemented in these case studies.

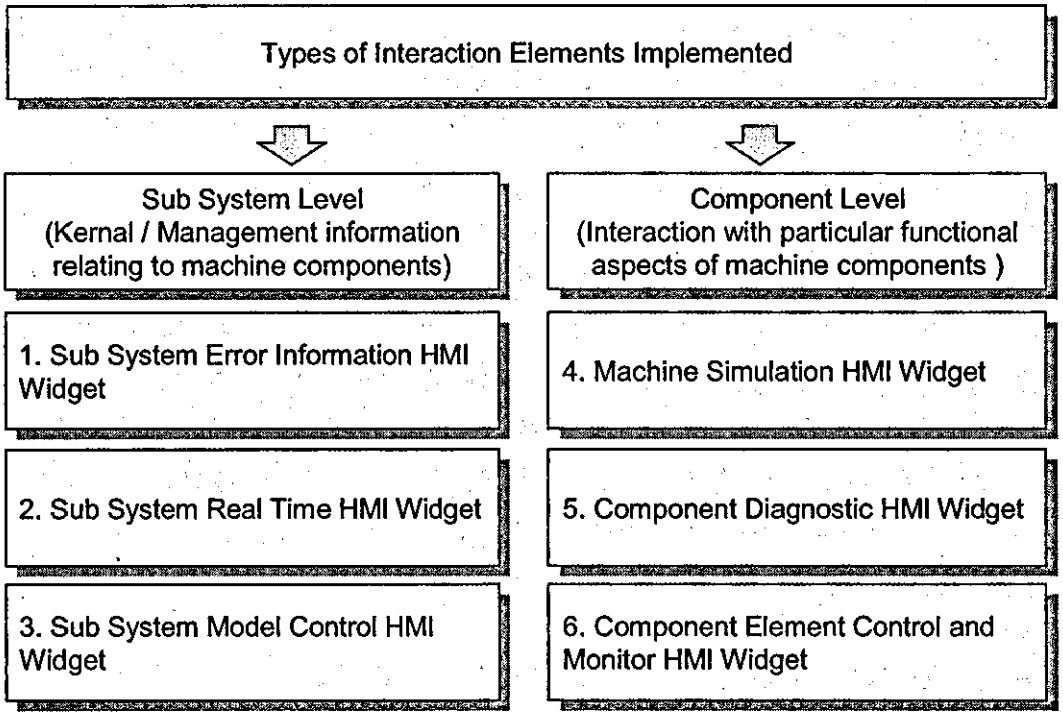


Figure 7.6 Types of Interaction Elements required in each case study

One or more HMI Widgets are combined to develop a HMI Task component. The association between the HMI Widgets and the HMI Task Components implemented in these case studies are detailed in tables 7.2.

The HMI functionality required in all machines implemented in the case studies is; 1) machine error information, 2) mode control, 3) real-time information, 4) simulation model, 5) diagnostic information and 6) monitoring and control.

The Machine Error Information, Mode Control and Real-time Information are implemented as HMI tasks that are directly reused in all three case studies described in this chapter. The Machine Error Information HMI Task provides three standard types of error information; 1) History of Errors, 2) Actual Errors and 3) the Frequency of Errors. A single Error Information HMI widget configured in the HMI task to provide the different type of error information (see section 7.5.5). The Mode Control HMI Task supports the five industry standard machines modes [88, 89]; 1) Automatic, 2) Manual, 3) Reset, 4) Step and 5) Initial Position. This Mode Control HMI Task consists of five instances of the Mode Control HMI Widget and each is configured to support the different machine modes (see section 7.5.5). The Real-time Information HMI task contains the Real-time Information HMI Widget and its configuration for the candidate machine.

The Solid Model Machine Simulation functionality implemented in each case study is unique to each machine's physical configuration and therefore a Solid Model HMI widget must be developed for each machine [63]. The Ford Test Rig Simulation HMI Task consists of the Ford Test Rig Solid Model HMI Widget, the Krause Simulation HMI Task consists of the Krause Solid Model HMI Widget and the Lamb Simulation HMI Task consists of the Lamb Solid Model HMI Widget. The machine's diagnostic HMI functionality is unique for each machine implemented in these case studies and each machine consists of a Diagnostic HMI Task unique for the machine application. To support the machine components control and monitoring two different HMI tasks are developed that that are dependant on the machine

type; 1) Transfer Line Machine Monitoring and Control HMI task and 2) Assembly Machine Monitoring and Control HMI task.

Table 7.2 HMI Client Software Components Implemented in Case Studies

			All HMI Widgets Implemented									
	HMI Tasks	Functionality	Error Information HMI Widget	Real-time Information HMI Widget	Component Monitoring and Control HMI Widget	Ford Test Rig Solid Model HMI Widget	Krause Solid Model HMI Widget	Lamb Solid Model HMI Widget	Mode Control HMI Widget	Ford Test Rig Component Diagnostics HMI Widget	Krause Components Diagnostic HMI Widget	Lamb Components Diagnostic HMI Widget
Ford Test Rig HMI	Machine Error Information HMI Task	Actual Errors	✓									
		History of Errors	✓									
		Frequency of Errors	✓									
	Mode Control HMI Task	Initial Position Mode							✓			
		Step Mode							✓			
		Reset Mode							✓			
		Manual Mode							✓			
		Auto Mode							✓			
	Real-time Information HMI Task	Real-time Information		✓								
	Ford Test Rig Simulation HMI Task	Solid Model				✓						
	Ford Test Rig Diagnostics HMI Task	Diagnostics								✓		
	Assembly Machine Monitor and Control HMI Task	Assembly Monitoring & Control			✓							

Table 7.2 continued

	HMI Tasks	Functionality	All HMI Widgets Implemented: Error Information HMI Widget	Real-time Information HMI Widget	Component Monitoring and Control HMI Widget	Ford Test Rig Solid Model HMI Widget	Krause Solid Model HMI Widget	Lamb Solid Model HMI Widget	Mode Control HMI Widget	Ford Test Rig Component Diagnostics HMI Widget	Krause Components Diagnostic HMI Widget	Lamb Components Diagnostic HMI Widget
J.A. Krause GmbH machine HMI	Machine Error Information HMI Task	Actual Errors	✓									
		History of Errors	✓									
		Frequency of Errors	✓									
	Mode Control HMI Task	Initial Position Mode							✓			
		Step Mode							✓			
		Reset Mode							✓			
		Manual Mode							✓			
		Auto Mode							✓			
	Real-time Information HMI Task	Real-time Information		✓								
	Krause Machine Simulation HMI Task	Solid Model					✓					
	Krause Machine Diagnostics HMI Task	Diagnostics									✓	
	Assembly Machine Monitor and Control HMI Task	Assembly Monitoring & Control			✓							
Lamb Technicon machine HMI	Machine Error Information HMI Task	Actual Errors	✓									
		History of Errors	✓									
		Frequency of Errors	✓									
	Mode Control HMI Task	Initial Position Mode							✓			
		Step Mode							✓			
		Reset Mode							✓			
		Manual Mode							✓			
		Auto Mode							✓			
	Real-time Information HMI Task	Real-time Information		✓								
	Lamb Machine Simulation HMI Task	Solid Model						✓				
	Lamb Machine Diagnostics HMI Task	Diagnostics										✓
	Transfer Line Machine Monitor and Control HMI Task	Assembly Monitoring & Control			✓							

Selecting HMI Task components for an End User HMI Client is dependant on the machines type and the HMI functionality that is required.

The example in figure 7.7 illustrates how the same machine function (machine component control and monitoring) requires different HMI Task components according to the end machine type (assembly machine or transfer line machine in this example). The difference between the two machines Component Monitoring and Control HMI Tasks is how the machine components element states are positioned in the HMI Task.

The sequencing operation of transfer line machine must be represented in the Control and Monitoring HMI Task. The transfer line Machine Component Control and Monitoring HMI Widgets that execute machine moves are positioned down each side of the screen and are placed in the sequence that the machine steps through during its actuation cycle. The left hand side buttons are termed "*goto work*", which could be for example a drilling spindle drilling into the work-piece and the right hand buttons are described as "return to home" which would involve the drill moving out of the work piece.

The requirement for Control and Monitoring HMI Task for an assembly machine is to follow the machines physical structure. Each Screen contains the sensors and actuators for the selected machine component. The navigation between screens follows the hierarchical structure to the machines design.

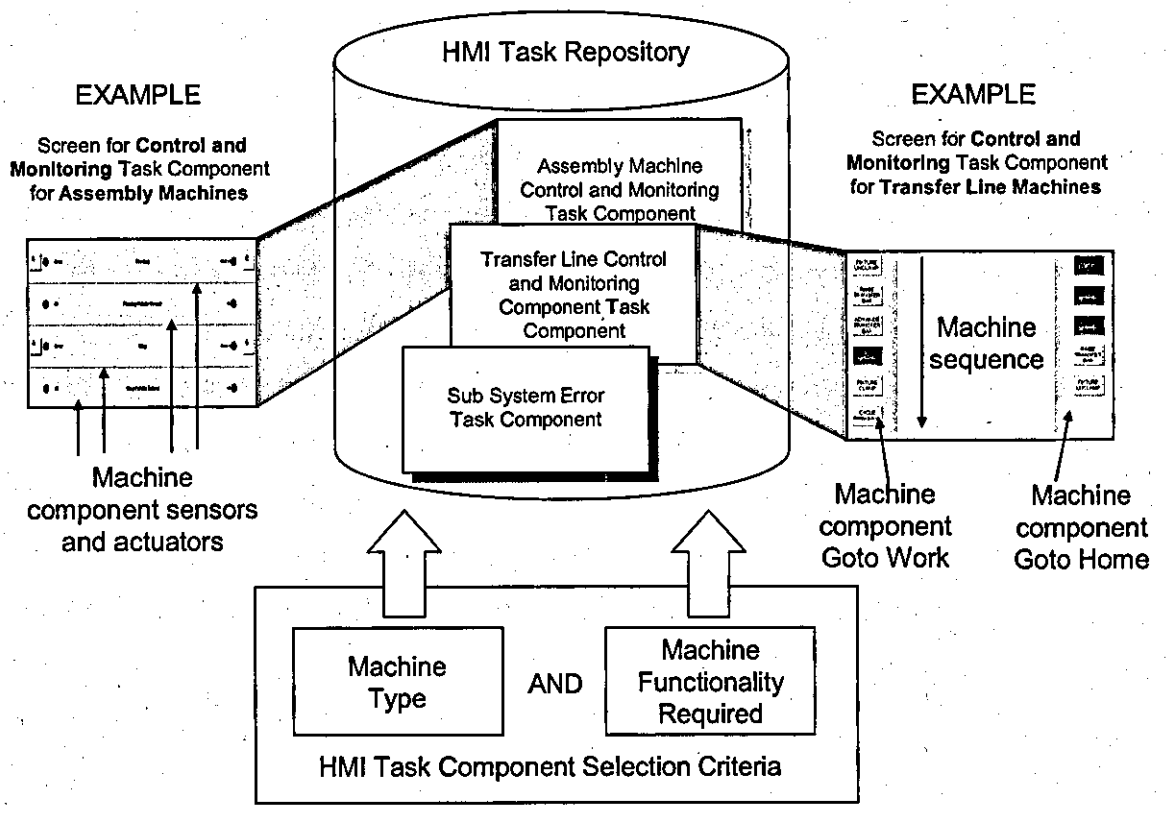


Figure 7.7 Criteria for selecting HMI Task components for a client HMI

7.3.3 HMI Composition

An End User HMI Configuration Tool has been developed to support the construction of new End User HMI Clients. One of the major concepts of the End User HMI Configuration Tool that it will support storyboarding HMI models (see section 7.4.4) to be directly entered and the composition of complete End User HMI Clients. The End User Configuration Tool is within an Integrated Engineering Environment that supports the design, install, testing, simulation and testing of manufacturing machine (see section 6.2). The End User HMI Configuration Tool supports the end users in selecting HMI Tasks from a HMI Task Repository to build new End User HMI Clients. This tool also supports the deployment of exiting End User HMI Clients to new machine applications.

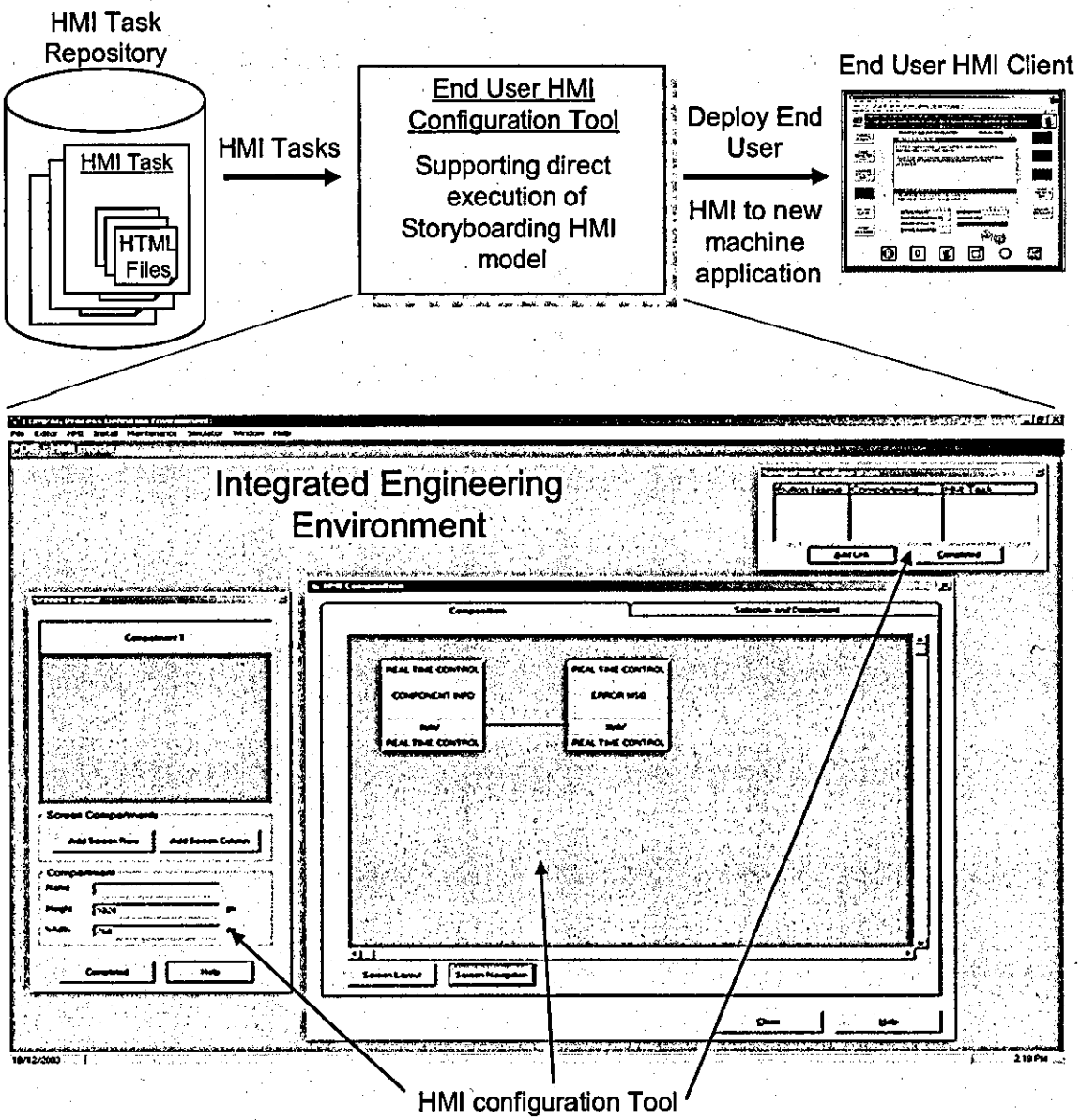


Figure 7.8 Client HMI Composition Tool developed in these case studies

7.3.4 HMI Server Implementation

The HMI System is based on Client Server architecture and its implementation in these case studies is illustrated in figure 7.9. Major elements of the server architecture (detailed in chapter 5 of this thesis) are; 1) a web server, 2) web applications and 3) a common machine data repository.

End User HMI Client's use a web browser to perform a HTTP requests and retrieve web pages from the web server. The web server implemented in these case studies uses server side scripting technologies (PHP scripting engine adopted in this implementation) to generate HTML pages based on machine configuration information in the common machine data

repository. PHP pages are used that contain scripting expressions that the server side scripting engine executes to generate the HTML page that is displayed in the HMI Client.

The HMI Servers Web Applications support the HMI Client with particular function, for example the Web Applications implemented in this case studies support the HMI Client in requesting; 1) machine configuration information used in the Control and Monitoring HMI Task, 2) machine error events used to provide information for the Machine Error Information HMI Task and 3) Machine Control Commands that are used in the Machine Mode Control HMI Task and the requests for to move machine components in the Machine Control and Monitoring HMI Task.

The Web Applications are initiated by HTTP requests from the HMI Client. The HMI Client contains a HTML Form and the form parameters (see section 7.5.5) determine which web application responds to the request. Figure 7.9 illustrates an example HTTP requests the machine command "RESET". The parameters in the HTML form are; 1) the action which in this example is "SSCOMAND", 2) the value which is "Reset". The machine Control Command Web application responds to this HTTP request and resets the machine mode.

The client HMI contains a Java Applet (stand alone application within the web browser) that have direct real-time communications with the Broadcaster web application via a TCP/IP socket connection which is used to provide information in the Real-time Machine Information HMI Task. The Broadcaster (see chapter 5 of this thesis) connects receives information to the Machine Dynamic Run-time Information in the common machine data repository.

The common machine data repository that has been implemented on the HMI server in these cases studies is a MYSQL database. The information in the data repository is divided into two areas; 1) Machine Dynamic Run-time Information and 2) Machine Configuration Information. The Machine Dynamic Run-time Information consists of; 1) Current Sub System Errors, 2) a Log of Errors, 3) Current machine components states and 4) the current sub system mode. All these different types of information are text data types. The Machine

Configuration Information related to the HMI system is the machines control systems elements. A machine is made up of sub-systems/stations, which are made up of modules containing one or more components. Each component contains one or more elements, each with their own specific state behaviour, interlocks. The names and structure of the control systems is stored in the data repository as textual data type. Each component contains a Diagnostic HMI Widget and a hyperlink reference is stored in the data repository to point to the URL of the supporting HMI widget. Each sub system has a corresponding Simulation HMI Widget and the data repository contains a hyperlink reference to point to the URL of the supporting HMI Widget.

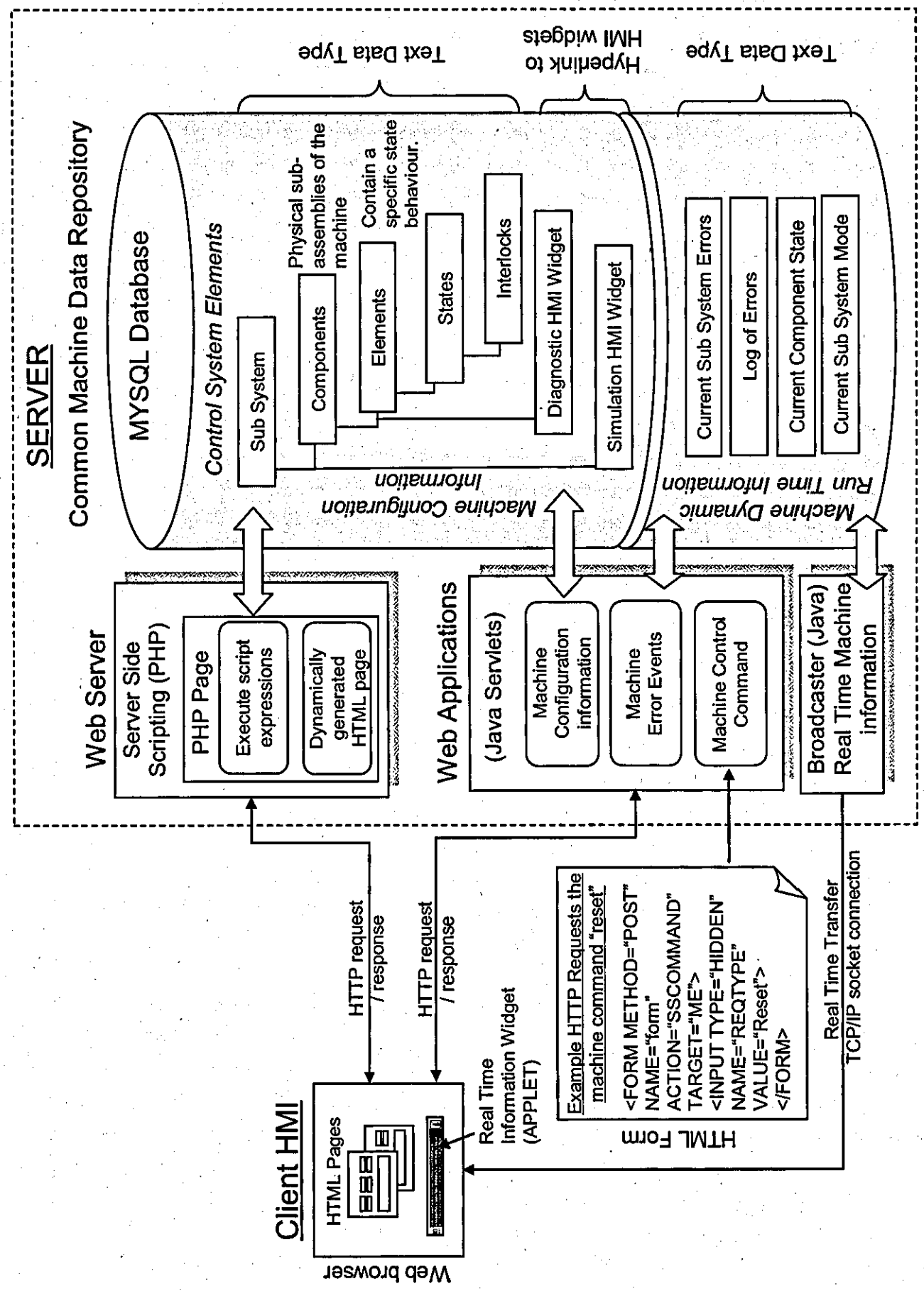


Figure 7.9 HMI Server Applications and Database used for in all case studies

7.4 Case Study 1 – Assembly Automation Machine

Assembly automation involves the design and manufacture of machine that enables physical components to be assembled into complete manufactured parts or sub assembles. The assembly operation can be divided into two basic types of operations [88, 89 and 90]; 1) Primary operations that directly add value to the assembly product, for example placing engine tappet components into a automotive cylinder head casting and 2) secondary operations that provide the supporting roles to the primary operations, for example transportation to and from assembly stations.

Automotive engine assembly systems require specialised assembly equipment to be built [91]. The example in figure 7.10 shows an engine block assembly plant. It consists of a transport system that links assembly and test stations. Base engine blocks are loaded onto empty pallets on the transport system at the "Start" and then carried into different assembly stations where engine parts such as pistons, conrods and the cylinder head are assembled to the engine block. At the end of the process the fully assembled engines are tested in hot test cells prior to being unloaded and sent to vehicle assembly plants [93].

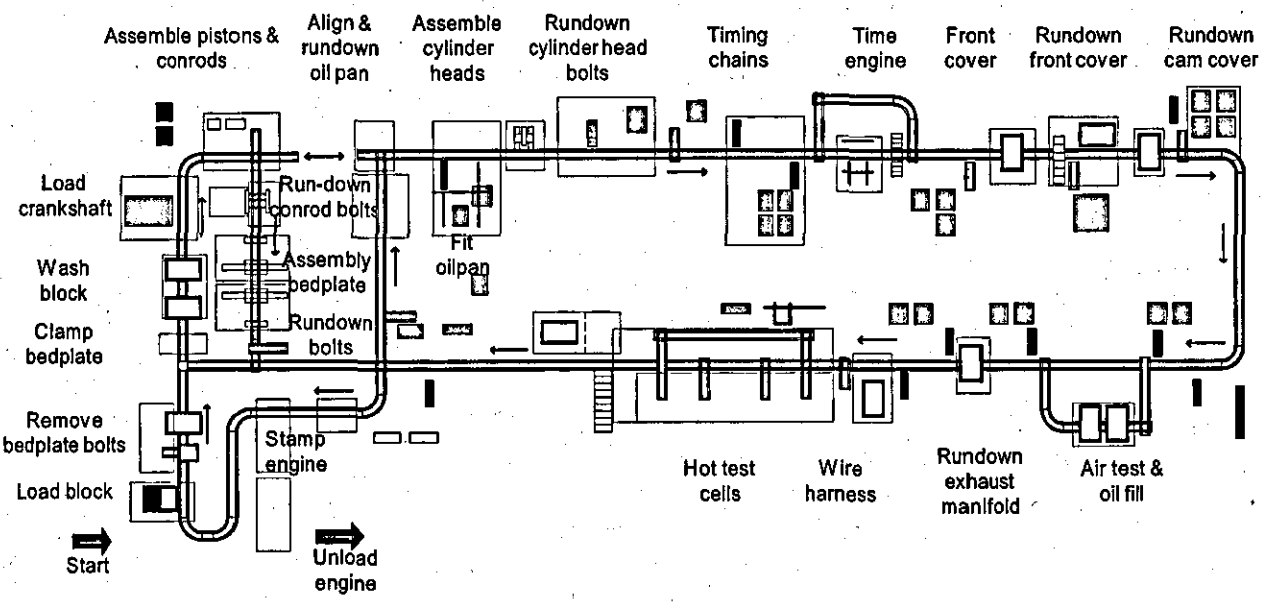


Figure 7.10 Example of Automotive Engine Assembly Plant. Source J. A. Krause Machinenfarik GmbH

7.4.1 Krause Test Rig Details

The case study at J.A Krause GmbH is based on the implementation of the C-B HMI concept on a full size machine used by the company for evaluating new control equipment and systems. The assembly task is to pick valve springs or tappets from feed trays and place them onto engine blocks mounted on a pallet. Figure 7.11 illustrates two sub systems that comprise the Krause test rig; 1) the Assembly Sub System and 2) the Transport Sub System.

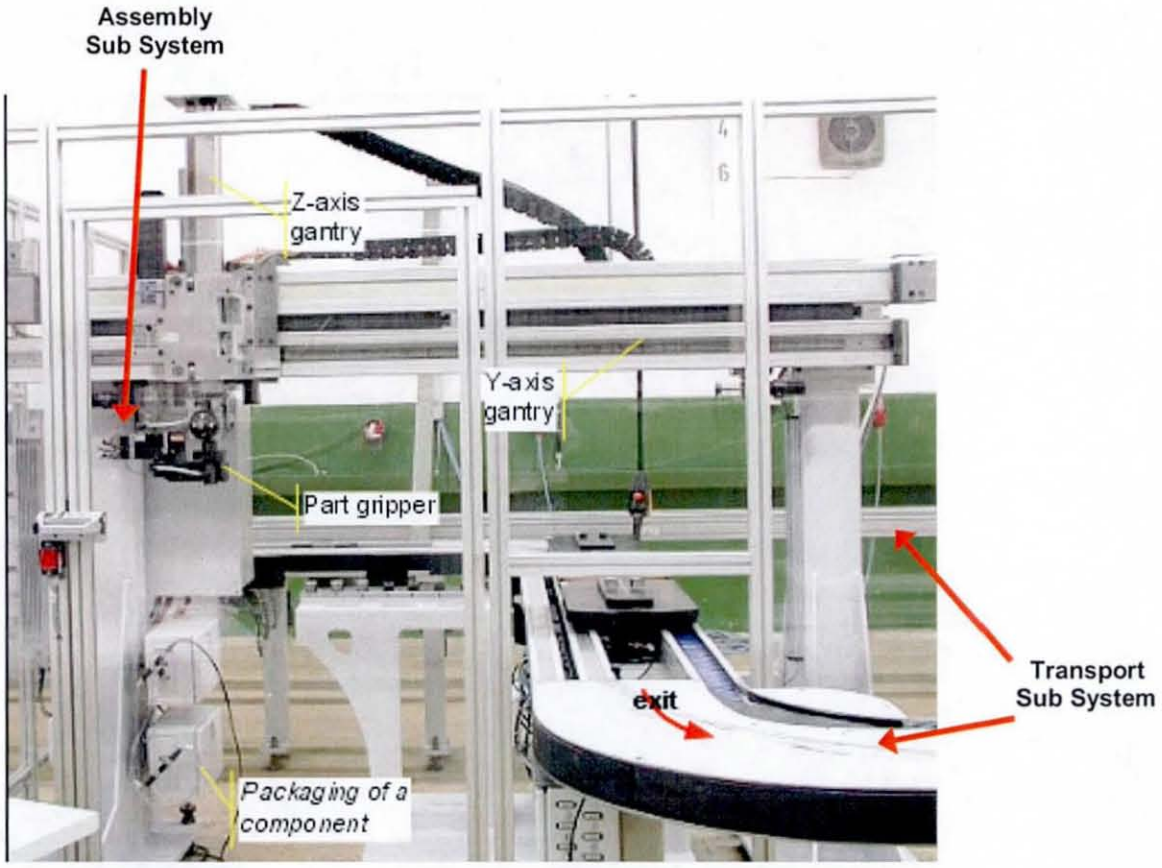


Figure 7.11 Actual J.A Krause Demonstration machine illustrating Assembly Sub System and major components and Transport Sub System

The Krause test rig schematic (figure 7.12) illustrates the machine components for each sub system. The Transport System consists of a number of conveyor belts drives, stops, Radio Frequency IDentification (RFID) readers and a conveyor line diverter. Pallets are sorted according to the RFID tag information mounted underneath each pallet. The Assembly System has a Feeding Unit and a Pick and Place Unit. The Feeding unit consists of a stop and pre stop, identification system, section monitoring and fixing Unit. The Pick and Place Unit has two servo drives, a pneumatic gripper and an ultrasonic sensor for checking the

availability of parts. Each sub system has an individual Human Machine Interface (HMI) for operation control.

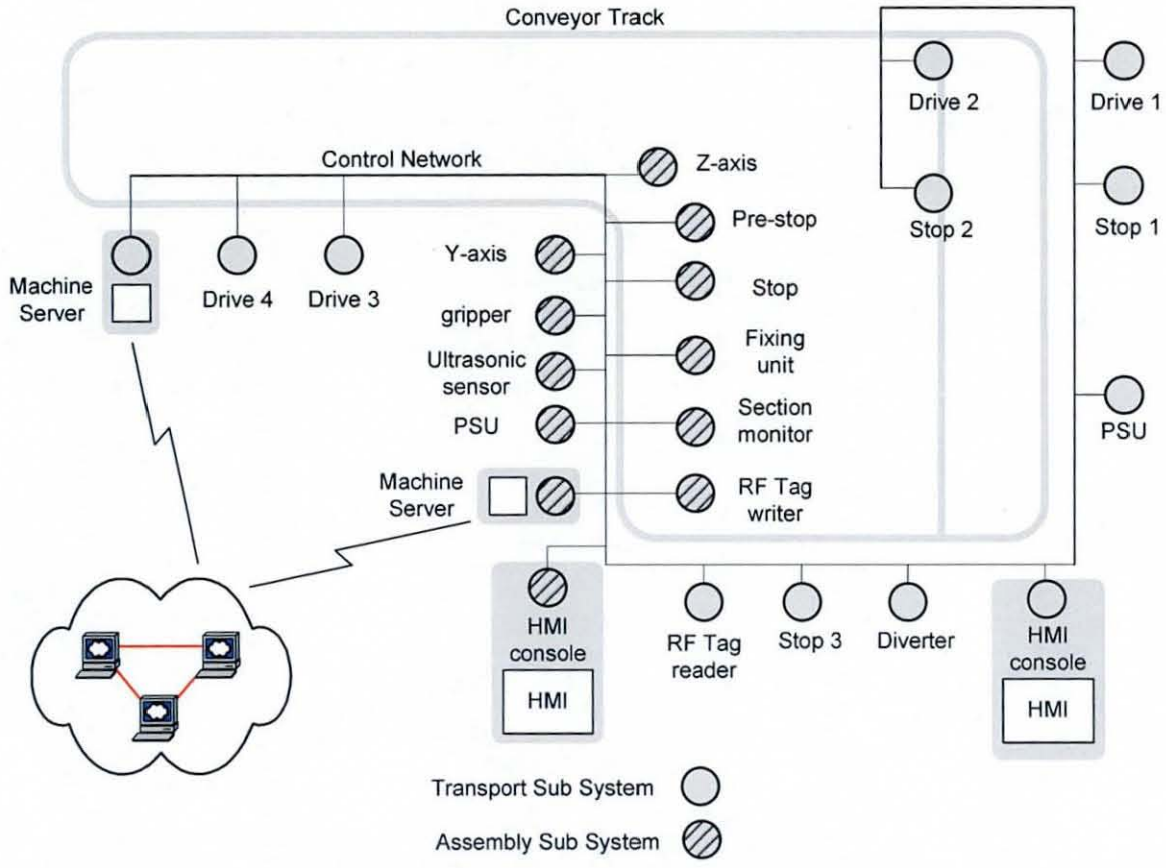


Figure 7.12 Schematic of J.A. Krause Case Study demonstration machine

7.4.2 HMI Implementation

The basic design and layout of the End User HMI Client for the automotive assembly machine is illustrated in figure 7.13. The HMI screen layout has been designed by dividing the screen into a number of compartments and determining the machine tasks to be contained in each compartment. The complete End User HMI Client contains many different HMI screens and hence the machine tasks to be displayed need to be determined. Generic HMI Tasks such as navigation must be displayed on every page together with real-time machine information and sub system control if this is required in the particular End User HMI Client. The main centrally positioned screen compartment in this case study End User HMI Client displays any functionality that is not required to be constantly displayed and user must navigate to. HMI functionality of this type are error information, component control and monitoring, component diagnostics and machine simulation models.

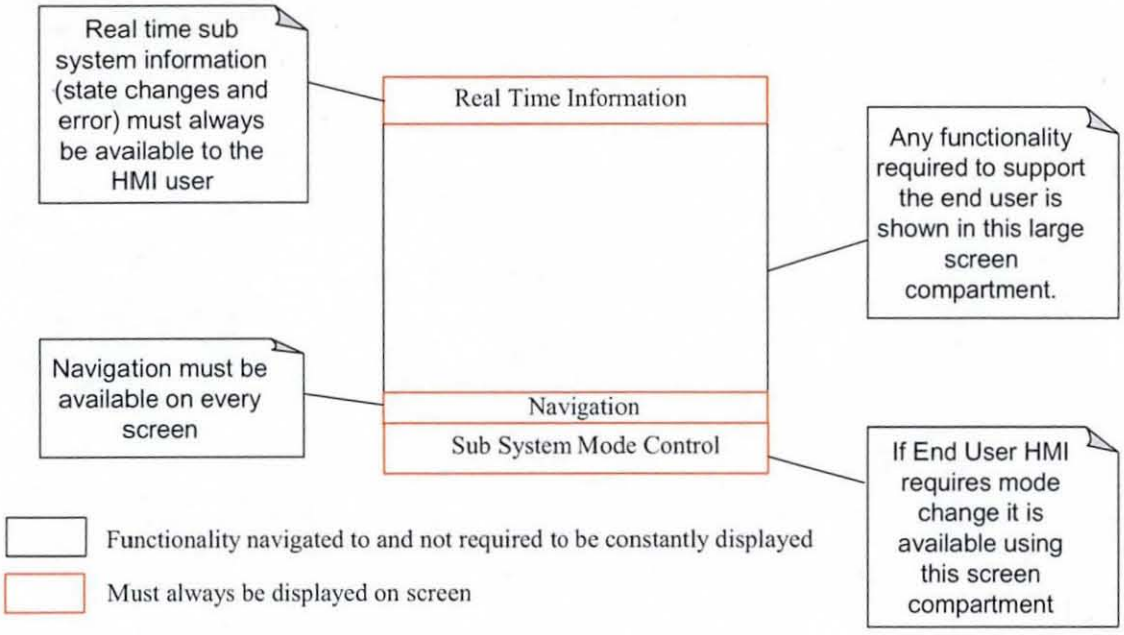


Figure 7.13 Design and layout of the End User HMI for the automotive assembly machine at Krause GmbH

Figure 7.14 displayed the real HMI console that was implemented on the J.A. Krause Case Study demonstration machine. Two HMI consoles, one for each machine sub system were implemented to support users performing machine tasks locally to each machine sub system. The HMI console consists of a touch screen based PC and a standard web browser with internet connectivity. Since the HMI implementation is based on WWW technology (see chapter 5 of this thesis) any end user HMI can be displayed on this machine, however its purpose is only to display the HMI that is required by the machine operators.

A storyboarding model (see chapter 4 of this thesis) of the Krause machines HMI is illustrated in figure 7.15. This model represents the screen functionality, the screen layouts and their navigation. The functionality for the End User HMI Client is encapsulated in HMI Tasks (see the functional architecture described in chapter 5 of this thesis) and the HMI Tasks implemented in the HMI to support the Krause machine are detailed in table 7.1. The Real-Time Information HMI Task, Navigational HMI Task and Mode Control HMI Task are required on every screen 88, 89 and 90]. From the Machine Monitor and Control HMI functionality (Assembly Machine Monitor and Control HMI Task) the operator can navigate to; 1) the Diagnostics HMI functionality (Ford Test Rig Diagnostics HMI Task) using the

Diagnostic button on the navigation (Navigation HMI Task) or 2) the machine mimic (Krause Simulation HMI Task) using the Mimic navigation button or 3) the machine error information (Machine Error Information HMI Task). From the machine diagnostic screen the operator can only navigate to the machine monitoring and control screen. The operator can navigate to the machine error information or the machine monitoring and control from the machine mimic. From the machine error information the user can navigate to the machine mimic.

Figure 7.14 Implemented HMI Console on J.A. Krause Case Study demonstration machine

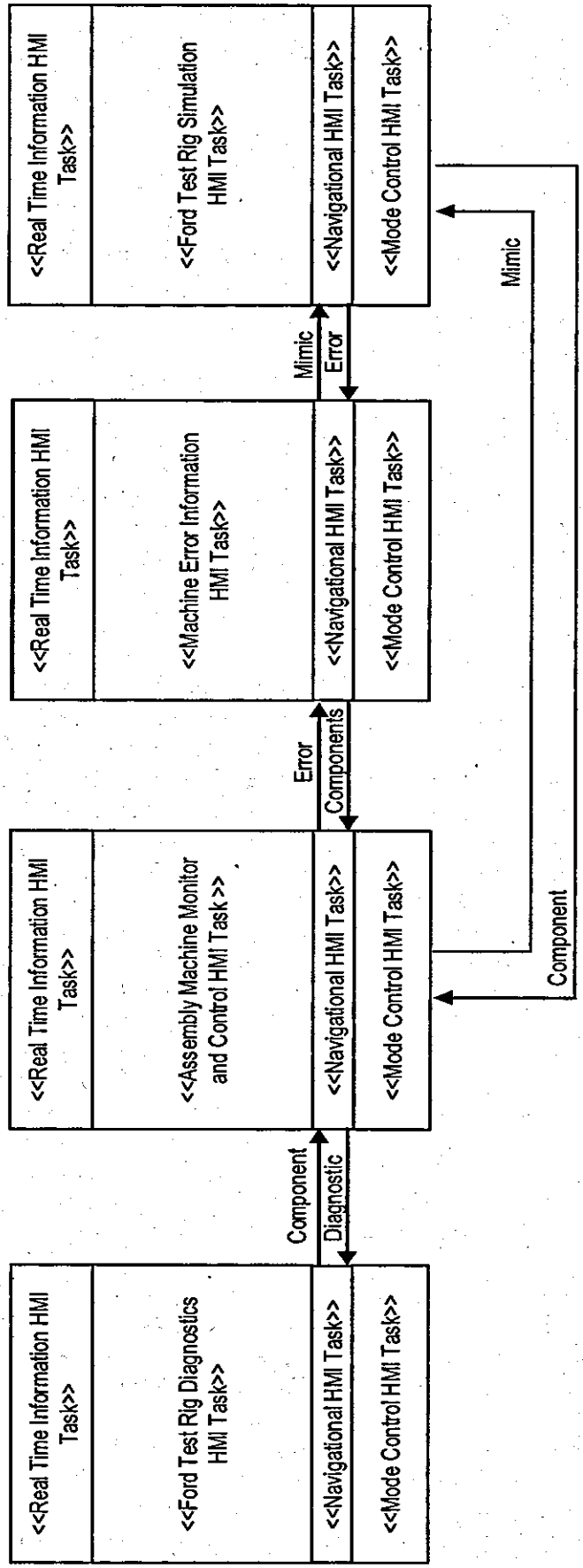


Figure 7.15 Storyboarding model for Automotive Assembly Machines (Krause and Ford Test Rig End User HMI)

7.4.3 System Attributes Testing

In this section expected benefits of utilising the C-B HMI approach are tested using scenarios appropriate to this case study. The benefits (see table 7.3) that are to be tested is the rapid HMI composition. The enabling system attributes that support the rapid HMI composition are a high degree of reusable HMI elements and a low skill set required to construct the end user HMI's. The implementation of the HMI system is based upon a directly executable HMI model that supports the low skill level enabling system attribute. The high degree of system re-use is implemented in the system through storing reusable elements, machine tasks components, the building blocks for end user HMI's and end user HMI in system libraries.

Table 7.3 Rapid Composition benefit tested using this case study.

Benefits Tested	Enabler	Implementation
Rapid HMI Composition	Low skill level	Directly executable HMI model
	System reuse	End user HMI stored in libraries
		Machine task components stored in libraries

7.4.4 Directly Executable HMI Model

Figure 7.15 illustrates a End User HMI Client for a Machine Engineer composed using the End User HMI Configuration Tool that supports the direct execution of an HMI model (see chapter 4 of this thesis) into a fully functional HMI. The model is constructed using the HMI Configuration Tool and requires no specialist programming skills. The HMI model is used to specify the functionality that is required in the End User HMI and the layout of the screens. The functionality in the HMI architecture is defined by the machine tasks. The layout of the HMI screens is described by decomposing the screen into compartments. The machine engineer HMI illustrated in figure 7.16 contains four different types of screens. The four types of screens in this example all have screen compartments that contain real-time sub system information and navigation tasks and in addition they contain a machine tasks to support their screen function.

Machine Component Monitor and Control Screen (illustrated in figure 7.16, 1) contains the component monitoring and control HMI Task. The component monitoring and control HMI

Task consists of two types of screen. One screen provides a listing of all machine components and allows the users to select a component and navigate to its elements. The selected components elements page contains the HMI Widgets for the associated elements and provides the user with the component element state and control. From the component monitoring and control screens the user can navigate to the component diagnostics, machine errors or the machine mimic.

Machine Mimic Screen contains the simulation model machine HMI Task (illustrated in figure 7.16, 2). The simulation model machine task provides a fully animated three dimensional model of the machine. From the machine mimic screen the users can navigate to the component control and monitoring or the error machine task.

Machine Components and Diagnostics Screen this contains the component diagnostics HMI Task (illustrated in figure 7.16, 3). The component diagnostic HMI Task consist of two types of screen. One screen displays all machine components and diagnostic information available to enable the user to navigation to the component diagnostic information shown on the second screen. The component diagnostic HMI Task can only be navigated to from the component control and monitoring machine HMI Task.

The Machine Error Screen contains the Machine Error Information HMI Task (illustrated in figure 7.16, 4). This HMI Task contains three types of industry standard error screen information; a) actual machine errors, b) history of errors and c) most frequent errors. The HMI Error Widget is used on each screen and the machine error task configures the HMI Error Widget to show the error type information. From the Machine Error Information HMI Task the user can navigate to the Component Control and Monitoring HMI Task and the Machine Mimic HMI Task.

The screens within the HMI Tasks that are selected using the End User HMI Configuration Tool and are hidden from the user when composing the End User HMI Client to simplify the process so that low skill users can compose HMI's. The End User is only concerned with the HMI Task they need to perform and not the internal working. It is the responsibility of the

machine process and HMI engineers to use their experience and knowledge of the machine to encapsulate the appropriate functionality within the machine task library.

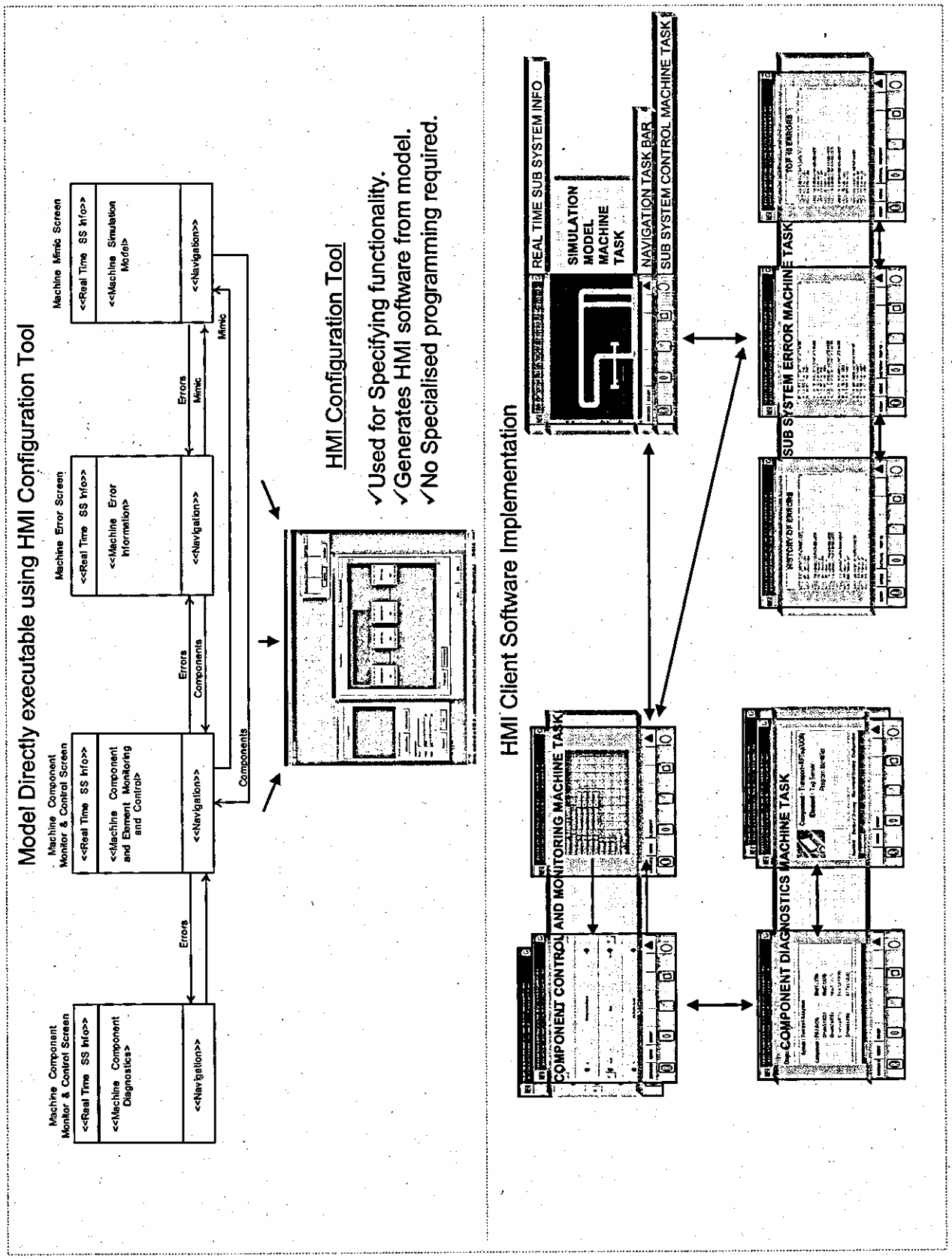


Figure 7.16 Machine Engineer HMI Client supporting rapid composition

7.4.5 System Elements Reuse

The concept of reuse of HMI Task components in many applications is illustrated in figure 7.17. The HMI Task component is implemented as a template pattern that specifies the screen layout and the HMI widgets required. In addition the HMI Task component dynamically generates at run-time (each time the page is refreshed in the web browser) browser HTML code from the machine data repository. With this system architecture where the task logic is separated from the machine configuration data, the machine task can be used within many applications that have different machine interlocking logic, machine physical design and / or machine component selection

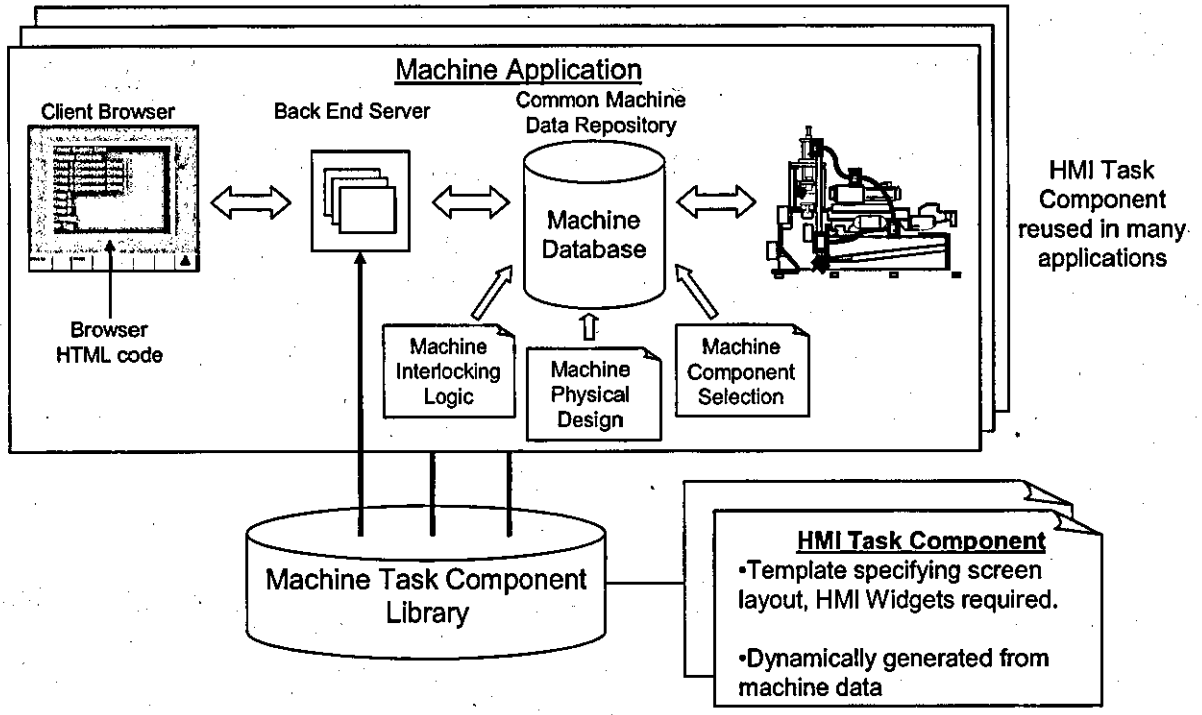


Figure 7.17 Concept of reusable machine task component across many applications

The Assembly Machine Monitor and Control HMI Task is used in this case study to demonstrate its reuse across different machine applications. This HMI Task is intended to be reused in different manufacturing assembly machine applications. Of the three case studies described in this chapter, the Ford Test Rig and Krause machines are both manufacturing assembly type machines and the same Assembly Machine Monitor and Control HMI Task is reused in each machine HMI System. Figure 7.18 illustrates a template for an Assembly Machine Monitoring and Control HMI Task. It's defined as a template storyboard because it

is not describing a particular machine configuration; it is a template that describes the types of HMI Widgets on each screen and the navigation between them. The Assembly Machine Monitoring and Control HMI task consists of two types of screens, the assembly machine component screen which displays all the components in the sub system and the assembly machine element screen which is navigated by selecting a component HMI widget and displays all elements in the selected component.

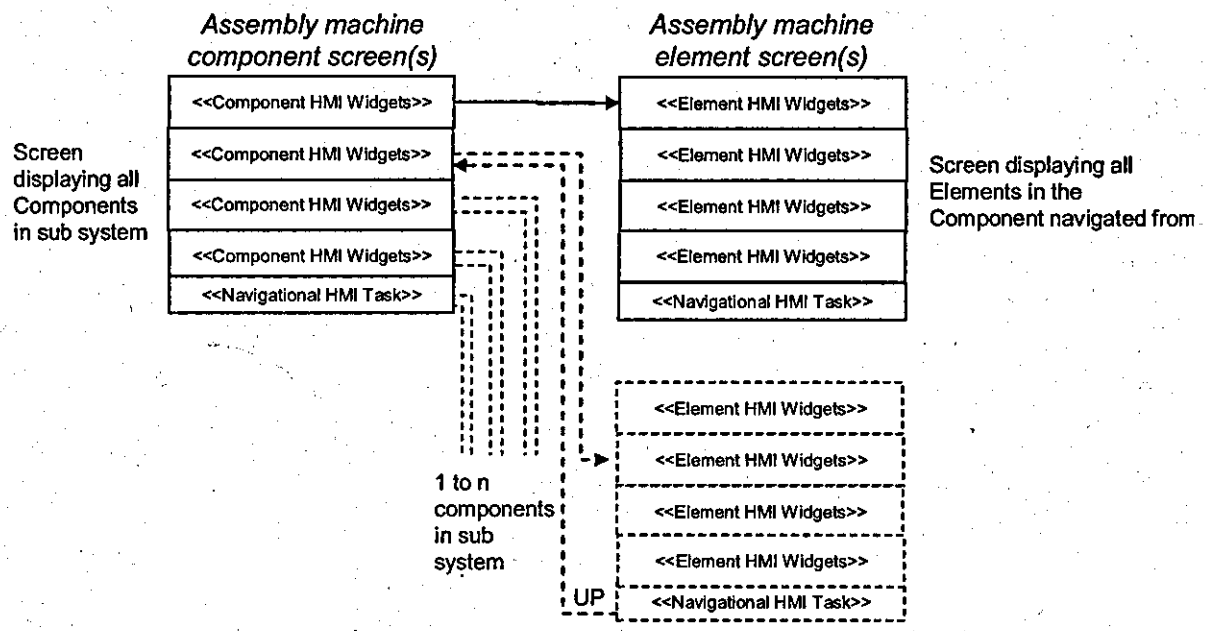


Figure 7.18 Storyboarding template for the Assembly Machine Monitoring and Control HMI Task

The assembly machine component screen(s) implementation supports reusability across many machine applications by generating the machine task screen from machine configuration information as illustrated in figure 7.19.

When the Assembly Machine Monitor and Control HMI Task is requested by the HMI Client a server side script is executed on the web server. This script connects to the machine database and retrieves all the machine component names in the sub system. The sub system component names and links to elements formatted on a HTML page which is displayed in the client web browser.

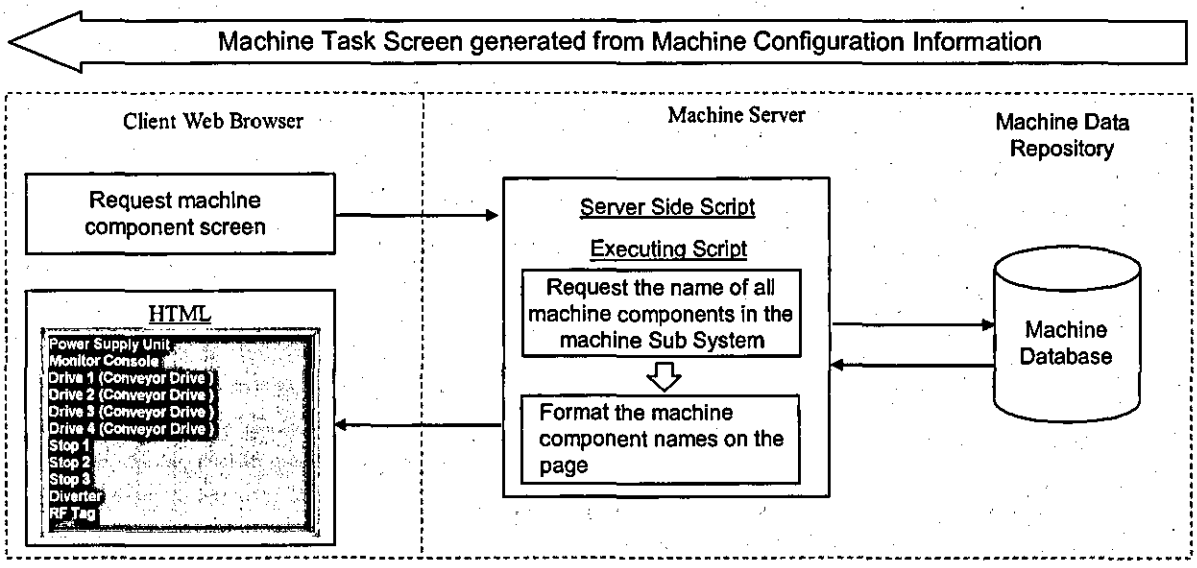


Figure 7.19 Component Assembly Machine Monitoring and Control HMI Task Component screen implementation to support reusable in many machine applications.

The implementation of element screens within the Assembly Machine Monitor and Control HMI Task is illustrated in figure 7.20. This screen is generated using a server side script to retrieve from the machine database the element name and their associated HMI widgets from the selected machine component in the previous task screen. The server side script then creates new instances of the HMI Widgets and configures them with the corresponding component element name. The HMI Widgets are then formatted using HTML in the client web browser.

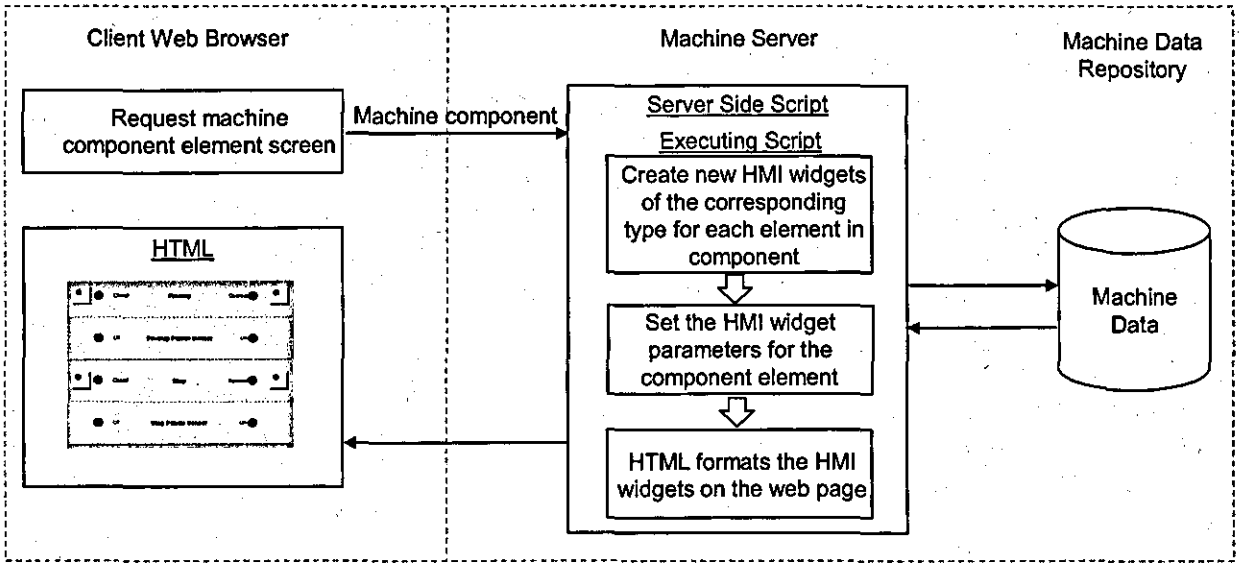


Figure 7.20 Assembly Machine Monitor and Control HMI Task element screen implementation to support which is reusable in many machine applications.

7.5 Case Study 2 –Transfer Line Automated Machine

A typical automated transfer line production machine designed to produce cylinder heads for engine blocks is type is illustrated in figure 7.21.

Transfer line machines generally consist of two major units, work stations and transfer mechanisms. Transfer mechanisms are used to move the work piece from one station to another to enable various operations to be performed on the part. Work pieces are transferred between stations using one or more of the following techniques; 1) rails along which the parts usually on pallets are pushed or pulled, 2) rotary indexing tables or 3) overhead conveyors.

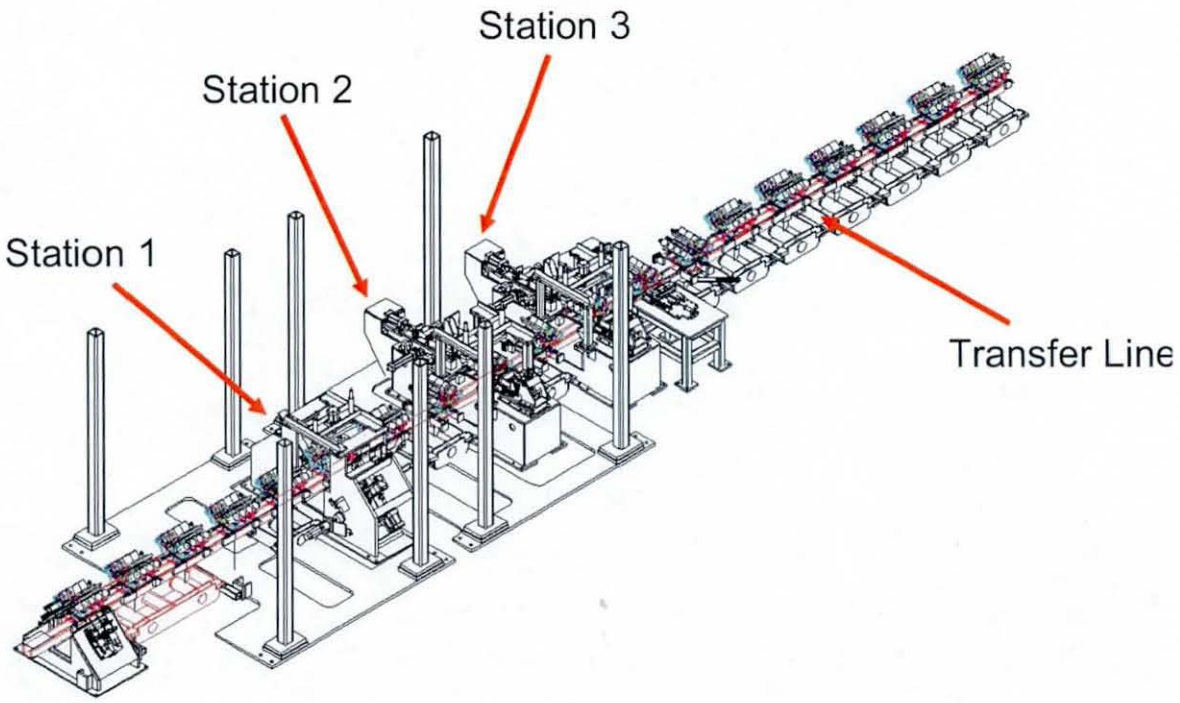


Figure 7.21 Transfer line machine system which consists of three work stations and a transfer line to move work piece between the stations [40].

7.5.1 Lamb Test Rig Details

Lamb Technicon is a major automotive machine supplier and has worked with automotive end users on many production machine engineering plants for the engine block and cylinder head lines. Lamb claims to have *"designed cost-effective machining systems that created the appropriate balance for the block and cylinder head lines' feature adaptability, machine component part commonality, and machine/process robustness"*. This case study is based on the implementation of the C-B HMI concept on a transfer line Machine used by Lamb

Technicon for evaluating new control systems. The Lamb Technicon machine consists of a single station from a transfer line for cylinder head machining. The test machine is made up of the transfer bar mechanism; the part-clamping fixture and a single wing base with a combination of end stop hydraulic and servo electric actuators.

The Lamb Machine consists of three sub systems; 1) Transfer Sub Systems, 2) Fixture Sub System and 3) the Wingbase Sub System (see figure 7.22). The Transfer Sub System consists of two machine components, the Transfer Raise / Lower and the Transfer Advance / Return. The Fixture Sub System consists of a single machine component, the Fixture clamp that has clamped or unclamped states. The Wingbase Sub System consists of two machine components; 1) the Wingbase Home / Cycle which performs an x-y axis cutting path on the work piece and 2) Wingbase Ready Dept that control the z axis (depth) of the cutting tool into the work piece that has two states, clear depth and depth.

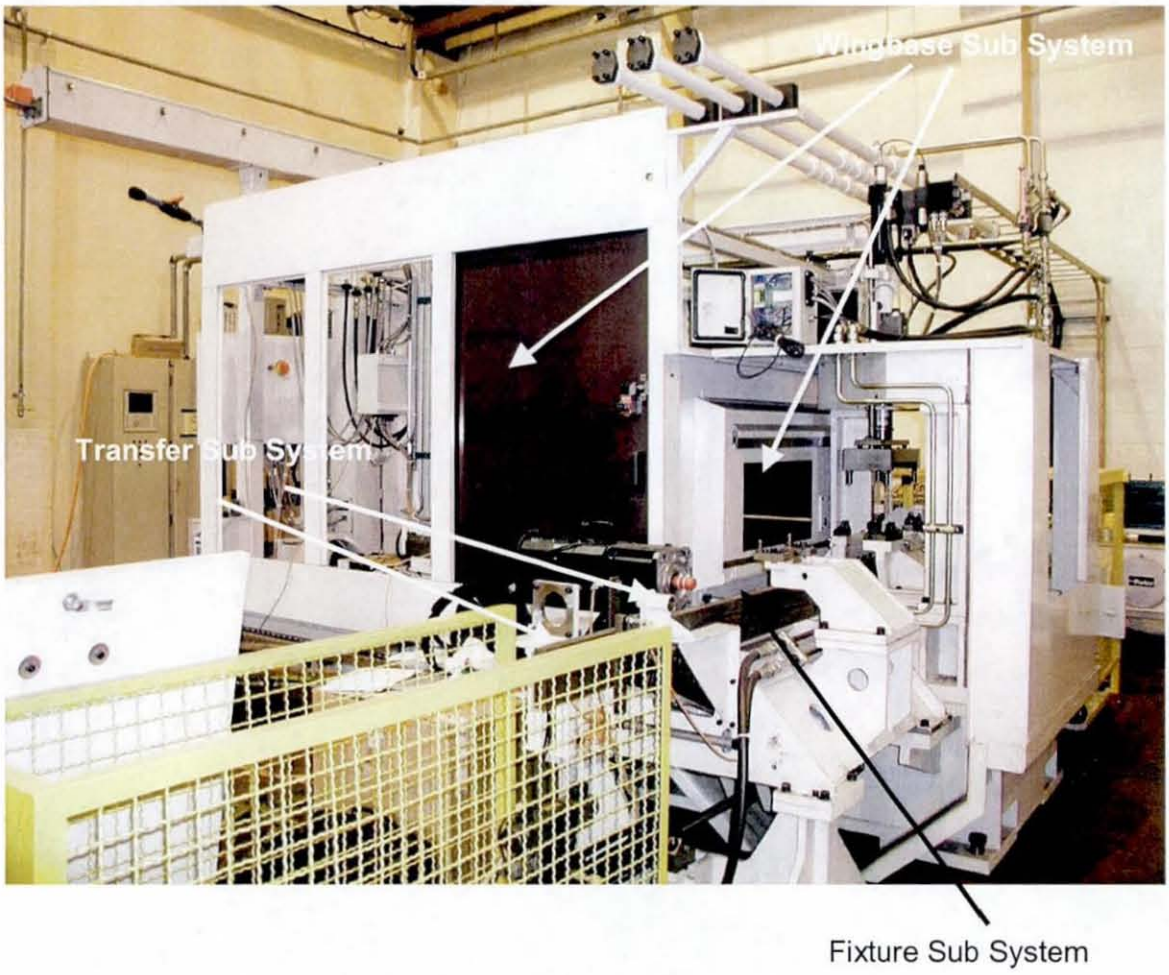


Figure 7.22 General View of the Lamb Machine

The machine actuation sequence of the Lamb Transfer machine is illustrated in figure 7.23.

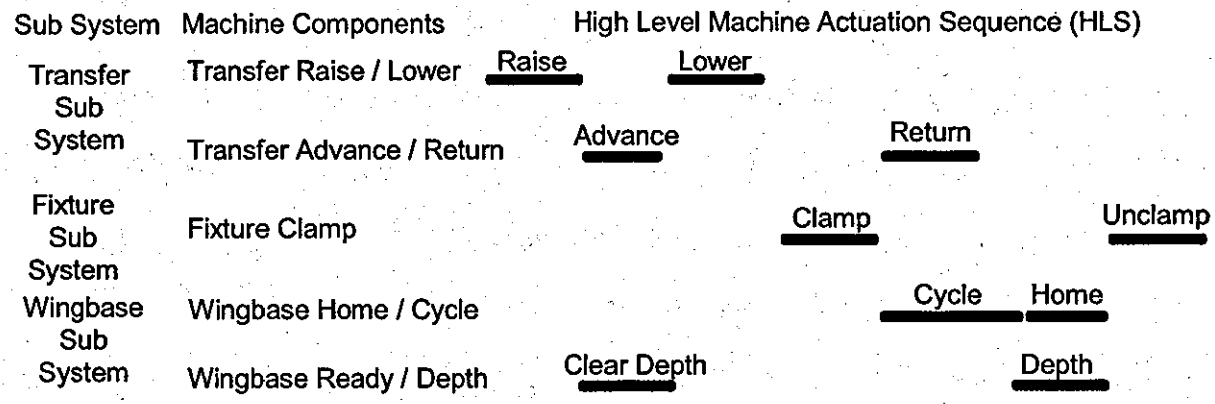


Figure 7.23 Lamb Transfer Line Machine actuation sequence cycle

7.5.2 HMI Implementation

Due to the sequencing nature of the transfer line machine the screen layout for the HMI system has different requirements to the assembly automation machine. The requirement to represent the sequence of the machine is implemented on the HMI screens by having the Component Control and Monitoring HMI Task always visible on the left and right hand sides of the screen (see figure 7.23).

The End User HMI Task has the buttons that monitor and control machine actuation down each side of the screen. The controls have to be ordered on the display in the sequence that the machine sequences through during its cycle. The machine control and monitoring HMI functionality for this case study has requirements that are unique to Transfer line machines and the software component implemented is a Transfer Line Machine Monitor and Control HMI Task.

The Real-Time information, sub system control and the navigation is displayed on the screen in a very similar way to the assembly automation machine. The basic screen layout for the HMI implemented in this case study is illustrated in figure 7.24.

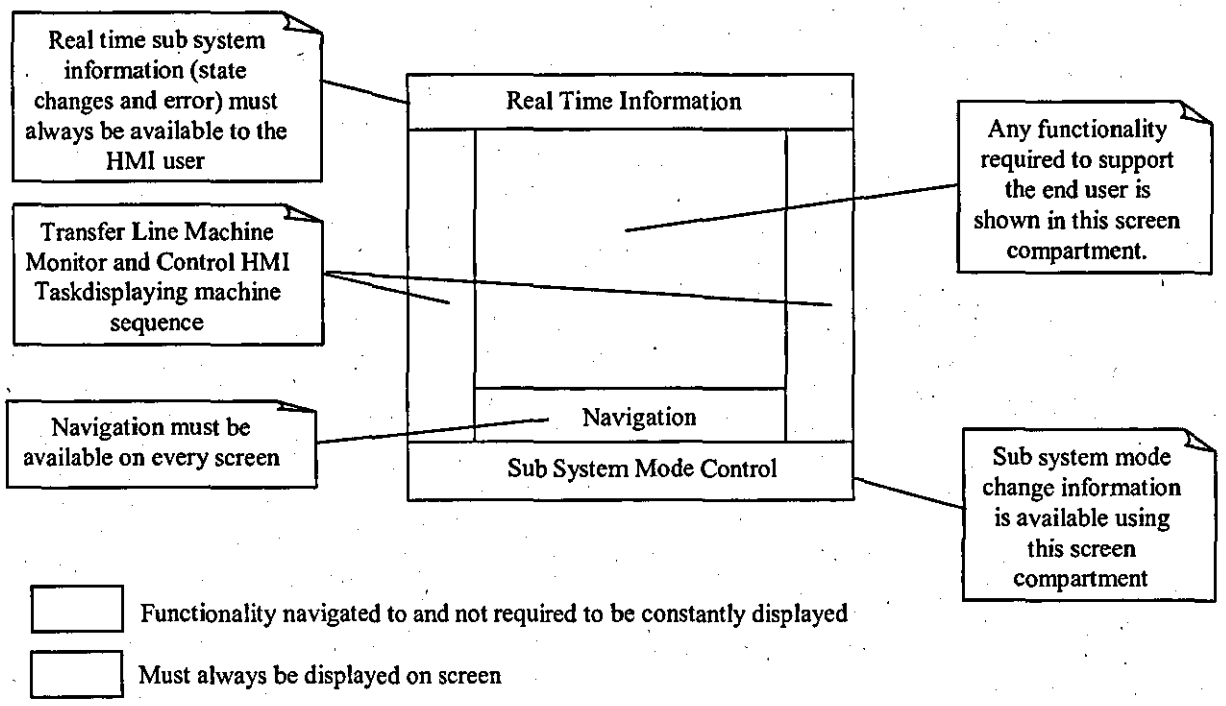


Figure 7.24 Layout of the End User HMI for the automotive transfer line machine

A storyboarding model (see model concepts in chapter 4 of this thesis) of the Lamb machine End User HMI Client is illustrated in figure 7.25. This storyboarding model represents the screen functionality, the screen layouts and the navigational links. The functionality for the End User HMI Client is encapsulated in HMI Tasks (see the functional architecture described in chapter 5 of this thesis). The Real-Time Information HMI Task, Navigational HMI Task and Mode Control HMI Task and the Machine Monitoring and Control HMI Tasks are required on every screen 88, 89 and 90].

Other types of information displayed in the End User HMI Client but not displayed on every screen that the operator can navigate to are; 1) Machine Errors Information HMI Task, 2) Lamb Simulation HMI Task and Diagnostics Information HMI Task.

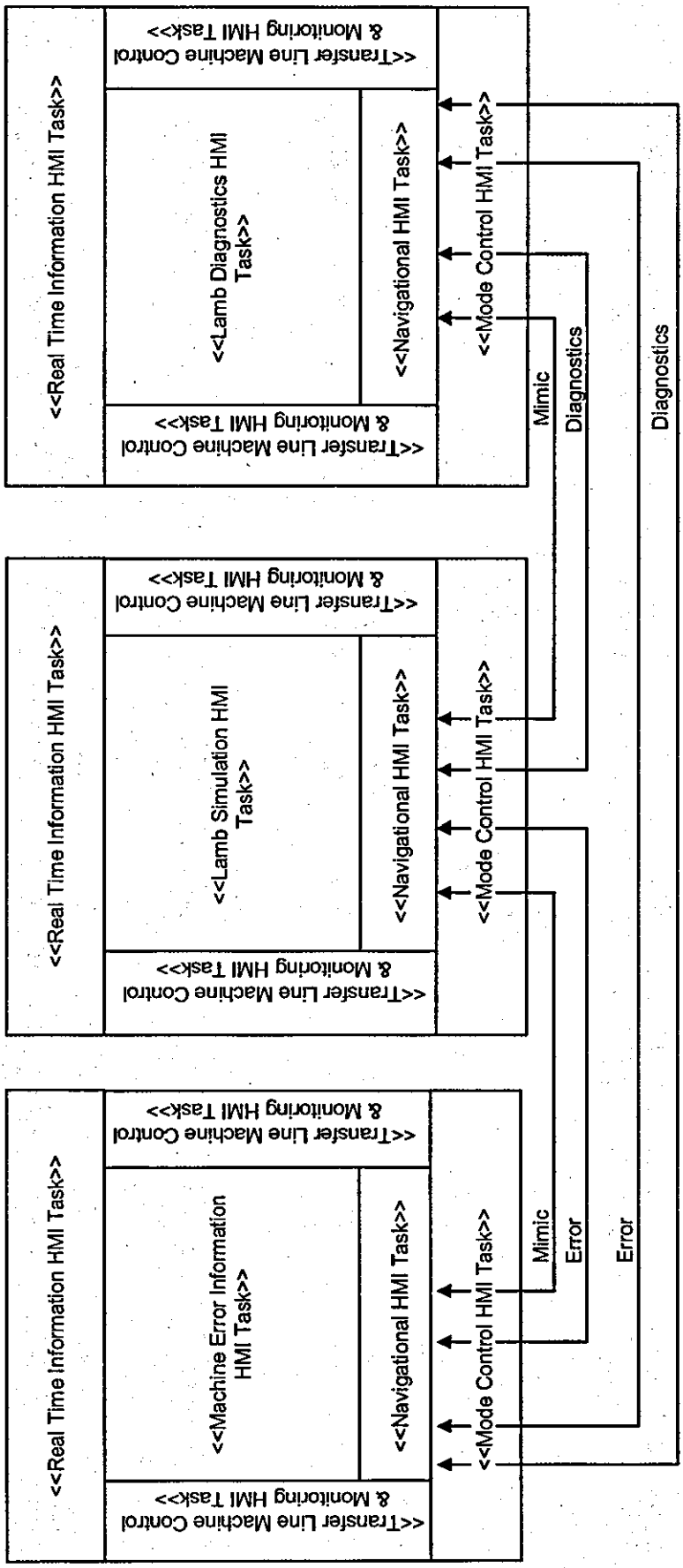


Figure 7.25 Storyboarding model of automotive transfer line machine HMI (Lamb Technicon)

7.5.3 System Attributes Testing

The major benefits has been investigated using this case study is how the system design ensures that the information generated by the machine is consistent in the HMI. This has been identified as a major issue in the correct design of HMI systems. This consistency in the HMI system is enabled through the use of a common data machine model. Section 7.5.4 describes how this case study is used to demonstrate how the machine information is requested from a common machine database and information in the end user HMI is updated automatically to the latest machine configuration.

The second significant benefit that this approach enables industry best practice to be embedded in the system and therefore system developers are less reliant on their experience in the design process. This benefit is demonstrated (in section 7..5.5) in the case study through the standard industry types of error information and mode control required in the machine system.

Table 7.4 HMI benefits demonstrated in this case study

Benefits Tested	Enabler	Implementation
Machine Consistent HMI Messages and Control	Common data machine model	Machine database where all HMI data is sourced
Industry best practice HMI	Industry best practice knowledge embedded in components	Industry standard error information embedded in system.
		Industry standard mode control embedded in system

7.5.4 Machine Consistent HMI Messages and Control

Figure 7.26 is a schematic diagram that illustrates how the HMI system supports consistency within the machine system during any reconfiguration activities. Machine re-configuration may involve machine physical design changes, alternative machine component selection and changes to the machine interlocking logic. Machine information that is changed during the reconfiguration activity is stored in the machine data repository. The machine HMI is always consistent with machine data since it is automatically generated from the machine data repository. Within the HMI systems architecture (see chapter 6 of this thesis) HMI Applications process information where it is then displayed in the client browser (see section 7.3.4).

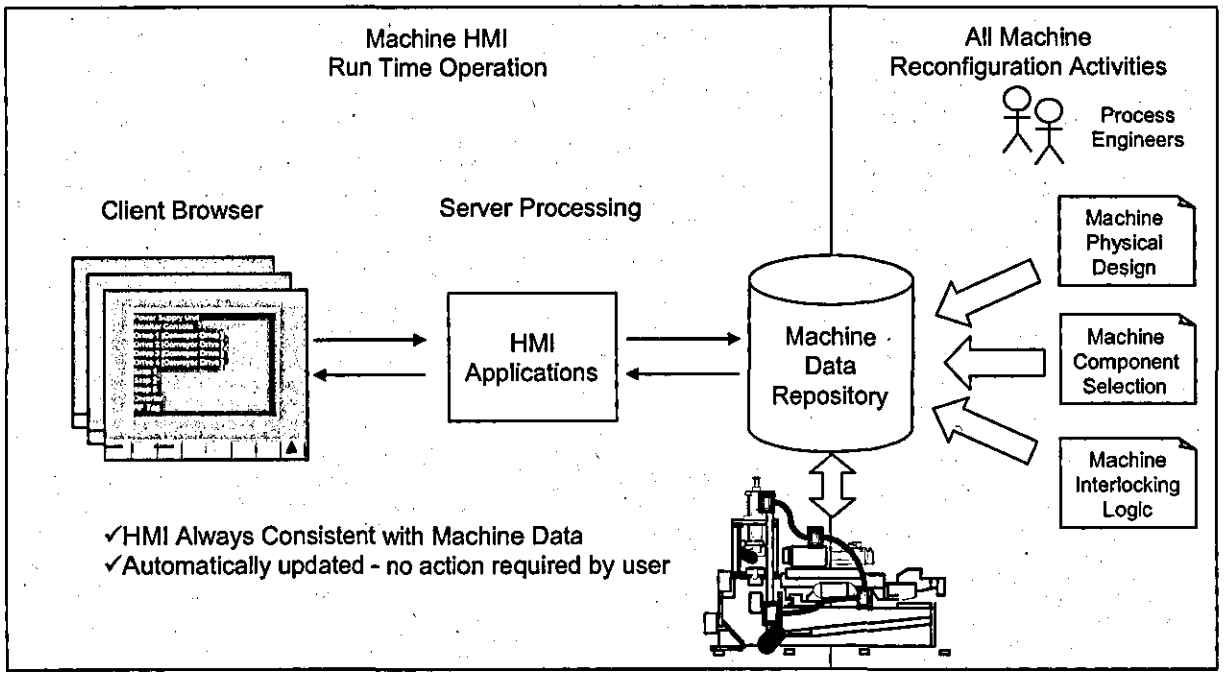


Figure 7.26 HMI system supports consistency with the machine system during any reconfiguration activities.

An example scenario of the HMI system providing accurate machine information throughout reconfiguration changes during its lifecycle are illustrated in figure 7.27. The scenario is that a change to the machine sensor is required. This change requires all associated machine sensor information to be updated so that the correct error, state message relating to the sensor is displayed in the HMI. Process engineers use the Integrated Engineering Environment which is a set of tools within the Engineering Software Environment that support the machine change. All machine information that is required in these engineering tools is extracted and stored in the machine common data model. Any events generated from the machine system are propagated to the HMI applications and HMI client via the machine common data model to ensure that the consistent information is used in the HMI system. The real-time machine information task is implemented in the form of a Java Applet Application that connects directly to the Broadcaster on the machine server. Hence the HMI always displays consistent machine sensor error information to machine operators. This provides a substantial advantage over traditional systems where a change in sensors on one machine would typically result in

the need to update three different databases, i.e. the programmable logic controller, operator panel and HMI system.

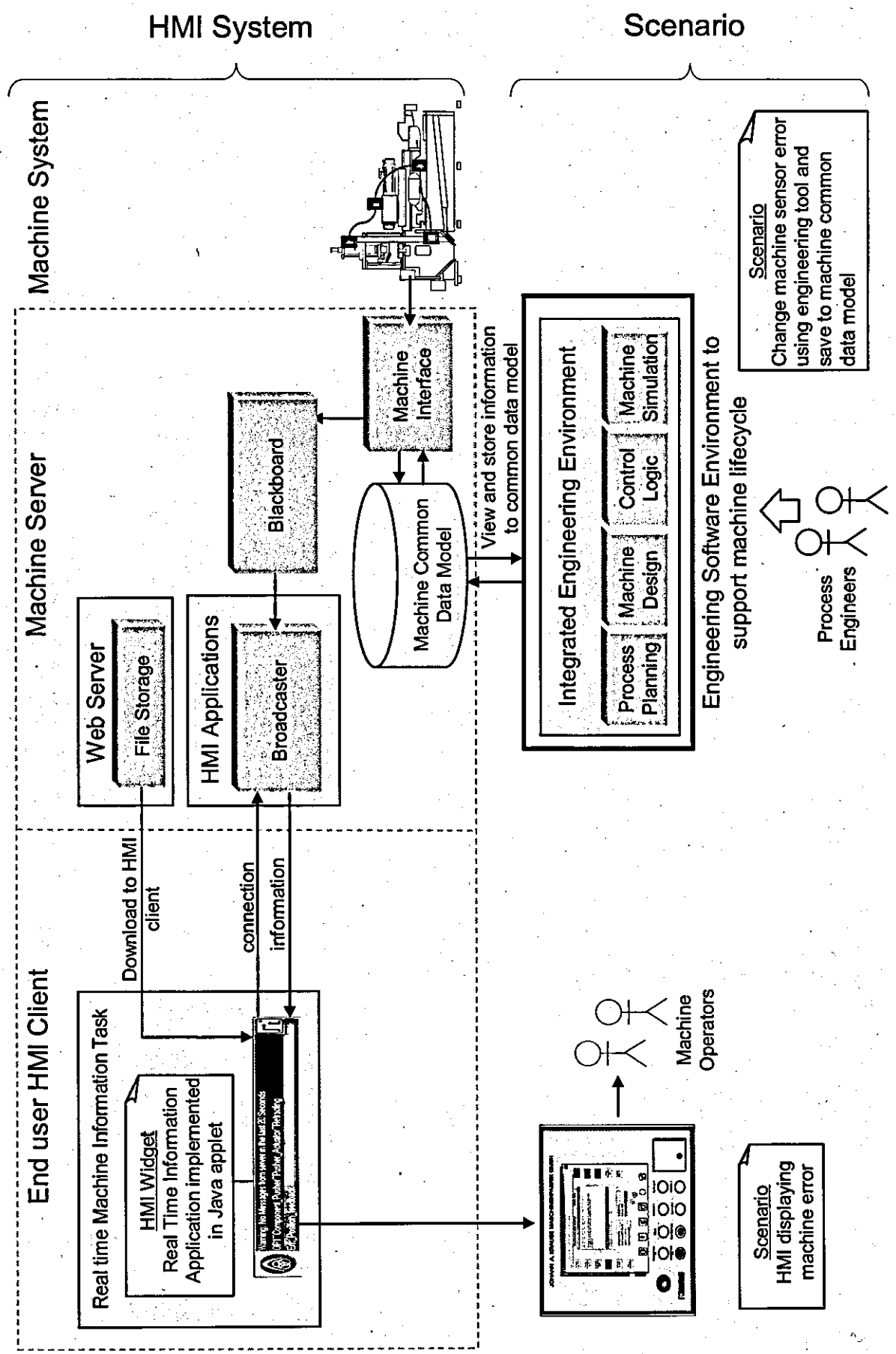


Figure 7.27 Scenario of the HMI system providing accurate machine information throughout reconfiguration changes during its lifecycle

7.5.5 HMI Industry Best Practice

Within the domain of automotive manufacturing there are established and proven methods of performing machine tasks [88, 89 and 90]. Any well designed HMI System must support this industry best practice to provide high usable HMI systems. Figure 7.28 illustrates two examples of industry best practice having been embedded within the HMI components (the HMI Tasks and HMI Applications software components) and subsequently reused in all future HMI implementations.

The first example of industry best practice that has been encapsulated in the HMI components are the three types of machine error information; 1) Actual Errors 2) History of Errors and 3) Most Frequent Errors

The second example of industry best practice that has been encapsulated in the HMI components are the five machine mode controls; 1) Automatic Mode, 2) Manual Mode, 3) Step Mode, 4) Return to Initial Position Mode and 5) Reset Mode.

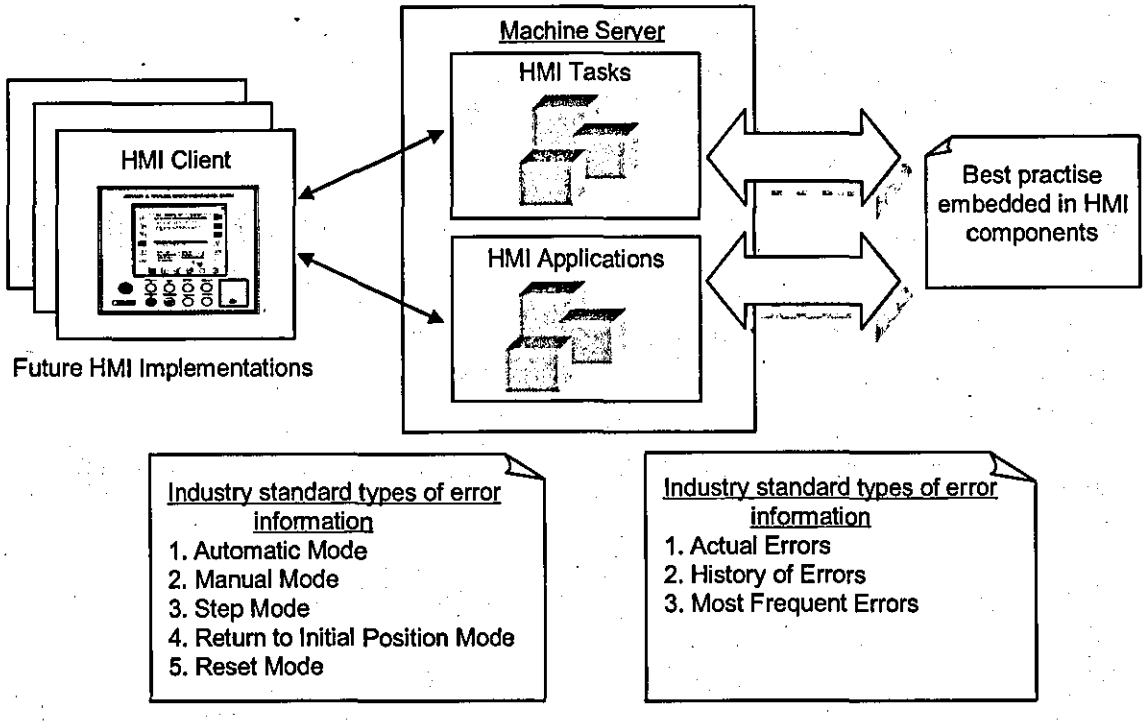


Figure 7.28 HMI system architecture supports encapsulating industry best practice

Figure 7.29 illustrates the Machine Error Information HMI Task that's consists of three instances of the same Error Information HMI widget. The machine Error Information HMI Task is requested from the machine server by the HMI client. The HMI Error Widget

displays the three different types of error information according to the HMI Widget configuration parameter setting in the Machine Error Information HMI Task which is displayed in the HMI Client. This Machine Error Information HMI Task hence has the industry best practice embedded with the error information HMI Widgets in terms of standard display of machine error information that is required by the HMI users and provides a consistent look and feel when reused and feel in different machine applications.

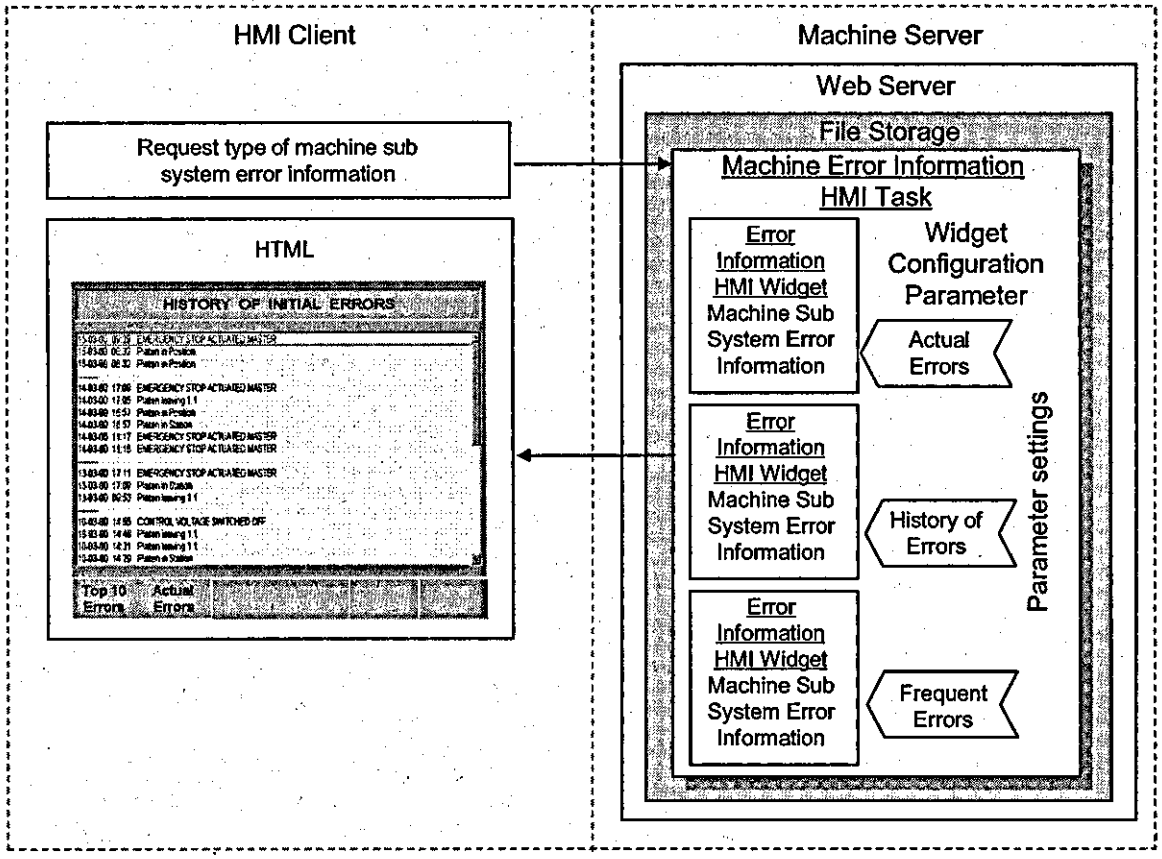


Figure 7.29 Three types of industry standard error information in sub system error machine task.

Another example implementation of the concept of a HMI Task containing HMI Widgets and their configuration parameter settings is used in the machine mode control HMI functionality. The implementation of the machine mode control HMI functionality is illustrated in figure 7.30 and is a Machine Mode Control HMI Task software component. The requirement for this HMI task is to support the industry standard control modes of the machine:

1. Automatic Mode Operation of the machine is carried out automatically according to the specified application behaviour without the need for intervention from the operator.
2. Manual Mode There is no predefined operational sequence in this mode. The machine operates upon specific commands issues by the operator.
3. Step Mode The machine operates in the control sequence as in automatic mode. However the system will pause after each control sequence and wait for the operator confirmation before proceeding to the next operation sequence.
4. Return to Initial Position This button is continually pressed and the machine components return to their initial home position. All machine components must be in their initial position before the machine can enter automatic mode.
5. Resetting the Sub System When errors have been generated in the sub system the error fault must be rectified and then the reset command clears the error in the control system.

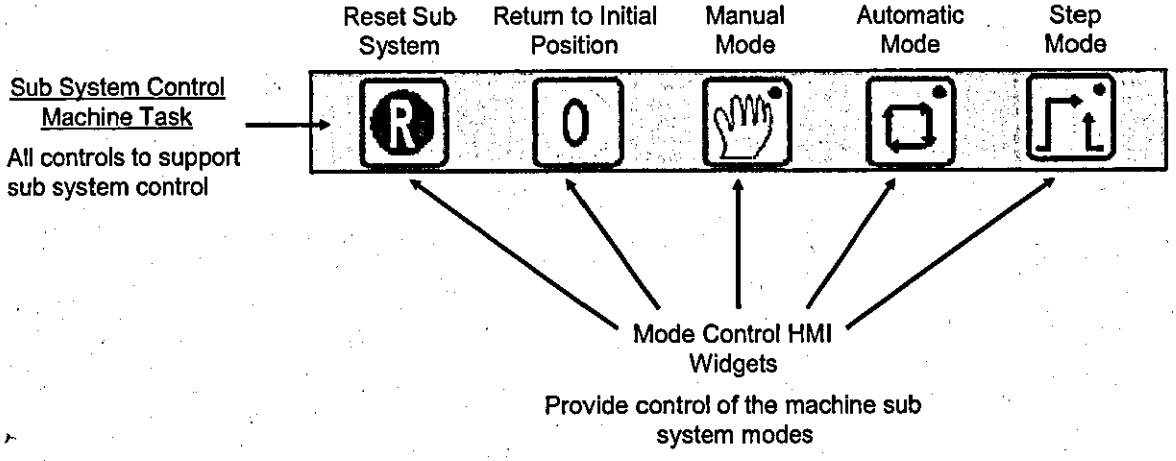


Figure 7.30 Implemented Machine Mode Control HMI Task software component

Each Mode Control HMI Widget in Machine Mode Control HMI Task is configured using HMI Widget parameters to specify the control mode allocated machine mode command. Figure 7.31 illustrates example code that is used in HMI Widgets and HMI Task components. The machine mode control HMI Widgets communicate with the machine HMI Applications using HTTP Requests. To make these HTTP requests the HMI Widget contains a HTML

form (see chapter 5 of this thesis). The HTTP form declaration contains; 1) a *NAME* which is this example is "HMIMode", 2) an *ACTION* which is the URL that the destination HMI Application for the HTTP request, 3) HTTP Form parameters which is this example is the value of the mode command. The machine mode control HMI Task accesses the HMI widget parameters to be set by referencing the parameters in the syntax detailed in figure 7.30

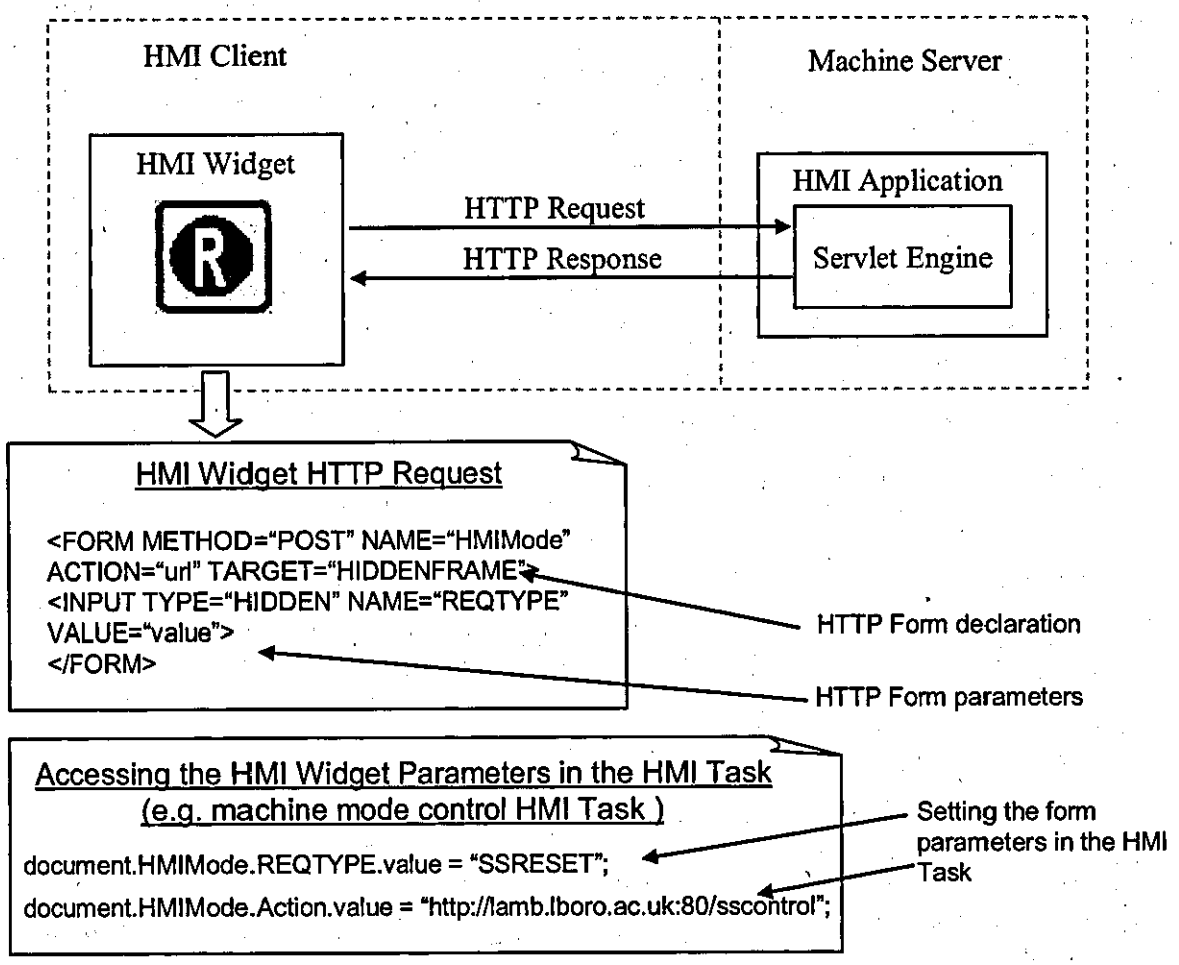


Figure 7.31 Example Implementation of a Machine Mode Control HMI Widget

The HMI Tasks discussed in this section aim to encapsulate the lower level HMI Widget functionality in higher level machine tasks that can be used in many applications enabling machine builders to concentrate better on their core competence of building machine automation system utilising pre built, integrated HMI components.

7.6 Case Study 3 – Ford Test Rig Case Study

The Ford Test Rig is a table top style test bed with the objectives to facilitate the advancement of the state-of-the-art in manufacturing machine control technology and collaborative demonstrations.

The Ford Test Rig, illustrated in figure 7.32 was developed in conjunction with Ford Motor Company and mimics general automation machines that exist within the organisation. The objective of the test rig is to investigate new approaches to automation systems in isolation and in a coordinated fashion. This investigation must be allowed to occur in a flexible environment where new ideas can be verified without the cost of lost productivity or scrap inevitable in a real production environment.

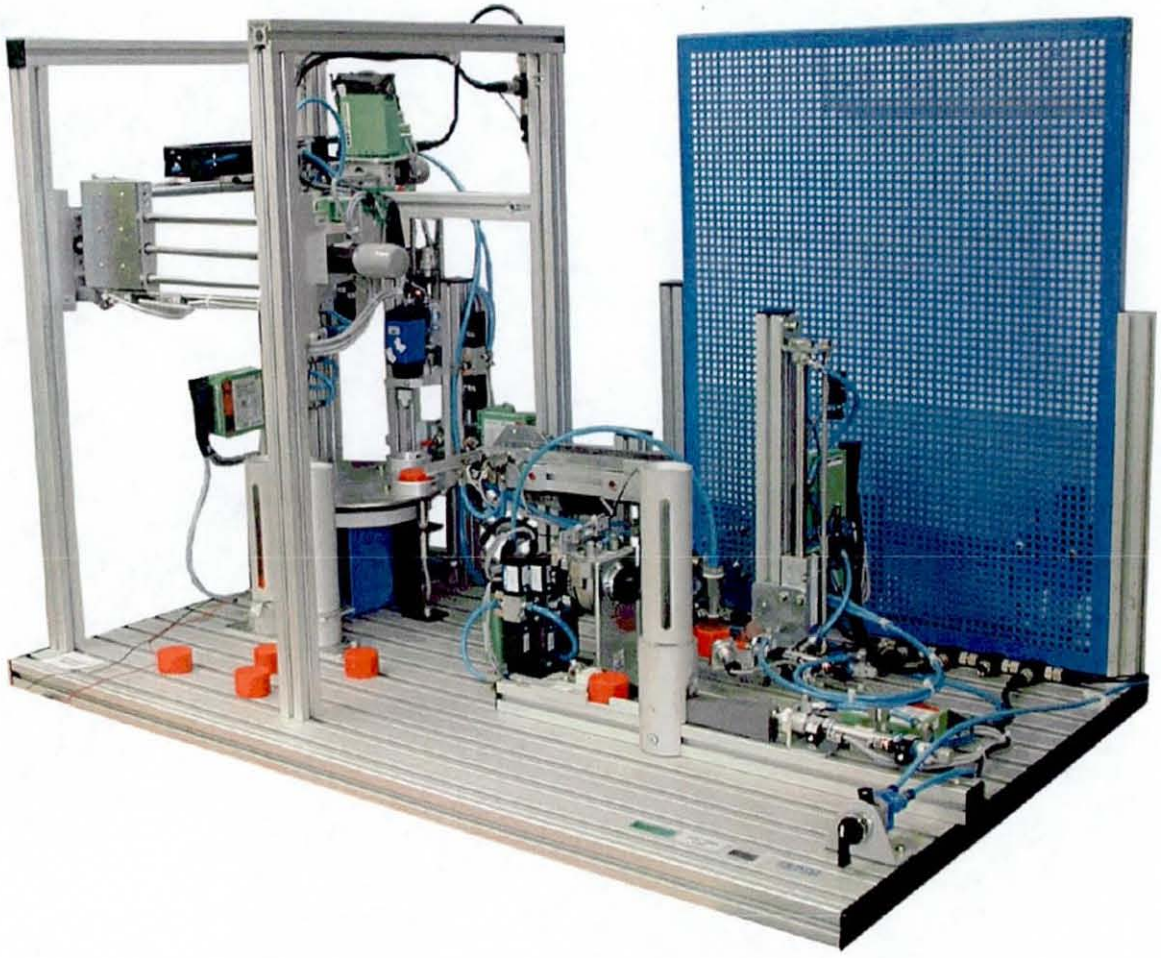
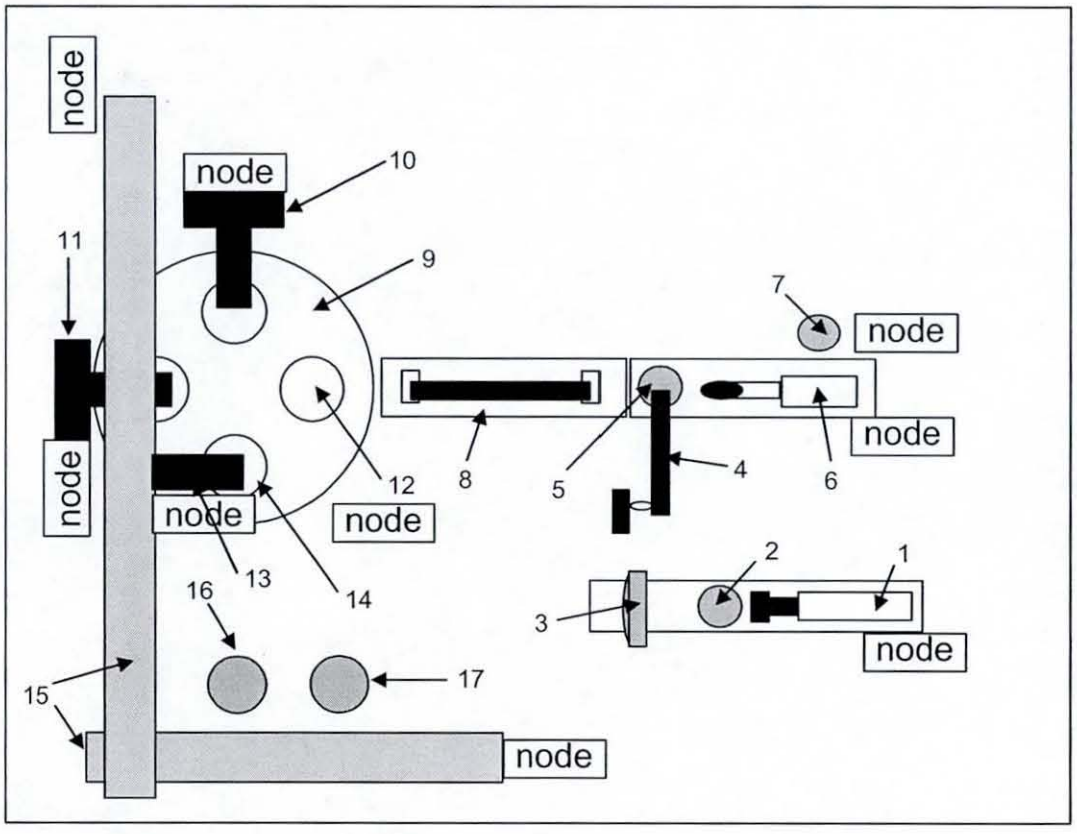


Figure 7.32 The Ford Test Rig to facilitate the advancement of state of the art in manufacturing machine control concepts

7.6.1 Ford Test Rig Details

This case study is based on the implementation of the C-B HMI concept on the Ford test rig. The Ford Test Rig consists of nine different machine components that are illustrated in figure 7.33 as nodes. Parts are loaded into the rig in the feed magazine (node 2) and the ejecting cylinder pushes (node 1) the parts to the rotary actuator (node 4). The rotary actuator lifts the part to the lifting cylinder (node 7) where they are then ejected to the conveyor (node 8). The conveyor moves the part to the rotary table (node 9) which moves the parts to different stations; the drilling station (node 10) and the probe station (node 11). The x, y gantry then picks the part and places them in either the goods part or the goods reject bin depending on whether a good / bad drilling operation is detailed at the probe station.



1 - ejecting cylinder
2 - feed magazine
3 - part present
4 - rotary actuator
5 - rotary actuator
6 - lift ejector
7 - lifting cylinder
8 - conveyor
9 - rotary table
10 - drilling machine
11 - probe station
12 - gantry cylinder
13 - gantry cylinder vacuum
14 - proximity sensor
15 - gantry
16 - goods part bin
17- goods reject bin

Figure 7.33 Schematic of the Ford Test Rig illustrating major components [85]

7.6.2 HMI Implementation

The End User HMI implementation developed for this case study is identical to one implemented for the Krause machine (see section 7.4.2). This is due to the fact that the Ford Test Rig is the same type of machine (automotive assembly manufacturing) as the Krause machine and the HMI was originally implemented to promote best industry practice through the collaboration with Ford Motor Company.

7.6.3 System Attributes Testing

There are three HMI System attributes that have been tested using this case study. The first is how early validation of the end user HMI is facilitated using a simulated machine model. The second attribute that is tested is how this HMI approach allows operators to be trained on the machines operation using a simulated machine model. The third system attribute tested is how remote diagnostics can aid engineers in diagnosing machine faults (see table 7.5).

Table 7.5 HMI benefits demonstrated in Ford Test Rig case study

Benefits Tested	Enabler	Implementation
Early HMI validation	HMI can monitor and control a virtual machine model	HMI is validated using a virtual model of the machine
Operator training		Using virtual simulation model to train end machine users
Remote Diagnostics	Web-based, distributed End User HMIs	Using a remote HMI with simulation model functionality machine faults are diagnosed

7.6.4 Simulation Model

The benefits as a result of early validation and improved operator training using the C-B HMI are mainly due to the incorporation of a simulation model within the HMI.

Using the C-B HMI it is possible to drive both the real machine and the simulation of the machine via the HMI, enabling engineers to be trained on the operation of the machine HMI prior to the real machine being completed [78].

Figure 7.34 illustrates activities involved during the ramp up of machine a new automotive production machine. The tools within the Integrated Engineering Environment support the machines mechanical design. All machine design information inputted during the activities supporting the machine over its entire lifecycle are stored in the Common Data Model that resides on the machine server. The information stored in the common data model can drive

either a Modelled or Real machine that is designed using the tools in the Integrated Engineering Environment. All machine information that the HMI system requires is collected from the common data model, therefore either the simulated machine or the real machine can be driven by the HMI. This enables the HMI system to be validated using a simulation machine model that is fully functional before the real machine is built simplifying the commissioning tests. Another benefit of the HMI system being able to drive either the modelled or the real or a combination of the two is that machine operators can be trained before the machine is built.

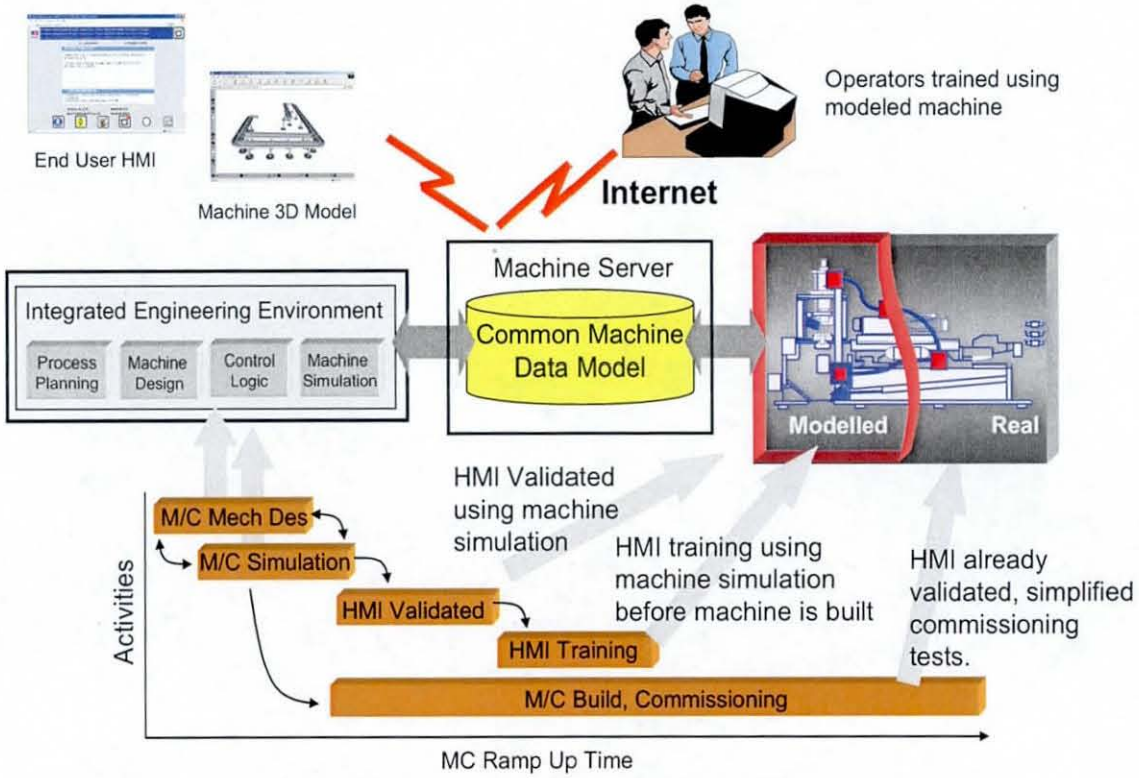


Figure 7.34 Simulation model provides benefits in HMI Validation and Operator training during the machine ramp up phase

7.6.5 Remote Diagnostics

Remote maintenance (or e-maintenance) provides the services to facilitate the monitoring and diagnostics and repair of machine equipment from remote location(s). The web based distributed End User HMIs are aimed at supporting this maintenance concept. The functionality of the HMI such as the three dimensional common model is envisaged to make major contributions to the efficiency of remote maintenance operations by enabling engineers

to visualise the exact status of the machine and not rely of mental models which is traditional practice.

To test the remote maintenance capabilities of the C-B-HMI system a number of faults scenarios for the Ford Test rig were designed.

Five different types of error was injected into the Ford Test Rig which were; 1) Air supply cut off, 2) Parts jammed on the conveyor, 3) Power supply cut off, 4) Sensor pairs check and 5) unplugging the network connection. The errors were diagnosed using three different approaches so that the performance could be benchmarked. The first approach involved only using a telephone to communicate between the machine user and the maintenance / control engineers in a remote location. The second approach involved involves using telephone and video support to solve the machine fault. The third approach involves using telephone, video and the remote HMI which includes the three dimensional machine model functionality.

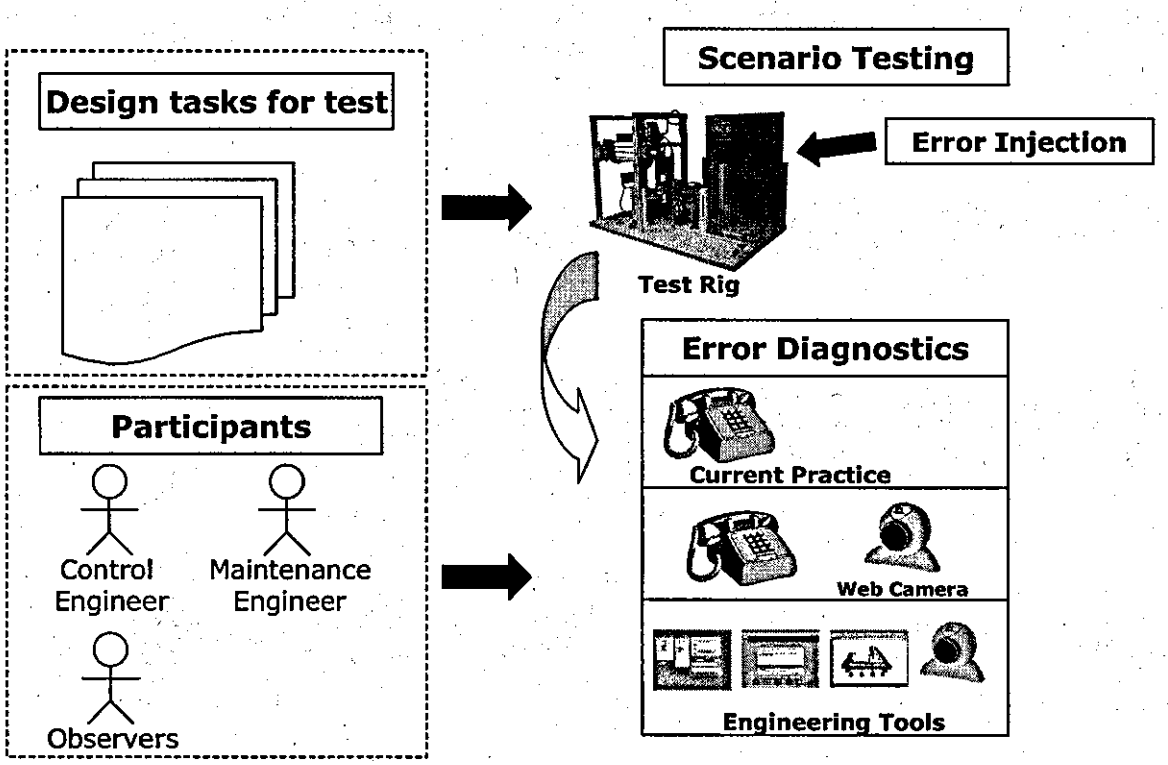


Figure 7.35 Three different approaches used to test the HMI Systems remote diagnostic capabilities [80]

The results from the fault scenarios injected into the Ford Test Rig that are tested in this case study is shown in table 7.6.

Table 7.6 Results from remote maintenance test conduct in this case study

Problem	Type of errors	Diagnostic tools	Time Taken (minutes)	Average Time Taken (minutes)
1	Abandoned			
2	Air supply cut off	Phone	7:05	
3	Power supply cut off	Phone	7:05	7:05
4	Pairs check	Phone + video	2:46	
5	Pairs check	Phone + video	4:27	
6	Power supply cut off	Phone + video	6:46	4:39
7	Air supply cut off	Remote HMI + 3D models + video	1:49	
9	Parts jammed on conveyor	Remote HMI + 3D model + video	3:21	
10	Power supply cut off	Remote HMI + 3D model + video	5:20	
11	Pairs check	3D model	3:26	
12	Network unplugged	Video + 3Dmodel	4:00	3:35

7.7 Summary

This chapter has described the actual implementation details of the concepts introduced in chapters 4, 5 and 6 of this thesis. The C-B HMI approach has been designed to meet to challenges faced by the automotive manufacturing industry. Three case studies are used to describe the C-B HMI implementation, two Automotive Assembly machines and one Automotive Transfer Line machine.

The Automotive Assembly Machine case study at J. A. Krause has been used to demonstrate rapid HMI composition. The rapid configuration capability is also illustrated through the reuse of existing machine HMI Task components. Furthermore this case study has also been used to demonstrate another core feature of the C-B HMI approach where the machine component structure is changed and the HMI is automatically updated to reflect this change.

The Automotive Transfer Line Machine case study is used to illustrate how machine HMI messages are generated from the machine common database. This ensures that the machine and its associated HMI are always consistent. Industry best practice is shown in this example by demonstrating industry standard types of error messages are embedded in sub-system components so they can be reused in many applications.

The benefit of remote monitoring and diagnostics through the use of internet technologies that the HMI system supports is illustrated using the Ford Test Rig case study. The three dimensional solid model is described and how this supports early HMI validation and operator training.

In this chapter the example case studies describing the implementation of the C-B HMI have been presented. The implementation will be qualitatively and quantitatively evaluated in chapter 8.

8 Evaluation

8.1 Introduction

The three C-B HMI case studies (detailed in chapter 7 of this thesis) explain the HMI system implementation and provide examples cases of testing the HMI system attributes. These system attributes are qualitatively and quantitatively evaluated in this chapter and the benefits associated with the C-B HMI are briefly discussed. The system has been evaluated from the perspective determining if the C-B HMI approach meets the requirements of global manufacturing. The qualitative aspect of the evaluation focuses on how well the new approach to building HMI Systems meets the needs of global manufacturing whereas the quantitative aspect evaluates the HMI systems performance in an global manufacturing environment by measuring key attributes, for example a comparison of the system build time between the current industry practice and the HMI system approach described in this thesis.

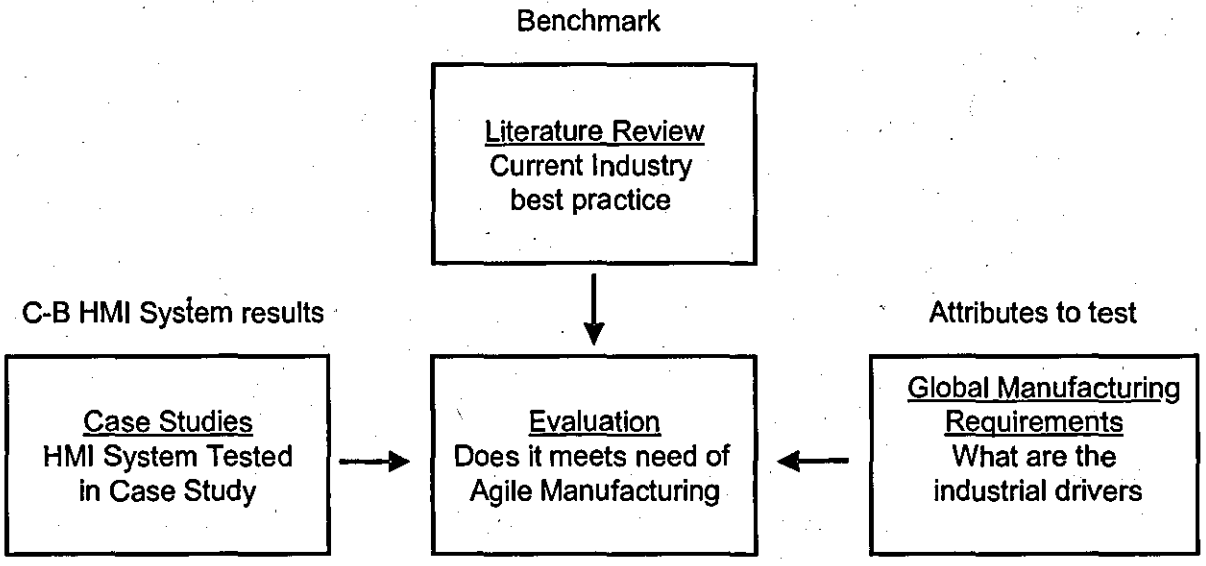


Figure 8.1 Evaluation test the results from the case studies against key demands in global manufacturing which are also benchmarked against current best practice

8.2 Fulfilling the Challenges of Global Manufacturing

This section aims to determine if the new approach to HMI systems described in this thesis fulfils the challenges of global manufacturing. Global manufacturing is demanding that enterprises to respond quickly to changing market demands and organisations must globally distribute manufacturing plants to remain competitive in the global marketplace (see section 2.2). There are three objectives of the C-B HMI system that have been evaluated to determine the effectiveness in meeting the requirements placed on the system by the pursuit of agility. Three objectives are; 1) faster machine ramp up time, 2) machine operating effectiveness and 3) increasing machine up time. Each HMI objective has a number of attributes inherent to the HMI that form the HMI system evaluation criteria. The HMI approach described in this thesis must also have the desired performance characteristics that are expected from HMI systems in terms of 1) industrial robustness, 2) fail safe operation and 3) real-time response.

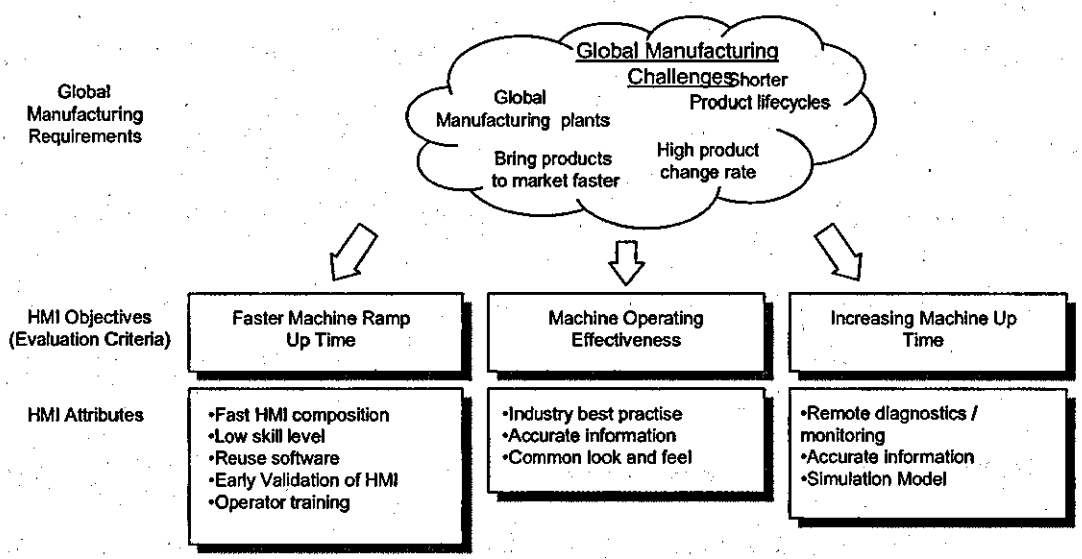


Figure 8.2 Fulfilling the requirements of Global Manufacturing

8.3 Evaluation of Machine Ramp Up Time

The first of the three HMI system objectives is the ability to provide a faster machine ramp up. Five HMI attributes have been identified that are instrumental to satisfying this objective which are; 1) Fast HMI composition, 2) Low Skill level required during composition, 3) reusable HMI software components, 4) early validation of the HMI and 5) operator training. A summary of these HMI attributes to support faster machine ramp up that is described in this section is detailed in table 8.1

Table 8.1 C-B HMI System attributes that support faster machine ramp up objective

HMI Attributes	Implementation	Result
HMI Development Time	The HMI is modeled using a storyboarding technique and directly input into a HMI configuration tool within the Integrated Engineering Environment that generates the executable HMI system. HMI's can be selected and deployed using a tool in an Integrated Engineering Environment that supports the machine throughout its entire lifecycle.	In the Krause case study the End User HMI was composed in less than 5 minutes. End User HMI's can be selected and deployed to a new machine application in 2 minutes. The only unique software developed for each machine application was the simulation model [87] and the diagnostics (approx 120 mins for 10 machine components)
Low skill level to build machine HMI systems	Storyboarding technique specifies the HMI at high level and no specialist programming knowledge is required. The component based structure of the HMI System allows the lower level higher skilled development to be encapsulated and reused via User Task Components.	Krause case study demonstrates how process engineers can potentially compose fully functional end user HMI's. Lamb Technicon case study proves industry standard error information and sub system mode control is encapsulated in system.
Reusable HMI software components	Clean component based design. High degree of reuse in software architecture at 3 levels; 1) the End User HMI, 2) HMI Task Components, 3) HMI Widgets.	The reusability of the HMI components developed in the case studies is evaluated. End User HMI have a 50% reusability index (see table 8.4). HMI Tasks have a reusability index of 58.6% (see table 8.4) HMI Widgets have a reusability index of 66.6% (see table 8.4)
Early validation of HMI	HMI can be tested using a virtual, real or composite machine. The machine simulation is developed earlier in the machine ramp up phase.	Ford Test Rig case study demonstrates the HMI system driving a virtual machine to support validation of HMI capabilities.

Table 8.1 continued

Operator training	Using virtual model remote to the machine build location	Ford Test Rig case study demonstrates the HMI system driving a virtual machine from a remote location to support operator training.

8.3.1 HMI Development Time

Reducing the machines associated HMI system development cycle time is a key attribute in achieving fast production ramp up [72]. Current industrial practice involves the requirements and specification stage to be completed by the process engineer. This is then passed to the HMI programmer who implements the HMI System software. In the C-B HMI approach, the End User HMI is modelled using a storyboarding technique (see chapter 4 of this thesis) and directly input into the End User HMI Composition Tool by the process engineer who then generates the executable End User HMI software (see section 7.4.4 of this thesis). The elimination of the HMI programmer can significantly reduce the development time of the HMI but this can only be realized if the end user client software components (HMI Tasks and HMI Widgets) have been developed (see section 7.3.2).

During the Krause case study the time taken to compose an End User HMI was measured as less than 5 minutes (see section 7.4.4). Once the End User HMI is composed it is stored in system libraries and it can be can be selected and deployed to new application in less than 2 minutes using the End User HMI Configuration Tools within the integrated engineering environment.

Due to the less structured, iterative and hence more complex process in developing a machine HMI system using the current industry technologies that involve indirect development activities (specification documenting, bespoke HMI system and machine PLC device interfacing) it is more challenging to determine an exact time scale for HMI System development. Nevertheless by combining the requirements specification and software implementation (illustrated in figure 8.3) the development cycle using the storyboarding requirements specification technique that is directly executable it is suggested that the overall

development process time would be reduced. The HMI development time would be shortened by the time taken for the HMI software to be implemented that is eliminated from the HMI development process.

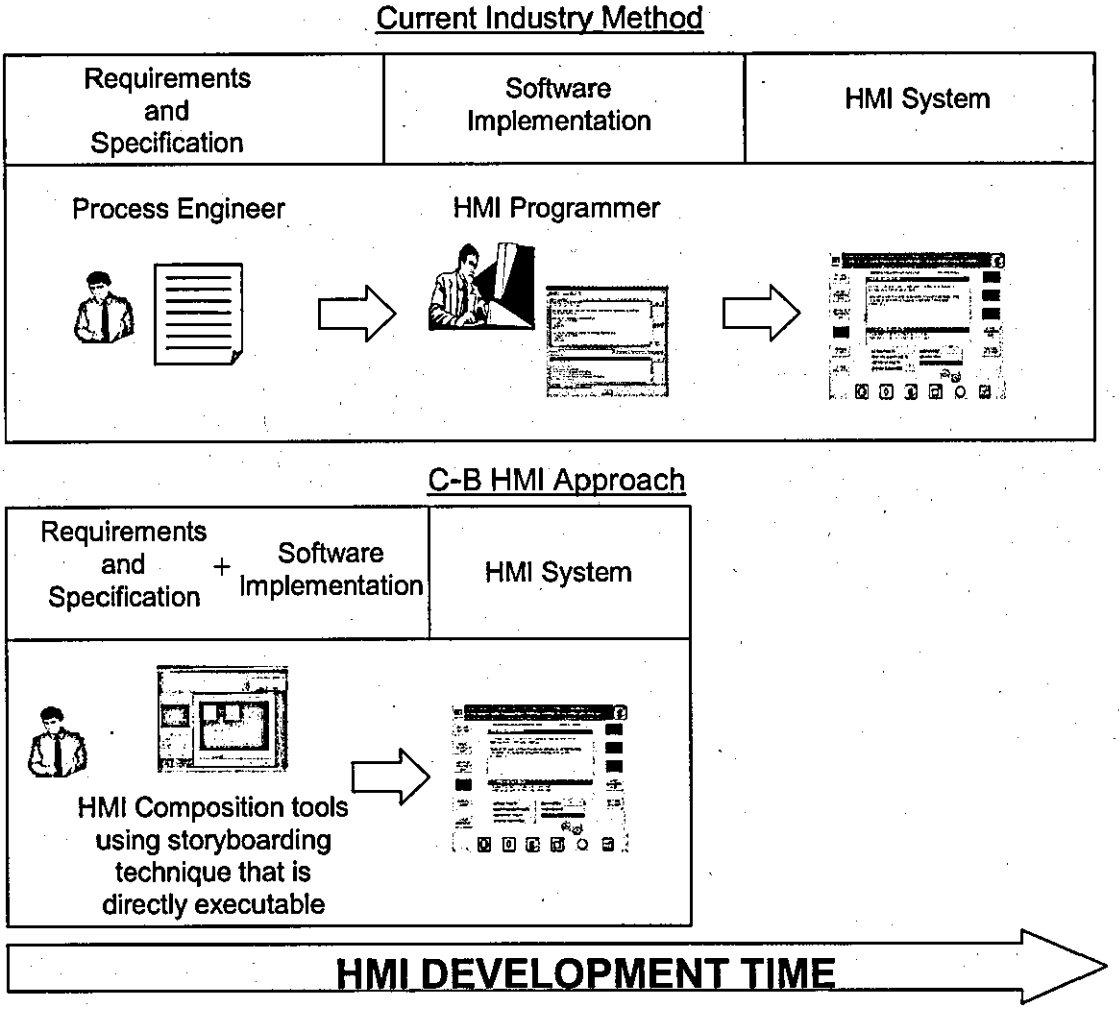


Figure 8.3 Elimination of the software implementation in the HMI Development process is suggested to shorten develop time

8.3.2 Low skill level to build machine HMI systems

The most technically demanding aspect in building a new machine HMI system is the software implementation process that is performed by specialist HMI programmers [90]. The requirements and speciation of the machine HMI system is completed by process engineers who have knowledge about; 1) the operation and functionality of the HMI and 2) the users and what tasks they must perform on the machine. However process engineers do not have the specialist knowledge to convert these requirements into fully functional HMI software applications.

The aim of the storyboarding technique used in the C-B HMI approach to describe the End Users HMI functionality eliminates the highly skilled aspect of the HMI development process by enabling complete End User HMI's to be directly generated from specifying the end user HMI functionality at a level that requires no specialist skills (see section 7.4.4). The HMI composition tool within the integrated engineering environment (see section 6.5) supports the input of the HMI requirements using this storyboarding technique by the process engineer in this case and automatically generates the HMI software. The process engineer requires no specialist HMI programming knowledge. The C-B structure of the HMI System partitions the system so the higher skilled development activities form End User software components (HMI Task Components and HMI Widgets) which the HMI programmers must develop and are subsequently reused to compose complete End User HMI's.

The Krause case study demonstrates how process engineers can potentially compose fully functional end user HMI's with no specialist skills required.

The Lamb Technicon case study proves how industry standard error information and sub system mode control is encapsulated in system by the HMI programmer.

8.3.3 Reusable HMI software components

The systematic reuse of previously implemented software components aims to achieve increased software development productivity as well as increased quality of the software [58, 59 and 60]. If previously tested components are reused in a new HMI software project's they are likely to contain fewer errors than if new components were developed for each new application. This has been reported to reduce the overall failure rate of the software project. Hence, high quality software can be built by incorporating reused components into the software project. Reuse offers the potential for lower cost of development through shorter development timescales and producing better quality software, however, reuse can also be a significant constraint actually slowing it down and making the overall design of the system unwieldy, and unstable. It can actually increase the long-term cost of the system [94].

The reuse rate is a factor that can be used to enable the assessment of the value of the reuse method in the systems approach. It is essential to assess the reuse rate to monitor the success of the software systems reuse methodology. According to Poulin [62], the de-facto standard for measuring the reuse level is:

$$\frac{Reused\ Software}{Total\ Software} \times 100\ \% \qquad \text{——①}$$

The software designed in the HMI approach described in this thesis has a structured layered architecture which promotes reuse by encapsulating functionality of the End User HMI components at three system levels; 1) End User HMI, 2) HMI Task Components, 3) and HMI Widgets. The remainder of this section evaluates the reuse of the End User HMI components implemented to support the three case studies described in chapter 7.

End User HMI Clients Re-use

Table 8.2 details the End User HMI's in each of the three case studies described in chapter 7 of this thesis to support the requirements of the three different (see section 7.3.1) end users identified; 1) Machine Operator, 2) Machine Engineer and 3) Machine Diagnostic Engineer. Six End User HMI's were developed to support the nine End User HMI's required (three per case study) which provides a reusability index of 50% (see table 8.3).

Table 8.2 End User HMI's in each of the three case studies described in chapter 7

Case Study (see chapter 7)	Implemented End User HMI	End User HMI Type and Storyboard Model
Krause	1) Machine Operator HMI Client 2) Machine Engineer HMI Client 3) Machine Diagnostic Engineer HMI Client	End User HMI for the automotive assembly machine (7.4.2)
Ford Test Rig	4) Machine Operator HMI Client 5) Machine Engineer HMI Client 6) Machine Diagnostic Engineer HMI Client	End User HMI for the automotive assembly machine (7.4.2)
Lamb Technicon	7) Machine Operator HMI Client 8) Machine Engineer HMI Client 9) Machine Diagnostic Engineer HMI Client	End User HMI for the automotive transfer line machine (7.5.2)

The dependant variables for the reuse of the End User HMI are; 1) the end user type and 2) the machine type.

Two different types of machine were implemented in the three case studies. The Lamb Technicon case study was based on an automotive transfer line machine and the Ford Test Rig and J. A. Krause case studies were based on an automotive assembly machine. The three assembly machine end user HMI's (Machine Operator HMI, Machine Engineer HMI and Machine Diagnostic Engineer HMI) implemented for the assembly type machine could be directly reused in the Ford Test Rig and J. A. Krasue case studies due to the machine being the same type. The HMI implemented for the automotive transfer line machine could only be used for the Lamb Technicon case study. The reusability index (using Poulin method [62]) is 50% (3 End User HMI reused / 6 End User HMI developed) for end user HMI's in these three case studies.

It could be suggested that this reusability index would be increased to 100% if this approach was adopted in industry due to a machine builder developing many individual machines applications of the same type for the same type of end users.

Table 8.3 End User HMI Software reused in chapter 7 case studies

System Architectural Layer	Dependant variables for reuse	Case Studies	Reusability = (no. reused / no. developed) %
End User HMI	Machine Type	3 user types 2 machine types	(3 reused / 6 developed) 50%

HMI Task Components Re-use

HMI Tasks software components were implemented and used to compose the End User HMI's clients.

The six different types of user functionality required (1. Diagnostics, 2. Simulation Model, 3. Control & Monitoring, 4. Real-time Info, 5. Mode Control and 6. Error Information) in the HMI's for the case studies are detailed together with their corresponding implemented HMI Task component in table 8.4.

The Machine Error Information HMI Task (supporting the error information user functionality), Mode Control HMI Task (supporting the mode control user functionality) and the Real-time Information HMI Task (providing Real-Time machine information user

functionality) were all reused in the Ford Test Rig, J. A. Krause and Lamb Technicon case studies and therefore each achieved 100% reuse.

The control and monitoring user functionality had different requirements (see section 7.4.2 for the assembly machine and 7.5.2 for the transfer line machine) which are fundamental to the operation of the type of machine. The sequencing nature of the transfer line machine required the component states that the machine normally executes to always be displayed down each side of the HMI screen, the *HOME* state of the machine component on the left hand side and the *WORK* state on the right side of the screen. In contrast the assembly machine type required the machine components state to be displayed in a hierarchical structure of the machine components and then machine elements for the selected component. Two different HMI Tasks (Assembly Machine Monitor and Control HMI Task and a Transfer Line Machine Monitor and Control HMI Task) were implemented to support the two different machine types (see table 8.4c) which results in a 50% reusability index (one HMI Task reused divide by two HMI Tasks developed).

The Solid Model Machine Simulation functionality (table 8.4b) implemented in each case study is unique to each machine's physical configuration and therefore Solid Model HMI Task must be developed for each machine [63]. The reusability index is 0% for this software component. However a machine solid model is developed from re-usable modules [87] but in this evaluation from the HMI perspective only the Solid Model HMI Task is considered – how the machine solid model is developed is outside the scope of this research.

The machines diagnostic user functionality (table 8.4c) is unique for each machine implemented in these case studies and each machine consists of a Diagnostic HMI Task unique for the machines application. The diagnostic HMI Task components implemented in the three case studies therefore have a reusability index is 0%.

The mean average reuse for the HMI Task components required in the three case studies is 58.6%

Table 8.4 HMI Tasks used in each case study (Chapter 7 of this thesis)

User Functionality	All Implemented HMI Task Components	End User HMI's implemented in Case Studies: Ford Test Rig HMI – Machine Operator HMI Client	Ford Test Rig HMI – Machine Engineer HMI Client	Ford Test Rig HMI – Machine Diagnostic Engineer HMI Client	J. A. Krause HMI – Machine Operator HMI Client	J. A. Krause HMI – Machine Engineer HMI Client	J. A. Krause HMI – Machine Diagnostic Engineer HMI Client	Lamb Technicon HMI – Machine Operator HMI Client	Lamb Technicon HMI – Machine Engineer HMI Client	Lamb Technicon HMI – Machine Diagnostic Engineer HMI Client	Reusability % no. reused / no. developed
f. Error Information	Machine Error Information HMI Task	✓	✓	✓	✓	✓	✓	✓	✓	✓	(1 reused / 1 developed) **100%**
e. Mode Control	Mode Control HMI Task		✓			✓			✓		(1 reused / 1 developed) **100%**
d. Real-Time Info	Real-time Information HMI Task	✓	✓	✓	✓	✓	✓	✓	✓	✓	(1 reused / 1 developed) **100%**
c. Control & Monitoring	Assembly Machine Monitor and Control HMI Task	✓	✓	✓	✓	✓	✓				(1 reused / 2 developed) **50%**
	Transfer Line Machine Monitor and Control HMI Task							✓	✓	✓	
b. Simulation Model	Ford Test Rig Simulation HMI Task	✓	✓	✓							(0 reused / 3 developed) **0%**
	Krause Machine Simulation HMI Task				✓	✓	✓				
	Lamb Machine Simulation HMI Task							✓	✓	✓	
a. Diagnostics	Krause Machine Diagnostics HMI Task					✓	✓				(0 reused / 3 developed) **0%**
	Lamb Machine Diagnostics HMI Task								✓	✓	
	Ford Test Rig Diagnostics HMI Task		✓	✓					✓		
										Mean Average Reuse	**58.6%**

HMI Widgets Re-use

The lowest level HMI End User elements in the HMI systems structure is the HMI Widgets. One or more HMI widgets are combined in a HMI Task component. The concept of the HMI Widgets is to support the direct interaction with a functional aspect of the machine. The functional aspects of the machine required to support end user HMI's have been identified in the case studies (see section 7.3.1) and are; 1) Machine Error Information, 2) Real-time Machine Information, 3) Machine component control and monitoring, 4) Machine Mode control, 5) Machine solid model and 6) machine diagnostics and their reusability is detailed in table 8.5.

The Error information HMI widget implemented supports the three standard types of machine error information (see section 7.5.5). This HMI widget is used to develop the Machine Error Information HMI Task component which is reused in all three case studies. Only one HMI widget is developed to support the Error information functionality which is reused in all case study implementations therefore the reusability index is 100% (one reused component divided by one developed component).

The Real-time HMI Widget provides real-time machine information to the End User HMI. This HMI widget is implemented as a Java Applet (see Figure 7.26) which is generic and reused in all three case studies. The reusability of this HMI widget is 100% (one reused component divided by one developed component).

Mode Control HMI widget used to provides the functionality to control the mode of the machine. Automotive machine in the case studies have five different modes of operation; 1) Initial Position Mode, 2) Step Mode, 3) Reset Mode, 4) Manual Mode and 5) Auto Mode which are all supported by configuring the mode control HMI widget. The reusability of this HMI widget is 100% (one reused component divided by one developed component).

The C-B Paradigm for Automation Systems [88] defines each individual machine components sequencing and interlocking logic using state transition diagrams. This consistent representation of all machine components allows a single HMI Widget to be adopted that

controls and monitors the machine components states. Application of the C-B paradigm to the three case studies allows the Component Monitoring and Control HMI widget to be reused across all machine components in all case studies. The reusability of this HMI widget is 100% (one reused component divided by one developed component).

The machine Solid Model HMI Widget is unique for each machine's configuration and a new HMI widget is developed for each case study. The reuse is 0% due to their being no reuse of this HMI widget between different machine applications in the three case studies.

The mean average reuse for the HMI Widgets implemented to support the three case studies is 66.6%

Table 8.5 HMI Widgets used in each case study (Chapter 7 of this thesis)

Functionality	HMI Widgets	Case Studies: Ford Test Rig HMI	Case Studies: J. A. Krause HMI	Case Studies: Lamb Technicon HMI	Remarks	Reusability
f. Error Info	Error Information HMI Widget	✓ (3)	✓ (3)	✓ (3)	Widget used to support 3 standards types of error information; 1) Actual, 2) History and 3) most frequent in all case studies in all three case studies (see section 7).	(1 reused / 1 developed) **100%**
e. Real-time Info	Real-time Information HMI Widget	✓ (1)	✓ (1)	✓ (1)	Real-time Information HMI Widget is reused in all three case studies.	(1 reused / 1 developed) **100%**
d. Machine Mode	Mode Control HMI Widget	✓ (5)	✓ (5)	✓ (5)	Mode Control widget used to support the five different operating modes for each case study.	(1 reused / 1 developed) **100%**
c. Control & Monitor	Component Monitoring and Control HMI Widget	✓ (17)	✓ (20)	✓ (10)	Widget is used to provide control and monitoring for all machine components used in all case studies.	(1 reused / 1 developed) **100%**
b. Solid Model	Ford Test Rig Solid Model HMI Widget	✓ (1)			Solid Model HMI widget is unique for each machine and is developed for each case study.	(0 reused / 3 developed) **0%**
	Krause Solid Model HMI Widget		✓ (1)			
	Lamb Solid Model HMI Widget			✓ (1)		
a. Diagnostics	Ford Test Rig Components Diagnostic HMI Widget	✓ (17)			The diagnostic widgets (electrical schematics, mechanical schematics) are unique to each machine and do not support any reuse.	(0 reused / 3 developed) **0%**
	Krause Components Diagnostic HMI Widget		✓ (20)			
	Lamb Component Diagnostics HMI Widget			✓ (10)		
					Mean Average Reuse	**66.66%**

8.3.4 Early validation of HMI

Early validation of the HMI system is a key attribute to achieving a faster machine ramp-up time. The current industry approach to building HMI systems does not allow the HMI system to be tested and functionality validated until the real machine has been physically built and the machine control software developed.

The HMI System approach described in this thesis facilitates early validation which is inherent to the HMI architecture adopted. The HMI system transparently interfaces (see section 5.7.7 that describes the machine interface) to either the real, simulated or hybrid (combination of real and simulated machine) machine. The process of constructing a new machine or re-configuration of an existing machine involves [88] simulating the machine's sequencing and interlocking logic behaviour using a three dimensional solid model. This three dimensional model is developed before the real machine which enables the HMI system to be tested before the real machine is built leading to the machine users being trained earlier in the machine ramp up phase contributing bringing a machine on line quicker.

The Ford Test Rig case study in chapter 7 of this thesis is used to demonstrate the End User HMI driving a simulated machine (see section 7.6.4) which can be used to support the early validation of HMI. The validation of the HMI involves performing a number of commissioning tests that ensure machine real-time component states and error diagnostic messages are displayed appropriately and clearly for the operator. The simulation model is used earlier in the ramp up phase to ensure that the HMI is exercising the intended machine components and all machine mode controls commands and error information is accurate.

8.3.5 Operator training

Training the operators on the operation of the machine HMI before the machine is fully installed and commissioned in the manufacturing plant is a key benefit that this HMI approach described in this thesis supports.

Using the current industry approach the operators are trained on the manufacturing machines on site in the final stages of the bringing the machine online. New machines are more likely to have process faults in the early stages and operators will be least trained for these

scenarios. Often operators are trained on similar machines enabling them to be trained earlier in the machine ramp up and then transferred to the new machine when it is fully commissioned. Problems with this approach are that the end users are not trained on the machine they are operating and although they may be the same machine type the are trained on (two machine types in the case studies, an assembly machine or a transfer line machine) their are differences between the machine they are trained on the machine they operate. For example the sequencing and interlocking logic behaviour and cycling times are unique to each machine application and the adoption of this HMI approach allows the users to be trained on the actual machine and understand the recovery form errors using the HMI diagnostics functionality.

Driving both the real machine and the machine simulation (demonstrated in the Ford Test Rig case study) via the HMI, enables the end users to be trained on the operation of the HMI prior to the real machine being commissioned and installed.

8.4 Machine Operating Effectiveness

Three HMI attributes have been identified that are key to supporting machine effectiveness; 1) Implementing industry best practice, in the HMI system 2) Accurate information and 3) common look and feel.

Table 8.6 C-B HMI System attributes that support faster machine ramp up objective

HMI Attribute	System Implementation	Result
Industry best practice	End User HMI C-B structure consists of three levels that encapsulate best industry practice: 1. HMI Widgets, 2. HMI Task Components and 3. End User HMI Clients	HMI Widgets encapsulates proven communication mechanism between the client and the machine server support interaction with the machine components that. The industry best practice is encapsulated in HMI Task components that consist of one or more HMI screens that are laid out with the correct types of machine interaction (HMI widgets) and navigation between them. Industry knowledge and experience of the HMI functionality required on each screen the navigation between the screens is encapsulated at this level in the HMI architectural structure.
HMI Accurate information	A novel feature of the C-B HMI System architecture is all machine information is stored in a common machine data repository ensuring there is no inconsistency in machine information being fragmented across different HMI's and machine control systems.	Scenario in case study proves accurate HMI machine error information by changing a machine sensor using the integrated engineering tools and this updated machine error information is propagated to the HMI System via the common machine data repository.
HMI common look and feel	Reuse of common HMI software components in End User HMI's provides a common look and feel.	Out of the 6 HMI Task components used in an End User HMI 4 are directly reused (see table 8.4). Through this reuse the HMI adopts a common look and feel.

8.4.1 Industry best practice

The C-B HMI system architecture allows the encapsulation of best industry practice in building high quality HMI Systems with encapsulated industry experience and knowledge. The structure of the End User Client and how it supports industry best practice is;

1) HMI Widgets support interaction with the machine components that encapsulates proven communication mechanism between the client and the machine server (see Figure 7.30 in section 7.5.5). This communication mechanism is encapsulated in the HMI widget and is pre-tested when integrated into HMI Task components.

2) HMI Task components provide a particular functional aspect of the HMI typically to support the user in performing a machine task (for example machine diagnostics, monitoring and control, real-time information, error information). The industry best practice is encapsulated in HMI Task components that consist of one or more HMI screens that are laid out with the correct types of machine interaction (HMI widgets) and navigation between them.

3) End User HMI Clients combine the different aspects of the HMI functionality using HMI Task components (see section 7.3.2) to provide a complete end user HMI. Industry knowledge and experience of the HMI functionality required on each screen the navigation between the screens is encapsulated at this level in the HMI architectural structure.

The standard practices of the HMI that are implemented across machines using this approach allow HMI users to be familiar with HMI systems on different machines. The HMI system component based architecture supports encapsulation of best practice within a component.

8.4.2 Accurate information

It is essential that the machine information used in the HMI system is accurate. A major issue with current industry HMI systems is that machine information (for example error messages and machine diagnostics (see section 2.3.3) that is required in the HMI system are fragmented across different systems, for example typically a HMI system database contains the machine error messages and are referenced by a machine PLC device. As changes are made to the manufacturing machine configuration which involves changing the machine PLC device

program the HMI system database must be simultaneously updated top ensure consistency accurate error information is displayed. If all the systems (machine HMI's and PLC devices) are not updated consistently the information displayed in the HMI is not accurate A novel feature of the C-B HMI System architecture is all machine information is stored in a common machine data repository (see common machine data repository implementation section 7.3.4) which ensures consistent accurate machine data is propagated to engineering tools in the machines lifecycle including the HMI. The Lamb Technicon (see section 7.5.4) case study details a scenario where changes are made to an error message associated with a machine sensor using the integrated engineering tools and this updated machine error information is propagated to the HMI System via the common machine data repository.

The benefits of having accurate machine information in the HMI system are to support more effective use of the manufacturing machine by the users acting on the correct machine information.

8.4.3 Common look and feel

A common look and feel is one of the key factors that will increase the usability of manufacturing systems and reduce operating costs of end user operations [98]. The "look" of an HMI is defined as the style of presenting information to a user. The "feel" is defined as how the HMI responds to an action taken by the user [42].

The HMI approach described in this thesis supports reusing common system components at; 1. Widget level (providing interaction with the machine), 2. Task level (supporting a particular aspect of HMI functionality) and 3. End User HMI level (complete fully functional end user HMI defined using an End User HMI Storyboarding model). The objective of reusing common HMI components between different machine applications is a common look and feel is inherent to systems built from reusing the same components.

Table 8.7 details the four core aspects of HMI functionality required in the HMI system which are; 1) Error Information, 2) Machine Mode Control, 3) Standard HMI controls to exercise and monitor machine components and 4) Standard real-time machine information.

Within the automotive industry a common look is required for each of these aspects of HMI functionality (detailed in table 8.7) which is encapsulated in the associated HMI Task component.

Table 8.7 HMI Task components provide common look

Aspect of HMI functionality where a common look and feel is encapsulated.		Associated HMI Task Component	Common look required in automotive industry
1.	Error Information	Machine Error Information HMI Task Component	Three types of industry standard error information are encapsulated in the End User HMI components; 1) Actual Errors, 2) History of Errors and 3) Most Frequent Errors (see section 7.5.5)
2.	Machine Mode Control	Machine Mode HMI Task Component	Five types of industry standard machine mode control are encapsulated in the End User HMI components; 1) Automatic Mode, 2) Manual Mode, 3) Step Mode, 4) Return to Initial Position and 5) Resetting the Sub System (see section 7.5.5)
3.	HMI controls to exercise and monitor machine components.	Machine Control and Monitoring HMI Task Component	Machine Component Control and Monitoring HMI Task encapsulates best practice for exercising and monitoring machine components.
4.	Real-time machine information	Machine Real-time HMI Task Component	Machine real-time information is displayed in the end user HMI in an industry standard way.

How the different aspects of HMI functionality (supported using HMI Task components) are laid out on the end user HMI screen layout is critical to achieving a common look and feel [22]. The screen lay out is defined in the End User HMI Storyboarding model which is directly executed using the HMI composition tool within the integrated engineering environment. Reusing the End User HMI in different machine applications ensures a common look to the screen layout.

8.5 Increasing machine up time

Increasing the up time of a manufacturing machine is a major industry driver for a HMI system [19]. The HMI system must provide the correct type of machine information and or machine control to the targeted users in their desired locations. Too often, a great deal of downtime and resources are consumed on simple operational problems for example work pieces becoming jammed in the machine, air or other miscellaneous machine supplies not being switched on [65]. Three attributes (detailed in table 8.8) of the C-B HMI System described in this thesis have been identified that are key to increasing the machine up time; 1) Remote diagnostics and monitoring, 2) Accurate machine information and 3) a local or remote simulation of the real machine.

Table 8.8 C-B HMI System attributes that support increasing machine up time

HMI Attribute	Implementation	Result
Remote diagnostics / monitoring	Web-based, distributed End User HMIs that utilise internet connected PC using standard web browsers are implemented to meet the demands of remote diagnostics and monitoring.	Errors injected into Ford Test Rig machine and diagnosed remotely using; 1) telephone, 2) telephone + live video and 3) telephone + live video + HMI, 3D model. Average time taken to diagnose a machine faults where 1) telephone – 7:05 min, 2) telephone + live video – 4:39 mins and 3) telephone + live video + HMI, 3D model 3:35 mins.
Accurate information	Novel feature of the HMI System in this is the machine integration platform where all machine information is stored in a common unified machine data model which ensures consistent accurate machine data to all machine lifecycle engineering tools including the HMI.	Ford Test Rig case study demonstrates accurate information throughout the remote monitoring and diagnostic testing.
Simulation Model	Simulated machine model functionality within HMI system.	Used to aid remote diagnostics. Tested in the remote monitoring and diagnostics HMI attribute.

Remote diagnostic and monitoring provides the services to support the maintenance, and repair of equipment from a remote location via information and communication technologies which enables assessment and or control of the equipments performance. The remote diagnostic capabilities of the C-B HMI system are demonstrated in the Ford Test Rig case study (see section 7.6.5). The HMI system is only one tool in the process of remotely diagnosing machine faults. It is envisaged that the remote capabilities of the HMI will compliment current industry practice which involves a telephone conversation and a live video link to the machine between remote engineer and the machine operator.

To demonstrate the remote diagnostic capabilities of the HMI system five different types of faults were injected into the Ford Test Rig. Three different approaches were used by the remote engineer to diagnose the machine fault using a different combination of the diagnostic tools; 1) a machine operator describing the fault to a remote engineer using only a telephone, 2) a video of the machine together with the telephone and 3) a video of the machine together with the telephone and the remote HMI.

The results from the remote diagnostic tests are tabulated in the case study (see table 7.6) and figure 8.4 illustrates the average time taken to diagnose faults for the three different diagnostic scenarios.

It used observed during these tests that were conducted that only using the telephone provided limited diagnostics and took the longest length of time to successfully diagnose the machine fault (425 seconds, see figure 8.4).

Having a live video of the machine available to the remotely located diagnostic engineer enables the engineer to see the machine status rather than continuously asking the operator for information relating to the machine status which speeds up the diagnosis time to and average of 279 seconds (see figure 8.4).

Using the remote machine HMI system allows the engineer to have more information about the status of the machine. The HMI displays the current state of all the machine components and the interlocking information so the remote engineer can determine what the machine

conditions required for the machine to continue normal operation. Addition to this the engineer can view all industry standard error information (actual, most frequent and history of errors) relating to the machine and the three dimensional machine model allows the engineer to visualise the machine state. It is these aspects of the HMI that assist the engineer bring the machine back online in a faster time-frame (215 seconds using the phone, video and HMI) and hence increasing the machines uptime.

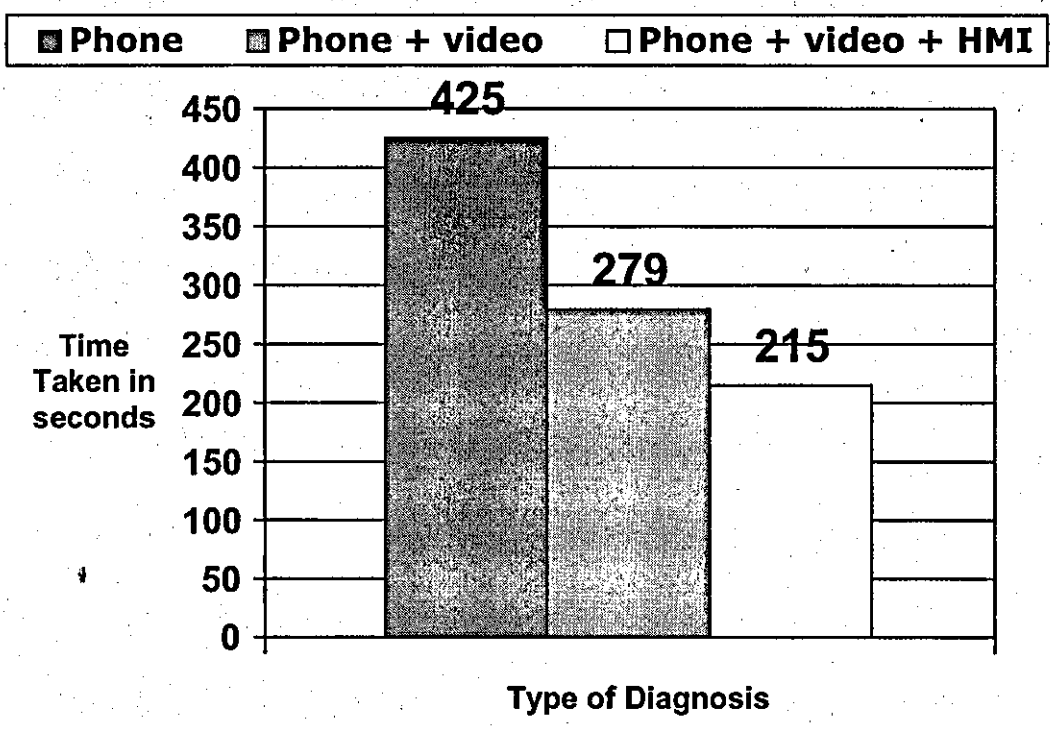


Figure 8.4 Average times taken in seconds to diagnose five different fault scenarios using three different diagnostic approaches [53]

8.6 HMI Performance

The HMI approach described in this thesis must also have the desired performance characteristics that are expected from industrial HMI systems in terms of; 1) industrial robustness, 2) fail safe operation and 3) real-time response.

8.6.1 Industrial robustness

The industrial robustness of the HMI system approach described in this thesis has been tested through the application to industrial collaborators real demonstrator machines. The machine's HMI systems are subjected to each respective company's machine commissioning checks that test the machines operational function.

In order to fully test and evaluate the C-B HMI approach from a perspective of industrial robustness the systems implemented in the case studies (see chapter 7 of this thesis) have been subject to standard commissioning tests from the respective companies commissioning engineers. The objective of these tests is to check to robustness of the C-B HMI system in a real industrial environment subject to typical disturbances. The full commissioning tests that production automation machinery is subject to by machine builder commissioning engineers are listed in table 8.9.

The commissioning tests checks; 1) the control of the machine and that is it sufficiently robust enough to handle typical disturbances, 2) the mechanical units on the machine are checked that they are securely mounted and correctly tuned and 3) the HMI is checked to ensure that status, error, diagnostic messages are displayed appropriately and clearly to the operator and the control of the machine from the HMI is correct. The commissioning tests for the Human Machine Interface that are relevant to this thesis are detailed in table 8.9.

Table 8.9 Types of Commissioning Tests for Automated Production Machinery.

Type		Objective
General machine tests	Control	To check the ability of the control system to operate normally as well as handle disturbances.
	Mechanical	To ensure that all mechanical units are securely mounted and all moving mechanical parts correctly tuned or conditioned to operate smoothly.
HMI aspect of machine commissioning tests	Human Machine Interface	To ensure that status and error diagnostic messages displayed appropriately and clearly for the operator. Ensure that all control of the machine is exercising the intended mechanical units. Ensure all mode controls commands are accurate.

Examples of three main commissioning tests that were conducted and are relevant to the C-B HMI system implemented in the case studies are summarised in the following sub sections.

Return to Initial Position

Scenario: The machine was stopped during its automatic operation and is required to return to it initial state.

Function of test: Machine HMI should execute the machine control command to return the machine to its initial position.

Expected behaviour: The machine must be set to "manual" or "step" mode by pressing relevant button on the machine HMI display. The operator then presses button "Return to Initial Position" button on machine HMI display. Each machine component returns to its initial position. When all the machine components have returned to there initial position the operator is notified via a message on the HMI stating "Machine in Initial Position"

Result: Once the Manual machine mode on the HMI display was selected and then the Return to Initial Position button was pressed the machine components are moved to their initial position the HMI displays notifies the operator.

Manual System Manipulation

Scenario:	To exercise the machine components manually via the HMI display
Function of test:	To ensure that the operator can manually select and move all the machine components to the desired state via the HMI display.
Expected behaviour:	The machine must be set to "manual" mode by pressing the Manual button on the HMI display. The machine components state are displayed on buttons on the HMI display and when clicked drive the machine component to the desired button state. If a machine component is attempt to be driven into a state which is not allowed in the machine configuration due to the manual interlocks then a message stating the machine component state that has been requested and the manual machine interlock preventing this movement.
Result:	All machine components were move to the desired states safely and manual interlocks were displayed on the HMI display.

Error Handling

Scenario:	An error is generated on the machine and is propagated to the HMI display.
Function of test:	To ensure the correct machine error information is displayed in the HMI display.
Expected behaviour:	An error is injected into the machine by disconnecting a limit switch on a closed loop actuator (e.g. a STOP component). The machine STOP component will generate an error message "No Feedback is detected". This error message must then be propagated to the HMI display and displayed.
Result:	The error message was displayed on the HMI screen.

8.6.2 Fail Safe Operation

It is critical that the HMI system operates in a failsafe manor. The failsafe mechanisms of the system are inherent to the architecture and the application implementation. Failure to fully supporting fail safe operation of the HMI system would render the system unsuitable for controlling industrial manufacturing machinery.

Client Server Architecture

The client server architecture that has been adopted in the C-B HMI approach ensures the machine will continue to operate if one or more of the clients fail. If a HMI client fails then another HMI client can be used to complete the machine task.

The automation machine is autonomous in its operation and all of the machine sequencing and interlocking logic is embedded in the intelligent machine components on the physical machine. The machine will continue to function if the HMI server is disconnected however the monitoring, diagnostics capabilities will not be functional.

Application Implementation

The application implementation relates to the mechanisms that are implemented within the application which are for example message timeouts and heartbeat messaging to manage the communications between the client and server. These mechanisms are required to ensure that if communications are disconnected between the client and the server then the server is aware and does not continue processing a command. For example this is partially critical if the user requests a manual machine component move and then the communication between the HMI client (where the manual move button has been pressed) and HMI server fails. The HMI server must terminate the manual move and not continue to move the machine component – this would lead to the machine being in manual mode and not under the direct control of the operator. Another example scenario of where it is critical to detect if this communication link has failed is if the HMI real-time messages do not update on the HMI display so the operator assumes that the machine components are not changing state but actually the communication link has failed therefore no HMI server messages are updating the client HMI. A heartbeat

mechanism exists where the HMI server must send out an heartbeat message every 1 second, if this is not received by the client HMI then a communications message is displayed on the screen indicating clearly that there is a fault within the HMI system.

8.6.3 Response Times

The response times of the HMI system are defined as the time taken for actions to be propagated though the HMI. For example, the time taken from a button is pressed on the HMI display to when the machine command is issued is the user to the machine HMI response time. Likewise when a machine state change occurs the time taken between the machine issuing the event on the HMI and it being displayed is the response time of the HMI from the machine to the user.

It is expected the machine state information must be propagated considerably faster than the machine takes to move. HMI systems within automotive assembly manufacturing must have a 500ms worst case response time [22]. The HMI was response time of the HMI systems implemented in the three case studies was tested to ensure they meet this criterion. The test involved setting a machine component state change and watching the real-time HMI Task Component change. No formal timing was conducted due to it being apparent (straight away the HMI responded to the machine state) that the 500 ms response was met. The communication mechanism for the real-time information implements TCP/IP socket technologies. The response times of these technologies are detailed in table 8.10. In addition to the communications between the server and client other overheads that relate to the performance of the real-time information transfer from the machine to the HMI display are the client and server processing time. The client's real-time HMI Task Component that handles this type of machine is implemented in the form a Java Applet which is an application program that runs within a web browser. The time taken to process the message and render the display is insignificant. The server side processing involves processing the machine information, storing the message in the machine common model and then updating

the broadcaster. Again this is a virtually insignificant overhead in the response time of the HMI system.

Table 8.10 TCP IP Performance Characteristics [104]

Response time	Jitter	Data rate
1ms	<1ms	100Mbit/s

8.7 Summary

In this chapter the C-B HMI is evaluated from a qualitative and quantitative perspective in meeting the requirements of global manufacturing. Three major requirements that a production machine must support to meet the needs of global manufacturing are described (see section 3.3) and each of these requirements has a number of attributes inherent in the HMI and evaluated in this chapter.

The first requirement for the HMI system is to support a faster machine ramp up time. The HMI software implementation development time (see section 8.3.1), skill level required (see section 8.3.2) and reuse of HMI software components (see section 8.3.3) are all attributes that are supported in the C-B HMI system that influence the time-scale associated with the development of a new machine HMI. In addition the HMI system must also support early HMI validation (see 8.3.4) and operator training in (see section 8.3.5) which is described from the HMI case studies (chapter 7).

Increasing the machine's operating effectiveness is a core driver for any new HMI system. The C-B HMI system architecture allows the encapsulation of best industry practice (see section 8.4.1) in building high quality HMI systems encapsulates industry experience and knowledge. Displaying accurate machine information in the HMI system is key to the end user's effectiveness in operating, maintenance or diagnosis of the machine. This accuracy is inherent to the HMI system architecture adopted here since a single common machine data repository stores all information ensuring information is not fragmented across different systems. A common look and feel of the end user HMI client is one of the key factors that will increase the usability of manufacturing systems and reduce operating costs for end user's. The HMI approach described in this thesis supports reusing common system components and each support a particular aspect of the end user HMI functionality. In reusing common system components HMI systems are developed that have a common look (see section 8.4.3).

Increasing the up time of a manufacturing machine is a major industry driver for HMI systems [19]. The results from some basic scenarios to test the remote diagnostics capabilities of the HMI system described in this thesis are reviewed in section 8.5. It can be seen that using the remote HMI system together with existing industry practices (telephone conversation and live video of the machine) achieved a significant reduction in machine diagnostic times.

9 Conclusions

9.1 Research Achievements

The research described in this thesis has contributed to the knowledge in the area of HMI systems used within automated manufacturing systems. The knowledge has been generated through the design, implementation and evaluation of;

- A novel modelling notation for the specification capture of HMI systems.
- A novel approach to the design and re-configuration of industrial machine HMI systems.
- An integrated system architecture that supports providing consistent machine information using a common machine data model.

The objectives of the research together with the finding are detailed in table 9.1. The innovative aspects of the research include;

- A modelling notation and technique that formally and unambiguously describes the HMI System functionality and layout that is easily understood by end users and provides sufficient information to enable either software programmers, or automated software generation tools, to compose fully functional HMI Systems.
- An HMI system architecture that meets the challenges in engineering manufacturing machines within in terms of; 1) faster machine ramp up time and rapid machine re-configuration, 2) increased machine operating effectiveness though encapsulating industry best practice and providing common HMI display look and feel, 3) increasing the machine up time through the provision of remote monitoring and diagnostics to all globally distributed engineering partners.
- The selection and integration of state of the art technologies to compose an HMI system that better meets the needs of global manufacturing demands.
- Description of the HMI systems construction process and stakeholder roles in the adoption of this novel HMI approach to the automotive manufacturing industry.

Table 9.1 Research focus and achievements fulfilled in this research

Primary Focus and Goals of research	Research Achievements and Contributions
Primary focus of the research presented within this thesis is to Conceptualise, Design, Development, Prototype and Evaluate a new approach to HMI systems that supports Run-time Monitoring, Diagnostic, Servicing and Control capabilities for Component Based Automated Manufacturing Production Machinery within an Global Manufacturing Environment.	• A novel C-B approach to the engineering of HMI systems to support manufacturing production machines within an global manufacturing environment has been proposed in this research.
Secondary Goals of research	
Study current industry best practice within manufacturing machine HMI systems and demands faced from Global Manufacturing.	• The emerging requirements that manufacturing machines HMI systems must meet are determined. • Limitations and inadequacies with current HMI systems in supporting global manufacturing systems are established.
A structure and technique that comprehensively describes the end users functionality in the context of manufacturing machine systems.	• The development of a storyboarding modelling technique that formally and unambiguously describes the HMI System functionality. • Storyboarding modelling notation that is easily understood by end system users and provides sufficient information to allow software programmers or automated software tools to compose fully functional C-B HMI Systems.
A System Architecture that supports the activities for the design and implementation of C-B HMI systems that meet the requirements of global manufacturing.	• A system architecture that contains a common machine data model containing all machine information that ensures the HMI always has machine accurate information. • The proposal of a client server architecture that supports globally distributed engineering partners who are involved in the engineering of automotive production machine.
Evaluation of the new approach developed in this research in an attempt to describe the benefits, characteristics and its significant improvements over current practices.	• Description of how approaches and concepts developed in this research aim to support the complete lifecycle requirements of C-B automated manufacturing machines.

9.2 Recommendations for Future Work

The ultimate goal of the research presented in this thesis is to provide a HMI system that fully meets the needs in engineering automated production machinery within the emerging global manufacturing environment. Due to the time and resource constraints on the research to date a number of areas relating to further research are suggested, in particular;

- More thorough evaluation and assessment of the approach could be undertaken. System evaluation by real industry stakeholders rather than academic researchers acting the role of the industrial stakeholders would give a more in depth study and enable a greater understanding and evaluation of C-B HMI approach.
- Although the C-B HMI approach described in this thesis has been implemented on industrial machines they were not live manufacturing production systems, only prototype machines used for proof of concept of new machine systems. Applying the C-B HMI approach to a live manufacturing system would generate new knowledge and complete the migration path to a new generation of industrial HMI systems.
- Application of the C-B HMI approach to in other industry sectors. In the research documented in this thesis the focus has been the automotive industry. Section 9.3 briefly discusses the application of the C-B HMI approach to other industry sectors.

9.3 Application of C-B HMI to other Industries

In this thesis the focus of the research has been on the automotive industry. From the application of the C-B HMI approach to this industry a particular set of benefits and advantages have been realised (see section 8.2). Other industrial sectors may however benefit most from other aspects of the new functionality and performance offered by the C-B HMI approach. For example in the automotive industry frequent machine re-configuration is considered to be critical requirement, but if the approach was applied to the petrochemical industry remote monitoring and diagnostics may be a more critical requirement.

Research on core requirements for HMI systems in a number of different industries, for example (see figure 9.1) petrochemical, packaging, semiconductor manufacture, electronics manufacture and textile manufacture would allow the suitability of the C-B HMI approach described in this thesis to be comparatively analysed across industrial sectors.

Implementing the C-B HMI approach in different industries to compose application specific end user HMI's would involve engineering some industry specific components whilst other components and aspects of the system would be remain generic. For example the HMI server architecture would remain generic across all industries together with the HMI composition tools that are core to the approach. The HMI Widgets and the composition of the HMI Widgets into HMI Task Components would be particular to the individual stakeholders in each industry.

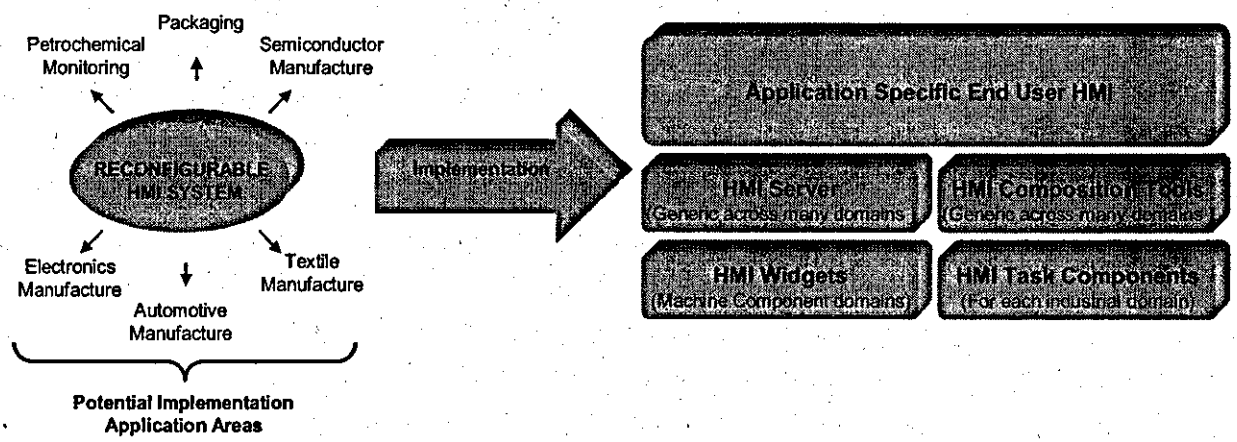


Figure 9.1 Application of C-B HMI to other Industries

References

[1] I. Henrik, D. Rudiger, M. Hägele, K. Arif, ABB Robotics - A white paper on the status and opportunities of the European Robotics Industry, European Robotics Forum (IFR ERF) & European Robotics Research Network (EURON), September 2001.

[2] L. Bainbridge, "Ironies of Automation", Department of Psychology, University College London, 1983.

[3] U. Berger, R. Lepratti, Advanced Man-Machine Interfaces for Industrial Robots in Automotive Manufacturing, Brandenburg University of Technology at Cottbus, Cottbus, Germany 2004

[4] [WR2000] United Nations – Economic Commission for Europe (UN/ECE)/International Federation of Robotics, World Robotics 2000, UN/IFR, Geneva, Switzerland.

[5] R. Harrison R, A. A. West, R. H. Weston, R. P. Monfared, Distributed Engineering of Manufacturing Machines, Proc Instn Mech Engrs, 212 (part B), 2000, p 217-231revised, 1993.

[6] E. W. Mellor, R. Harrison, "A Component-Based Human Machine Interface System For Automotive Manufacturing Machines", Proceedings of 7th Biennial ASME Conference: Engineering Systems Design and Analysis July, 2004, Manchester, UK

[7] R. Harrison, "Component Based Distributed Control Systems For Automotive Manufacturing Machinery Developed Under the Foresight Vehicle Programme", S.A.E. Conference 2001.

[8] Anon., "Human-Machine Interface (HMI) Worldwide Outlook", http://www.arcweb.com/research/auto/hmi.asp, October 2002.

[9] R. B. Hee, (1995, October). Knowing the basics of plcs—part 1. Electrical Construction and Maintenance 94(10), 20-28.

[10] K. R. Brookings, (1991, March). State logic: a paradigm for integrated control in the 90s. Control Engineering 38(5), 36-39.

[11] B. G. Coury, C. M. Pietras, (1989,November). Alphanumeric and graphic displays for dynamic process monitoring and control. Ergonomics 32(11), 1373-1389.

[12] K. Crater, (1997, June). Plc programming: state language is the way to go. Instrumentation & Control Systems 70(6), 91-92.

[13] S. R. Dartt, (1984, February). Programmable logic controller survey: comparison of top systems. Pulp & Paper 58(2), 54-58.

[14] Louis [online]. Available WWW URL: http://www.softis.is> (1995).

[15] W. Eckerson, "Three Tier Client/Server Architecture: Scalability, Performance, and Efficiency in Client Server Applications." Open Information Systems 10, 1 (January 1995): 3(20).

[16] H. Edelstein, "Unravelling Client Server Architectures." DBMS 7, 5 (May 1994): 34(7).

[17] Anon, Information Technology for European Advancement (ITEA), "Potential for integration of conditioned-based information", September 2004, Page 22 – 25, http://www.proteus-iteaproject.com/index.php?p=publications

[18] K, Liebstückel, Information Technology for European Advancement (ITEA), "A generic platform for e-Maintenance", 2001, p. 13

[19] Anon. Information Technology for European Advancement (ITEA), Potential for integration of conditioned-based information, 1999, p.11

[20] V. Stoll, (1999): Technological trends for the monitoring and diagnosis of complex systems, in: Westkaemper, e/it, w/tender, S. (Hrsg.): Maintenance management in new organization forms, Berlin and others 1999, p. 121 – 141

[21] R.H, Weston, "Manufacturing Process Integration and Information Support, Methods Models and Tools" , Beijing International Symposium on CIM , Wu, C., Xiong, G. and Fan, Y. (eds), CIMS Expert Group of China, ISCIMS '92 , Tsinghua University, Beijing, China, June 1992, pp 203-213

[22] Interview in J.A Krause Commissioning Engineers, Breman, Germany June 2001

[23] D. W. Thomas, "A Process Definition Environment For Component Based Manufacturing Machine Control Systems Developed Under The Foresight Vehicle Programme", S.A.E. Conference 2001.

[24] R. Harrison, A. A. West, Component Based Paradigm for the Design and Implementation of Control Systems in Electronics Manufacturing Machinery, J of Electronics Manufacturing, 10 (1), 2000, p 1-17

[25] R. P. Monfared, A.A. West, R. Harrison M. Wilkinson, A User-Oriented Interface Methodology For Automotive Manufacturing Machines Developed Under The Foresight Vehicle Programme, Society of Automotive Engineers, 2002.

[26] R. Lewis, Modelling Control Systems Using IEC 61499, The Institution of Electrical Engineers, Control Engineering Series 59, 2001.

[27] Steps Specification, Structured Transfer-Machine Eddi Programming System, Ford Motor Company.

[28] M. R. Lucas, D. M. Tilbury, "A study of current logic design practices in the automotive manufacturing industry", International Journal of Human-Computer Studies pages 725–753, 2003.

[29] United States National Science Foundation - Executive Overview, "Next-Generation Manufacturing, A Framework for Action", January 1997.

[30] E. Templ, A. Stritzinger, G. Pomberger, "Process Visualization with Oberon System 3 and Gadgets", Institut F. Wirtschaftsinformatik Ch. Doppler Laboratory for Software Engineering, Johannes Kepler University Linz, Austria, December 2001

[31] D. A. Carr, Toward More Understandable User Interface Specifications, Department of Computer Science and Electrical Engineering, University of Luleå, S-971 87 Luleå, Sweden 2001

[32] A. I. Wasserman, "Extending State Transition Diagrams for the Specification of Human-Computer, Interaction", IEEE Transactions on Software Engineering, Vol. SE-11, No. 8, August, 1985, pp. 699-713.

[33] Y. Y. Wong, Rough and ready prototypes: Lessons from graphic design. In Short Talks Proceedings of CHI '92: Human Factors in Computing Systems, Monterey, CA, May 1992, pp. 83-84.

[34] P. Pinheiro da Silva, N.W. Paton, "User Interface Modeling with UML," Information Modeling and Knowledge Bases XII, H. Jaakkola, H. Kangassalo, and E. Kawaguchi, eds., IOS Press, 2001, pp. 203–217.

[35] P. Markopoulos, P. Marijnissen, "UML as a Representation for Interaction Designs," Proc. Australian Conf. Computer-Human Interaction, CHISIG, 2000, pp. 240–249.

[36] N. J. Nunes, J. Falcão e Cunha, "Wisdom: A Software Engineering Method for Small Software Development Companies," IEEE Software, vol. 17, no. 5, Sept./Oct. 2000, pp. 113–119.

[37] F. Paternò, "Towards a UML for Interactive Systems," Proc. 8th IFIP Working Conf. Eng. for Human-Computer Interaction (EHC 01), Springer-Verlag, 2001, pp. 7–18.

[38] I. Crnkovic, M. Larsson, "Component-Based Software Engineering – New Paradigm of Software Development", Department of Computer Engineering Mälardalen University Box 883, 721 23 Västerås, Sweden

[39] I. Crnkovic, M. Larsson, , "A Case Study: Demands on Component-based Development", Department of Computer Engineering Mälardalen University Box 883, 721 23 Västerås, Sweden

[40] Jam Cad Designs Inc 2001, http://www.jamcaddesigns.com/

[41] J. P. Baartman, Automation of assembly operations on parts, PhD thesis, Delft University of Technology, Delft, The Netherlands, 1995, ISBN90-370-0119

[42] Anon, OMAC Baseline Architecture Functional Requirements, Version 1.0 OMAC Architecture Working Group Jan 2002

[43] J. R. Moyne, D. M. Tilbury, K. Sukerkar, H. Wijaya, An Integrated Distributed Software System for Reconfigurable Manufacturing Engineering Research Center for Reconfigurable Manufacturing Systems, University of Michigan

[44] Y. Koren, U. Heisel, F. Jovane, T. Moriwaki, G. Pritschow, A. G. Ulsoy, and H. Van Brussel, "Reconfigurable Manufacturing Systems," CIRP Annals, Vol. 48/2/1999

[45] J. Moyne, J. Korsakas, D. Tilbury, "Reconfigurable Factory Testbed (RFT): A Distributed Testbed for Reconfigurable Manufacturing Systems, Japan-USA Symposium on Flexible Automation, Denver, Colorado, July 2004.

[46] G. Behm. Enabling integration of APC with factory services using middleware technology. In AEC/APC Symposium XIII Proceedings, Banff, Canada, October 2001.

[47] J. Moyne, J. Korsakas, C. Milas, T. Hobrla, T. Hong, H. Kim, J. Priskorn, K. Sukerkar, H. Wijaya, N. Agarwal and D. Tilbury, "A Software Infrastructure for Reconfigurable Manufacturing Systems," 2nd CIRP Reconfigurable Manufacturing Conference, Ann Arbor, Michigan, September 2003

[48] J. Lee, N. Jun, K. Muammer, Tether-free Technologies for e-Manufacturing,e-Maintenance & e-Service, NSF Industry/University Cooperative Research Center (I/UCRC) for Intelligent Maintenance Systems (IMS), University of Wisconsin-Milwaukee (UWM) / University of Michigan (UM)

[49] M. W. Gertz, A Visual Programming Environment for Real-Time Control Systems, PhD Thesis, Department of Electrical and Computer Engineering Carnegie Mellon University Pittsburgh, Pennsylvania 15213-3890, Nov 1994

[50] P. Kruchten, Architectural Blueprints—The "4+1" View Model of Software Architecture, published in IEEE Software 12 (6) November 1995, pp. 42-50

[51] D. Garlan & M. Shaw, "An Introduction to Software Architecture," Advances in Software Engineering and Knowledge Engineering, Vol. 1, World Scientific Publishing Co. (1993).

[52] C. North, B. Shneiderman, Snap-Together Visualization: Can Users Construct And Operate Coordinated Visualizations? Int. J. Human-Computer Studies (2000, 53 p715-p739)

[53] M. H. Ong, A. A. West, S.M. Lee, R. Harrison, Evaluating Remote Maintenance of Production Machinery in the Automotive, MSI Research Institute, Loughborough University, UK

[54] M. H. Ong, Documentation on Lamb Technicon - General. Project Documentation, 2002, MSI Research Institute, Loughborough University

[55] J. Lee, Teleservice Engineering in Manufacturing: Challenges and Opportunities. International Journal of Machine Tools and Manufacture, 1998, 38, 901-910.

[56] R. Yu, B. Iung, H. Panetto, A Multi-agents based E-maintenance System with Case-based Reasoning Decision Support. Engineering Application of Artificial Intelligence, 2003, 16, 321-333

[57] M. A. Rothenberg, Systems Development with Systematic Software Reuse: An Empirical Analysis of Project Success Factors, School of Accountancy and Information Management, Arizona State University, PhD thesis

[58] J. E. Gaffney Jr, T. A. Durek, (1989): Software reuse - key to enhanced productivity: some quantitative models. Information and Software Technology 31(5): 258- 267.

[59] R. Banker, R. J. Kauffman, (1991): Reuse and Productivity in Integrated Computer-Aided Software Engineering: An Empirical Study. MIS Quarterly 15(3): 374-401.

[60] V. R. Basili, L. Briand, W. L. Melo, (1996): How Reuse Influences Productivity in Object-Oriented Systems. Communications of the ACM 39(10): 104-116.

[61] W. Frakes (Virginia Tech), C. Terry (Incode Corporation), Software Reuse: Metrics and Models, ACM Computing Surveys, Vol. 28, No. 2, June 1996

[62] J. S. Poulin, Measuring Software Reuse: principles, practices, and economic models. Reading, MA, Addison-Wesley. 1997

[63] D.A. Vera, A Component based Approach to the Design and Implementation of a Virtual Prototyping Environment (for manufacturing systems), PhD thesis, MSI Research Institute, Loughborough University 2004

[64] S. M. Lee, A Component based Distributed Control Paradigm for Manufacturing Automation Systems), PhD thesis, MSI Research Institute, Loughborough University 2004

[65] D. A. Vera, A. A. West, R. Harrison, D. W. Thomas, Virtual visualisation and prototyping environment for Component base production machinery, 31st North American Manufacturing Research Conference, Hamilton, Ontario, Canada 2003

[66] M. H. Ong, Modularity analysis o Transfer Line Machines in Lamb Technicon, MSI Research Institute, Loughborough University 2003

[67] M. Biehl, E. Prater, R. John, Remote Repair, Diagnostics, and Maintenance, McIntyre Communications Of The ACM, Vol. 47, No. 11, November 2004

[68] P. Jonsson, Company-wide Integration of Strategic Maintenance: An Empirical Analysis. International Journal of Production Economics, 1999, 50-61, 155-164.

[69] E. W. Mellor, Harrison R, A. A. West, "Reconfigurable User Interface's to Support Monitoring and Diagnostic Capabilities within Agile Automated Manufacturing System's", proceedings of IEEE Conference on Robotics, Automation and Mechatronics (RAM), Singapore, December 2004.

[70] D. W. Thomas, A. A West, R. Harrison, A Process Definition Environment for Component Based Manufacturing Machine Control Systems Developed Under the Foresight Vehicle Programme. Proceedings of the Society of Automotive Engineers World Congress and Exposition, Paper no. 2002-01-0468, Foresight Vehicle Technology: Supply Chain & Manufacturing/Machine Manufacturing Control (Part C&D), 2002.

[71] E. W. Mellor, The Application of Component Based Human Machine Interfaces within Automotive Manufacturing Machines. Mechatronics and Robotics Conference '04, Aachen, Germany, 2004.

[72] M. Koc, J. Ni, J. Lee, P. Bandyopadhyay, Introduction of e-Manufacturing. 31st North American Manufacturing Research Conference, Hamilton, Ontario, Canada, 2003.

[73] J. Lee, Teleservice Engineering in Manufacturing : Challenges and Opportunities. International Journal of Machine Tools and Manufacture, 1998, 38, 901-910.

[74] J.Grundy, J. Hosking, Developing Adaptable User Interfaces for Component-based University of Auckland, New Zealand, Systems, Interacting with Computers, vol. 14, no. 3,

[75] B. Myers, S. E. Hudson, R. Pausch, Past, Present and Future of User Interface Software Tools, Human Computer Interaction Institute, Carnegie Mellon University, Pittsburgh, Sept 1999

[76] M. Baron, P. Girard, SUIDT : A task model based GUI-Builder: Task MOdels and DIAgrams for user interface design Laboratoire d'Informatique Scientifique et Industrielle, ENSMA, In Proceedings of TAMODIA, July 18-19, 2002, Romania, Bucharest

[77] M. Abrams, C. Phanouriou, A. Batongbacal, S.William, J. Shuster. Uiml: An appliance-independent xml user interface language. In WWW8 / Computer Networks 31(11-16): 1695-1708, 1999.

[78] R.P. Monfared, A.A. West, R. Harrison M. Wilkinson, A User-Oriented Interface Methodology For Automotive Manufacturing Machines Developed Under The Foresight Vehicle Programme, Society of Automotive Engineers, 2002.

[79] S. M. Lee, R. Harrison, A. A. West, "A Component-based Distributed Control System for Assembly Automation", 2nd IEEE International Conference on Industrial Informatics INDIN 04, 24-26 June 2004, Berlin, Germany.

[80] M. H. Ong, R. P. Monfared, S. M. Lee, A. A. West, R. Harrison, "Evaluating the Implementation of the Component-Based System in the Automotive Sector", proceeding of ESDA 2004, 7th Biennial ASME Conference, Engineering Systems Design and Analysis, Manchester, United Kingdom

[81] R. Harrison, A. A. West, "Component-based paradigm for the design and implementation of control systems in electronics manufacturing machinery", Journal of Electronics Manufacture, 10(1), December 2000, pp 1-17, ISSN 0960-3131.

[82] R. Harrison, A. A. West, C. D. Wright, "Integrating machine design and control", International Journal of Computer Integrated Manufacturing, 13(6), November 2000, pp 498-516, ISSN 0951-192X.

[83] R. P. Monfared, R. H. Weston, "A Method to Develop Semi-Generic Information Models of Change-Capable Cell Control Systems," Int.J. Computers in Industry, Issue 41, vol.3, April 2000, pp 279-294.

[84] A. A. West, B. A. Bowen, R. P. Monfared, and A. Hodgson, "User-responsive interface generation for manufacturing systems: a theoretical basis", Proceedings of the Institution of Mechanical Engineers Part B: Journal of Engineering Manufacture, 214, 2000, pp 379-392, ISSN 0954-4054.

[85] R. Harrison, Proposed Collaboration with Lamb Technicon, MSI Research, Loughborough, Internal report, April 2003

[86] C. C. Huang, Overview of Modular Product Development, Proc. Natl. Sci. Counc. ROC(A) Vol. 24, No. 3, 2000. pp. 149-165, August 1999

[87] D. Vera, Virtual Model to support Agile manufacturing, PhD thesis, Loughborough University, 2004

[88] COMPAG, COMponent Based Paradigm for AGile Automation, Loughborough University research project.

[89] GEMM, Global Engineering of Manufacturing Machinery, Loughborough University research project.

[90] COMPANION, COmmon Model for PArtNers in AutomatION, Loughborough University research project.

[91] IEC61131-3 PLC IEEE standard

[92] Baartman JP. "Automation of assembly operations on parts" PhD thesis, Delft University of Technology, The Netherlands, 1995.

[93] S.M. Lee, A Component Based Approach to Manufacturing Automation Systems, PhD thesis, Loughborough University, 2004

[94] T. Lister, F. McGrath, E. Yourdon, "The Little Book of Testing, Volume II, Implementation Techniques", SPMN Pocket Guidebooks, E-Book - http://www.iceincusa.com/16CSP/content/resource/resfrm.htm

[95] J. Niederst, "Web Design in a Nutshell", O'Reilly; 1st edition, December 1998, ISBN: 1565925157.

[96] B. Shneiderman, "Designing the user interface :strategies for effective human-computer interaction", Harlow : Addison-Wesley, 1998, ISBN: 0201694972.

[97] N. Koch1, H. Baumeister, R. Hennicker, L Mandel, "Extending UML to Model Navigation and Presentation in Web Applications", Ludwig-Maximilians-University Munich, Germany, http://www.pst.informatik.uni-muenchen.de/personen/kochn/ExtendingUML.pdf , 2002.

[98] H. Baumeister, N. Koch and L. Mandel, "Towards a UML Extension for Hypermedia Design", Proceedings UML'99 Conference, France, R., Rumpe, B. (Eds) , LNCS, Vol. 1723, Springer Verlag, 614-629, 1999.

[99] D. Sano D: Large-Scale Web Sites – A Visual Design Methodology. Wiley Computer Publishing, 1996.

[100] J. Preece, H. Rogers, D. Benyon, S. Holland, T. Carey: Human-Computer Interaction. Addison Wesley (1994).

[101] A. Cechich, M. Piattini, A Vallecillo, Component-Based Software Quality: Methods and Techniques, Springer-Verlag Berlin and Heidelberg GmbH & Co. K, August 2003.

[102] K. K. Lau, Series on Component-Based Software Development - Vol. 1, The University of Manchester, UK, ISBN 981-238-828-1, March 2004.

[103] P. Kruchten. "The 4+1 View Model of Architecture," IEEE Software, vol. 12, no. 6, pp. 42-50, November 1995.

[104] Anon, IAONA Handbook Industrial Ethernet in Third Edition, August 2005, EBook request from; handbook@iaona.org

[105] E. W. Mellor, Harrison R, A. A. West, "Component Based Human Machine Interface System To Support Agile Manufacturing", proceedings of 12th IFAC Symposium on Information Control Problems in Manufacturing, Saint-Etienne, France May 2006.

[106] Interview in Lamb Technicon, Project Engineering Team, Mildenhall, UK July 2003

[107] Wikipedia http://en.wikipedia.org/wiki/User_interface 2005

[108] Anon http://www.matrikonopc.com/resources/dictionary.aspx May 2006

[109] T. Sturgeon, R Lester, "The New Global Supply Base: New Challenges for local Suppliers in East Asia", Global Production Networking and Technological Change, Oxford University Press 2004

[110] G. H. Lee, Designs of components and manufacturing systems for agile manufacturing int. j. prod. res., 1998, vol. 36, no. 4, pp1023 - 1044

[111] P Leitão, F. Restivo, A Layered Approach to Distributed Manufacturing. In: Proceedings of ASI'99 International Conference, Leuven, Belgium, 21-23 September 1999

[112] Y. Koren, U. Heisel, F. Jovane, T. Moriwaki, A. Pritchow, G. Van Brussel, H. Ulsoy, 1999, "Reconfigurable Manufacturing Systems," CIRP Annals, Vol. 48, No. 2,

[113] Y. Sun and Z. Zhang 'A Methodology to Form Agility Strategy – A Case Study', Proc of the 2nd Itnl Conf on Manufacturing Research, 7-9 September 2004, Sheffield, UK:

[114] K. C. Thramboulidis, An Architecture to Extend the IEC61499 Model for Distributed Control Applications, 7th International Conference on Automation Technology, Taiwan May 2003

[115] K. C. Thramboulidis, Development of Distributed Industrial Control Applications: The CORFU Framework, 4th IEEE International Workshop on Factory Communication Systems, Vasteras, Sweden, August 2002

[116] V. Vyatkin, J L. Martinez L. J. Christensen OOONEIDA: An Open, Object-Oriented Knowledge Economy for Intelligent Distributed Automation, Tampere University of Technology, Finland, HOLOBLOC, Inc

[117] D. Norman, S. Draper, (editors), User Centered System Design: New Perspectives on Human-Computer Interaction, pp. 87-124, Lawrence Erlbaum Associates Inc., 1986.

[118] L. Birnbaum, R. Bareiss, T. Hinrichs, and C. Johnson. 1998. Interface Design based on Standardized Task Models. In Proceedings of the 1998International Conference on Intelligent User Interfaces. San Francisco, CA: Association for Computing Machinery, pp. 65-72, ACM 1998.

[119] C. Resnick, Need for Real-time Production Information Drives Growth, ARC Market Outlook Study, September 2004

[120] A Degani1, M Heymann2, "Formal Verification Of Human-Automation Interaction", Nasa Ames Research Centre, Mountain View, California1, Department Of Computer Science Technion, Israel Institute Of Technology2, June 1999

[121] M. D. Lozano, F. Montero, Pascual González, A Usability and Accessibility Oriented Development Process, 8th ERCIM Workshop "User Interfaces For All", European Research Consortium for Informatics and Mathematics, Vienna, Austria. June 2004

[122] C. Stephanidis, User Interfaces for all: new perspectives into HCI. In User Interfaces for all- concepts, methods and tools. Lawrence Erlbaum Associates, Mahwah, NJ. pp. 3-17, 2001.

[123] A. Puerta, MOBILE: User-Centered Interface Building, in CHI99, ACM Conference on Human Factors in Computing Systems. Pittsburgh, ACM press, May 1999.

[124] E. Schlungbaum, Support of Task-based User Interface Design in TADEUS, CHI'98 Workshop, From Task to Dialogue: Task-based User Interface Design, ACM SIGCHI, Nov 2001.

[125] P. Silva,User Interface Declarative Models and Development Environments: A Survey, In Interactive Systems: Design, Specification and Verification (7th International Workshop on Design, Specification and Verification of Interactive Systems), Limerick, Ireland. LNCS Vol. 1946, pp. 207-226, Springer-Verlag, June 2000.

[126] G. Fischer, User Modelling in Human-Computer Interaction. User Modelling and User-Adapted Interaction, 2000.

[127] A. Martikainen, An XML-based Framework for Developing Usable and Reusable User Interfaces for Multi-channel Applications, Report, Department of Computer Science, University Of Helsinki, May 2002

[128] J. Vanderdonckt, F. Bodart, Encapsulating Knowledge for Intelligent Automatic Interaction Objects Selection, in ACM Annual conference on Human Factors in Computing Systems, pp. 424-429, 1993, ACM Press.

[129] K. Stirewalt, S. Rugaber, Automating UI Generation by Model Composition, In Proceedings of Automated Software Engineering (ASE'98), 13th IEEE International Conference, IEEE 1998.

[130] L. Birnbaum, R. Bareiss, T. Hinrichs, C. Johnson. 1998. Interface Design Based on Standardized Task Models. In Proceedings of the 1998International Conference on Intelligent User Interfaces. San Francisco, CA: Association for Computing Machinery, pp. 65-72, ACM 1998.

[131] Mohammed Elkoutbi, Ismaïl Khriss, Rudolf Keller, User Interface Prototyping using UML Specifications, Report, Université de Montréal, Montréal, Canada.

[132] H. Baumeister, N. Koch, L. Mandel, Towards a UML extension for hypermedia design.InProceedings UML'99, France, LNCS, Vol. 1723. Springer- Verlag (1999) 614-629.

[133] J. Rumbaugh, Ivar Jacobson, Grady Booch, The Unified Modeling Language Reference Manual, Addison Wesley Aug 2004.

[134] The Object Primer 3rd Edition, Agile Model Driven Development with UML 2, Cambridge University Press, Mar 2004

[135] J.A Landay, B.A. Myers, Interactive sketching for the early stages of user interface design. In Proceedings of CHI '95: Human Factors in Computing Systems, Denver, CO, May 1995, pp. 43-50.

[136] J. Lin, A visual language for a sketch-based UI prototyping tool, Conference on Human Factors in Computing Systems, CHI '99 Pittsburgh, Pennsylvania, Pages: 298 – 299, 1999, ISBN:1-58113-158-5

[137] E. Lank, et. al. "Interactive System for Recognizing Hand Drawn UML Diagrams". Proceedings for CASCON 2000, Mississauga, Ontario, Canada, Nov 2000

[138] J. Lin, M.W. Newman, J.I. Hong, and J.A. Landay, DENIM: Finding a Tighter Fit Between Tools and Practice for Web Site Design. CHI Letters: Human Factors in Computing Systems, CHI 2000, 2000: p. 510-517.

[139] J. Fleming, Web Navigation: designing the Use Interface, Sebastopol, CA: O'Reilly, 1998

[140] D. J. Mayhew,. The Usability Engineering Lifecycle. Morgan Kaufmann, San Francisco, CA. 1999

[141] L. Constantine, L. Lockwood, Software for Use: A Practical Guide to the Models and Methods of Usage-Centered Design. Addison-Wesley, New York, 1999.

[142] M. F. Costabile,. Usability in the Software Life Cycle. Handbook of Software Engineering and Knowledge Engineering. World Scientific Publishing, pp. 179-192. Singapore, 2001.

[143] Y. Koren, U. Heisel, F. Jovane, T. Moriwaki, G. Pritschow, A. G. Ulsoy, and H. Van Brussel, "Reconfigurable Manufacturing Systems," CIRP Annals, Vol. 48/2/, 1999

[144] J. Moyne, J. Korsakas, C. Milas, T. Hobrla, T. Hong, H. Kim, J. Priskorn, K. Sukerkar, H. Wijaya, N. Agarwal and D. Tilbury, "A Software Infrastructure for Reconfigurable Manufacturing Systems," 2nd CIRP Reconfigurable Manufacturing Conference, Ann Arbor, Michigan, September 2003.

[145] Y. Koren, A.G. Ulsoy, , "NSF Engineering Research Center for Reconfigurable Machining Systems," Proc. of the NSF Grantees Conf., Monterrey, Mexico, 1997.

[146] M. Mehrabi, A. Ulsoy, Y. Koren, "Reconfigurable Manufacturing System and Their Enabling Technologies," International J. of Manufacturing Technology and Management, Vol. 1, No. 1, pp. 113-130, 2000.

[147] Y. Koren, U. Heisel, F. Jovane, T. Moriwaki, G. Pritchow, H. Van Brussel, A. Ulsoy, 1999, "Reconfigurable Manufacturing Systems," CIRP Annals, Vol. 48, No. 2, (keynote paper), 1999

[148] J. Moyne, D. Tilbury, K. Sukerkar, H. Wijaya, An Integrated Distributed Software System for Reconfigurable Manufacturing, Engineering Research Centre for Reconfigurable Manufacturing Systems, University of Michigan, 2002

[149] J. Pascual, M. Masso, P. González López, Model-Based Design And New User Interfaces: Current Practices And Opportunities, Louise Research Group Instituto De Investigación En Informática De Albacete (I3a) Campus Universitario, Albacete 2002

[150] Engineering Research Center for Reconfigurable Manufacturing Systems of the National Science Foundation under Award Number EEC-9529125.

[151] P. Johnson, "Human Computer Interaction: Psychology, Task Analysis And Software Engineering", Mcgraw Hill Book Company, 1992.

[152] P. Stevens, R. Pooley Using UML: Software Engineering with Objects and Components, Addison-Wesley (Object Technology Series), ISBN: 0-201-64860-1.

List of Publications

1. E. W. Mellor, Harrison R, A. A. West, "A Component-Based Human Machine Interface System For Automotive Manufacturing Machines", Proceedings of 7th Biennial ASME Conference: Engineering Systems Design and Analysis July, 2004, Manchester, UK.

2. E. W. Mellor, Harrison R, A. A. West, "The Application of User Orientated Human-Machine Interfaces within Automated Automotive Manufacturing Machines", Proceedings of MECHROB 2004 Conference, IEEE Industrial Electronics Society, Aachen, Germany, 2004.

3. E. W. Mellor, Harrison R, A. A. West, "Reconfigurable User Interface's to Support Monitoring and Diagnostic Capabilities within Agile Automated Manufacturing System's", proceedings of IEEE Conference on Robotics, Automation and Mechatronics (RAM), Singapore, December 2004.

4. E. W. Mellor, Harrison R, A. A. West, "Component Based Human Machine Interface System To Support Agile Manufacturing", proceedings of 12th IFAC Symposium on Information Control Problems in Manufacturing, Saint-Etienne, France May 2006.