Development of a Toolkit for Component-Based Automation Systems

By

Charles Stuart McLeod

Submitted to Department Manufacturing and Mechanical Engineering, Loughborough University

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

Loughborough University

June 2013

From the earliest days of mass production in the automotive industry there has been a progressive move towards the use of flexible manufacturing systems that cater for product variants that meet market demands. In recent years this market has become more demanding with pressures from legislation, globalisation and increased customer expectations. This has lead to the current trends of mass customisation in production.

In order to support this manufacturing systems are not only becoming more flexible[†] to cope with the increased product variants, but also more agile[‡] such that they may respond more rapidly to market changes. Modularisation[§] is widely used to increase the agility of automation systems, such that they may be more readily reconfigured[¶]. Also with globalisation into India and Asia semi-automatic machines (machines that interact with human operators) are more frequently used to reduce capital outlay and increase flexibility. There is an increasing need for tools and methodologies that support this in order to improve design robustness, reduce design time and gain a competitive edge in the market.

The research presented in this thesis is built upon the work from COMPAG/COMPANION (COMponent- based Paradigm for AGile automation, and COmmon Model for PArtNers in automatION), and as part of the BDA (Business Driven Automation), SOCRADES (Service Oriented Cross-layer infrastructure for Distributed smart Embedded deviceS), and IMC-AESOP (ArchitecturE for Service-Oriented Process – monitoring and control) projects conducted at Loughborough University UK.

This research details the design and implementation of a toolkit for building and simulating automation systems comprising components with behaviour described using Finite State Machines (FSM). The research focus is the development of the engineering toolkit that can support the automation system lifecycle from initial design through commissioning to maintenance and reconfiguration as well as the integration of a virtual human. This is achieved using a novel data structure that supports component definitions for control, simulation, maintenance and the novel integration of a virtual human into the automation system operation.

Keywords: Virtual Prototyping, Manufacturing System Engineering, System Architecture, Component and Component-based Design, Virtual Reality Modelling Language (VRML).

† May cope with a wide product mix
‡ May be easily adapted to cope with new unforeseen changes
§ Build from independent units that can be constructed in different ways
¶ Change behaviour by changing defined parameters not reprogramming

Acknowledgements

From the outset of this research to its completion, the work has not been easy, it has provided me with a number of valuable experiences and taught me many invaluable lessons along the way. Throughout this I have enjoyed the help and support of family and friends as well as the guidance and belief of my supervisors who have freely given me their support both practically and academically, whilst giving me the freedom and trust to explore technologies and methodologies to support my research.

I would like to express my gratitude to my supervisors Dr Robert Harrison and Dr Andy West for continually encouraging me through this process and helping me make the research a success, and Dr Les Lee who supported my work and provided case studies and access to relevant personnel at Ford to make my research relevant and focussed. Also my colleagues at Loughborough University both past and present that have helped and taught me so much, especially Rev. David Thomas, Dr Daniel Vera, Dr I. Coutts and Dr Atul Jain.

I would also like to thank my wife Helen and my children Cameron and Lucy, who have loved and supported me throughout. Finally I would like to thank my mother Joyce McLeod, whose attitude towards life and family has always inspired me.

Table of Contents

List of Figures

Chapter 1 **Introduction**

1.1 **Introduction**

During the early years of mass production the aim was to provide high quality products to a growing consumer market. The consumer has become more sophisticated and their requirements and expectations have changed demanding high quality products that are customized to their needs. Companies have been forced to follow this market trend to remain competitive becoming more responsive and flexible by adopting an approach for "Agile Manufacturing".

The hypothesis of this thesis is that a component-based approach taken throughout the lifecycle of an automation system, that includes manual, semi–automatic and automatic systems will result in agile automation systems that support the manufacture of products that meet the customer's requirements. Tools and methodologies are required to support building component based systems such that benefits may be realised by the enterprise (from conception to decommissioning). The design, development and deployment of these tools to support collaborative working between departments and industrial partners in the supply chain based upon detailed end user requirements (see Chapter 3) are the foci of this thesis, in particular the novel integration of a virtual human into the automation system control.

1.2 **Focus on Domain**

Automation can be applied to many domains such as packaging, warehousing, processing and building automation, each of which have their own domain specific requirements. However since the author has been involved in projects for discrete part manufacture within the automotive industry this was chosen as the domain of the research.

The automotive industry provides a well-established, high speed, high volume and global business adopting innovative technologies and methods to solve production and business problems.

1.3 Identification of the Real Requirements

In this thesis the development of a component based toolkit to support modelling, virtual prototyping and simulation of automation systems that meets the requirements of the industrial partners involved in the design and implementation of Automation Systems (specifically for power train operations (i.e. engine manufacture)) is described. The main area for evaluation is the visualization of problems and solutions to promote collaboration between the partners in the supply chain each of which have their own domain specific knowledge, viewpoints and perspectives [95][97][98].

There has always been pressure to minimize the time to market, which in turn requires the compression of project timelines, ultimately requiring the compression of the development / reconfiguration of the automation system. To facilitate this, a high degree of simultaneous engineering is required between the distributed partners in the supply chain and the manufacturing customer. Therefore communication and sharing of ideas and designs is imperative requiring engineering tools to enable distributed discussion and resolution of issues as the product and manufacturing system evolve.

1.3.1 Selection of Case Studies.

Two case studies (detailed in Chapter 5) have been identified to demonstrate the results and benefits of the approach adopted in this thesis within the automotive industry. The first case study uses a training rig (i.e. the Festo Test Rig), used to demonstrate and teach students the fundamentals of automation, PLC and distributed control programming. The Festo Test Rig provides a target platform that has been used to validate the outputs from the engineering tools.

The second case study resulted from the early adoption of the approach by one of the project partners (Ford), who commissioned 15 real production machines to be modelled so they could evaluate the tools as part of their simultaneous engineering processes in the production of a new engine assembly line. A semi automatic machine has been chosen (i.e. OP60 Block Load) that encompasses manual operations, interaction with non-automated equipment (a manual hoist and dowel insertion tool) and automated behaviour (assembling the engine block to a pallet), to highlight all the relevant aspects of the design tools research. The machine includes automatic assembly components as well as a complex sequence of operations that the

human operator must follow. The use of the tools was undertaken by a multidisciplinary team of engineers across the supply chain. The goal was to demonstrate the tool to co-ordinate and support the inter-working and decision-making of distributed engineering teams in such a way that the multiple viewpoints of team members can be considered from the early stages in the system lifecycle from design right thought to commissioning and beyond.

1.3.2 State of the art Automation System Development Tools

There are many CAD applications that provide mechanisms to develop 3D simulation of automation systems that may potentially allow the development of process control, system testing and validation. These tools provide mechanisms for virtual prototyping, system visualisation and analysis of "what if" scenarios [97]. The 3D graphical representations of automation systems are one element of a "common model" of the system being designed that is central to facilitating collaboration between engineers from different domains [10].

Whilst CAD is invaluable in product design, these virtual models still remain in the design office and require skilled CAD developers to create manipulate and maintain. As a result many companies do not use the automation system design aspects of the tools and many do not get a significant return on their investment in these models due to the cost involved in maintaining the models throughout the automation system lifecycle.

There are areas of manufacturing where the use of CAD/ CAM has allowed the product design to be directly linked to the manufacture of that product such as CNC machines and electronic PCB / chip design. In this case, any change in the product is reflected directly in changes to the manufacturing output.

It may be deduced that there is a need for a toolset that is aligned to the requirements of manufacturing. The user of the toolkit must be able to describe the operation of an automation system, using a combination of textual information, diagrams and a "digital mockup", and use it to facilitate machine validation and the output of the toolkit must capable of controlling a "real system". If changes to the automation system are required the toolkit should provide functionality to alter and validate ther system behaviour, before the changes are implemented and run on the "real system". These virtual models can be regarded as "living models" that reflect operation of the real automation system through its lifecycle from conception to decommissioning.

Furthermore there are many CAD based tools that allow detailed development of human behaviour in the virtual world. These tools are mainly focused on highlighting manual operations and hence require the use of powerful computers operated by skilled CAD engineers. These tools currently do not integrate well into the operation of the automation system, i.e. changes in the automation system operation are not reflected in the virtual human operation. Therefore once developed, the update of these models throughout the automation system lifetime is limited due to the excessive cost and time to maintain them.

The investment in CAD required for automation system design is usually via proprietary packages, which tend to be large (i.e. require a large amount of processing power and system memory) and complex. The investment in software, training and support ultimately results in to vendor "lock-in". In order to share designs and collaborate using a common virtual prototyping approach will therefore require the industrial partners to enter into an inflexible long-term partnership [116]. This long-term partnership does not readily support multi partner collaboration and integration using a virtual environment, here the requirement is typically to provide short-term adaptive and responsive collaboration between the partners in the supply chain, without a large investment in software training and IT infrastructure.

1.4 Research Objectives

This research is based upon the work supervised by Prof. Robert Harrison and Prof. Andrew West on component-based engineering and fully distributed control. The goal is to support current automation trends in the automotive industry with specific evaluation of requirements developed from business process research carried out at the Ford UK motor company.

From this research it was identified that there are limited solution options for the building and maintenance of component-based systems that may be integrated into the automation system specification, prototyping, development, deployment and maintenance lifecycle phases. In addition it was noted that most approaches for the development of virtual prototyping systems were focused on standalone CAD with the automation output being only considered as a plugin.

Figure 1 illustrates current state of the art tools with high levels of functionality and complexity. Note: these tools require additional investments in IT resources (e.g. large storage, and processing power), cost and training.

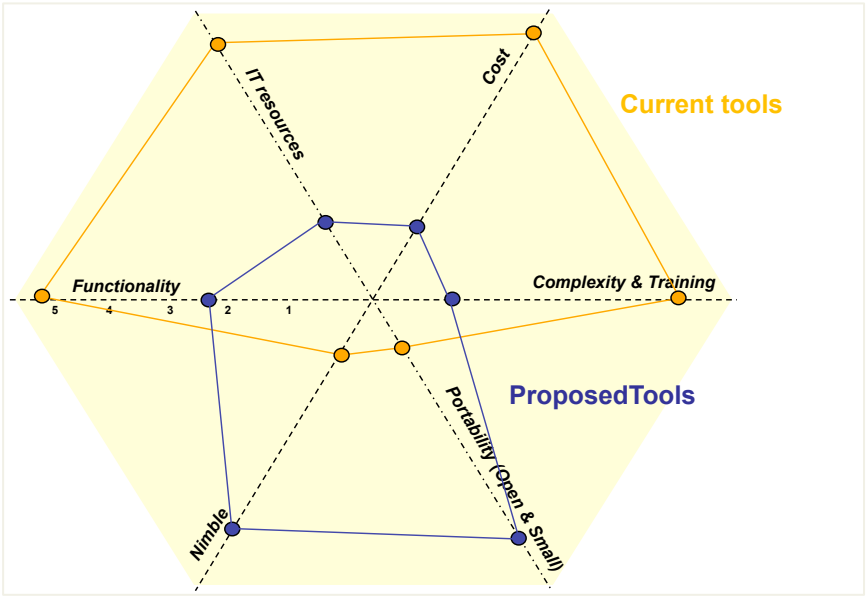


Figure 1 Radar Chart of current tools with respect to the proposed tools objectives

The Proposed tools shown by the blue line in Figure 1 reported in this thesis have been designed to be relatively low cost, easy to use and require only standard specification computer platforms to operate. The supported functionality has been focused upon the requirements of the domain resulting in readily editable, small-scale models that can support design collaboration and promote shared understanding between departments and industrial partners.

The following research objectives have been identified:

- **The development of a Common Model** – The Common Model is essentially a single data model employed to describe the behaviour of automation systems. Common models can be extended to include more detailed information as and wen this becomes available. All of the information required to describe and operate the automation system is maintained in the common model (i.e. no functional information is determined from 3D models as is generally the case). 3D information is used only in visualising the operation of the common model and plotting its behaviour.

- **The support of Model Reusability** – Model reusability is understood as the storage and reuse of existing automation system lifecycle knowledge so that it may be readily used on subsequent design projects.

- **The development of System Simplicity** – Simplicity allows multidiscipline teams to develop complex automation systems without extensive training. In addition this

objective supports the ability of any person within a single organisation or a group of industrial partners to be able to manipulate and interact with the simulation with minimal training.

- **The support of System Deployment functionality**– It is vital to be able to store and distribute models such that any engineer will be able to view them on their local computer via distribution over supported networks.

- **The development of a Vendor Neutral toolkit** – The tools should not be tied to single vendors. Hence "views" of the common model have to be created to support integration with proprietary solutions. For example for one implementation the CCE Tools were required to support deployment of the automation system definition onto a Schneider Modicon series PLC, then the same model was used to support deployment to a Siemens PLC.

- **The integration of the human into the machine control system** – The tools should allow the interactions between human operators and machines to be modelled such that if either the human process or the machine process is altered in any way, the impact on the overall performance of the automation system can be evaluated.

1.5 **Thesis Structure**

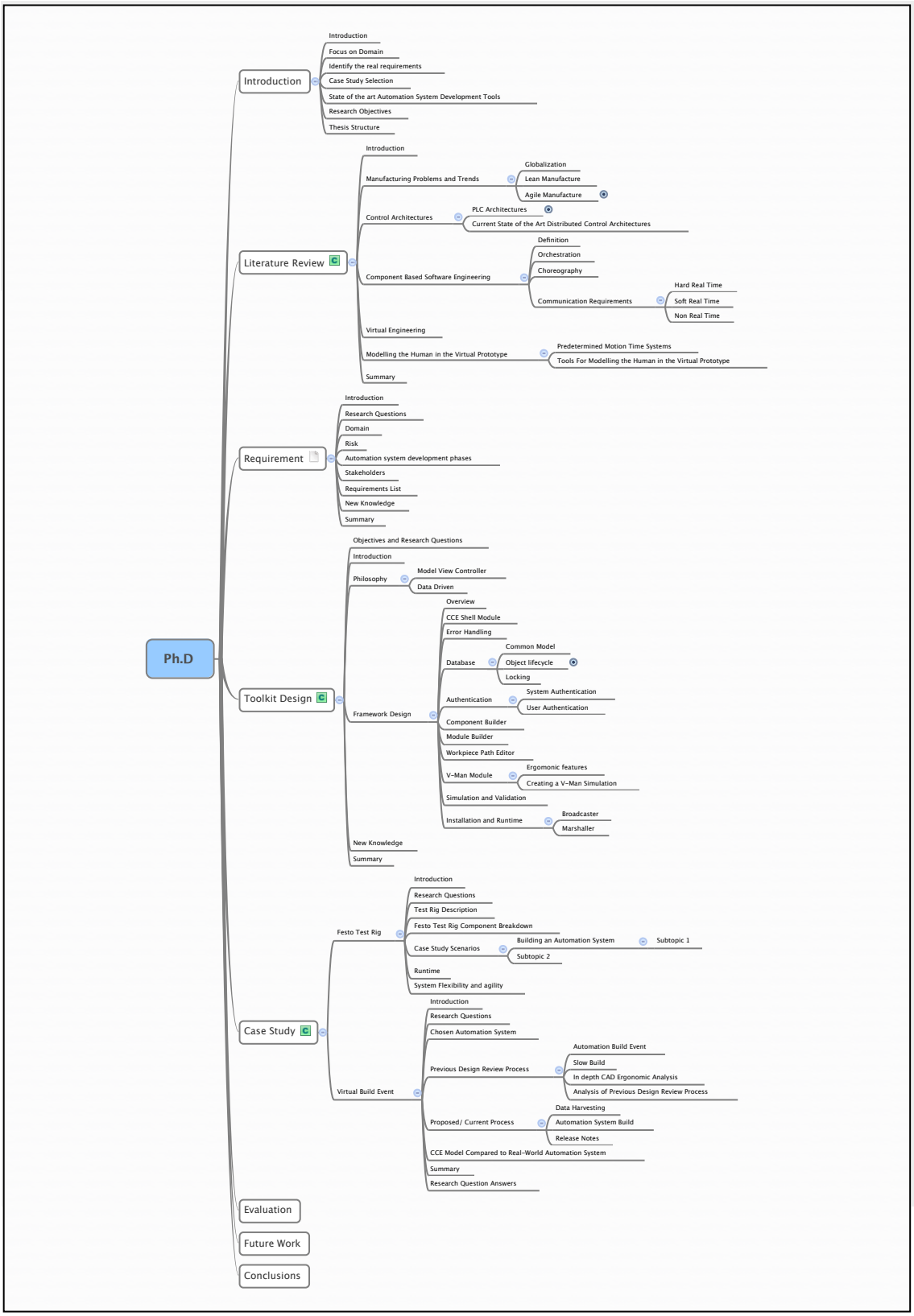


Figure 2 Thesis Structure

The overall structure of the thesis is illustrated in Figure 2. In Chapter 2, a review of the current manufacturing problems and trends within the automotive powertrain domain is detailed, with emphasis on issues resulting from globalisation and the support of lean and agile manufacturing paradigms. The typical control architectures are discussed covering a range of solutions from centralised PLC's to fully distributed controllers. The component-based software engineering paradigm is then detailed along with the requirements for control distribution and communication. Current trends in virtual engineering/ prototyping are reviewed with focus being placed upon modelling the role of the human in the prototype system.

In Chapter 3 the research carried out to determine the business requirements for automation systems, with particular emphasis on the management of risk during the development of an engine assembly line is outlined. The development phases of the automation system are described along with the linking these phases to the stakeholders from the supply chain. The results of this research have been used to create a list of requirements that have to be addressed by the CCE tools detailed in Chapter 4.

The design and implementation of the CCE Tool is detailed in Chapter 4. The design philosophy is described, followed by the framework adopted for the application. Software modules required to create components, build automation systems, integrate the virtual human, simulate and validate the machine operation and install an execute the created automation system on the "real" hardware are discussed.

The case studies that demonstrate the deployment, testing and reconfiguration an automation system to highlight the validity of the approach and its use within an automotive company as a simultaneous engineering / collaboration tool are presented in Chapter 5. Evaluation of the performance of the case studies with respect to the objectives outlined in Chapter 1 is detailed in Chapter 6.

A summary of the contributions to new knowledge and the opportunities for future research and exploitation activities is highlighted in Chapter 7 as a conclusion to the thesis.

Chapter 2 The Need For Change in the Automotive Industry – Literature Study

2.1 Introduction

The aim of mass production was to provide high quality products to a growing consumer market. Global competition and the demands of consumers for cheap, high quality, customised products has required manufacturers to seek solutions supporting "Agile Manufacturing Paradigms" in order to remain in business in this environment of constant change.

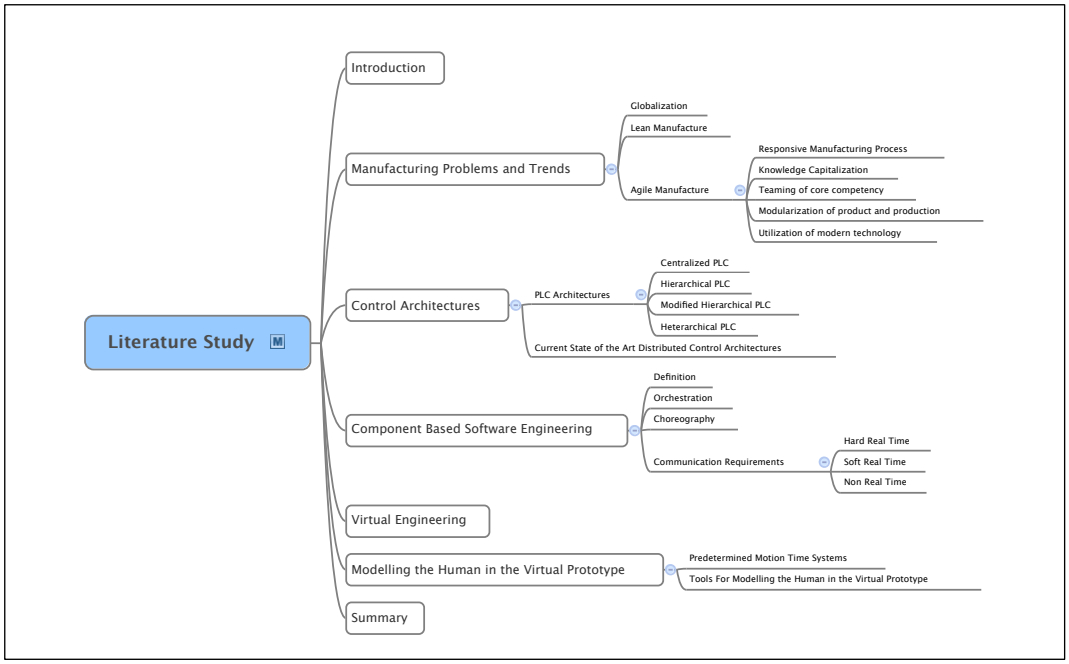


Figure 3 Literature Study Overview Mindmap

The breakdown of the research presented in this Chapter is illustrated in Figure 3, highlighting the discussion of the following areas:

- The business requirements for change.
- The current state of the art in control architectures to support the business requirements
- The move towards component-based software engineering.

- The state of the art in Virtual Engineering / Prototyping.
- Issues associated with the integration of the human into virtual prototyping environments.

The challenges that face companies within global are detailed. The current approaches adopted by the automotive industry to develop manufacturing systems in order to remain competitive are highlighted. In order to establish a benchmark for this research, the historical background and current state of the art of automation engineering solutions are outlined. Suitable control architectures to support change are discussed by firstly examining traditional approaches, in particular the ISA-95 Functional Model [22,117,118], and the NIST 7 layer Model [14].

Manufacturing organisation have made large investments in PLC based control from centralised control, distributed input and output, fieldbus networks and didtributed PLCs. These are discussed in this chapter and the problems associated with using them to develop automation systems are detailed [114,119]. PLC systems are contrasted with the current state-of-the-art distributed control architectures focusing specifically on non-vendor specific solutions and the tools available to support them. Competitive automation solutions, especially in emerging markets, require the integration of cheap skilled labour (the most agile components) to complete many tasks in a flexible way. This necessitates the integration between automation and manual systems to be addressed.

The challenges and the changes that companies have to respond to are detailed. Currently attempts to address these issues are outlined, along with a number of projects that are researching state of the art solutions to component-based control systems and the tools required to support their design and development within a virtual environment where human operation may be integrated with machine operation.

2.2 Manufacturing Problems and Trends

Globalisation and the thrust towards a more flexible customer driven manufacturing policy has resulted in a fiercely competitive marketplace where companies must provide high-quality products that evolve and change very quickly in a cost-effective manner in order to maintain a competitive edge [99][43][64][80][113].

Figure 4 Assembly Line at Ford Motor Company 1908

In September 1908 the first production Model T developed by Henry Ford was built. This is thought to be the first mass-produced automobile built on a production line (Figure 4). Manufacturing processes and customer requirements and expectations have progressed significantly since then. The modern consumer wants to be treated as an individual and demands products that are customized to their needs whilst still of high-quality. This has forced companies to move towards lean manufacturing and management paradigms to "allow an assembly plant to absorb greater complexity with little adverse effect on performance" [17] as long as the variety was contained within the product mix.

These factors have had an impact on the product lifecycle, reducing both development times (in order to maintain a technical edge) and production lead times (to assist in lean manufacture and market responsive manufacturing). Since batch sizes have to be reduced, traditional mass production manufacturing techniques are no longer appropriate due to their inherent inflexibility to change. Manufacturers are increasingly looking to migrate their manufacturing paradigms towards a more agile, flexible and adaptive approach. This forces the supply chain to become more agile, as

without capable suppliers and supportive supply chain management practices, the end use company will fail to achieve their goals [20-21].

Early research on flexibility tended to refer to two main capabilities of a manufacturing system [100]: The range of manufacturing operations that can be supported and the engineering effort required to "adapt" the system. The definition of manufacturing flexibility has been decomposed to 10 types discussed later in this chapter.

Flexible Manufacturing Systems (FMSs) by definition tend to be more complex and less reliable than machining lines dedicated to a single product (i.e. dedicated manufacturing systems (DMSs)) and as such are more suited to small batch sizes with a high number of variants.

Reconfigurable Manufacturing Systems (RMSs) were conceived to provide enterprises with a more flexible production capacity compared with DMSs. RMSs have to support a suitable range of production functionality (in order to support change) and technological openness to allow the ready integration of new machine technologies [101]. The Research on RMS's changed the focus from flexibility towards being able to adapt to change by allowing the system to be reconfigured quickly to accommodate new or different products, and leads naturally to the research on Agile manufacture.

When addressing business changes, enterprises have to consider the effect of their actions in a wider context. Relevant "stakeholders" within an organization e.g. customers, employees, investors, suppliers, the environment and society as a whole have to be identified [44]. Within this thesis only customers, employees, investors and suppliers are considered since these are key to automation design, development and deployment.

Two definitions of Agile Manufacturing (AM) are cited below:

"Agile manufacturing is the ability of accomplishing rapid changeover from the assembly of one product to the assembly of another product. " [35]

"The concept of Agile Manufacturing is also built around the synthesis of a number of enterprises that each have some core skills or competencies which they bring to a joint venturing operation, which is based on using each partners' facilities and resources." [36]

The concept of agility is at the heart of manufacturing systems integration within a group of companies required to perform collaborative tasks aiming to deal with the unpredictable and rapid changes (e.g. changing product designs, addressing customers demands and manufacturing new types of products) in both business and production plants [10]. In an ideal situation, this group of companies form "partnerships" resulting in an integrated enterprise containing a range of the best available resources for the business opportunities or opportunities of interest.

The Next Generation Manufacturing System outlined in [23] details recommendations that manufacturing enterprises should adopt to remain competitive and agile. Enterprises have to adopt the following capabilities:

Responsive manufacturing processes: Manufacturing processes should be responsive to market changes. The responsiveness of a manufacturing process can be viewed in terms of change capability and change capability rate [25]; change capability can be described as a function of agility and flexibility. ElMaraghy [24] has identified ten types of manufacturing system flexibility:

1. Machine flexibility: The range of operations performed without set-up change.
2. Material handling flexibility: The Number of used paths / Number of material flow paths between machines.
3. Operational flexibility: The number of different processing plans available for part fabrication.
4. Process flexibility: The sets of part types that can be produced without major changes in set-up.
5. Product flexibility: The ease of introducing products into an existing product mix.

6. Routing flexibility: The Number of feasible routes of all part types / Number of part types.

7. Volume flexibility: The ability to accept the product-mix whilst the system runs virtually uninterrupted.

8. Expansion flexibility: The ease of adjusting capacity and / or capability though physical chance to the system.

9. Control program flexibility: The ability of the control programs to run with only minor configuration changes.

10. Production flexibility: The number of all part types that can be produced without adding major capital equipment.

Note: This thesis is focused on supporting the flexibility demanded by types 5, 7, 8 and 9.

Knowledge capitalization: It is critical that any enterprise has the ability to capture and re-use the knowledge encapsulated in its products, processes and personnel. These *best practices* need to be stored in a universal format that can be readily accessed throughout the enterprise, the product lifecycle and by future projects. By learning from past projects, designers can avoid repeating previous errors. Several methods have defined techniques to "memorize" lessons and experiences from previous projects in the form of a project memory [37]. The exploitation of knowledge requires that content be described in an explicit way, so others can readily understand the data format (i.e. lessons learned need to be described in terms of metadata, drawings and categorization of documents according to their content).

Teaming of core competency: Businesses competing in a global market have to focus on their core competencies. The increasing complexity of products and the associated manufacturing processes mean that it is unlikely that a single company will possess all the expertise and equipment to design and manufacture a given product [20]. Alliances with other companies have to be developed to create and maintain a competitive edge.

Modularisation of product and production: Modularisation [26] has been identified as a method of handling of complexities in manufacturing. It allows the production of a wide variety of products (e.g. mass customisation) based up on the combination of a finite number production modules [27].

Modularisation has been applied to automotive production systems in automotive supply chain companies (e.g. Krause, Giddings and Lewis and Lamb Technicon) who use production modularity as an every day part of their design philosophy. The objective is the development of system capabilities to support "plug and play" manufacturing. Pre-designed modules can be assembled, interchanged and configured to build custom production equipment. To achieve, automation system modules should be built with a high degree of autonomy, which requires loosely coupled, distributed, co-operative, intelligent components [28, 29].

Utilization of modern technology: Information Technology has become pervasive throughout manufacturing enterprises. For example, Ethernet networking is installed in every Ford plant not only in the business areas but also supporting the manufacturing production lines. Networking provides the opportunity for integration of business management with the shop floor PLCs to support the management, integration and distribution of information from the real-time production systems to: (i) plant maintenance, (ii) production monitoring and (iii) remote machine diagnostic systems.

Harrison et al [120] have characterised three classes of "Change capabilities" :

1. *Programmability* is the ability to program system behaviour and / or composition so that a system can readily reach a range of pre-defined states.

2. *Reactivity* is the ability to react to change of an unpredictable nature by readily modifying system behaviour or composition.

3. *Proactivity* is the ability to predict and anticipate change requirements in uncertain environments and prepare system changes (i.e. behaviour or composition) accordingly.

The move towards agile manufacturing requires flexible, reconfigurable and extensible manufacturing systems. The focus of this thesis is on the tools to support the development of agile automation systems; in terms of introducing new products, ability to access and adjust the system capacity, control program flexibility, and production flexibility, by creating a toolkit that can quickly reconfigure, test and deploy automation systems in a structured controlled manner.

2.3 Manufacturing Automation Systems

The implementation of traditional control architectures and PLC architectures is detailed in this section. The problems associated with adopting these paradigms is described followed by a brief description of the current state of the art in non vendor-specific approaches to distributed control architectures.

2.3.1 Traditional Control Architectures

Machine control systems can be categorized according to their physical instantiations e.g. PLC's, motion controllers. These are programmed individually to execute sequences of commands to complete specific tasks. Communication between individual control systems is typically facilitated by a central system in a hierarchical network structure.

These traditional approaches present major issues when adopted as the basis of an intelligent manufacturing control architecture. Figure 5 shows how a typical automation system may be represented as an automation pyramid [5,22], comprising of a four level hierarchy of automation functions.

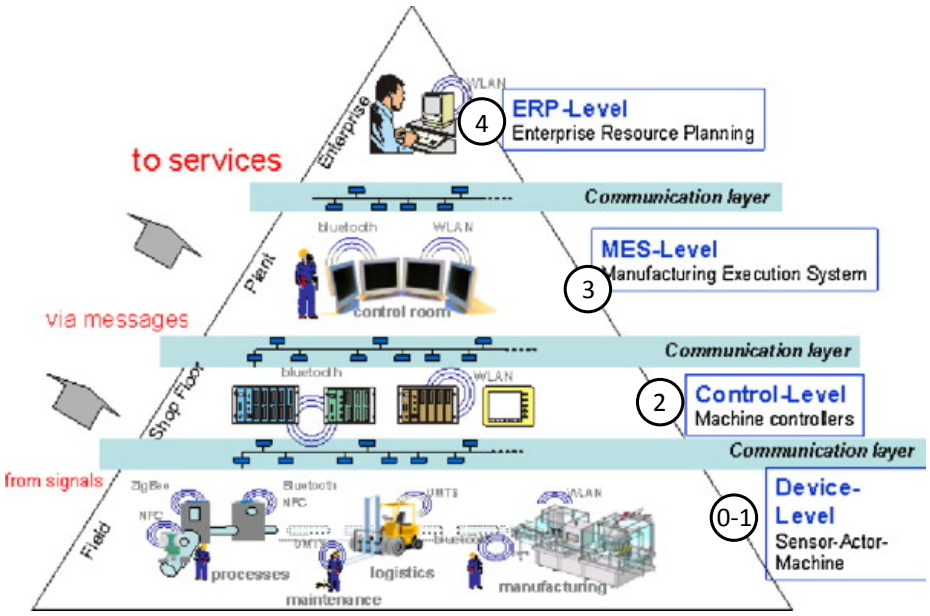


Figure 5 ISA 95 Automation Pyramid after [5,22,120]

The **ERP Level** is concerned with the control of enterprise-level finance and human resource planning, order management and fulfilment, manufacturing resource planning and distribution. The **MES Level** is primarily concerned with the monitoring of the automation process as well as manufacturing resource allocation, detailed scheduling, data acquisition, order tracking and quality management. **The Control Level** is focused on the industrial manufacturing process. This will typically comprise of automation systems such as PLC's and micro-controllers, with operator interaction using Human Machine Interfaces (HMI). The lowest level is the **Device Level**. This level comprises of devices that perform basic data acquisition and control. This thesis is concerned with the Control level, the creation of Devices, and presenting information to the MES level.

Systems at Level 2 and 3 of the ISA-95 pyramid may be partitioned into four basic elements (e.g. task decomposition, world modelling, sensory processing and value judgment) [14]. These elements are used to describe the computational nodes of the system, which can themselves be arranged into layers. Using this approach a real-time control system can be broken down into 7 Layers as shown in the table below:

No	Name	Task decomposition	World Model contents
7	Shop	Control Cell / Line activities.	Material inventory Tools and Machine routine
6	Cell	Schedules control activities.	Material inventory Tools and Machine routine
5	Workstation	Machine/ Group of machines.	Parts, tools and work flow buffers for the workstation.
4	Equipment Tasks	Elemental tasks such as move, grasp, cut etc.	Part details such as sizes and tolerances, material characteristics.
3	Elemental Move	Point to point moves.	Part features e.g. surface holes
2	Primitive	Trajectory of tools/ manipulators and their acceleration and deceleration profiles.	Edges and lines, trajectory segments and vertices.
1	Servo	Converts from tool path to actual coordinates.	Values of state variables e.g. joint positions, velocity, sensor readings and switch positions.

Table 1 NIST 7 Layer real-time system breakdown.

The 7 layer decomposition is relevant when considering traditional as well as state of the art systems [14]. In contrast with the pyramid model, the 7-layer model is focused on the task breakdown of the real-time control system rather than the overall manufacturing system.

2.3.2 PLC Architectures

PLC's are still dominant in the automation industry today. Their computing power, flexibility and the investment in engineering expertise ensures that the PLC will remain dominant in the short to medium term. However PLC vendors have been evolving the technology and in order to maintain their supremacy in the automation industry they have had to provide more flexible open approach by introducing firstly distributed IO and more recently distributed PLC's. Architectures used by PLC's have been classified into four groups: Centralized, Proper Hierarchical, Modified Hierarchical and Heterachical [46] (see Figure 6). This classification although based in 1991 still holds today.

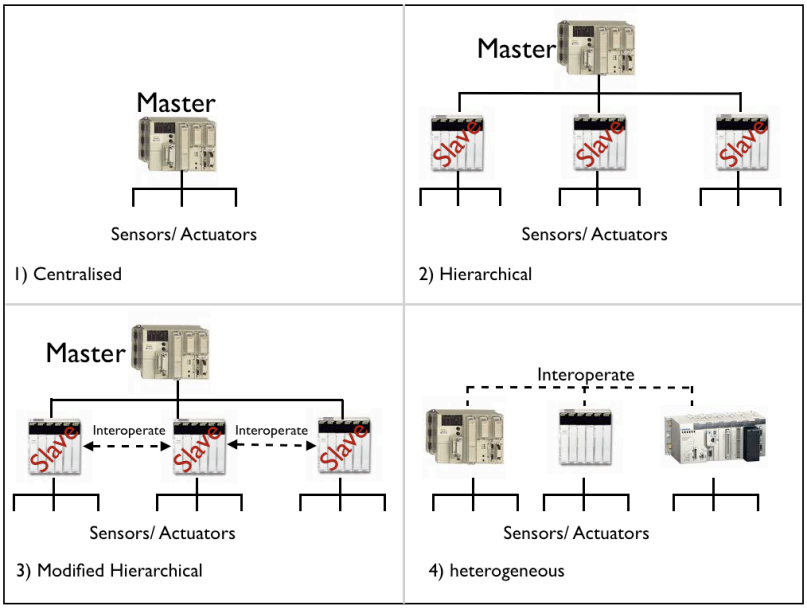


Figure 6 Dilts Control Architectures [46]

The Centralized PLC Architecture offers fast reliable control for applications with complex IO requirements. As the PLC may contain a complex program controlling a large part or the whole of the machine, any changes / reconfigurations that are required will necessitate a skilled engineer with complete knowledge of the system / process making to make modifications to the hardware / software.

The Hierarchical PLC Architecture is more flexible than the centralised architecture in that the master PLC (see Figure 6) is used to control job management (i.e. the process control level in the automation pyramid, level 5/6 of the 7 layer real-time system breakdown). This provides greater stability and allows for a higher

degree of flexibility as the Master PLC may direct the operation of the slaves based upon the cell / line requirements. Since each slave PLC has a higher degree of autonomy, improved system performance may be achieved at the expense of limiting the robustness to a single point of failure. However, due to the rigidity of the hierarchical architecture it is not easy to incorporate late changes or reconfigurations [49].

The Modified Hierarchical PLC Architecture provides improved flexibility by allowing peers to communicate directly rather than by the master, offering the ability to develop loosely coupled, collaborating PLCs. Supervisory control and sequencing is still maintained by the master PLC. This master/ slave relationship still has the drawback of a single point of failure and does not lend itself easily to late changes and reconfiguration [49].

The Heterarchical PLC Architecture is currently being lauded as the next generation of automation control system. Heterarchies allow each loosely coupled PLC to co-operate autonomously in achieving the overall systems goals. As intelligence is embedded in the lower level / smaller devices performing local operations and monitoring, the system should become more flexible as it does not rely on a central controller. In addition the failure of a control node may not affect the overall performance of the system as the knowledge and controls are distributed throughout the network, thus increasing robustness [49,50].

The heterarchical architecture has been reported to be unpredictable in terms of response time with the consequence that the overall system performance is hard to predict [49]. These criticisms are currently being addressed by thorough simulation and testing as well as the incorporation of Quality of Service (QoS) management systems, to ensure sufficient communication resources are available to meet the demands of distributed devices.

2.3.3 Current State of the Art Distributed Control Architectures

There are many proprietary control architectures for distributed control. State-of-the-art control architectures have been developed in research projects such as RiMacs, OSACA, SIRENA, and SOCRADES and in commercial solutions for communication frameworks such as OPC, OPC UA, and CORBA.

Project Name	Project Date
OSACA	1993-1995
SIRENA -ITEA	2003-2005
RiMacs	2005-2008
SODA	2006-2008
SOCRADES	2006-2009

Table 2 Distributed control architecture projects and dates

The current state of the art in distributed control architectures, built from components, describes the overall system process in terms aggregation of component behaviour (i.e. in terms of the propagation of component states). For example a Conveyor may be described with the following properties:

- Name
- Manufacturer / other static details
- Conveyor On /Off
- Conveyor Speed (m/S)
- Conveyor Component Arrival Sensor On/ Off
- Conveyor Component Exit Sensor On/ Off
- Output for each conveyor error

The control system must therefore be built in terms of the functionality offered by its components. Hence in order to achieve overall process goals (for example inserting the valves into a cylinder head) a higher-level component is required to orchestrate (Arrange and stimulate components to perform the desired goal (see section 2.4.2 on Page 27)) the behaviour of the overall system.

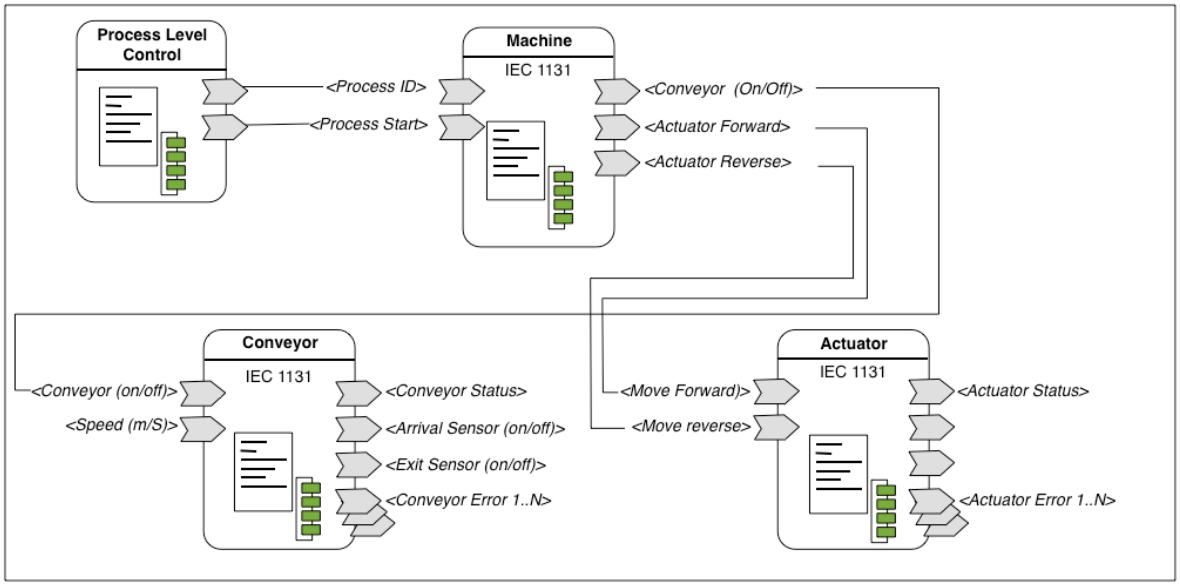


Figure 7 State of the art component breakdown (RiMacs)

Figure 7 shows how three layers of components can be used to represent component breakdown as defined by the RiMacs project [6].

Embedded Production Systems
Embedded Machine
Embedded Component

The embedded component is a functional or logical part of a system that in most cases is closely related to a physical component i.e. either the mechanical or mechatronic part.

The Embedded Machine is a higher-level component that can either be abstracted from the mechanical parts or the low-level control aspects. The Embedded Machine may: (i) use sub-components to handle the low-level control functionality (e.g. aggregation), (ii) provide machine level intelligence to components or (iii) be used as part of another embedded machine. This hierarchical architecture encourages not only reuse of components, but also modularisation and reuse of groups of components that provide higher-level functionality. For example a conveyor lifting station may comprise of several actuator components (i.e. to stop and raise a pallet) and a multitude of sensors to indicate the pallet location and possibly product variants. The embedded machine may also encapsulate and represent the functionality of a set of components as a simple set of command inputs. Thus a specific group of components may be aggregated into an embedded machine and instead of linking the high-level

control to a possibly complex set of components, it may communicate with an abstracted component to control the orchestration of the low-level components.

Embedded production systems generally orchestrate the behaviour of Embedded Machines providing an additional level of abstraction to facilitate simple development of production level control. This behaviour aims to manage machine routing / line balancing and throughput and product variation.

The SIRENA (http://www.sirena-itea.org 2003-2005) project was part of the ITEA initiative, and advocates the use of Service Oriented Architectures (SOAs) to specify and develop distributed applications using heterogeneous real-time embedded components. A set of common services that provide functionality for device discovery and "plug and play" have been defined. The aims are similar to the RiMacs project in that web services are utilised to define an abstract interface that can be used for component communication as well encapsulating autonomous "smart" devices [5].

The SODA (2006-2008) project was a development of the work achieved in the SIRENA project. The objective was to create a service-oriented framework for high-level communications between devices based on the SOA paradigm.

The SOCRADES (2006-2008) is a European project aimed to develop SOA technologies for the next generation of industrial automation systems. Web services were deployed on DPWS-enabled devices (components) to provide distributed control with an orchestrator controlling their behavior. Communication between the devices and the orchestrator was achieved using HTTP and XML base SOAP (Simple Object Access Protocol) Messages [121]

ESPRIT III OSACA (1992-1995) (Open System Architecture For Controls within Automation system) was an European project that has developed a platform to allow the interaction of object-oriented devices to facilitate the data exchange between software modules comprising control systems. The OSACA reference architecture defines the types of modules within the architecture and the tasks performed. This is achieved in a vendor-neutral manner through Application Programming Interface (API) development which includes the definition of the interface to the specific operating system as well as the communication protocol for the data exchange. [56]

The OPC Foundation (i.e. Object Linking and Embedding (OLE) for Process Control) is non-profit corporation that has defined a set of OLE / COM interface protocols to promote interoperability between automation systems and devices and business applications. OPC is specifically targeted at the process control industry. Essentially OPC provides an definition of how information can be obtained from low level devices / applications using a standard interface so that it may be incorporated into business or high level applications written for the Microsoft Windows platform.

In recent years OPC has been redeveloped by a consortium of companies (including Siemens and Microsoft) and has now been re-launched as OPC UA (OPC Unified Architecture). Instead of using OLE /COM (DCOM), OPC UA leverages the interoperability of Web Services. By using Web Services, OPC coupling with Microsoft Windows is broken, rendering OPC UA as platform and language independent and enabling integration with other SOA solutions such as those used in RiMacs and SIRENA. For improved performance OPC UA may use binary XML for communication, with built-in additional functionality for handling lost messages.

The Object Management Group (OMG) is a consortium of companies and research groups (including Microsoft, IBM and formally MSI) that has proposed the CORBA (Common Object Request Broker Architecture) architecture as a network *middleware*. Essentially CORBA provides functionality for object discovery and remote invocation in a heterogeneous environment. Object interfaces are defined in the OMG's IDL (Interface Definition Library).

Web Services (WS) are emerging as the preferred technology for deploying automated interactions between distributed and heterogeneous applications. WS are currently being supported by Schneider Electric (using the DPWS implementation), and Siemens (as OPC UA) as well as other vendors in the automotive and process industries.

Web Services represent device functionality that use the Web Services Definition Language (WSDL) to describe the behaviour available with that device in terms of the states and parameters (including any definition of the parameters), that can coexist on a network within WS environments (i.e. the infrastructure for functionality such as service discovery)

The standard communications protocol with WS environments is SOAP which is comprised of XML messages. Under the WS paradigm, SOAP messages are transmitted over the Ethernet. In large scale applications performance can be severely limited by the bandwidth of the network. Representing data using XML constructs usually results in a substantially larger messages (typically 400% larger) size than by representing the same data in binary format. This increase of the message size creates a critical issue when data have to be transmitted quickly, effectively resulting in a significant increase in the data transmission time. One of the ways to improve efficiency is to compress the XML messages, especially when the CPU overhead required for compression is less than the network latency.

2.4 Component Based Software Engineering

In the late 90's and early 00's object-oriented development targeted increased software reliability through reuse. The failure to achieve this has been attributed mainly as due to the objects being too detailed and specific for (re)use in the wider context. The decomposition of engineering systems into functional or logical components needs to be at a higher level of abstraction than the object [3], i.e. systems need to be decomposed into components.

A standard definition of a component is required along with how a control system may use components to achieve its automation goals. Component definitions must encompass the communication requirements as well as the factors associated with the communication Quality of Service (QoS).

2.4.1 Component Definitions

Component-based software engineering is defined as the practice of building software from pre-existing smaller products. These are generally called software components and when built into an existing framework such as DCOM Components, they are described as component models. [1] [3]

A software component is generally described as:

"A unit of composition with a contractually specified interface and explicit context dependencies only, that can be deployed independently and is subjected to composition by third parties" [30].

An ideal automation component should be capable of providing the following properties [31]:

- Service provision
- Validation
- Error containment
- Reuse
- Design and maintenance

A component within a distributed system should be a "self-contained computer with its own hardware (e.g. processor, memory, communication interface, interface to controlled object) and software (e.g. applications, programs, operating system), which performs a set of well defined functions within a system" [31].

The building of a manufacturing system (e.g. automation control system) requires the integration of components as well as overarching *orchestration* or *choreography* (see below) in order to achieve the system goals.

2.4.2 Orchestration

Orchestration is defined as the arrangement and stimulation of components to achieve the desired system goals [12]. For automation systems the orchestrator remains an integral part the control system and is responsible for the correct stimulation of the components during the system operation.

2.4.3 Choreography

Choreography is defined as the sequencing of the steps required to achieve the desired system goals. This requires that each component is aware of its behaviour with respect to the states of other components in the control system in order to collaborate in fulfilling the system goals. The choreographer does not need to remain as part of the system, as once the component interaction is configured the system should operate as defined. A choreographed system requires a component(s) for managing system level control such as the "health" of the system, modes of operation, error handling, user interaction and safety.

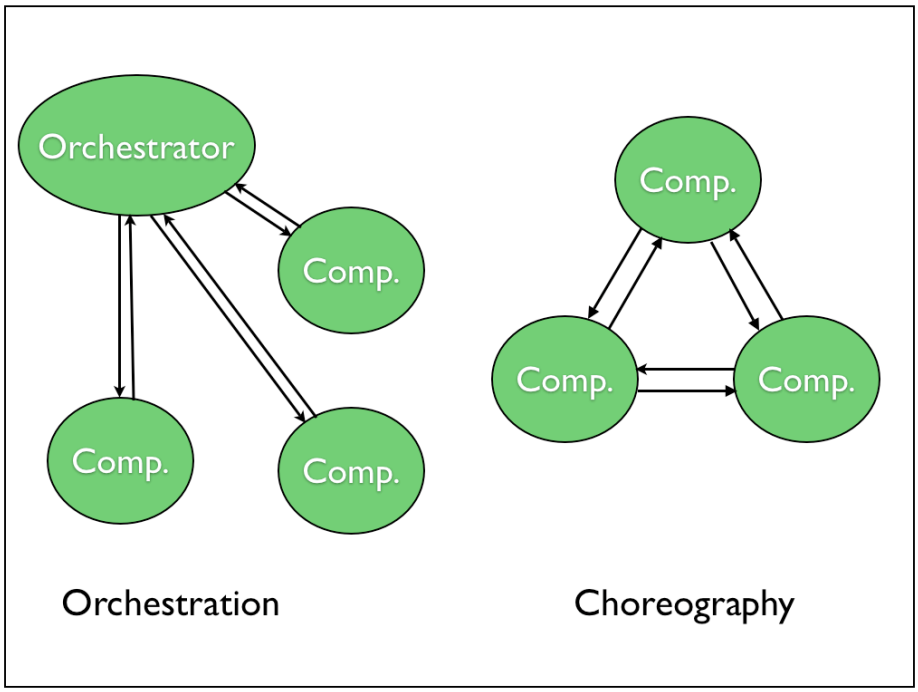


Figure 8 The difference between Orchestration and Choreography

Figure 8 illustrates the difference between orchestration and choreography in that orchestration describes the process flow between services, controlled by a singe party whereas choreography tracks the sequence of messaging between collaborating parties where no one party truly "owns" the conversation [12].

2.4.4 Communication Requirements

Real-time systems have historically been defined as systems with functionality that is "fast enough", (e.g. a simulation that could proceed at a rate that matched that of the real process it was simulating [3]). Since then the automation industry has utilised the term "real-time" to describe either communication, interaction and operation, further classification is required into hard / soft and non real-time systems.

2.4.4.1 Hard real-time

The operation of a hard real-time system depends not only on the correctness of the operation, but also the timeliness of the operation. This is especially important when synchronizing devices, or where critical failures and possible physical damage may occur if a deadline is missed. An example of this is a motor drive being controlled from a signal from a sensor (i.e. the drive is to stop when a sensor switches on). If the response time of the sensor is 50 ms then the maximum displacement of the drive can

be calculated as 0.05 * velocity. If the propagation delay is greater than this then the drive may stop too late or if the signal is only available for 60 ms then any delay in the sensor state propagation could cause the signal to be missed completely.

Deterministic systems are different to real-time systems in that each operation time is specified so that the user will be able to understand exactly how long the operation will take (within a certain tolerance).

2.4.4.2 Soft real-time

A soft real-time system is one that requires an update, but delayed update periods will not be catastrophic. Delayed updates may result in degradation of quality but the system will still perform. It is important to realize that when used in automation systems, soft real-time performance requires that funcationality and message delivery completion is 100% guaranteed.

2.4.4.3 Non real-time

A non real-time system is one that does not require any time constraints to be placed upon the operation or message delivery period. Examples of these within automation systems are production-monitoring systems displaying number of good and bad units produced and factory-monitoring systems displaying machine utilization, up and down time.

Once the level of control has been defined for the automation system, the QoS must be defined. It should be noted that several levels of QoS might be required for different aspects of the runtime, such as software download, configuration, commissioning and runtime. QoS includes factors such as timeliness and prioritisation of messages [7].

2.5 Virtual Engineering

The development of automation systems involves many partners in the supply chain from customers such as automobile manufacturers, machine builders, controls vendors and system suppliers. These partners are characterised by heterogeneous engineering domains, processes and infrastructures [102][103]. Complex automation

systems require partners to collaborate and thus there is a need to share information / knowledge and deploy distributed engineering tools.

Collaboration requires the linking of functional elements across the partners, which in turn requires communication. Computer Integrated Manufacture (CIM) emphasises the dependencies that exist between computer and network technologies and the need to achieve interworking of enterprise resource systems, which leads to integration and interoperation of computers, machines and people.

Vernadat [104] describes levels of CIM integration associated with levels within the enterprise IT infrastructure. A basic level of integration may be achieved by deploying basic IT infrastructure between partners to support Electronic Data Interchange (EDI), email and multimedia communication such as video conferencing and screen casting. In Vernadat's architecture the *application integration layer* focuses on the integration of Information Systems IS. IS integration is used to enhance communication at the engineering level, such as the deployment of distributed / shared data sources that underpin common software solutions. These shared data allow cooperation between the partners, although vendors servicing multiple customers often struggle to maintain multiple systems where each customer is using different software solutions. *Business level integration* between the partners is utilised to provide tightly coordinated business processes and decision support systems e.g. knowledge management tools, process modelling, simulation tools and decision support mechanisms.

Virtual engineering (VE) is defined as integrating geometric models and related engineering tools such as analysis, simulation, optimization and decision making tools, within a computer-generated environment that facilitates multidisciplinary collaborative product development [3].

In the area of product design and development, the aim of VE is to lever the best from the objectives of any product development cycle [73][80] (i.e. Product development speed, product cost, product performance, development cost (Faster, Cheaper and Better)).

Improved data transfer speeds, increased usage of network-based storage and increased processing power has increased the amount and variety of information

available to the enterprises. This has seen the development of large heterogeneous data management tools (as provided by SAP and Technomatics) encompassing: (i) CAD tools to share designs and develop of libraries of standard items that my be used to generate new products faster and with less risk, (ii) rapid prototyping techniques that may be used to reverse engineer products and create early physical prototypes and (iii) Virtual Reality (VR) environments that enable a user to be immersed into a 3D environment to support visualisation and interaction with products and systems. These management tools may be utilised throughout the complete life cycle of the product, from the earliest design stages to manufacturing, assembly, use, and maintenance phases [73].

The digital / numerical models that are created within these systems are used in virtual engineering environments mainly for dynamic visualisation, interaction between automation system devices and the workpieces, collision detection and tracking of the interface between the human and the machine. This thesis considers the definition of Virtual Reality (VR) as "a high end user interface that involves real-time simulation and interactions through multiple sensorial channels. These sensorial modalities are visual, auditory, tactile, smell, taste, etc." [74]

The focus of this thesis is on Virtual Prototyping (VP):

"Virtual prototype, or digital mock-up, is a computer simulation of a physical product that can be presented, analyzed, and tested from concerned product life-cycle aspects such as design/engineering, manufacturing, service, and recycling as if on a real physical model. The construction and testing of a virtual prototype is called virtual prototyping (VP)."[75]

Whilst VR can enhance the VP it is not seen as fundamental to the definition. Currently VR requires expensive specialist equipment for both the VR environment (i.e. VR gloves with force feedback, 3D VR Goggles) and the infrastructure to support them (i.e. the computer hardware capable of calculating all the aspects of the virtual environment and graphics hardware to display the results). It is acknowledged that in the future VR will become an indistinguishable part of VP.

The main suppliers of virtual system design tools development are UGS Tecnomatix (eM-tool suite) [76] and Dassault System (Catia V5 engine-based Delmia automation

tool suite) [77]. Both provide integrated environments (via central database systems) and module-based software architectures that allow mechanical and control design data to be edited and integrated into dynamic models [78]. Products come in the form of complete solutions (e.g. direct import of Catia model into the Delmia automation environment). Products such as Visual Component/V-SIM [79] are more specific. Their functions focus on 3D modeling and do not support control logic editing, but provide direct connection to OPC servers enabling simulation and testing of PLC logic against the 3D model.

All commercial products currently use proprietary formats for both 3D modelling and control logic editing which differs from the approach taken as part of this research where open formats (e.g. XML, VRML) are used right up to the final phase of the design lifecycle (i.e. deployment to physical hardware). Companies such as Dassault Systems are investigating the potential for lightweight models via products such as Virtools / 3Dvia for enabling online and desktop-based large-scaled interactive digital mock-ups. Open modelling formats such as X3D are now being used by large companies (e.g. EADS Innovation Works metadata management system using a 3D model-centric) in order to develop a collaborative platform to manage design data across distributed partners who use different platforms.

It is important to consider that with the development of SOAs and distributed control environments it is advantageous that design tools are vendor neutral so that they may be used to build and configure control systems in a heterogeneous runtime environment (i.e. interoperability between heterogeneous devices and over heterogeneous networks (wired and wireless)).

2.6 Modelling the Human in the Virtual Prototype

Since F.W Taylor began time study or work measurement, evaluating human behaviour has always been a well-debated topic, from the early stages of time and motion studies to today's complex ergonomic studies and VR simulations.

Ergonomics is defined as the scientific study of the human beings under work environment. It refers to the natural laws of the physical and psychological aspects of human beings under work situation with a view to have better compatibility and effectiveness [83].

Whilst stop-watch based evaluation of manual operations was widespread it is viewed as non predictive and anti motivational. From these studies has developed Predetermined Motion Time Systems (PMTS) which taken times from the analysis of work so that movements can be classified according to the nature of each movement and the conditions under which it is made. These data have been included into tables so that valued estimates of operation time may be calculated based upon the tasks to be carried out. The operation times calculated should allow the operator to repeatedly complete the task as required for the time set out by their shift. Frederick Taylor states "The greatest production results when each worker is given a definitive task to be performed in a definite time and a definite manner"

2.6.1 Predetermined Motion Time Systems

The principle of relating work to specific basic human actions such as reach and grasp was first published in the 1920's from the research carried out by F. Gilbreth. In the 1930's stopwatch based measurement of work was banned in the United States, so these methods became essential for planning and managing manual operations. From this Methods-Time Measurement MTM was devised, describing each action in tables that may be used to build up the human actions required to complete a task. These actions include Reach, Grasp, Move, Release, Walk, Bend, Read, Sit and Speak.

In order to decrease the time to develop MTM task information tables for different levels of detail have been created [85]:

1. Most detailed systems: MTM and Detailed Work-Factor were developed in the 1930s and contains 460 time values.

2. Second level systems: MTM-2 and Ready Work-Factor (abridged versions) achieved usually by the four methods of combining, statistically averaging, substituting and / or eliminating certain basic motions containing 30-50 time values.

3. Third level systems: MTM-3 and Abbreviated Work-Factor (even more abridged) "higher level" systems, usually times for complete activities.

The table below shows a simple MTM analysis chart for the assembly of an R.F. transformer to its base plate.

MTM Analysis					
	Job description:		**Analyst:**		**E J H**
	Assemble r.f. transformer to base-plate		**Date:**		**3 May**
El	**Description**	**LH**	**tmu's**	**RH**	**Description**
1	Move hand to washer	R14C	15.6	R14B	Move hand to transformer
2	Grasp first washer	G4B	9.1	G1A	Grasp transformer
3	Move hand clear of container	M2B	---	---	Hold in box
4	Palm washer	G2	5.6	---	Hold in box
5	To second washer	R2C	5.9	---	Hold in box
6	Grasp washer	G4B	9.1	---	Hold in box
7	Move washers to area	M10B	16.9	M14C	Transformer to plate

Table 3 MTM Chart Example [85]

The codes in the LH and RH column of Table 3 i.e. R14C is translated as "Reach 14 in. to an object jumbled with other objects in a group, so that search and select occur" (Class C reach). R14B is translated as "Reach 14 in. to a single object in location which may vary slightly from cycle to cycle." The tmu (time measurement unit is 1/100000 of an hour) [85].

Ford's Direct Labour Management System (DLMS) has been used and developed since the early 1990's for predicting and documenting labour times, generating the sequence of steps that a worker at the assembly plant must perform in order to accomplish this task and calculating the length of time that this task will require.

Resulting Work Instructions Generated by DLMS For Line 20
LOOSEN HEATER ASSEMBLY TURNSCREW USING POWER TOOL GRASP POWER TOOL (RT ANGLE NUTRUNNER) <01M4G1>
POSITION POWER TOOL (RT ANGLE NUTRUNNER) <01M4P2>
ACTIVATE POWER TOOL (RT ANGLE NUTRUNNER) <01M1P0>
REMOVE POWER TOOL (RT ANGLE NUTRUNNER) <01M4P0>
RELEASE POWER TOOL (RT ANGLE NUTRUNNER) <01M4P0>

Figure 9 Output from the DLMS System (SLANG) with associated MODAPTS Code [81]

The output of the DLMS system is a list of detailed work instructions that are required to implement to complete the tasks (see Figure 9). These work instructions, known as "allocatable elements", are associated with MODAPTS (MODular Arrangement of Predetermined Time Standards) (described below) codes that are used to calculate the time required to perform these actions [81][87].

Another output of DLMS research has been the acceptance and use of Standard Language to describe assembly instructions for the operators. "This has dramatically increased productivity by reducing ambiguity and confusion between our General Office engineers and the people at our assembly plants" [81]. The Standard LANGuage (SLANG) developed is intended to represent unambiguous and precise actions so that each sentence can only generate one unique set of work instructions.

Global Study and Allocation System (GSPAS) was also developed from DLMS in order to streamline the process of taking the process sheet data (output from DLMS) and checking using the GSPAS Artificial Intelligence / knowledge based system to ensure there are no ergonomic issues [87].

MODAPTS was developed by Mr G Chris Heyde, using the ideas behind MTM and it was introduced in 1966. It is used to estimate the required time to do an operation based upon a "fair days work" [86]. MODAPTS codes are widely utilized as a means of measuring the body movements that are required to perform a physical action and have been accepted as a valid work measurement system [81]. For example, the MODAPTS code for moving a small object with only a hand is M2, moving the arm gives a code of M3. These MODAPTS codes are combined to describe an entire sequence of actions to complete a task.

There are 3 types of MODAPT code

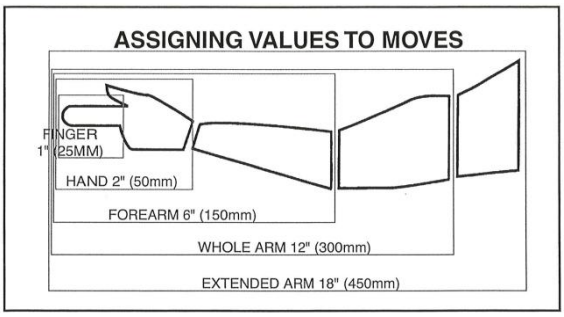
1. Movement – Movement through space by finger/ hand/ arm/ shoulder/ trunk.
2. Terminal – Activities done at end of move with close proximity to workpiece.
3. Auxiliary – All other actions such as Read, Write, Speak and Walk.

MODAPT Codes comprise two parts (i.e. a Letter and a Number). The letter refers to type of action as listed in Table 4.

Movement	Terminal	Auxiliary
M = Move	P = Put G = Get	R = Read J = Juggle X = Extra Force W = Walk F = Foot B = Bend S = Sit/ Stand C = Crank V = Vocalise U = Use E = Eye Control (e.g. Focus) H = Handwrite

Table 4 MODAPTS Action Classification and Codes

The MODAPTS number indicates the time in MODS to complete an action (where 1 MOD = 0.129s). Each actions code may have a number of standard actions associated with it, or overridden by the operator (e.g. A single step has a time of 5 MODS therefor the code is W5, or the time could be calculated using the calculation 7.75 MODS / metre). For the Movement classification the distance moved is an indicator as to which body part is being moved e.g. < 50 mm could be the Hand/ finger. Figure 10 shows the standard times associated with a movement.



Finger	1 MOD	Avg. distance 1” (25 mm)
Hand	2 MODs	Avg. distance 2” (50mm)
Forearm	3 MODs	Avg. distance 6” (150mm)
Whole arm	4 MODs	Avg. distance 12” (300mm)
Extended arm	5 MODs	Avg. distance 18” (450mm)
Trunk	7 MODs	Avg. distance 30” (750mm)

Figure 10 Definition of Modapts Code

The international MODAPTS association [122] have described the limitations of the MODAPTS:

1. When the output of a task is completely determined by the operation of the machine, then the MODAPTs analysis can only display the time that the operator is occupied within the machine lifecycle.

2. When the output of a task is determined by a combination of machine time and operator time then care must be taken to ensure the correct and timely interaction between the automation system and the operator is calculated.

2.7 Tools For Modelling the Human in the Virtual Prototype

There are many tools for completing ergonomic studies or simulation a human within an environment. This thesis is focused on the most used tools within the automotive industry.

SAMMIE CAD (System for Aiding Man-Machine Integration Environment) [89] is one of the earliest human modelling tools. It uses 18 "pin joints" connected by 21 rigid links with 23 Body segments to simulate the human motions. Each pin joint is constrained to emulate realistically the range of motion for a human and can be used to indicate if the joint movement is within the humans comfort range (see Figure 11).

SAMMIE CAD targets the ergonomic aspects of human modelling in great detail and has the ability to display up to 8 different human models with different anthropomorphic characteristics. As there are many ways to "reach" a target using the links described, SAMMIE-CAD has a fix posture facility to lock the models hip and shoulder joints in position so that a realistic position can be achieved more easily.

SAMMIE CAD has been used for many applications such as power station console layouts, visibility for underground trains and cockpit designs for cars / planes / helicopters and applications for evaluating access and use for disabled and older people.

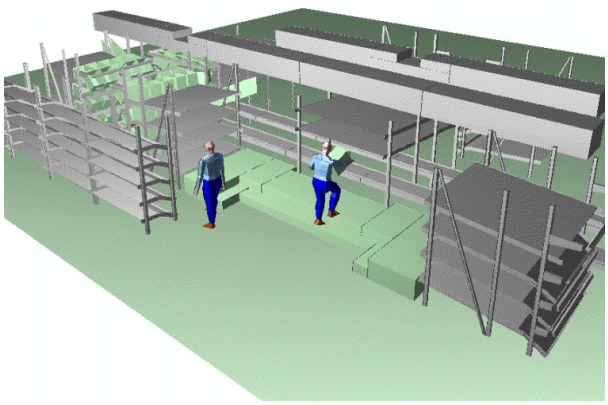


Figure 11 Industrial example of SAMMIE CAD[89]

Technomatix Process Simulate (TPS) and JACK: Originally developed at the University of Pennsylvania TPS has taken data from several sources for its definition e.g. joint limits from NASA data, body dimensions from 1988 anthropomorphic survey of U.S. Army Personnel (ANSUR). The JACK human model has 68 joints, 69 segments, 16 segment hands, 17 segment spine and coupled shoulder clavicle joints which makes it suitable for many applications, see Figure 12.

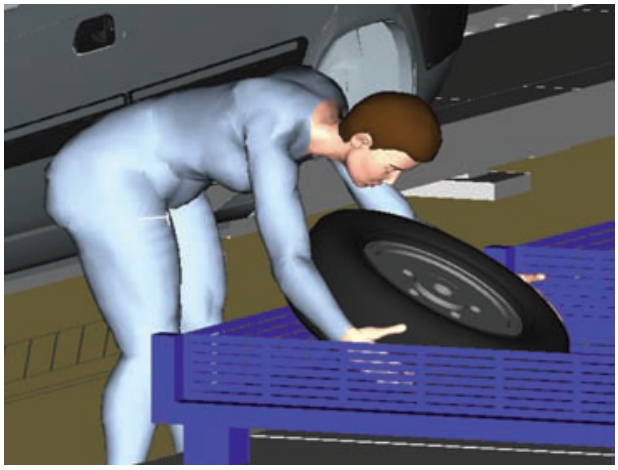


Figure 12 Example showing Siemens Tecnomatix Process Simulate Human tool [90]

Siemens have created a JACK plugin for the process simulate toolkit (called Process Simulate Human (PSH)), so that JACK may be integrated with machine behaviour providing a rich environment for in depth ergonomic studies. PSH allows the selection of different size, shape and gender of the virtual human (taken from National Health and Nutrition Examination Survey (NHANES)), and uses inverse kinematics to facilitate positioning of body parts as well as pre-set postures such as bend, squat and sit that allow the virtual human to be quickly positioned.

PSH also provides other ergonomic analysis tools such as National Institute for Occupational Safety and Health (NIOSH) lifting equation to access physical demands on the back and integrates Rapid Upper Limb Assessment (RULA) to assess the performance of the upper body. Using the University of Michigan's 3D Static Strength equation, strength requirements may be analysed. JACK also uses MTM-1 to analyze the time requirements of an operation and can directly output MTM codes from the simulation.

Dassault Systems, DELMIA Virtual Ergonomics suite provides a similar product to JACK as part of the Virtual Ergonomics (VE) suite. VE provides a "Human Builder" (HB) that allows the user to create "male and female standard manikins. (Name, Gender, 5th, 50th, 95th percentile)" [91]. The manikin has 99 independent links with articulated hand, spine, shoulder and neck, which accurately reproduce natural movement. Inverse kinematics for the definition of manikin motion are also provided.

An interesting feature of VE is the ability of the manikin to grasp and follow the path of an object automatically. This will allow simulation of the process whilst the automation system is moving e.g. on a constantly moving assembly line where the human operator may be inserting a bolt to the work piece as it passes though a station.

As with JACK, HB uses NIOSH for lifting analysis and RULA to assess the upper body movement. Snook and Ciriello Tables are also integrated into the tool to enable the analysis of the handling of weight and forces [92][93].

Figure 13 Santos Human example screenshot

SANTOS Human Inc [123] is a product from University of Iowa that has created a "physics-based and physiological response-based human modeling and simulation, toolkit with respect to task-based human performance assessment" (Figure 13). SANTOS models include dynamics as well as kinematics so it is possible to simulate the human response to outside actions (such as lifting a heavy weight from a shelf). The user interface allows for workspace evaluation, posture predictions and dynamic motion prediction. Using "Intelligent Kinematics" the user may position the virtual human so as to simulate task-based behaviour. Dynamic motion prediction enables the movement to be simulated based upon the physics and physiological response of the human to the situation (i.e. the model will move differently if carrying a heavy weight or not). The tool has been used in various applications in the automotive and aerospace industries as well as military applications as part of the Virtual Soldier Research (VSR) program. There are many biometric features that may be used to predict injury and the ability to calculate the total metabolic energy to complete a task.

Other Tools: There are a number of tools for analyzing humans within specific environments such as automobiles and aircraft including RAMSIS from Human Solutions (used by more than 70% of the automotive industry), COMBIMAN

(COMputerized Biomechanical MAN Model, a model for an aircraft pilot) and CREW CHIEF (CAD for an aircraft maintenance engineer).

In addition, several solutions have been created as add-ins for AutoCAD such as ANYBODY, Ergo Shape and HUMAN for workspace design and Mannequinne for 2D design layouts, but their use in automation simulation is currently limited.

2.8 Summary

The need for businesses to adopt a more flexible / agile approach has been discussed in this Chapter. The recent trends in terms of meeting the customers requirements by becoming more responsive and agile by knowledge capitalisation, teaming of core competencies, modularization and using new technologies has been reviewed.

The investigation into current trends in manufacturing automation systems has highlighted the investment into distributed control that is migrating towards service-oriented architectures. These approaches naturally lead to the development of modular / component based systems. Component based systems offer the ability to build automation systems either using centralised or distributed control. Using component-based systems, heterogeneous interactions and code reuse can be promoted. As increased autonomy is embedded within components progression from orchestrated systems to choreography based systems will occur.

In order to realize the benefits of component based systems, engineering tools are required to support their lifecycle from their design and development to deployment and maintenance. Virtual engineering / prototyping for automation has been reviewed with special focus on the tools and constructs suitable for modelling the human as part of the system.

Chapter 3 Requirements for an Component-based Engineering Toolkit

3.1 Introduction

In this chapter the focus of the thesis in terms of application domain and end user requirements for an engineering toolkit and to what extent these requirements address are discussed. Much of this research has been based on "real world" requirements to ensure the results are relevant to the automotive domain in the short, mid and long term.

3.2 Research Questions

What are the industrial drivers for this research?

1. What is the scope of the toolkit? (Domain)
2. What are the typical problems seen in the development of automation systems in the automotive domain, and how will this research address them? (Risk)
3. What are the typical workflows for automation systems in the automotive domain?
4. Who are the stakeholders (e.g. End User or Machine Builder) in the system lifecycle and what is their remit?
5. What are the problems with the current automation development process?
6. How will this research influence the automation system design lifecycle?

3.3 Domain

There are many application domains that the research outlined in this thesis could be appropriate for such as the chemical process, packaging and electronics industries. The focus of this research has specifically been limited to the automotive industry and in particular discrete part manufacture within power train (e.g. engine / gearbox manufacture and assembly). Addressing the set of requirements will enable the

development of a solution capable of resolving real issues, tailored to the domain specific problems in a prescriptive and structured manner, rather than a generic solution that may not completely address the domain issues.

Automotive production is one of the largest and most complex manufacturing sectors in the world [2, 9]. Whilst there is over capacity in the industry there are still large profits to be made by having the right mix of products in the marketplace at the right price and at the right quality.

The end-user must consider the supply chain associated with the domain since the solution will potentially change the relationship between the supply chain partners. The interaction between the partners will be discussed in the following sections.

3.4 Automation Domain - Risk.

3.4.1 Introduction to Risk in The Automotive Domain.

Automotive companies such as Ford invest a large amount of resources predicting market trends, and developing new products. As such there is a huge risk associated with the development of automation processes and systems to manufacture these products. Engineering tools are used/required to allow as much analysis and development to be achieved before actually building physical the infrastructure and machines. This section outlines the major risks associated with companies in the automotive domain and identifies how and where the engineering toolkit proposed in this thesis may address them.

In order to build and maintain a successful business, the risks associated with that business need to be understood and addressed. A significant policy for large businesses typically found in the power train area of the automotive domain is the reduction of risk whilst maintaining or increasing profit. If a company's exposure to risk can be minimised then profitability can be more accurately quantified and addressed. Risk can be defined as the combination of the probability of an event and its consequences [62]

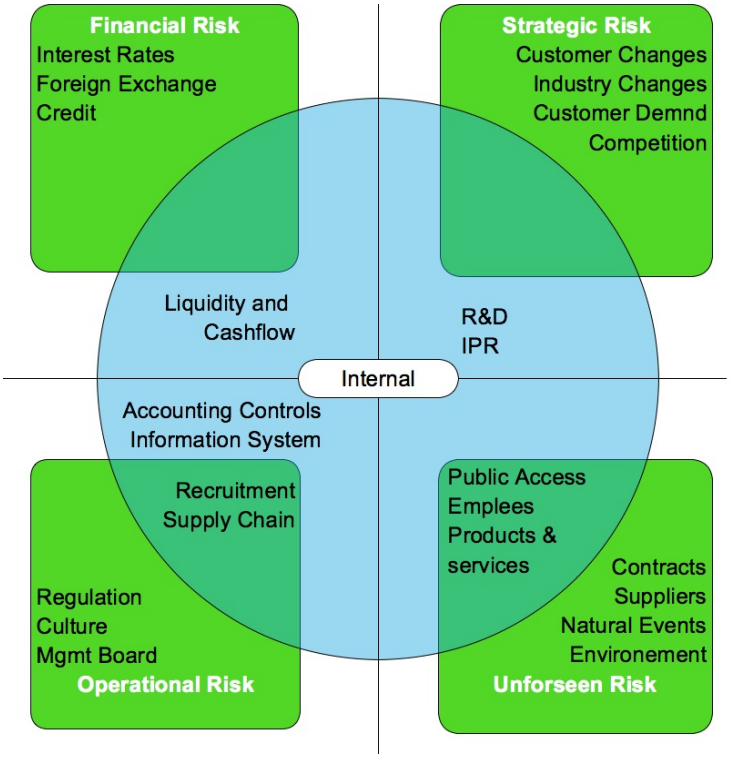


Figure 14 Risk associated with the enterprise after [61]

3.4.2 Types of Risk

A simple categorisation of risk as described by the Institute for Risk Management is illustrated in Figure 14. There is internal risk arising from decisions and changes within the business, and external risk where change is forced upon a company that may be beyond their influence. Some specific risks can have both external and internal drivers and are therefore shown in the overlapping areas (e.g. recruitment and employees).

Risk can be divided into four categories:

1. **Strategic Risks** tend to be longer term risks associated with the location of the business, availability of capital and interest on that capital. Externally this may be affected by changes in the market place and customers purchasing trends, competition in the market place and advances in the industry leading to new different, cheaper or superior products.

2. **Operational Risks** are associated with the day-to-day running of the organisation and represent short-term risk. Externally this may be the

management of the supply-chain. Internally operational risks can arise from the effectiveness of the accounting/ engineering management systems.

3. **Financial Risks** address the control of the finances of an organisation and the external factors such as exchange rates. Internally these risks may be concerned with the cash flow of the organisation.

4. **Unforeseen Risks** may relate internally to knowledge management developed in the company for both the product and the manufacture of the product. Externally this is affected by IPR abuse, competition or political changes (e.g. emission rate change or taxes). Internal factors relate to knowledge management systems and their use and loss or gain of key employees. Natural events such as power shortage, fire or flood could be other examples.

The automotive industry manages risk in many ways. For example manufacturing globally reduces the single political / social risks but increases the foreign exchange and transportation risks. The results of the work detailed in this thesis will impact the four main areas of risk. Strategic risk will be addressed by providing a more flexible and potentially agile system in order to react quickly to changes in the market place. Operational risk will be addressed by making the automation system more open and accessible to the supply chain and provide a mechanism for communicating concepts, requirements, problems and solutions. Financial risk will be addressed by allowing the development and testing of automation solutions more quickly so investment may be more focused and lead times can be reduced to increase cash flow. Unforeseen risk is difficult to quantify, however, the reduction in the overall project time and in particularly the reduction in the building and commissioning times will reduce the exposure to changes both internally and externally.

In this thesis, focus will be on the management of strategic and operational risk, as well as responding effectively to unforeseen risk. Figure 15 shows a simplified graph of the breakdown of expenditure against time during the release of a new engine in Ford motor company. (Data taken from work carried out in Loughborough University by Dr R. Monfared, Dr M. Ong and Dr I. Haq.)

Figure 15 displays the cost associated with development of the automation system (after the initial product design is complete). Initially there is a large investment in

the product, with design for manufacturing, creation of the specifications required to build the product and investment in evaluating the competing strategies of the engineering suppliers (in this case, machine builders). Subsequently the cost of the process development (the design and layout of the assembly process for the engine) is incurred. Process development includes:

1. Mechanical Design
2. Facilities Design (e.g. hydraulic, electrical and pneumatic)
3. Logistics Design (e.g. material flow)
4. Process Design
5. Productivity (human interaction with the process e.g. operator assembly)
6. Health and Safety
7. Plant layout / Building design

Costs peak during commissioning. The final remaining cost is unexpected costs.

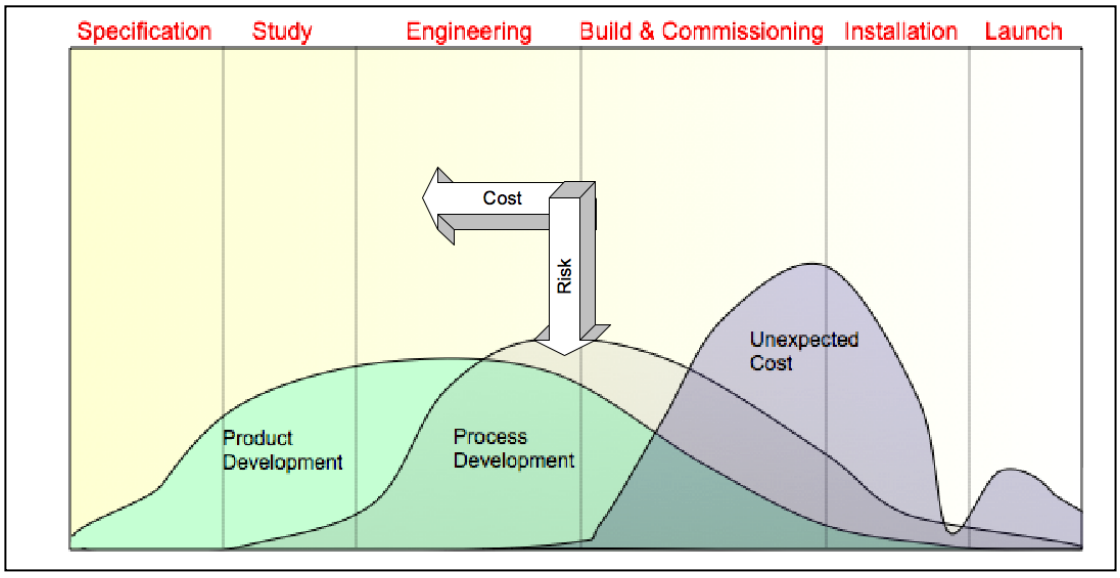


Figure 15 Cost v/ Risk for a power train manufacturer (M.Ong)

Unexpected changes can be attributed to the following [63]:

1. Market pressure can sometimes result in immature products being built, which inevitably leads to late, unexpected product and process changes.

2. Verification of the automation systems is still only achieved when both the hardware and software systems are assembled. Late changes have been observed during the building of an automation system for a recent engine program. In this case two weeks before commissioning the control code was still unavailable from the machine builder.

3. Delays in the product and process design mean verification and commissioning times are squeezed. This increases the risk of integration problems causing increased cost and delays in the delivery of the automation system.

4. There is very little reuse of software / hardware from program to program. Lessons will potentially have to be relearned.

5. Missing parts at the late stages of the automation system build often means compromises are made to "tweak" the machine so it can run without them. These parts are often integrated during commissioning or even on "ramp-up" (when the automation system cycle time is being increased to meet the required production rate)

3.4.3 Modified V Development Lifecycle

The lifecycle model used within Ford is a modification of the V Development lifecycle [65] (Diagram courtesy of Wikipedia [66]) illustrated in Figure 16.

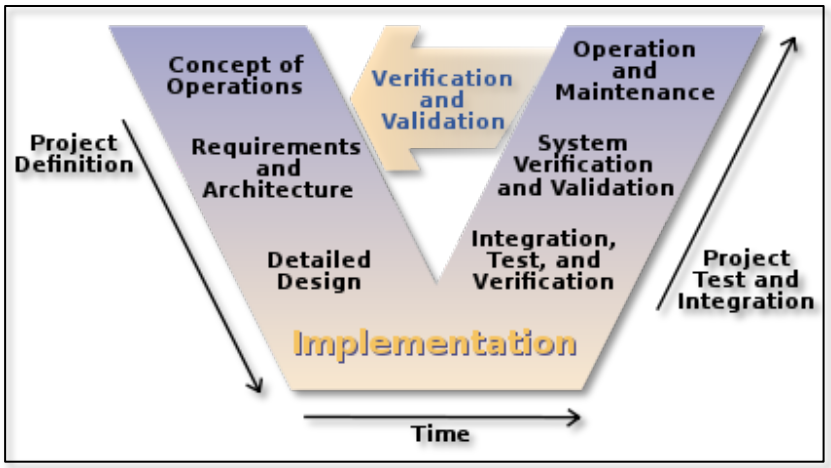


Figure 16 Generic V Model Diagram (for Software)

The lifecycle model shown in Figure 16 is can be applied to users in many domains. It is composed of two primary phases firstly related to the planning and definition of the project followed by the realisation of these plans in building testing and integration. This diagram does not directly show the changes to the Detail Design, Requirements and the Concept that will obviously occur during project test and integration.

The use of the V model does indicate the difficulty in changing the *Concept* at the *Operation and Maintenance* phases of the lifecycle as these two items are widely separated and any change in the *Concept* may require alterations in the *Requirements*, *Detailed Design*, *Integration*, and *System Verification* in order for the change to be correctly implemented. Conversely the impact changes to the *Detailed Design* during the *Implementation*, *Integration, Test and Validation* is relatively simple (as indicated by their proximity). The panacea for reducing the risk of change later in the lifecycle, is to narrow the gap between the different phases (i.e. reduce the time/ effort required to facilitate the change).

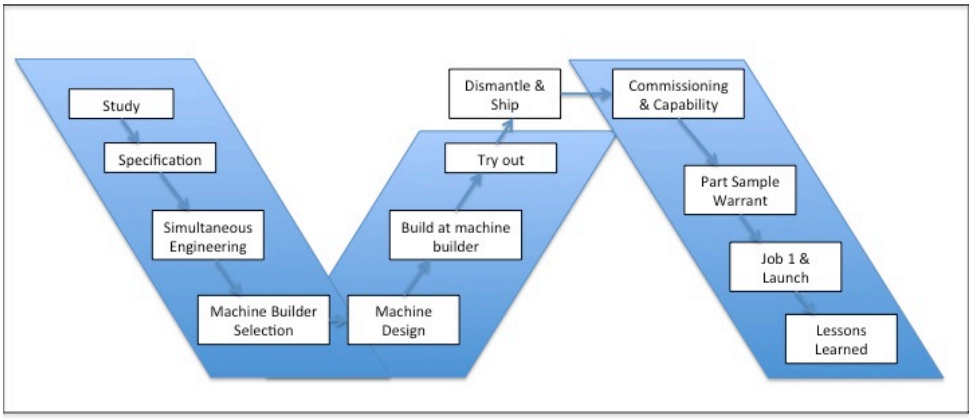


Figure 17 Modified V Diagram (Used by FORD)

The modified V diagram used by Ford to describe their engine assembly line programs is illustrated in Figure 17. The information described is similar to the traditional V-model with the exception that it contains an additional "arm" detailing the final commissioning process. The reason for this is once the machine has been assessed during the *Try out* phase the design should be effectively "locked" with only minor changes being approved to improve quality (through *Part Sample Warrant*) and performance (*Job 1 & Launch*). The final task is to document the *Lessons Learned* for the project to feed back to the next project.

3.5 Automation system development phases

Table 1 describes the development process from concept to realisation identifying the key phases of the automation lifecycle.

Phase	Description	Area
Study	Strategic Intent; Objective setting; Create Short List of Suppliers	End User
Specification	Selection of control components; External specifications for programs; legislative and internal specs; Program approval.	End User Machine Builder Controls Vendor Board of Directors
Simultaneous Engineering	Team formation; Communication of targets; Breakdown of machine aspects –software hardware and network architecture	End User Machine builder Controls Vendor
Site Selection	Decision of plant location	End User
Vendor Selection	Commitment to use particular partners for machines and automation etc.	End user
Machine Design	Detail design to suit the vendors implementation	End user Machine builder Controls vendors Tooling
Build at vendor	Build and run at OEM	End user Machine builder Controls vendor Tooling
Try Out	Test machine and prove quality and functionality	End user Machine builder
Dismantle and ship	Move the line to the end users plant	Machine builder
Commissioning	Install in plant and OEM produce first part for analysis	End user Machine Builder Controls Vendor
Part sample warrant (PSW)	Product tested with emphasis on quality using accelerated lifecycle tests. Once [assed the line is further optimized to reach full production rates and these parts are tested to ensure quality.	End user Machine builder Controls vendor
Job 1 and launch	Start production ramp up and improve system reliability	End user Machine builder
Lessons Learned	Detail the lessons learned and report for future	End user Machine builder Control vendor

Table 5 Development phases of a typical automation system development.

In order to capture the user activity requirements and, from these, their application engineering needs, the activities for the actors involved from each organisation are described below:

3.5.1 Study

In this phase, the end user defines the strategic intent of the program, sets objectives, defines the outsourcing strategies and creates a supplier shortlist.

3.5.2 Specification

The end user as well as the control and process engineers define the internal and external specifications and legislative framework to be used on the programme. Work plan and milestones are specified.

At this phase, the controls vendors must specify the appropriate control technology and ensure that the necessary control components are available in line with the work plan and the milestones.

3.5.3 Simultaneous Engineering

An engineering team is formed by the end user. Representatives from the machine builders and controls vendors are invited to join this team. The end user is responsible for the definition and communication of targets and the development of a detailed manufacturing process description.

The machine builder is required to contribute a breakdown of machine related aspects. This includes solutions for the combination of handling, assembly and fitting operations to one or more line sections (plots). The assembly process for the product is decomposed into manufacturing steps able to meet expected cycle times related to target production volumes of sub assemblies and required product mix. The decomposition process takes into consideration:

1. Manufacturing process user requirements
2. System engineering.
3. Lifecycle support.

The controls vendor is required to define suitable software, hardware and network architectures to meet the end user and machine builder requirements.

3.5.4 Site Selection

The end user defines the final location of the new installation inside the plant. Special consideration will be given to transport and material flow aspects and existing production facilities, which perhaps need to be integrated into the new production automation concept.

3.5.5 Vendor selection

The end user makes a final selection of suppliers. The selected vendor will then take over the line building and machine and the control engineering part of the programme.

3.5.6 Machine and Line Design

The machine builder starts with design activities. They are performed sequentially beginning with mechanical engineering followed by electrical and control system design. To reduce the time for these activities opportunties to reuse previous designs are considered. CAD tools and a virtual engineering environment are required to support these activities.

At this phase, the overall production line configurations are finalised and detailed mechanical design is carried out by the mechanical engineers at the machine builders. It is important to ensure the system design includes focus on modularity, fast re-configurability, and scalability of the system. Human interaction aspects, home positioning of the automation in case of failure or configuration alteration during automatic run are also considered.

Based on the definition of the mechanical requirements, the electrical systems are specified by the electrical engineers at machine builders at this phase.

The control system needs to be designed for modularity, re-configurability, and scalability of the machinery components of the line by the controls engineers at the machine builders.

Control engineering activities have to be carried out with special emphasis on resulting hardware and software requirements. The control system software development has to be defined with regards to system requirements and user standards in terms of standardised structures, modularity, controller interfaces, data formats, data transfer lines, programming, user interfaces, monitoring demands, and error recovery procedures.

The controls vendor must define the control system components in detail. Controls vendors also have to cope with the integration of robot, sensor and process technology into the overall control concept. This will include the final specification and selection of control system components, process and control strategies, networks, data base structures with related import / export procedures, sensing functions and user interfaces in line with user requirements.

The end user is responsible for monitoring and coordinating the design process. He has to check for conformance with the specification and appropriate external and internal standards.

3.5.7 Machine-Build at Vendor Site

Mechanical, electrical, control and commissioning engineers at the machine builders are responsible for building and commissioning the machine at the vendor's site before strip down and installation at the end users site. The functionality of machine sections in the final production line is tested individually in order to reduce costs and save set-up and ramp-up time at the end user plant.

The controls vendor is required to support hardware and software development, including testing and diagnostics. Engineering efforts are focussed on material flow control aspects, sensing functions, quality control, process monitoring, data transfer,

CAD link, robot program generation, and error handling. Virtual engineering of the control system behaviour would offer the potential to make substantial savings in this process, particularly if this could be integrated with the machine build activities thus enabling a complete virtual machine build.

3.5.8 Try-out

End user process and control engineers test all machines sections at the machine builder's site with parts provided from the product suppliers in order to prove both production rate and production quality.

3.5.9 Dismantle and ship

During this phase, the machine builder is responsible for the dismantling, shipping and configuration of the production line at the selected end user production facility.

3.5.10 Built at end of user site

The entire line constructed, installed and commissioned at the end user site by machine builder commissioning engineers. Functional tests are performed to check the proper operation of line sections. The line has to be integrated into the surrounding production facilities. This includes activities like implementation of data transfer from the CAD system, safety engineering, engineering of material flow and transportation requirements, adjustments and calibration.

The controls vendor integrates the control hardware and implements the control software. The interaction of control systems, sensors and actuators installed has to be tested to determine whether they fulfil the expected control functionality. Interoperability checks related to the data exchange between control software and technology database are also undertaken. The user interface is subsequently configured. Default values to run specified functionality are selected during the test phase.

3.5.11 Testing and Commissioning

Commissioning engineers at the machine builder are responsible for producing the first sub assemblies in accordance to required tolerance bands during the ramp-up of

the line with the produced parts are assessed for quality by the end user product and process engineers.

End user plant engineers are responsible for the provision and arrangement of plant services and enterprise connectivity e.g. IT, electrical, hydraulic, and coolant systems, welding consumables, as required. Controls vendor support engineers are required to ensure successful process control at the user site.

3.5.12 Part Sample Warrant (PSW)

At this phase, while product quality testing is carried out, the line undergoes further testing and optimisation by the machine builder's and controls vendors. For example, tests will be carried out running the line at full-specified production capability, i.e. full production rate machine speeds and feeds. Failure situations will be simulated, for example, weld restart or manual interaction and emergency stop.

If successful the production manager gives green light to run the full installation at production conditions.

3.5.13 Test production and Launch

At this phase, the system requirements are validated against the real production line. These include line monitoring, validating the production rate and individual machine monitoring to ensure that system reliability targets are being met. All of the supply chain partners are involved. The initially trained operators will now provide "on the job training" to other operators.

3.5.14 Lessons Learned

After sales service with a number of scheduled meetings will be used to learn from the activity and to identify where future project improvements can be made. From interviews with key personnel it was interesting to note that this is described as "Lesson Documented" rather than learned. This quite often attributed to the fact that during this phase of the project the next project (usually under the management of another group) is well under way and due to the long lifecycle of the project key personnel often change and in many cases the following program often will repeat the errors.

3.6 Stakeholders

It is important to appreciate that many stakeholders share this machine lifecycle across several organisations in the supply chain. Each stakeholder will have their own product lifecycle, operating procedures and management structure, but without a common understanding, approach or view of the Manufacturing Customers' (End User e.g. Ford) product there are inevitable misconceptions in the design and operational requirements of the machine.

Figure 18 The use of the CCE tools by the automation system stakeholders

Therefore the Engineering toolkit (Core Component Editor (CCE)) tools as developed in this thesis must provide a level of support / visualisation to all the stakeholders (see Figure 16). The use of these tools will therefore affect the relationship between the *Manufacturing Customers'* internal departments such as mechanical and controls design as well as the relationship between partners in the supply chain. A brief description of the typical users of the system from each of the stakeholders is given in Table 6. It is envisaged that *Component Suppliers* will create automation components directly using the CCE environment for integration into automation systems use by the other stakeholders.

Manufacturing Customer (End User) Stakeholders	
Simultaneous Engineering team	Consisting of CAD Specialist, production automation manager, process engineers, productivity engineers and mechanical engineers.
CAD Specialist	Responsible for digitally modelling the components of the product
Process Engineer	Responsible for the design of the manufacturing process.
Product Automation Manager	Responsible for the coordination of the design activities and the site
Productivity Engineers	Responsible for the design of manual aspects of the manufacturing process
Machine Builder Stakeholders	
Mechanical Engineering	Responsible for the engineering of the automation systems mechanical structure
Electrical Engineering	Responsible for the electrical systems, wiring and networking requirements
Control Engineering	Responsible for the implementation of the control system and HMI.
Automation Manager	Responsible for the coordination of work at the machine builders, ensuring the system complies to the end user specification and commissioning of the system.
Controls Vendor Stakeholders	
Controls Engineers	Responsible for the development of the control system tool, technical support and system interoperability.
Component Supplier Stakeholders	
None	Currently all integration is carried out by the Machine builder. In future these may include Mechanical, Electrical and Controls Engineering.

Table 6 Stakeholders for organisation in the supply chain

There has been much research in this area by the Loughborough team (e.g. Harrison[40,43,60], Monfarad[25], Haq [105] et al). This is only briefly discussed in this thesis, as the main focus will be the design of the Machine / Process control and the lifecycle support of the machine (see Figure 19). The complete development of a typical powertrain line from product release (Program Start) to Job 1 the release,

and the points of interaction with the end-user and the machine builder is illustrated in Figure 19. Typical observed errors, with their categorisations, have been loosely linked to the machine build process, indicating the types of problems that can occur.

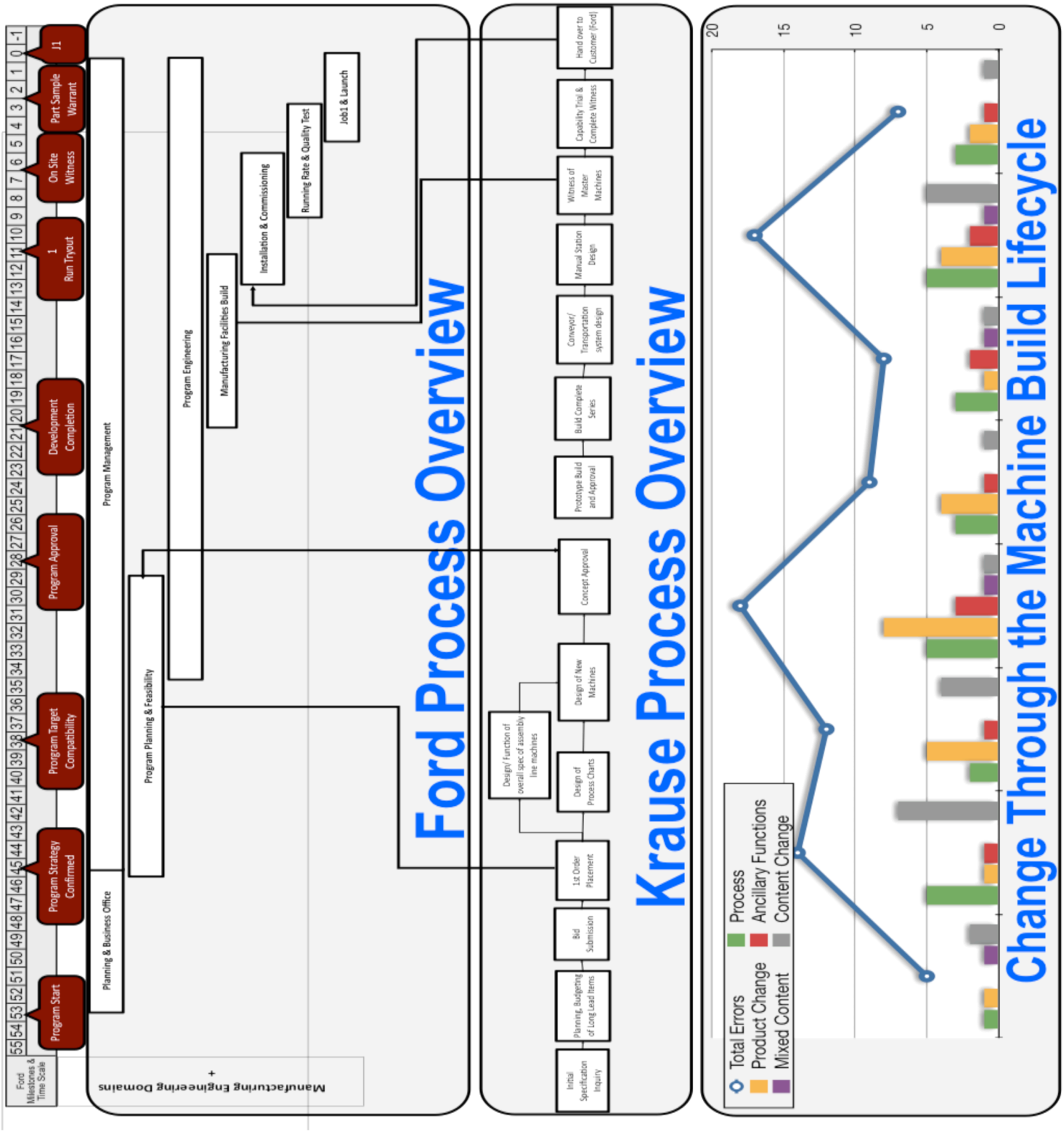


Figure 19 Workflow Integration between the End User and the Machine Builder- showing the lifecycle of a typical end user and machine builder with the breakdown of changes to the automation system during the machine building and commissioning.

The end user and multiple machine builders work in parallel from program start to create the strategy and specification of the automation line (including the transport and pallet handling system). Once these have been agreed and the machine builder(s) have been selected, the design of the machines is created by the machine builder with some guidance from the end user. Once the *Program* has been approved, detailed design of the stations, plant layout and transport system are developed and the machines are developed at the machine builder's site. Finally the machines are tested on site (*witness of machine operation*), before they are dismantled and shipped to the End User plant for installation, commissioning, ramp up and handover.

The communication of concepts and ideas from program start to program approval is mainly through a mix of 2D / 3D CAD (some new, some reused from pre-existing lines). This can cause misconceptions on both sides from translation errors and incorrect assumptions. The problems are exacerbated as the design and manufacture is completed on a global scale with the product design, process design, and machine design often being done in different countries or even continents.

Figure 19 also illustrates the tracking of the 5 top types of changes (after [63,106]) that occurred during a particular machine design and build project. As can be seen, the product, when released for manufacture, is relatively immature as highlighted by the number of product changes. It is expected that the incidence of late product changes are liable to increase with the current global market competition and over production in the industry. These changes not only affect the machine builder, but also impact to the end user as well, both in terms of late delivery to the market of the product and associated loss of revenue. Other changes include content change (i.e. adding or removing content required to be assembled by the process), process changes (i.e. how the product would be assembled or handled) and ancillary functions (e.g. parts delivery and storage). The total number of changes reduces around the program approval phase, but there is a marked increase in changes as the system build heads towards commissioning and try out. It is feasible that having greater amount of communication and visualisation of ideas and solutions could make a marked reduction in process changes. However, this increased understanding must be available from concept through to Job 1 to be effective.

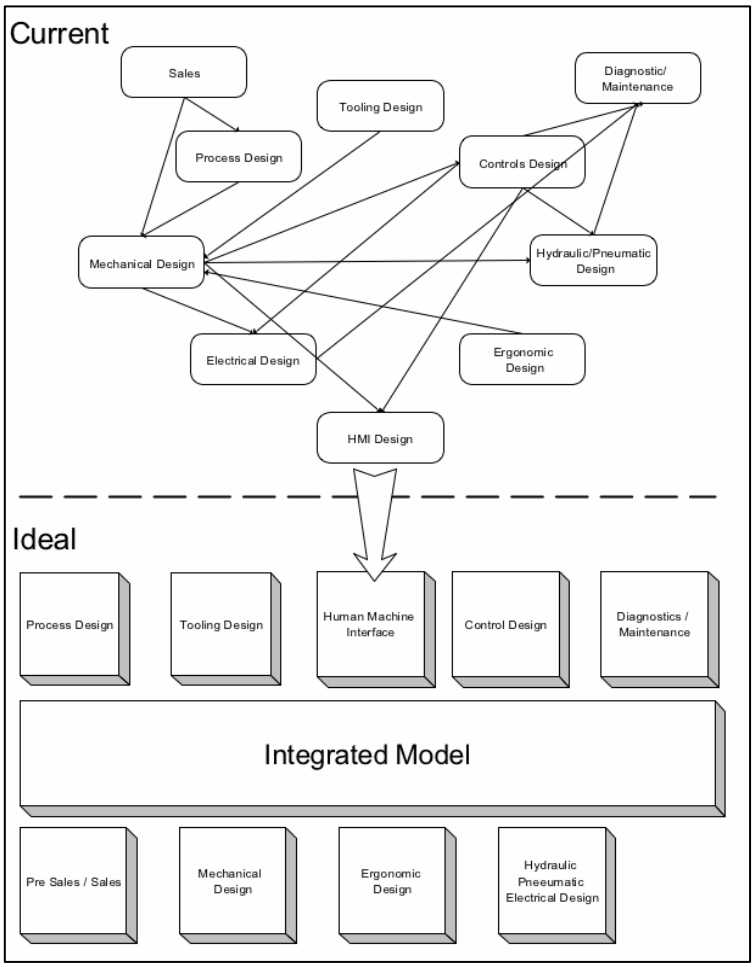


Figure 20 Reduce Errors through Communication Simplification

The main problem with the development process is communication. Figure 20 displays how the current development system has many disparate functions and departments each with their own systems, such as 3D CAD used by the mechanical design, Tooling programs, Process diagrams and combination charts used by productivity to describe human machine interaction. Conversion from one format to another may not necessarily cause a problem until (if undertaken in a structured well-defined manner) change is introduced during the process (illustrated in Figure 19). Unless the process is automatic there is always the possibility of things being missed or not implemented correctly.

The end user is also required to manage multiple systems from a variety of suppliers, to facilitate this it is important that they can use a common toolset. If the tools are not common, then systems integration and maintenance issues will be a continual problem during the working life of the automation system.

The nirvana for the development process would be for the information to be built into an integrated *common model* that all the departments use. This model would propagate changes as they occur to the "views" of the *common model* highlighting when and what work is required to incorporate the change in the model, thus ensuring communication and translation errors are kept to a minimum. The main barrier to this approach is that no one solution is capable of satisfying the needs of each of the groups. Each group uses their chosen "best in class" solution to satisfy the needs of their role in the most reliable and efficient manner. Even if there was a concept-to-implementation solution available then it is likely to be seen as a compromise in some functionality. Typical issues associated with CAD systems used throughout the design and build processes are; that expert users are required to operate them and with the large models (in terms of size and complexity) that have to be produced, high performance hardware is required. This restricts the potential benefits of the models, as their ability to be used throughout the machine lifecycle is limited.

Any potential solution would impact the business in many ways. These include how the *internal* departments are formed to support the design of the automation systems and *externally* how it will affect the supply chain in that they will be required to use the same models to gain the benefit of sharing information.

Organisational Impact	
Area	**Impact**
Machine / Process Control	Creation of Automation Systems
Lifecycle Support	Sales/ Desgin/ Manufacturing/ Comissioning/ Operation Reporting/ Maintenance
Enterprise integration	ERP MES Integration
Supply Chain Interaction	Supply of automation components & modular machines.

Part of Research

Table 7 Organisational Impact

The highlighted cells in Table 7 shows that the focus of this thesis will be on Machine / Process Control and Lifecycle Support. Any solution has to satisfy many of the issues highlighted with the current process such as early visualisation and agreement

of the machines / process, communication problems due to translations from different formats and languages and disciplines [105][106].

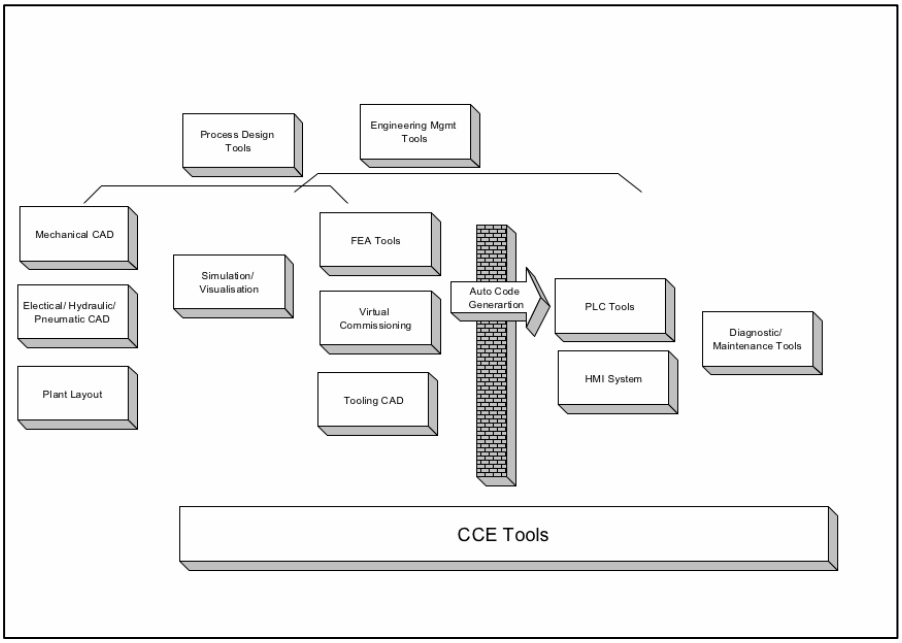


Figure 21 Lifecycle impact of the CCE tools

Figure 21 shows the area of impact of the CCE tools. It should be noted that the toolkit is not aimed at replacing CAD systems, instead it offers complimentary functionality to extend the CAD models availability to systems and processes later in the machine lifecycle, thus gaining more "value" from the models.

The tools should input the CAD models and attach functionality and information to support the design phases by offering simulation, validation and enable potential problems that may require further investigation to be highlighted. Further to this, the CCE tools should support the generation of e.g. PLC code (specific to the target platforms requirements (e.g. Siemens or Schneider)), and provide diagnostics and reporting tools to enable the machine behaviour to be analysed so that performance information can be fed back into the design to make the components/ modules of the system more robust. This in turn will offer the ability to change the machines behaviour and be "confident" that the changes made in the tools will be reflected in the machines behaviour.

3.7 Requirements List

Based on the interviews with the stakeholders and on the business process analysis set out in this chapter, a set of desired attributes for automotive manufacturing systems have been identified in Table 8, along with the section of thesis where the requirement is addressed in the thesis. (NB: 4.x 5.x referrers to all subsections of

No.	Description	Where Addressed
1	To be able to respond and react quickly and accommodate changes throughout the lifecycle.	4.3, 4.4.3, 44..7
2	A high degree of reuse is desired throughout the lifecycle of machine design and build. This includes reuse of automation hardware, control software, engineering knowledge and best practices acquired from previous projects.	4.4.5
3	A common data representation is required to support the various phases of the lifecycle. A consistent data representation would help to reduce repetitive work of interpretation and translation of design specification. This common representation seems all the more important when the partners in the projects have differences in (i) geographical location, (ii) levels of experience and understanding, and (iii) cultural and language backgrounds.	4.3
4	Support integration of a simulated human with the control system.	4.4.9
5	Use simulated human to provide accurate timing of manual and semi automatic station.	4.4.10
6	Provide Modapts/MTM output sheet to describe the behaviour of the simulated human.	4.4.9.2
7	Visualisation of system behaviour through modelling and simulation prior to installation is desired. This would enable the control engineer to validate the system before the physical assembly.	4.4.10
8	Support for virtual engineering, system try-out and commissioning.	4.4.10, 4.4.8
9	There is also a need for an approach that will enable the machine design and the associated control behaviour to be available to all interested parties throughout the lifecycle. It has to be in a format in which the users can easily relate to.	5.2.5
10	A more integrated support for the system diagnostics and maintenance is required.	4.4.10, 4.4.11, 5.1.8
11	Business process and functional benefits of innovations, new technical architectures and approaches need to be appreciated readily by non-technical managers to ensure commitment, uptake and investment are achieved.	Not addressed
12	Ability to (re) configure machines built from reusable modules. Maximise reuse of machine design. In order to maximise manufacturing agility at minimum time and cost, it is vitally important to be able to reconfigure production machinery easily and quickly.	4.x 5.x
13	Any system should be easily integrated with higher level enterprise systems.	4.4.11
14	During the lifecycle of the machine the models created shall be visible and supported by the supply chain partners.	5.2
15	Provision for integrated production monitoring, for process management.	4.1
16	Any solution should be vendor neutral encouraging open systems, only specialising in where required (e.g. control hardware)	4.x
17	High level machine configuration capability	4.x
18	High level process description	4.x
19	Plant layout support	5.2.5
20	Capture “Lessons Learned”	5.2.5
21	Inherent compliance with standards	4.x
22	Lifecycle support from engineering tools	4.x
23	Support for globally distributed engineering teams	5.2.x

Table 8 List of user requirements as obtained from research and a typical end user

3.8 New Knowledge

New knowledge gained from the analysis of two key players in the domain has shown:

- There is still a significant gap in understanding between the stakeholders in the supply chain and using 3D simulation models shared between them will significantly reduce this gap. This will rule out many CAD tools currently available as they are too complex, costly and require high specification computers to be effective.

- During the design phase the biggest challenge faced was with semi-automatic machines. Semi-automatic operation determines the interaction between a human operator and the machine, especially when there are several points in the cycle where the machine waits for the operator or the operator waits for the machine. This can cause significant under or over utilisation of the operator or excessively long cycle times. Early visualisation using the tools developed in this thesis has allowed this operational mode to be optimised at the early stages in development (see Chapters 5 and 6). With the additional benefits coming from automatic job description and training aids for the operator, the potential impact of the research is high and the benefits can be readily obtained.

There are several CAD systems that allow the building of component-based control systems as discussed in Chapter 2, as well as CAD systems that offer human simulations . However, (at the time of writing) there are currently no known systems that integrate component-based automation system operation that are directly linked with the behaviour of a virtual human.

This integration of automation system operation and human operation, with detailed human operation times based upon MODAPTS, will ensure that accurate cycle times for semi automatic machines may be predicted.

3.9 Summary

This chapter has outlined the requirements of a component-based engineering toolkit to support the machine design and build process and lifecycle support within the automotive industry. It details a typical process for the development of an automation system from concept to "Job 1" and highlights the areas where major improvements in terms of time and quality can be achieved.

There is a requirement to compress the time for machine build and related activities wherever possible through concurrency and the reuse of previous designs and physical components. The use and reuse of tried and tested components can significantly reduce the risk associated with the design process also concurrently providing a common understanding of the machine behaviour between engineering departments and even companies is of equal importance for the reduction of risk.

Open, vendor neutral systems in simulation are advantageous, particularly to the end-user and machine builders. The use of an open system will have significant impact through all machine lifecycle phases, as it is not tied to a specific vendor until it is really necessary. Also by being vendor neutral the potential benefits could significantly reduce costs, training requirements and interfacing problems.

Any potential solution will be required to support all aspects of the machine lifecycle, from a high-level process engineering capability, information for site layout and logistics planning to the provision of the creation of low-level control for output to the PLC. In addition to the requirements laid out in this chapter it would also be advantageous for any solution to be integrated with the end users product lifecycle management (PLM) system, and any solution should inherently support and enforce compliance of standards (such as IEC 1131 for diagramming, safety and quality).

It should also be noted that from an end user perspective, current automation systems are often too complex and too general. It is often quoted in industry that current engineering simulations are too slow to deliver and too costly to be effectively used in the automation system development and just provide simple visualisations. The ability to accept CAD models from any source and build lightweight models that are available for use outside of the CAD simulation are important goals.

An effective virtual engineering environment is required to support such reuse activities effectively, whilst allowing the capture of lessons learned from program to program. The aim is to reduce risk, development time, inconsistency and cost (chapter 4).

Specifically this chapter's objective was to answer the research questions:

1. *What is the scope of the toolkit i.e. which domains are and industries will be covered?*

   The scope of the toolkit is the automotive domain in particular discrete part manufacture and the powertrain division.

2. *What are the typical problems seen in the chosen domain, and how will this research address them?*

   Described in this chapter are the challenges faced regarding lack of reuse and lifecycle support, information exchange between tools and departments / engineering disciplines and the lack of a common understanding of the problem. The CCE tools offer a solution that allows models to be developed through the lifecycle of the automation system (from design to maintenance), that is component-based (aiding reuse) and provides a common view of the machine to all the stakeholders (reducing information exchange problems and increasing the common understanding).

3. *What are the typical phases of the lifecycle and workflow within the automation design in the domain?*

   This chapter has outlined the main phases of the machine lifecycle and highlights the workflow between the stakeholders (see Figure 19), and how the CCE tools can be integrated into the current workflow of automation system design.

4. *Who are the stakeholders in this lifecycle, and what is their remit (e.g. End User or Machine Builder)?*

The stakeholders have been identified as the manufacturing customer (End User), the machine builder, component supplier and the controls vendor. The End User is responsible for specifying the automation system requirements and managing the overall process, the machine builder is responsible for the machine development and documentation, the controls vendors is responsible for supplying standards based tools to permit the development of the automation system control code and the component supplier is responsible for providing automation components to the machine builder.

5. *What are the problems with the current development process?*

   Although a lot of emphasis is placed on concurrent engineering, current development processes still resemble a waterfall style development lifecycle with each stakeholder discipline feeding the next with limited feedback between lifecycle phases. There are many areas where there is translation of information into different formats where errors can be made and changes to the product and process become difficult to implement. Moreover there is a lack of a common understanding of the complete machine design between the stakeholders and supply chain that leads to interpretation and communication errors.

6. *How will the research impact the automation system design lifecycle?*

   It is envisaged that the outcome of this research will significantly impact the simultaneous engineering process, allowing multidisciplinary teams of engineers to examine and resolve issues more effectively by having a common understanding / view of the automation system. Also by integrating the human into the automation system design, the current problems with predicting operator loading and cycle times in semi automatic machines may be reduced significantly. Finally, documenting the machine behaviour (and lessons learned) in an easy to understand manner, will aid the development of new automation lines by using existing lines as a reusable template, as well as supporting the rapid reconfiguration of existing systems.

Chapter 4 Component-based Engineering Toolkit Design

4.1 Objectives Research and Research Questions

The main objective of this chapter is to outline the design philosophy behind the design tools that enable them to meet the system requirements (chapter 3), and show how the tools are built to fulfil each specific need.

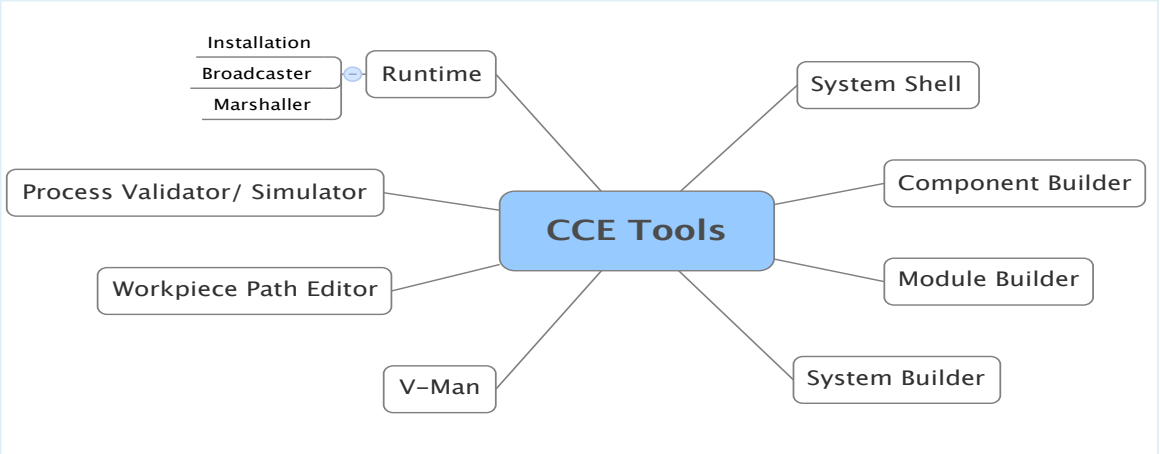


Figure 22 CCE Tools Functional Map showing the major functional aspects of the CCE tools

The CCE Tools have been developed from functional modules, starting with the *System Shell* that binds the user modules, providing database and user access (see Figure 22). The *Component Builder* facilitates the building of reusable components that are catalogued and placed into libraries. The *Module and System Builders* allow these components to be integrated into sub systems and systems accordingly. In addition to this there is the *V-Man* module that integrates human behaviour into the systems, and the *Workpiece Path Editor* and *Process Validator / Simulator* that enable the sensor inputs that will be triggered as a workpiece (i.e. product) is manufactured by the system to be defined and are vital to enable the behaviour of the system to be simulated and validated.

Once the desired machine behaviour has been defined, the system is downloaded onto the runtime platform (such as a PLC or distributed control platform for example one based on Web Services) via the *Runtime* component. The *Installation* module provides the component and input / output sensor and actuator mappings, the

Broadcaster is used to distribute the system state to support e.g. remote diagnostics and web based Human Machine Interface (HMI) panels, and the *Marshaller* provides secure / controlled access to control the machine or request information.

Each of these modules is discussed in detail in this chapter, along with the design philosophy and implementation. The overall aim of the chapter is to show how the development and utilisation of the CCE Toolkit may address the following research questions:

4.1.1 Research Question 1.

Is it possible to support a multidiscipline team working to develop an automation system whilst providing a common understanding of the machine layout, performance and operation?

4.1.2 Research Question 2.

How the toolkit can support the development of "reusable components" that are non-vendor specific in terms of hardware and software?

4.1.3 Research Question 3.

How can human behaviour be integrated with machine behaviour, without adding so much complexity to the system that it becomes either specialist or unwieldy?

4.2 Introduction

Since the mid 90's Loughborough University's MSI Research Institute has been researching modular / component-based distributed control systems. The author has been involved in the specification and development of the toolkit from its inception for the development of distributed control within the automation industry. The project **COMP**onent based architecture for **Ag**ile manufacture (COMPAG) saw the initial development of the toolkit [13]. The current research has been undertaken to expand the features in the intial tools and integrate more real world business requirements such as V-Man human modelling components into the toolkit. During this development the name of the tools has changed to the Core Component Editor

(CCE) to reflect the development of the components for the automation system lifecycle.

The CCE provides support for the specification, simulation, configuration and maintenance of the automation systems for both the initial design and final maintenance phases of the lifecycle. This has been achieved by "breaking down" the automation system into reconfigurable reusable components.

The basic concepts adopted for the development of the toolkit are:

1. The components behaviour is to be described using Finite State Machines.
2. The assembly of an automation system should:
    a. Require very little training.
    b. Be in line with the role of the mechanical / process engineer
3. The CAD should be taken from existing CAD toolkits, which should be first simplified to remove items such as nuts and bolts and cabling that are superficial in terms of observing the behaviour of the system.
4. All imported CAD should have its geometry simplified to provide lightweight models for distribution and simulation efficiency.

Central to the CCE toolkit is the representation of the control logic as a set of consistent timing (c.f. Gantt charts) and state transition diagrams. System "stakeholders" can use the tools at different stages of the machine lifecycle for different purposes as outlined in Figure 23.

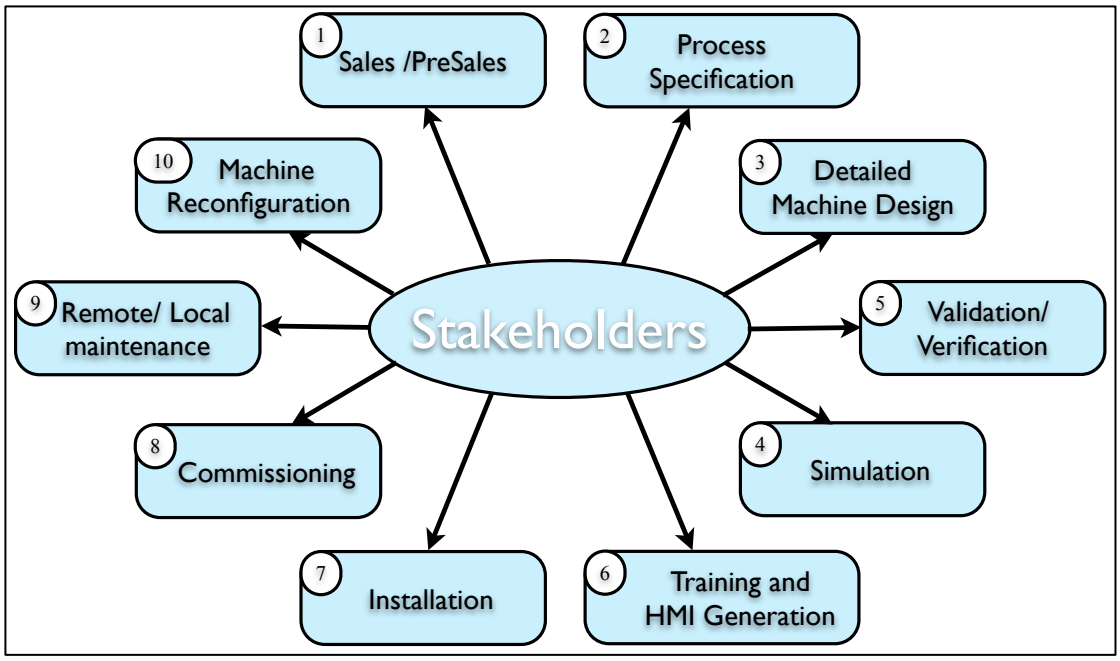


Figure 23 How the CCE tool Stakeholders are aligned to the automation system lifecycle stages

In order to achieve this lifecycle support, the integrity of the system and the information that is being displayed is paramount. To achieve this the CCE component architecture is based upon an "integrated model". Ensuring that this model is maintained and enhanced throughout the lifecycle of the machine will maintain the system integrity and reliability.

The integrated model is used during the complete development of the automation system from initial concept / specification to machine reconfiguration. The model is made "more complete" during the development of the automation system. For example, at the process specification phase a simulation can be made of the machine running in an automatic mode (i.e. without human interaction) to ensure the machine process is correct. Later the manual interlocks and part variant handling interlocks can be added to the existing model to complete its behaviour. Further to this the choice of control system, its installation configuration and diagnostic and maintenance procedures can be added to the model which may then be used to generate a working HMI automatically [101]. As the integrated model is an enhancement of the model created in the early lifecycle stages this model may still be used to convey the machine concept and operation without change.

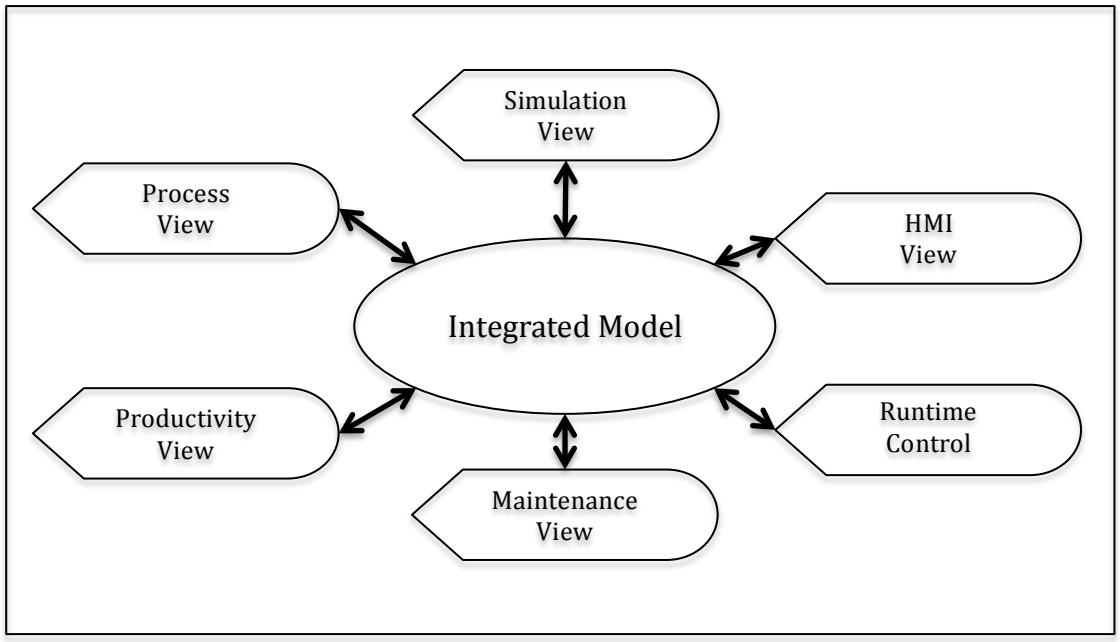


Figure 24 Examples of the currently available views of the CCE Integrated Model

Central to the CCE philosophy is that the physical control network on the real automation system uses the same definition of the application as the simulation thus reducing the likelihood of errors being introduced into the application at the commissioning stage, as there are no translation issues [102].

The integrated model has been implemented using relational database technology (see 4.4.3) to provide secure concurrent access (currently Microsoft Access for testing and SQL Server for multiuser implementation) for the stakeholders at the different lifecycle phases.

During the machine lifecycle each stakeholder will be presented with views tailored to their specific requirements (see Figure 24), although it is envisaged that there will be some views that will be used throughout the lifecycle by the majority of stakeholders, such as the state transition diagrams included e.g. in the process and simulation views (see 4.4.4.2.)

4.3 Design Philosophy

The research focus has been based on providing a toolkit to support component-based engineering of automation systems, not on the architecture and implementation of the

underlying software design that supports this. However, in order to satisfy the requirements of the toolkit in terms of generating a robust environment to integrate a distributed industrial supply chain a component-based engineering approach to the CCE software design was taken. The aim of this was to modularise the software and provide consistent application interfaces.

The system has been decomposed into three separate areas: (i) Model, (ii) View and (iii) Controller. This approach is recognised as a fundamental software "design pattern" [41] that separates objects into one of three categories:

- **Models** for maintaining data.
- **Views** for displaying all or a portion of the data.
- **Controllers** for handling events/ actions that affect the model or view.

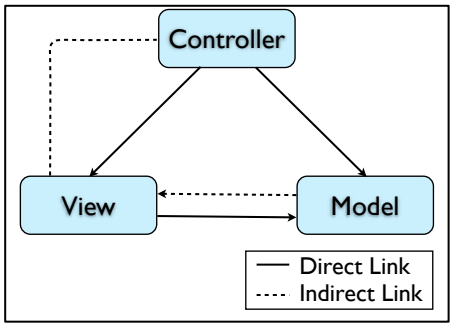


Figure 25 Model View Controller

The *Model* is directly linked to the controller and the view to allow the direct update of the model (see Figure 25). The *Model* handles system persistence and ensures that data integrity is maintained. By implementing data integrity in the *Model* ensures that illegal operations are not carried out by the *Views* (that may be written by a 3rd party), and also provides a simple interface to allow the retrieval and storage of data. The *Controller* has a direct link to both the *Model* and the *View*, such that if either changes it may notify the other. The *View* may create and link parts of the *Model*, and it may be dynamically updated through the *Controller*. Many *Views* may be created all linked through the *Controller* such that the *Views* will receive events affecting the *Model*.

Other design patterns used are:

- **Singleton** – This ensures that only single instances of an object are ever created by the application. For example, there is only one database module handling updates to the database.

- **Observer** – This pattern is used by the *Controller* to ensure that events are passed to all the *Views* using a "publish and subscribe" mechanism.

- **Façade** – Used to hide the implementation of the shell and the other objects associated with it.

- **Factory Method** – The Factory Method is used to load / build the relevant set of modules / classes to create the user *Views*.

The toolkit is designed to be extensible. For example:

- The toolkit shell is a framework that allows modules to be added through the use of the shell application programming interface (API) and database without recompilation.

- The loading of the software modules is done via the database, and hence supports configurable access to modules.

- The Integrated Model structures maintain the integrity of the model so new *Views* and *Controllers* may be added without impacting on the integrity of the *Models*. Further to this, any output from the tools are defined using the eXtensible Markup Language (XML) to allow consistent import / export of *Models* to and from external systems.

The Integrated Model is central to the architecture of the CCE toolkit. This model provides all of the data required for the build and simulation of the automation system including the basic kinematic description of the component behaviour. This is significantly different to tools such as provided by Delmia and Technomatics where kinematic information is integrated into the 3D model. The Design Schema of the integrated model is such that: (i) operation on any relational database (ii) management

of the automation system component lifecycle and (iii) management of locking and release of automation system components are readily achieved.

4.4 Framework Design

4.4.1 Overview

The development of any large system requires decomposition into component parts in order to reduce the complexity of the problem. The first level of this decomposition is to identify the functional modules that are required. This functional breakdown not only offers a set of discrete software modules, but it also helps in the providing access to users based upon their requirements (e.g. a productivity engineer will not be required to build automation components, see section 4.4.2).

The CCE decomposition has identified 6 distinct modules that may be used for the development of automation systems:

1. Component Builder
2. Module Builder
3. System Builder
4. Work-piece Route Editor
5. Process Validator/ Simulator
6. System Install

The **Component Builder** is used to define state behaviour of the virtual or real elements, build geometry and animate the physical representation of the system components. The component builder is also used to define the error conditions for a component.

The **Module Builder** module provides a logical combination / assembly of different elements to group and animate their behaviours (e.g. a pallet clamp that may have part seated sensors and clamp actuators). The Module Builder supports the generation of a list of parameters to configure the module behaviour, geometry and module level interlocking.

The **System Builder** supports a logical assembly of different components to represent operational / system level requirements. At this level interlocks can be created to define the system operation for the control modes (e.g. auto / manual). The System Builder module incorporates error / safety handling and supports peer-peer communication between the components to propagate their state behaviours to the higher-level systems.

The **Workpiece Route Editor** is used to define the state behaviour of a workpiece (i.e. product) as it progresses through an automation system and how the workpiece interacts with the system e.g. when cylinder block moves from a conveyor to a work station the Workpiece Route Editor is used to define which sensor states are changed.

The **V-Man Editor** enables the simulation of a human being interacting with the behaviour of an automation system to be defined. This is achieved by describing human behaviour using finite state machines, which are interlocked with the machine behaviour. The V-Man Editor describes the "essential" human behaviour without the complexity of moving fingers, heads and feet and without the details of animation (e.g. walking models) as displayed in other human simulation tools. It has been designed soley for the purpose of visualising the process effectively without the need for specialist CAD stations and CAD trained operatives. The editing of a V-Man sequence may trigger an additional full-scale ergonomic study. These are time consuming and expensive, so is best reserved for specific cases where it is essential to understand the human capability and safety issues.

V-Man simulations are also interlocked directly to the machine behaviour. This functionality is designed by the process engineer by describing the manual operation process at a high level of abstraction (e.g. using FSMs (Finite State Machines)) in the same way as the inter and intra component behaviour is interlocked. As such V-Man models may be integrated into the machine process by interlocking the V-Man Finite State machine with those running in the automation system. Each dynamic state may then be populated with the virtual human movements and exported as either MTM or MODAPTS sequences.

The **Process Validator** is used to verify and validate the system through simulation based on the output of the Workpiece Route Editor (or using manual sensor

stimulation). It may also be used to check visually for potential clashes between the components and confirm the system cycle time. In addition, commissioning tests can be performed with the virtual system to simulate component failure by forcing sensors / actuators to fail and ensure correct operation error reporting. The Process Validator may also be used to test the progression of a virtual part through an integrated *virtual* and *real* system, allowing the full testing of a system performance before the build has been completed i.e. only a few of the real components have been completed. (Note: Commissioning is currently undertaken after the real automation line is completed at the machine builder when end user teams visit to test prior to shipment and installation).

The **System Install** module facilitates the process of installing the machine configuration to the target platform. This allows the granularity of the installation to be chosen e.g. either all of the components installed on one PLC or each component installed on a single controller. As a general rule the distribution granularity should be no finer than individual components located on single controllers.

A framework has been devised to provide controlled access to the modules such that they may operate in consistently and provide unified user interactions. This framework comprises of the following modules:

1. CCE Shell Module
2. Database Module
3. Authentication Module
4. Error Management Module
5. User Management Module
6. 3D Model View Module

The **CCE Shell** provides the window management system and allows the user to access the functional modules in a consistent manner. The **Authentication** module firstly ensures the registration of the tool installation using a system wide software

registration key and secondly authenticates that users have access to the automation system library.

The **Error Management** module handles the display and reporting of all information and error messages. This not only improves the user experience, but also provides better diagnostics of system performance. The **User Management** module allows users to belong to user groups. A user group will dictate what access the user has to the system, ensuring that the access is tailored the requirements of the user.

By associating a **3D Model View** with each component of the automation system, a complete model of the component behaviour may be created. This model is used to show the physical form of an automation machine (or part of the machine) and allow them to simulate its behaviour based up on the FSM diagrams created during component building. Virtual Reality Modelling Language (VRML) has been used for this as it provides the following characteristics:

1. VRML is an open standard, which most CAD software is capable of importing and exporting as surface models and is under control of the World Wide Web Consortium (W3C)

2. Lightweight 3D Models (i.e. small file / memory size) can be represented in VRML so that the models may be used on general PCs and distributed on the Internet. VRML was initially developed as a web XML document to display 3D models on the Internet.

3. VRML provides the mechanisms for building complex movements using assemblies of objects in a hierarchy i.e. the movement of one piece of geometry can be linked to its child geometries so the children will move with the parent i.e. end effectors (child) can be linked with transport arms (parent) and animated together.

4. VRML facilitates the integration of kinematic representations of geometrical items.

5. Commercially there are System Development tooKits (SDK's) that support the creation and manipulation of VRML 3D models.

In the CCE toolkit, the VRML models are assembled and their behaviour is "interlocked" with other component's (s') behaviour(s) to produce a fully working 3D virtual model of the "real" machine.

The CCE system has been built to allow distributed teams of engineers to view and edit the performance of automation systems simultaneously and consistently by ensuring that only single users may edit a part of the system at any one time. This is achieved using a central relational database to support user edit locking. Edit locking is supported centrally to ensure that only one person can edit a system, component, or module at a time. The database structure is discussed in more detail later in this chapter.

4.4.2 CCE Shell Module

The CCE Shell Module controls the generic behaviour of the system and provides: (i) user access, (ii) control of the initiation of other system modules such as *System Editor* or the *Component Editor*, (iii) maintenance of the CCE's window lifecycle and (iv) error handling and reporting.

The schematic shown in Figure 26 illustrates how the *Shell Module* binds all the aspects of the CCE toolkit together. It authenticates the individual CCE software modules and enables the user to create a list of views of the model that the user is authorized to use. The *Window Builder* creates the selected views based upon information extracted from the database and manages the lifecycle of the windows based through a simple interface (e.g. via a façade). Each part of the system communicates with the *Shell Module* through this interface so that errors may be handled in a consistent way and the *View* may gain information from the *Shell Module* regarding e.g. the user preferences. The *Shell Module* also maintains a list of windows, dictating how they operate and manages the clean up of destroyed *Views* / windows.

The *Database Module* (see section 4.4.3) provides access to the core database that contains component and machine libraries whilst maintaining the integrity of the data.

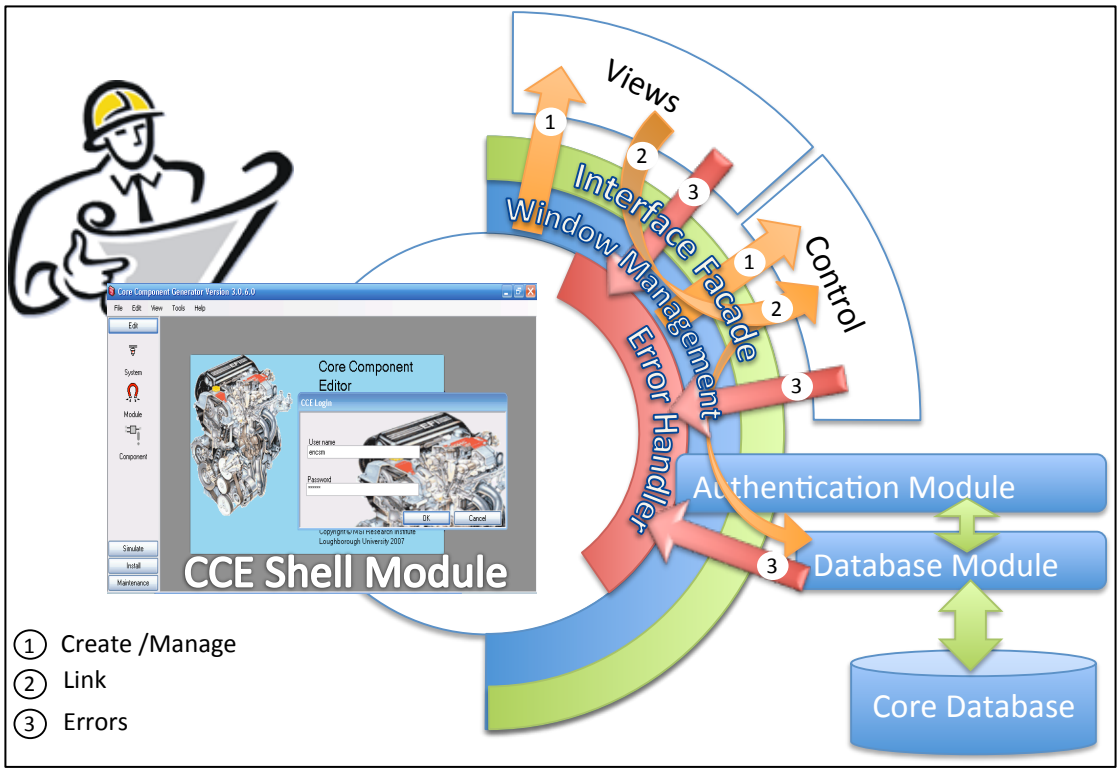


Figure 26 Schematic of the system Shell showing the interaction with the modules, authentication, database and error handling.

During the machine lifecycle many types of user will potentially use the system. To allow this, the toolkit offers the ability to present the user with views tailored to their specific requirements, although it is envisaged that there will be some views that will be used throughout the lifecycle by the majority of users (e.g. the state transition diagrams).

Figure 27 shows a simple "use case diagram" for the main stakeholders of the CCE system (e.g. marketing, process, plant maintenance, control, installation engineers and program manager) and how their access to the modules may be configured. The two main stakeholders of the system are the control engineer and the process engineer. The control engineer's role is to create focussed run time control programs that support the components within the automation system. The control engineer needs access to the *Component Builder* (allowing them to build components for the component library (discussed in 4.4.3)). Whilst this is their main role they may be required to support alternative aspects of the machine lifecycle such as providing support to the process engineer during system building as well as provide shop-floor support during installation and remote maintenance (via the *System Install* and

Remote Monitor modules). However it is unlikely that the control engineer would be involved in the defining the route of the workpiece through the system (via the *Workpiece Editor* module described in 4.4.8). The process engineer's role is to use the pre-defined components to build and validate complete automation systems, as well as provide remote support for the machine operation, requiring they have access to the *System Builder,* the *Workpiece Route Editor,* the *Process Validator* and the *Remote Monitor* modules.

The installation engineer only requires access to the *System Install* and the *Process Validator* modules, whilst the plant maintenance engineer may require additional access to the *System Builder* to "tweak" the operation of the installed machine. Finally the marketing engineer may wish to prototype a visualisation of machine for a customer or prospective machine builder (via the *System Builder* and *Process Validator* modules) whilst the program manager should just be able to view the created models to validate the progress of the program via the *Process Validator* module.

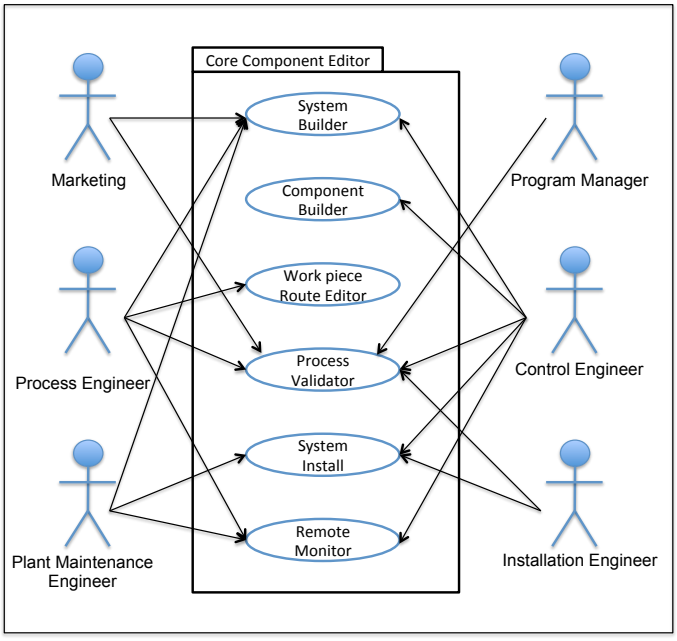


Figure 27 Use case diagram detailing the proposed User Access Requirements [13]

Access to the tools is controlled on a per user group basis and a user may reside in one or more groups. In this way, access to the *Views* may be controlled thus providing each user only with the appropriate tools consistent with their roles.

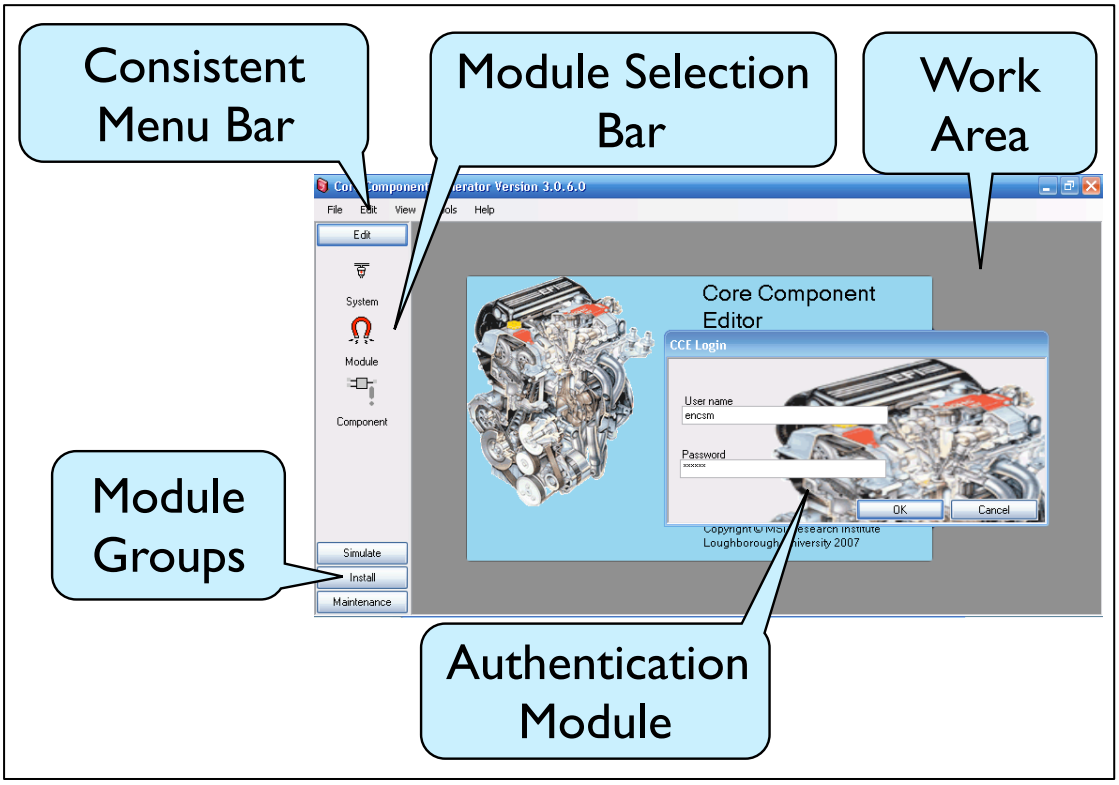


Figure 28 Shell Screen Layout

Figure 28 shows the layout of the CCE shell screen as seen by the user. The *Authentication Module* is integrated into the *Shell Module*, and is used to authenticate both the application and the user (i.e. the user is required to login using their username and password to gain access to the system and the CCE tools must be authorised for use on their computer). At the top is a set of menu's that control the generic functions of the application so that the user is presented with the same menu functions whichever module they are using. The *Module Selection Bar* is broken down into groups of related modules (e.g. Edit, Simulate, Install and Maintenance) and is configured as required in the database (see 4.4.4). Finally there is the application *Work Area*, where the selected module will be displayed in a maximised window. The CCE *Module Selection Bar* is hidden (and the authentication disabled) when a module is chosen to ensure that the user is only provided with relevant information when focused on system design, implementation and evaluation activities.

4.4.3 Database Module

The database module is a *Singleton* that provides a single access point to the underlying core database. The role of the *Database Module* is to allow a list of valid users to be maintained and provide controlled access to the CCE modules and automation systems, whilst ensuring the integrity of these data. This does not imply that the information is complete or the automation system will work correctly. However it ensures that the data contain valid information without references to removed or changed items. The *Database Module* also provides mechanisms for importing and exporting data from remote sources i.e. XML files or ccZ files (which are Core Component editor Zip files that contain all the required information to "completely" describe an automation system (see CCE Viewer below)).

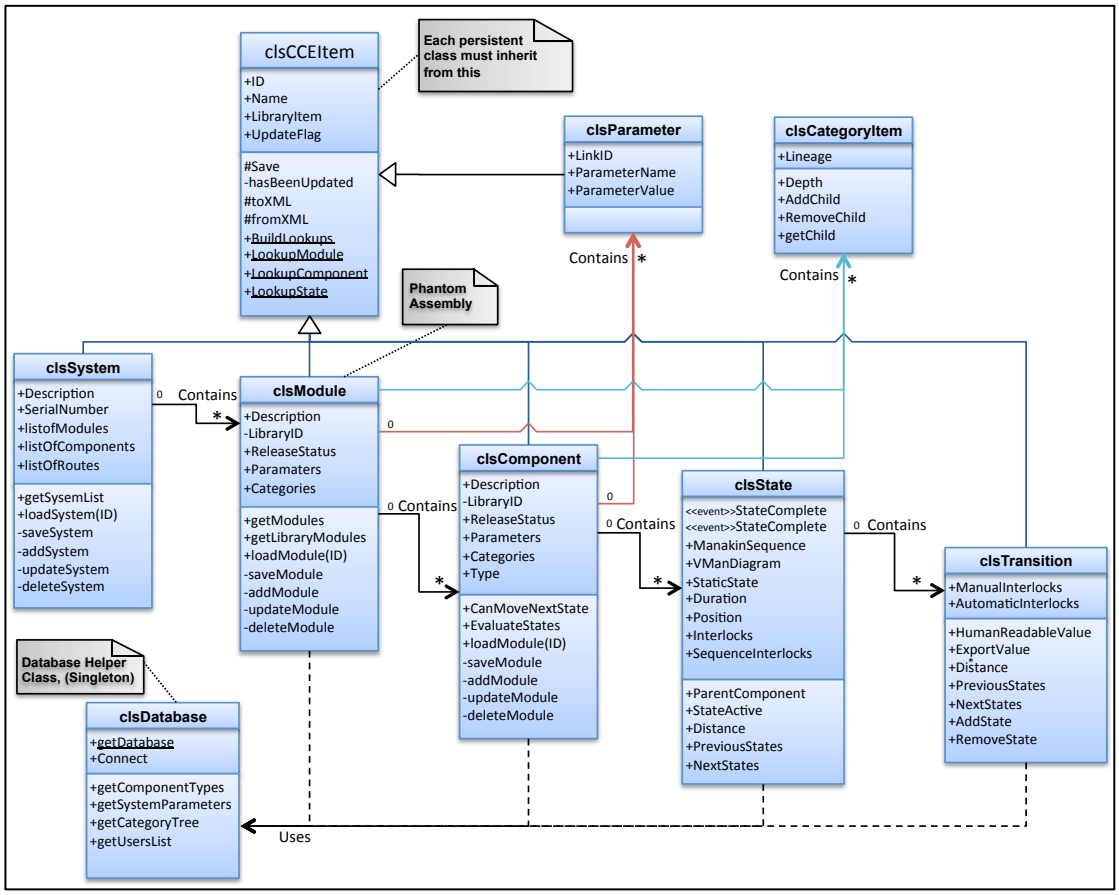


Figure 29 Database Module Class Diagram

Figure 29 illustrates the main classes in the automation system object model for the database module. The main class is *clsCCEItem* from which any persistent item inherits its properties. *clsCCEItem* is an abstract base class (i.e. it cannot be

instantiated) with several virtual functions that manage the storage in a consistent way:

- Each item must have a unique ID so that it may be referenced by the automation system. A GUID (Globally Unique IDentifier) has been chosen for this as it is unique across machines and networks. Additionally the persistent item adds a prefix depending on its type (e.g. components add "C-", modules add "M-" and system adds "S-"). Whilst the program does not require this prefix it significantly aids identification and the debugging process.

- Each item must have a name to make it easy for the creator / user to identify it. The default name is the original name given to the component when it was stored in the library.

- Library items are reusable components / modules that may be used to create automation systems. The Library item flag identifies if the database item resides in the library tables or the automation system tables.

- To identify the "saved" status of the database item an *UpdateFlag* is used, which is required by the save function to the appropriate actions are undertaken (i.e. add, remove or update).

There are four virtual functions that must be overridden by each database item class:

- *Save*: to save the item to the database.

- *hasBeenUpdated*: which is a flag to indicate if the item or the child / items it uses requires saving (i.e. based upon the status of the update flag).

- *toXML*: to ensure that each item must be able to output its information as an XML node.

- *fromXML:* to ensure that each item must be able to create itself from an XML node.

Finally there are some shared functions and properties that are available to all the objects in the hierarchy. These are used to identify build "lookup tables" of all the items in the automation system object hierarchy and helper functions to search the lookup tables. These lookup tables provide "random" access to any of the items within the automation system hierarchy and are used by objects within the hierarchy as well as externally (e.g. if the system wants to retrieve the parent component of a specific state there is a lookup that gets *Component ID* from the *State ID*).

Each database item uses the *Singleton clsDatabase* class to access the database. The connect function returns either a new connection, or an existing connection to the database. In this way alternative database connections can used without affecting the database item classes.

Components and modules also have a list of parameters associated with them that allow the recording of associated information. This may be physical parameters such as payload or length, or control parameters such as speed 10- 20 m/s or temperature measurement range. Ultimately these parameters can be linked to component properties such as dynamic speed dictating the timing performance of the component, or the length of a conveyor dictating the physical parameters of the model.

The database has been designed using a relational database schema so that access can be obtained using standard SQL 97 and the relevant .NET drivers to support access to the underlying relational database management system (RDBMS). This ensures that the CCE is not limited to a single database vendor. Also to avoid vendor lock in, stored procedures and other database specific functionality have not been employed. The database solutions that have been tested include Oracle, Microsoft SQLServer, Microsoft access and MySQL.

Entity Relationship Diagrams (ERDs) for the library components, modules and system items are illustrated in Figure 30 and Figure 31. These ERD's shows only the main table interactions and cardinalities that support the database module. Figure 30 shows the library structure containing all the library components and modules and Figure 31 show the automation system structure (e.g. System – Components – Parameters – States – Transitions).

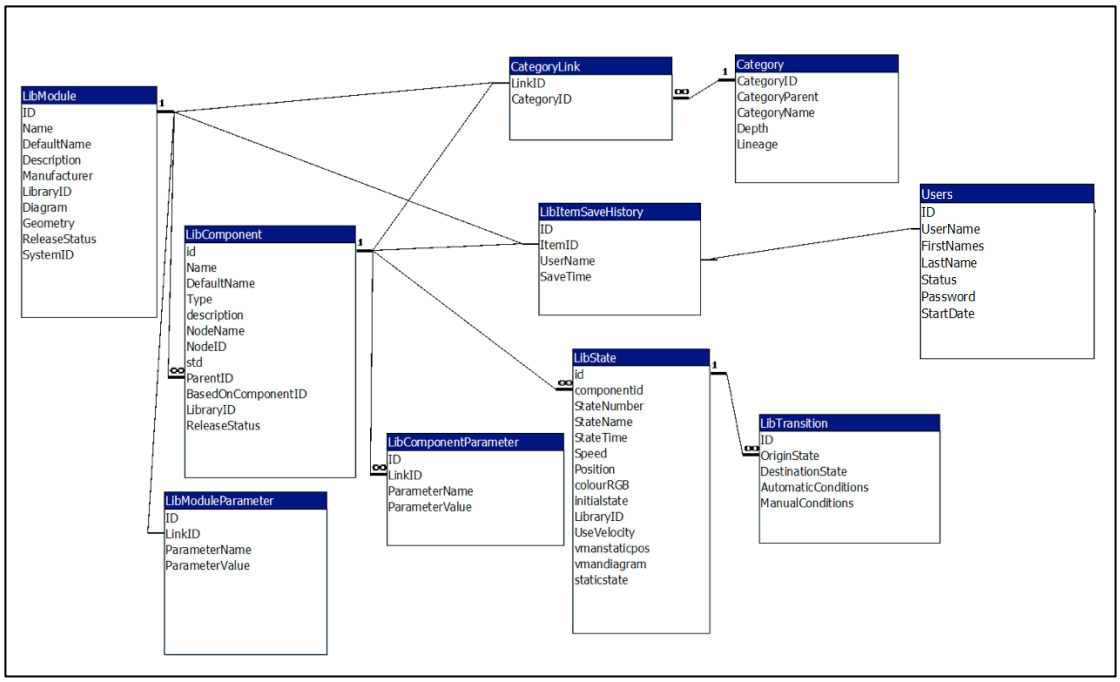


Figure 30 Library Items Entity Relationship Diagram

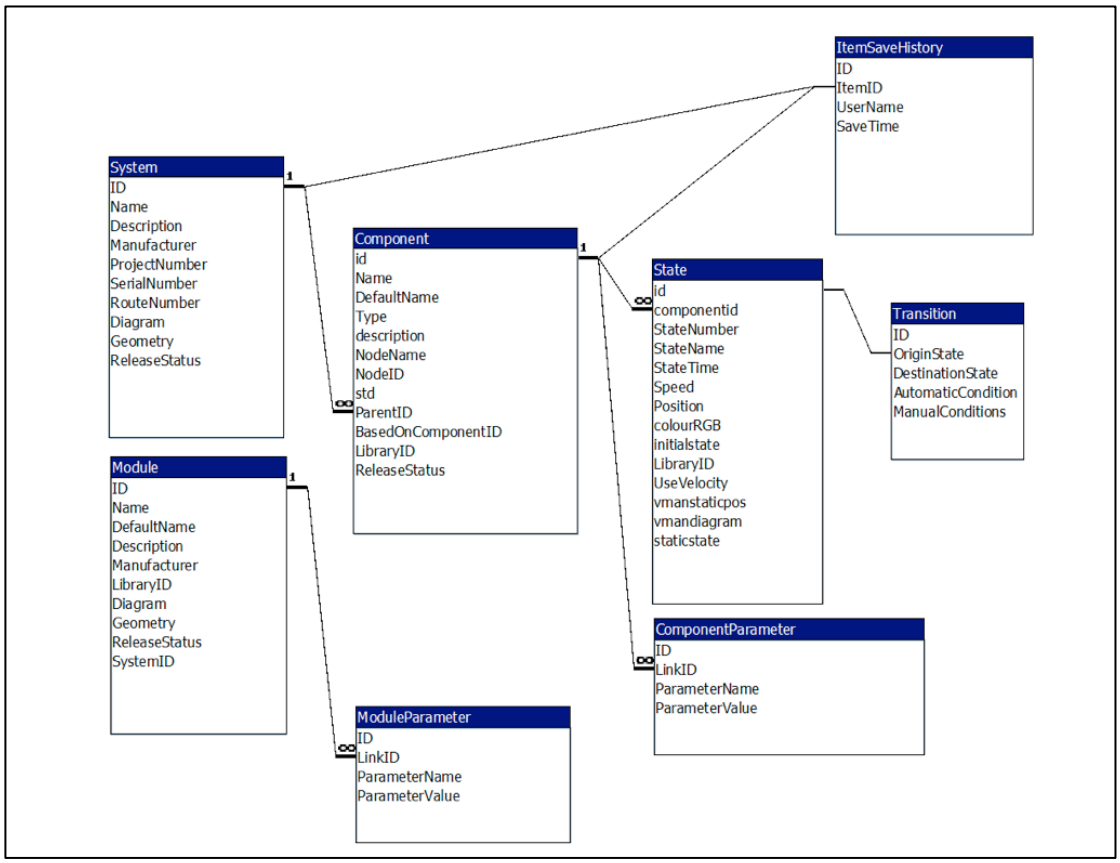


Figure 31 System Items Entity Relationship Diagram

4.4.4 Authentication Module

The *Authentication Module* is responsible for ensuring that the CCE toolkit has been authorised for use on that computer (i.e. System Authentication) as well controlling the user login access to the tools and constituent modules of the CCE (i.e. User Authentication).

4.4.4.1 System Authentication

System Authentication ensures the CCE is authorised for use on a computer by storing and checking a hardware unique key with an unlock key provided through registration. The hardware key is generated from 2 of 4 hardware unique ID's (e.g. Hard disk ID and Ethernet MAC address), which is provided to the CCE administrator by email or phone. The administrator then generates an unlock key based on the hardware ID which once entered will allow the CCE to start correctly.

4.4.4.2 User Authentication

The CCE toolkit has been designed as a multiuser tool and as such requires that each user logs onto the system for validation, access control and edit locking control. Only minimal details are stored about the user (e.g. username, password, full name, email address and a list of groups that the user is a member of that link to the tasks that provide access to the modules of the CCE).

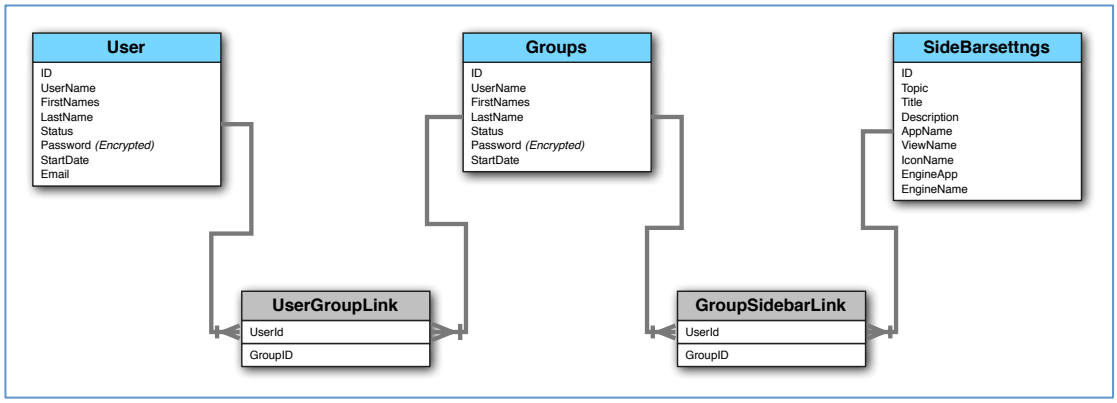


Figure 32 Database representation of the user access rights

The user access rights are configured in the database such that the user can be a member of different groups (e.g. Library Editor, Maintenance or Process

Engineering). Note: since a group may also be used by many users the *UserGroupLink* table is used to create a many-to-many relationship.

The *Groups* table links to the selected *Side Bar* items, such as *Edit System, Edit Component* and *Install System*, which are grouped as per their topic (Note: a *Topic* relates directly to a *Module Group* e.g. Edit, Install or Maintain (see *Module Groups* shown in Figure 28)). A group may have many *Side Bar* items and *Side Bar* items may belong to many groups so the table *GroupSidebarLinks* is used to create a many-to-many relationship.

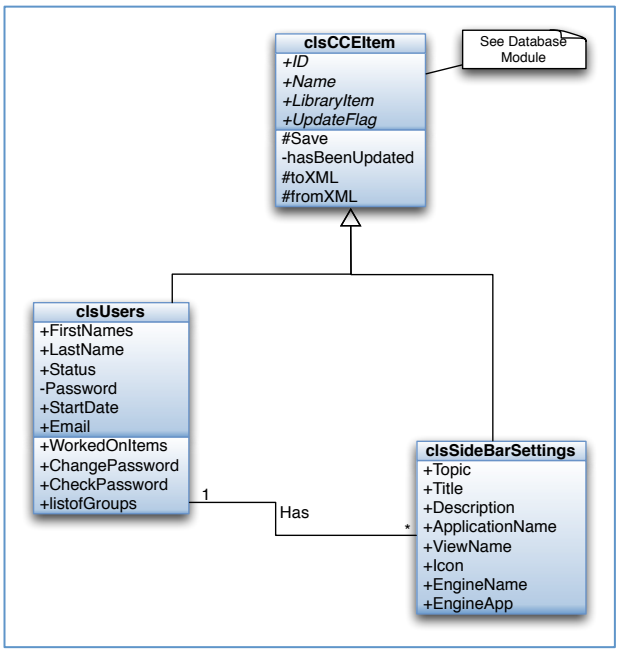


Figure 33 UML diagram of user access rights

Figure 32 shows how the user table (inherited from *clsCCEItem* to provide persistence) has 1-n *Side Bar* items, which will be grouped using the *Topic* property to create the *Module Groups*, with sub-items of the *Views* they may choose from (see Figure 32).

The *User* class also maintains a list of all the items a user has worked on. This class is linked to the saved history for *Component*, *Module* and *System* classes.

4.4.5 Component Builder Module

The *Component Builder Module* is used to create libraries of commonly used components. A CCE component is defined as a unit of composition that has a contractually specified interface and explicit dependencies only, which can be deployed independently and is subject to compositions to build automation systems, by providing a service, validation, error definition / reporting, and design / maintenance information.

There are different types of components used in the CCE tools:

1. **Actuators** are defined as mechanical components that accept data signals and perform actions based on those signals. Examples include a 2 position cylinder that may be used to block a workpiece from progressing through the automation process.

2. **Sensors** are components that detect external stimuli and respond in a distinctive manner (e.g. by changing state from OFF to ON).

3. A **Virtual** component has no physical output but has behaviour that may affect the automation system such as a timer or a routing algorithm that based upon its inputs, controls the flow of workpieces through the automation system.

4. A **Non-Control** component has no inputs or outputs but is used to build automation systems providing a capability to represent, for example, a machine's standard framework.

5. A **Manakin** is a virtual human component with reverse kinematics functionality associated with the limbs (see later in this chapter).

6. **Process Logic** is defined as a component that controls the overall process of a system, module or group of components. For example whilst a machine may comprise multiple components its overall process could be described as the following sequence: (i) Receive Workpiece, (ii) Clamp Workpiece, (iii) Undertake Operation, (iv) Unclamp Workpiece and (v) Release Workpiece

A well-defined component will enable reusability through the use of a well-defined FSM and parameter list, as well as the encapsulation of all the hard real-time communications needed for consistent and correct operation. The hard real-time interaction of components may not be successfully linked through the component interface, as latency and determinism of messages cannot be guaranteed without the complete understanding of the software/ hardware performance as well as the software distribution.

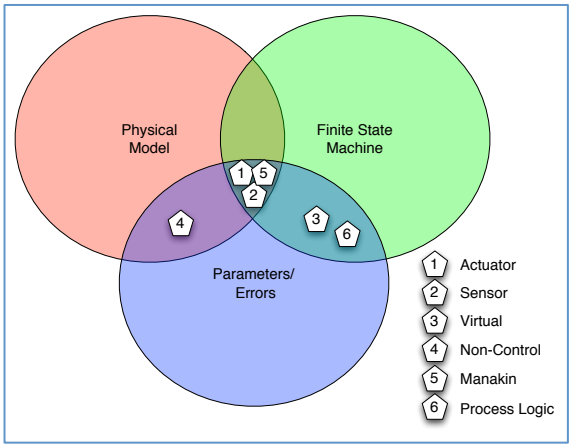


Figure 34 Constituents of each type of component

Actuators, Sensors and *Manikins* may all have a physical model, state machine and a set of parameters / errors (see Figure 34). They do differ in that *Actuators* and *Manikins* are controllable devices, whereas a sensor only represents the result of external stimuli. In addition *Manikins* have predefined geometry that has to be configured prior to simulation using the V-Man tools.

System state progressions required by *Virtual* and *Process Logic* utilise FSM representations and may have a list of Parameters / Errors. *Non-Control* components have a physical representation and some Parameters / Errors, but do not require FSM to represent the logic encapsulated within them.

In order to build *Actuators, Sensors, Non-Control* and *Manakin* components a physical model is required. As the CCE tools are not designed to compete with 3D CAD packages (many of which are large complex systems employing highly advanced modelling techniques, such as Delmia, Catia, Nx and Technomatics) physical model data must import data must be imported from all sources. To avoid having to import CAD data from many sources, VRML, an export format that is

common to all of them, was chosen. The exported models are surface models only, which for the purpose of visualisation is sufficient.

The process of building lightweight models (required to meet the tools objectives) is described in Figure 35 below.

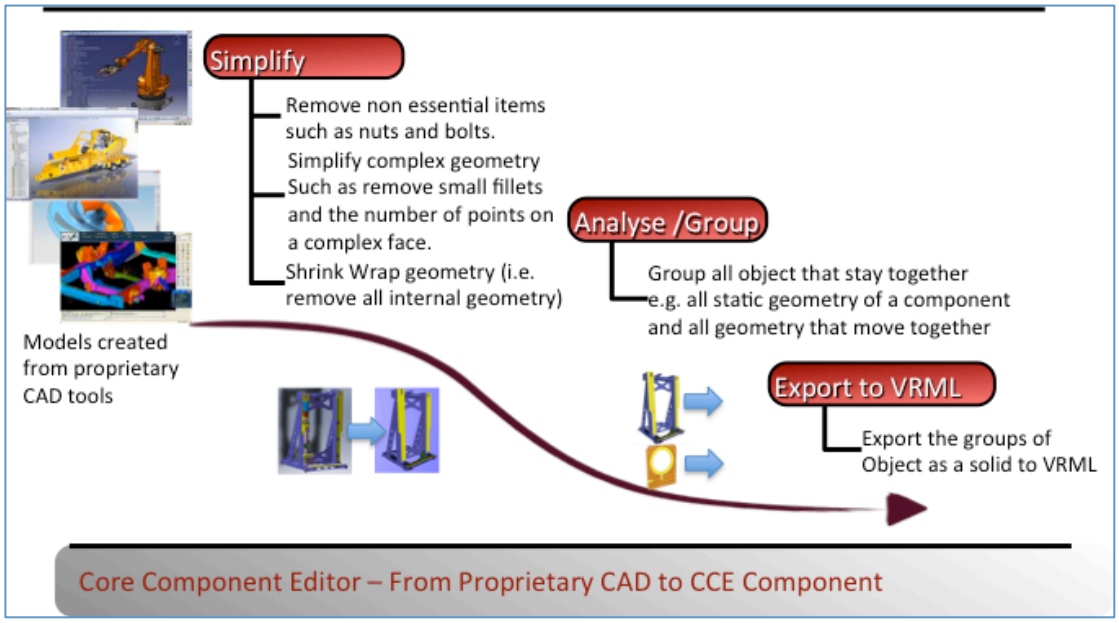


Figure 35 Data taken from proprietary CAD simplified and grouped and exported to VRML

Before the models are exported from CAD it is good practice to simplify the model to produce a "lightweight" version. This involves removing non-essential items (e.g. washers, screws, nuts & bolts, cables and clamps) from the assembly to reduce the complexity. In order to provide an animated model the original CAD must be split into the following:

1. **Static Geometry** e.g. (1) Guarding, baseplates and frames to create *Non Control* components and (2) *Actuator* casing/ mount points or a clamp base to create *Actuators* and *Sensors*.

2. *Dynamic Geometry* – e.g. *Actuator* piston / end effector or moving parts of a clamp.

Each isolated piece of geometry is then exported to a separate VRML file for post processing. Once exported a proprietary tool may then be used to simplify the

geometry i.e. remove detail from the model such as small fillets and holes and reduce the number of points used to describe surface features (without losing the visual quality of the model). Finally the mode is "shrink- wrapped" to contain only the external faces i.e. any internal geometry is removed such as oil ways and cavities.

Experience has shown that using these techniques a model of 140 MB may be reduced to approximately 3-4 MB without losing quality of the surfaces and providing enough detail for simulation and visualisation purposes. (See Case Study 3, Chapter 5).

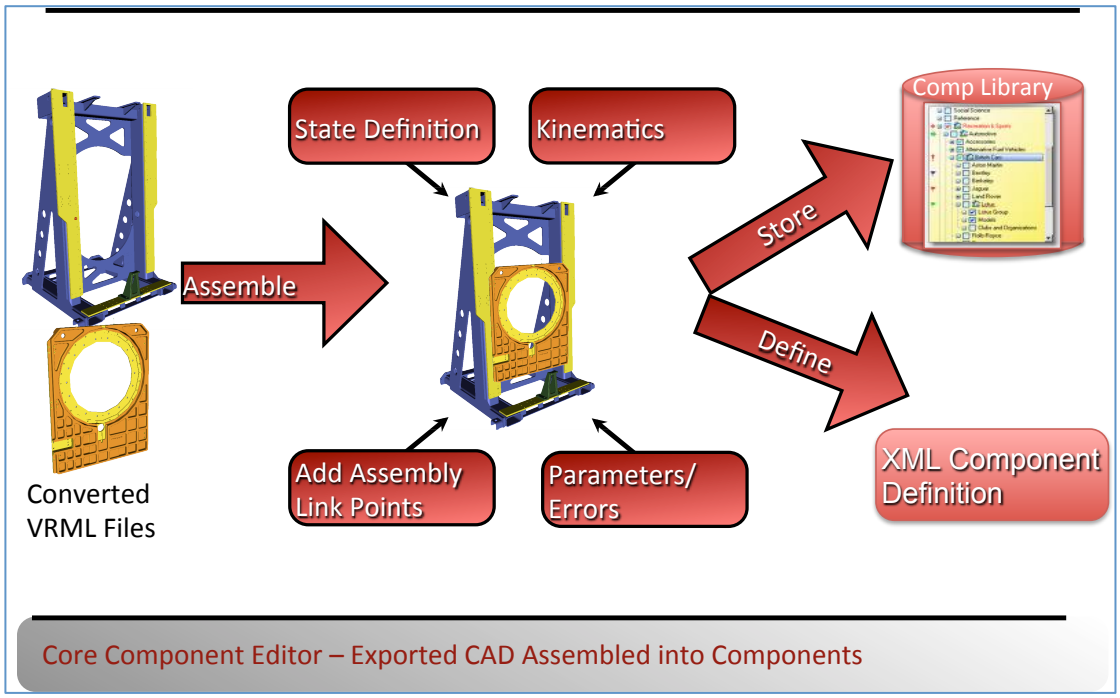


Figure 36 Building components from exported CAD

Figure 36 illustrates the process of building components from exported CAD files. These files are then imported into the CCE *Graphic Library* (currently this is a manual operation achieved by copying files into a predefined file structure). The CCE *Component Editor* uses the geometry in the library to enable the assembly into components using *Link Points*. *Link Points* are simply location points within the model space of the parent geometry (i.e. the position of the link point is related directly to its parent so as the parent moves so will the link point)

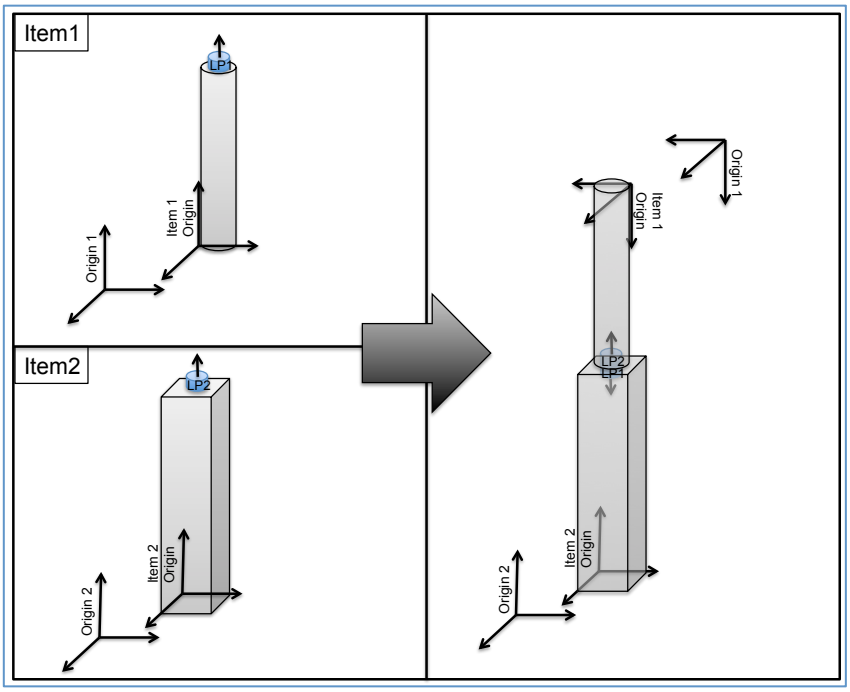


Figure 37 Link Point assembly connecting two CAD models in 3D space

Figure 37 illustrates how two graphical items (Item1and Item2) can be linked using the link points LP1 and LP2, whilst each has their own origin and therefore model-space (this is separate into the overall origin and model space). When a *Link Point* LP1 is associated with link point LP2, LP2 becomes a child of LP1 and its model space is referenced with respect to the model space of Item1. This results in any movement associated with LP1 also affecting Item2 and any children Item2 may have. As the model space of Item2 is adjusted, any relationship that Item2 has with its children will not be affected, as they are encapsulated by Item2's model space.

Figure 38 below shows a typical component from a modular machine built at Krause GmbH as part of their Intelligent Manufacturing System (IMS) product range. The lift has two electric drives that raise and lower the lift plate (on the front of the frame). The lift plate has attachments for static and rotating clamps. Whilst the drives may be configured to stop at many points along their vertical travel, the Finite State Machine (FSM) shows how the three configurable static states (Home, Pos 1 and Pos 2) have been designed. Movement to these states is controlled by the dynamic states (Move Home, Move Pos 1 and Move Pos 2). The positions of the states are configured by the static state positions and the velocity parameters defined in the dynamic states properties configures the speed profiles used by the system between the static states.

The red thick line shows the states that can be achieved from the home position. Selecting the other states will highlight the states they are connected to and, if required, interlocked with.

There are some fundamental rules associated with creating FSMs for the CCE Tools:

1. Each FSM shall have an initial state. This state must be static.
2. Static states can be connected to one or multiple static, and dynamic states.
3. Dynamic states can only be connected to a single following static state, i.e. dynamic states will always "move-to" a single designated following state

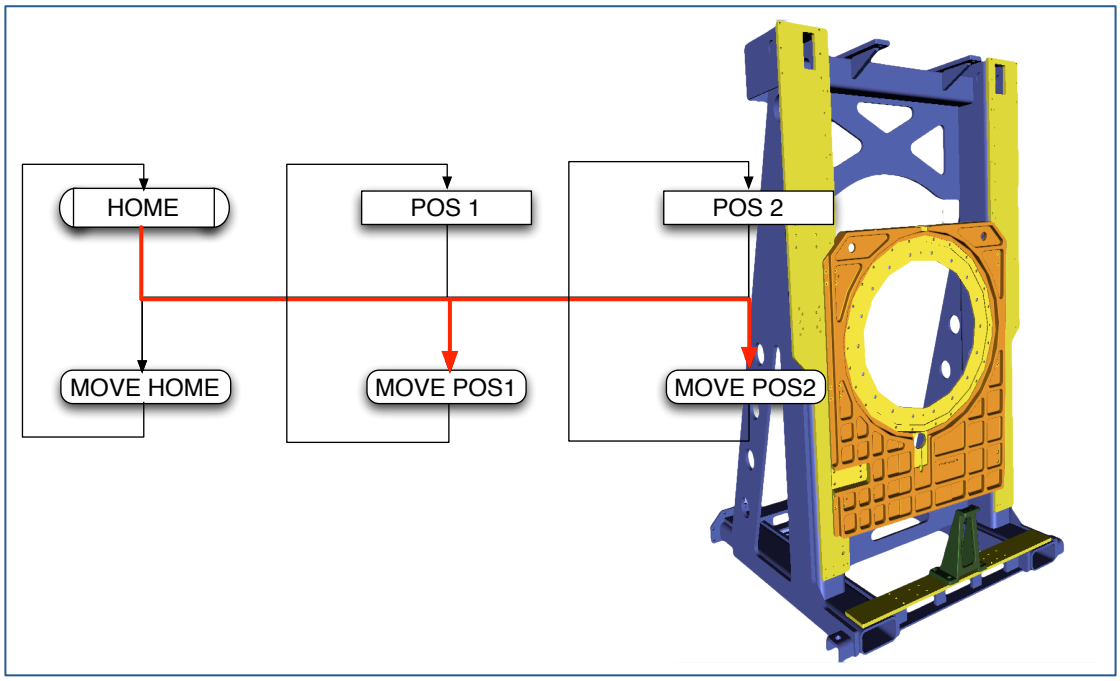


Figure 38 Example of a Component (Lift Actuator from the Krause IMS machine) and the corresponding FSM.

It is conceivable that variants of a component may be required e.g. the above lift may have multiple instances with a different number of defined positions e.g. (2 Position actuator, 3 Position actuator, 4 Position actuator). Each instance may use the same hardware, with the component FSM being designed configured to provide the required flexibility for the automation system operation. These instances will require separate components to be built and stored with different state behaviour.

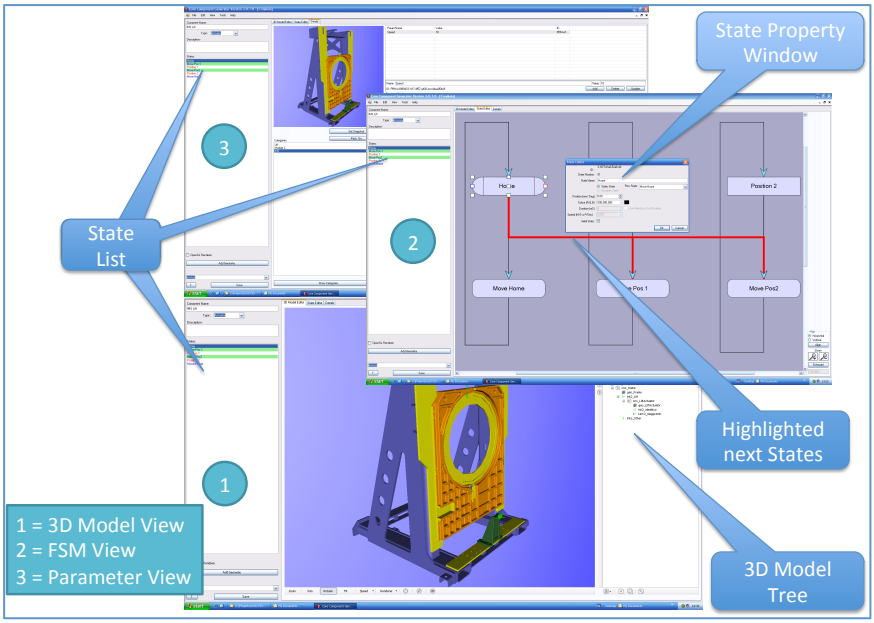


Figure 39

Component Model Views. 3D Model, FSM and Parameter View.

Figure 39 show the 3 *Views* provided for creating a component (i.e. (1) 3D model, (2) FSM and the (3) Parameter View). The 3D model *View* displays the CAD model for the component. On the right hand side the 3D model tree provides access to all the 3D construction items such as geometry and link points required to build the model as well as the translation and rotation parameters of the graphical items (i.e. their kinematic behaviour).

On the left of the screen is the component general description, the save button and the list of states associated with the component. The blue highlighted state shows the current state of the component and the green displays the next available states (based upon the FSM). Selecting the green items exercises the behaviour of the component, which can be readily visualised by observing the movement of the VRML model.

The second screen shows the FSM editor. The FSM editor ensures that: (1) there can only ever be 1 initial state (which is as static state) and (2) dynamic states may only be linked to a single static state. For example a dynamic state "move to work" has to proceed to the static "at work" state.

The FSM editor allows the position for a static state, and either the time or the speed of the dynamic state (with the calculated distance to be moved allowing the time to be calculated) to be entered. These times and positions are used to drive the kinematics of the model. It should be noted that any information pertaining to the operation of

the system is always stored in the system *Integrated Model* (i.e. not embedded in the CAD or runtime systems). This provides a more flexible storage and development of the components / automation systems as different views of the model can be created to suit the users requirements. For example if a user wants to edit the parameters and the state behaviour without changing the CAD model then a simple view can be created to achieve this, without the added complexity of interacting with the 3D model.

The final *View* of the component is the Parameter View. In the Parameter View the component can be:

1. Categorised, to aid component selection, from a user defined category tree.
2. Parameters can be recorded such as speed and power.
3. PDF documents may be linked to component to aid understanding and maintenance.

Once the component saved it becomes part of the component library, and available for use in the definition of automation systems.

4.4.6 Module Builder Module

A Module is defined as a logical grouping of components that achieve a defined goal. The *Module Builder* Module provides the facility to create groups of components that are required to work together in a specific orientation. These modules may be items such as a lift attached to a rotary plate and a clamp. The lift component may be configured specifically for an operation (such as an engine lift and rotate operation, or a lift and rotate to orientate for valve insertion).

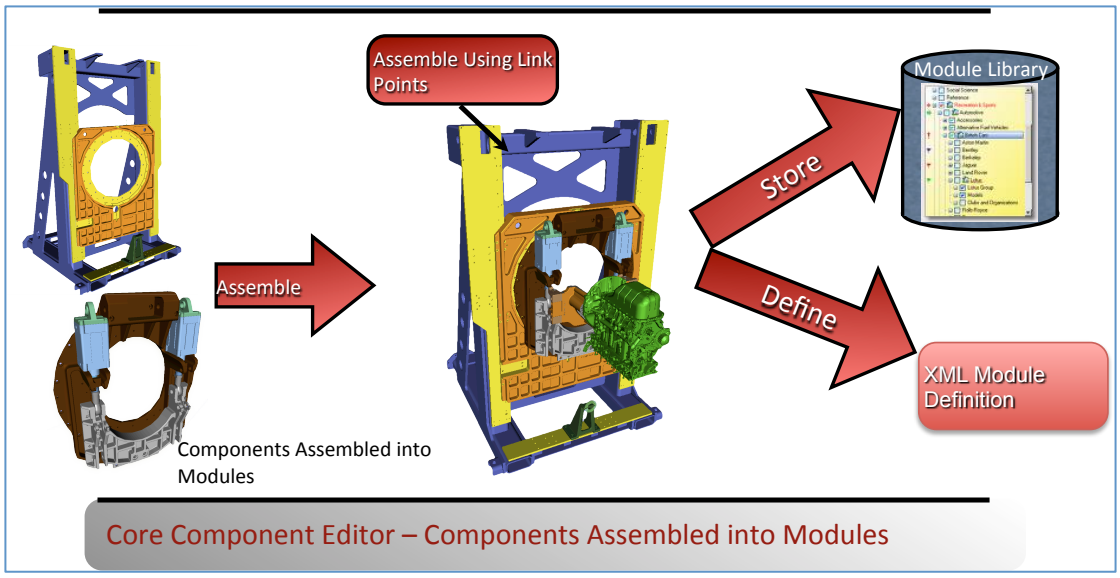


Figure 40 Process of creating modules assembled from components.

The example shown in Figure 40 shows two components (a lift actuator and a rotating clamp), that are assembled using the link points (see Figure 37) to produce a module. The assembly is then configured, categorised and stored in the module library for future use.

Modules are extended to include (1) interlocks with other modules and (2) workpiece routing logic hence providing core functionality required by the automation system.

4.4.7 System Builder Module

The *System Builder* Module is used to create an automation system that includes component interlocks to define how each component will interoperate with other components in the system. An automation system requires all of the configuration information for the following:

1. Components /Modules
2. Component Interlocks for each mode of operation.
3. Workpiece Routing Information (See 4.4.8)
4. V-Man Behaviour / Timing (See 4.4.9 V-Man Module)

Components / Modules are added to the system assembly (see Figure 41) by selecting them from the library. Once inserted they are assembled using link points (see Figure 37). This 3D model may be animated though the use of the *State Viewer* to prove the correct assembly and operation of the automation system.

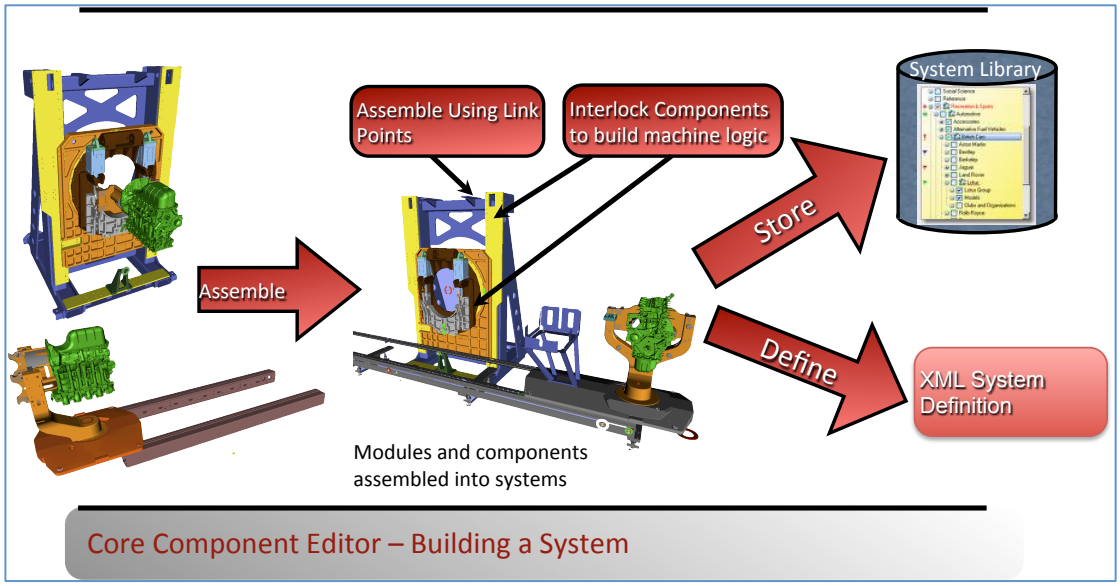


Figure 41 Creating a system from modules and components.

Interlocks are added to states of each component to define how they will interoperate with respect to the behaviour of other components within the system.

There are 2 different types of interlocks

1. **Internal Interlocks** define what is required for a state to remain in that state. If the conditions are not met then an error is raised. For example, for the lift to move to any position the Clamp must be in the "clamped" state otherwise an error will be raised.

2. **Sequence Interlocks** define the conditions that stop the Component from moving from one state to the next. For example, a clamp (shown below) will not move from "Unclamped" to "Clamping" unless the interlock conditions are satisfied. One safety caveat is that an actuator component without any interlocks will not move from its home position and this will stop a non interlocked component from rapidly moving through its states as soon as the machine is switched on to automatic.

This is achieved as shown below.

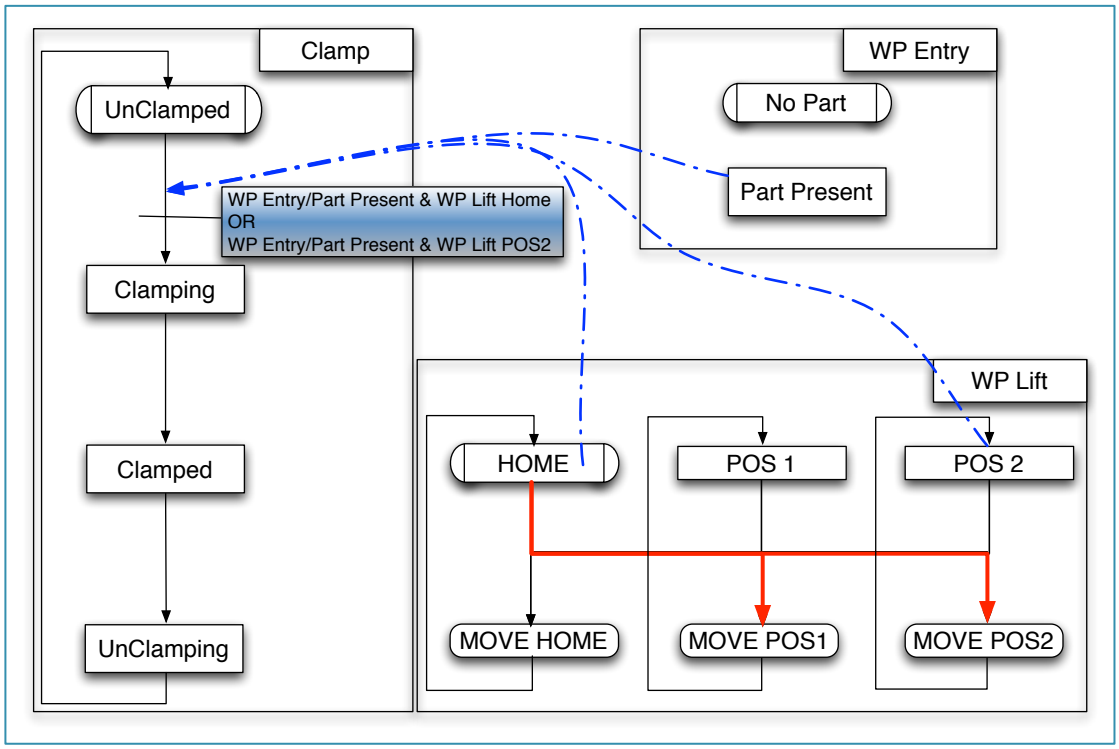


Figure 42 Component Sequence Interlocks

Figure 42 shows three components, 1 sensor (WP Entry), 2 Actuators (WP Lift and Clamp) and the condition defined to allow the clamp to progress from the "Unclamped" to "Clamping" states. The interlock entered for the clamp to start "Clamping" is:

(WP Lift/Home AND WP Entry/Part Present) OR (WP Lift/Pos 1 AND WP Entry/Part Present)

Interlocks are entered as condition groups. Each condition *within* the group is an AND *between* each group is an OR.

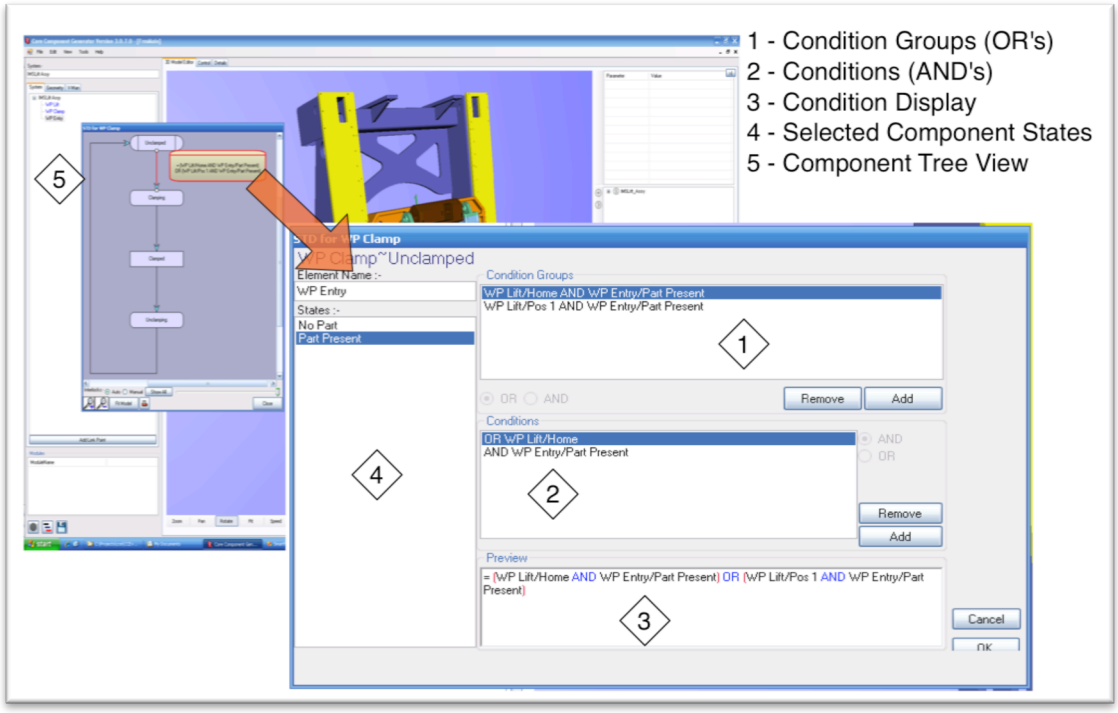


Figure 43 Entering Conditions

The entry of conditions is simply achieved by double clicking on the component from the system tree (see 5 in Figure 43), which brings up the FSM view. Highlighting the transition line between the states displays the yellow interlock box (which displays the interlock for this state change). Double clicking the yellow box displays the interlock-editing window (foremost window). To create each interlock:

1. The component that is to be interlocked is dragged from (5) to (4), to display the selected components states.

2. The selected state is dragged into (1) to enable the interlock to be created.

3. Steps 1 and 2 are repeated either dragging the state into (1) to create an OR or box 2 to create an AND with the condition group selected in (1).

As the Conditions are added the complete interlock is shown in (3). The interlocks are built to support the operation modes listed in Figure 44.

Modes of operation

Manual	Interlocks to prohibit the components from starting operations that may cause physical damage.
Automatic	Additional interlocks to above, that dictate the desired sequence of operations to occur to fulfil the required task, without user input
Single Step	As above but stopping after each control sequence is completed and wait for the operators confirmation to proceed to the next sequence
Return to initial Position	Use manual interlocks to automatically return each component to its initial condition.

Figure 44 Modes of operation.

Manual interlocks are in operation at all times thus ensuring the safety of the machine. Manual interlocks are generally based upon locations of actuators (e.g. the lift may not raise until the locking mechanism is released).

When the operator selects the "automatic" mode of operation, the machine will employ the automatic interlocks to provide sequencing. Automatic interlocks are generally based upon process logic and sensor input (e.g. if the machine is waiting for a workpiece, and a workpiece arrives and triggers the workpiece sensor then the clamp will close).

The combination of manual and automatic interlocks makes the automation system safe as no operation should be allowed to occur that could damage the machine since the progression of system states is under operator control. Additionally sequence changes can be made relatively easily without affecting machine safety. Components may also be integrated into the automation system with less risk as it is only the interaction of the new component that has to be considered without having to check the whole application.

4.4.8 Workpiece Path Editor Module

In order to simulate the behaviour of an automation system, external real world inputs are required. These inputs enable the effect of a part as is enters, moves through and exits the system to be simulated and visualised. All major commercial applications for simulating automation system behaviour provide real world input through the use of

virtual sensors (e.g. Delmia uses ray beam sensors to pick out when an object breaks the beam). This may be very useful for selecting sensor placement and simulating the effects of passing many different shaped workpieces through an automation system (e.g. a parcel sorting application), but can very processor intensive. The CCE approach is to use the knowledge gained from the CAD models and engineers' experience to predict which sensors will be triggered as the workpiece travels through the system. This process ensures that the knowledge of the automation system can be maintained in the "integrated model", without the need for 3D CAD for simulation. The advantages are that:

1. There is no knowledge embedded into the 3D model. The 3D model is only a view of the integrated model.

2. It allows simulation information to be communicated as text, so that 3rd party applications may view, edit and use it. For example a virtual commissioning tool may take the information to evaluate the behaviour of the machine if a sensor / actuator fails.

3. The cycle time for the system can be calculated in real-time as the user creates the system. Therefore the effect of any change to the machine logic can be seen immediately, without running the application.

In the CCE tools the above functionality is achieved by creating a FSM that sets sensor values moves workpieces from *Link Point* to *Link Point* and sets workpiece visibility. Figure 45 shows a typical workpiece route in the *Workpiece Route Editor*. The green and red circles indicate the start and end of the workpiece route (i.e. workpiece entry and workpiece exit). The process boxes attached to the states represent the sensor states to be activated (Note: only sensors are allowed to be activated).

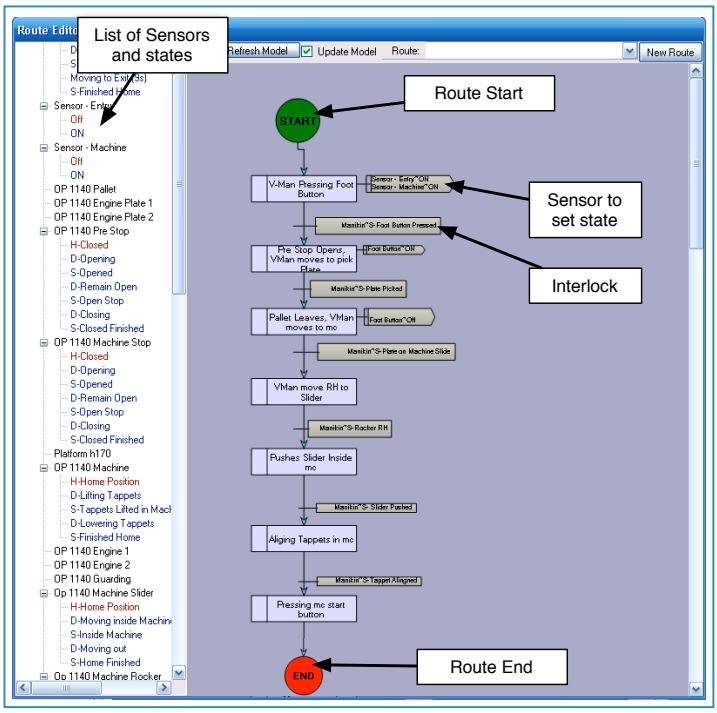


Figure 45 Screenshot of the Workpiece Route Editor

To simulate workpiece variants a mechanism to change the workpiece geometry has been implemented e.g. if Part A is inserted then geometry A is used otherwise if part B is inserted then geometry B is used (and so on). The workpiece routing logic may include a decision box, in the routing chart, to direct flow based upon the workpiece inserted. This redirection of the workpiece routing will allow different workpiece variants to trigger different sensor states without the need for separate routes for each workpiece variant. This allows the user to see the effect of introducing different workpieces sequentially to the automation system.

As part of the properties of each workpiece routing state there is the ability to change workpiece visibility to simulate the input and output of workpieces to the system. Also the ability to move a component from being associated to one link point to another link point has been added to emulate the workpiece progress through the system. For example, as a workpiece arrives it is attached to a pallet. When in position, the workpiece is clamped then lifted from the pallet. In this case the link point has to move from the pallet to the clamp so that the workpiece will move with the clamp.

4.4.9 V-Man Module

With the globalisation of powertrain operations to include the low wage economies of Asia and South America, cost decisions have to be made on complexity of machine / cost versus on-going cost of employing people to assemble engines. As discussed in Chapter 2, in high labour cost areas the level of automation is significantly higher than in low labour cost areas, so tools are required to support the design of systems that can accurately predict cycle times for Automatic, Semi-Automatic and Manual Stations. To achieve this requirement, a simulated human (*V-Man*) needs to be integrated into the automation system to prove the operation of the operator and their interaction with the automation system. The general requirements for the simulated human are displayed in Figure 46 below.

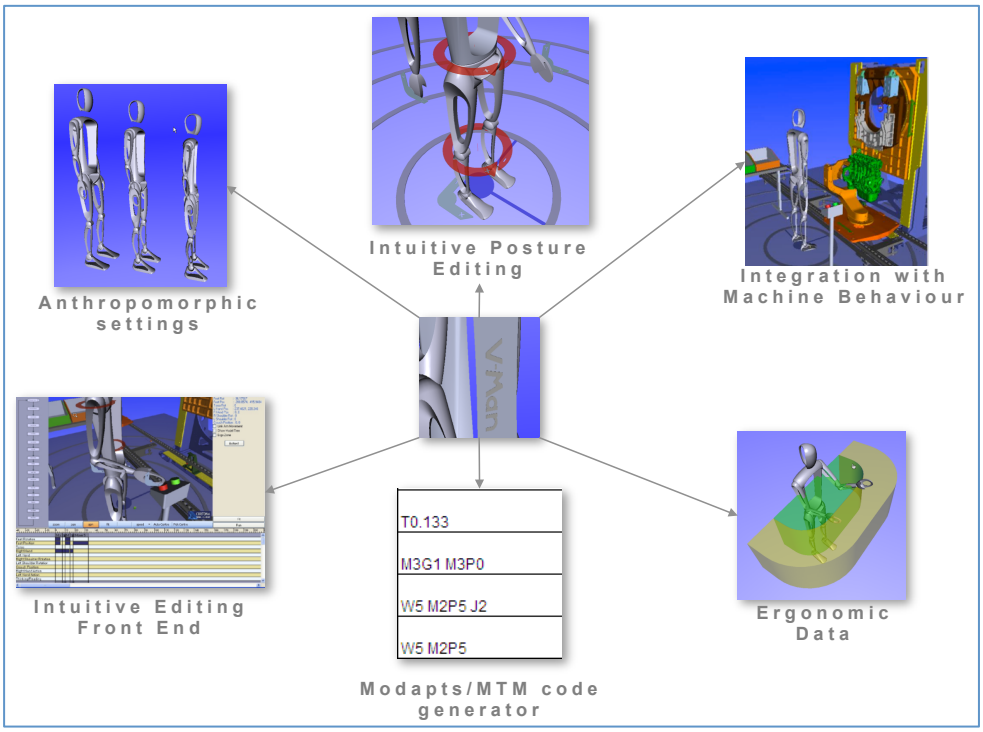


Figure 46 V-Man requirements

The *V-Man* module offers intuitive posture editing and move sequence editing (see Figure 47, for a list of movements), complete integration with machine behaviour using FSM, different sized human models and visual ergonomic feedback to help evaluate operator safety and fatigue. In addition the *V-Man* exports to MODAPTS sequences to support the current productivity team operations.

MODAPTS was chosen for the initial implementation of the *V-Man* module, but the code has been modularised to allow MTM to be integrated into the tools (see chapter 2).

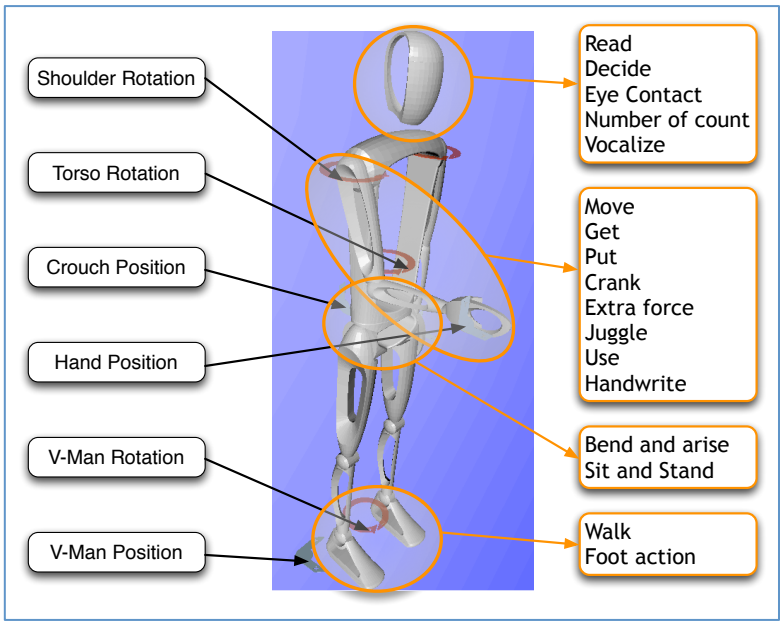


Figure 47 V-Man Manipulations

The CCE approach is to provide a lightweight approach to modelling human operation within an automation system (see Figure 46). The human (i.e. *V-Man*) operation is described by defining an FSM that outlines the process the *V-Man* will follow to complete the overall goal. The example shown in Figure 48 is a very simple FSM that describes the general process required to start a machine and return to the home position. This *V-Man* FSM may then be readily interlocked with other components within the automation system (in the same manner shown in Figure 42) whilst the *V-Man* operation for each state within the FSM will simulate each part of the operator's process.

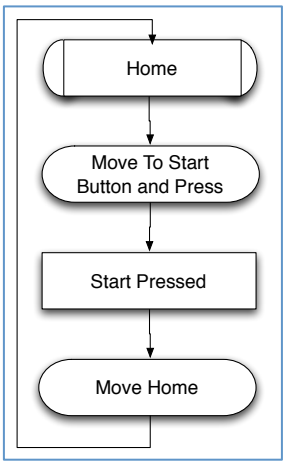


Figure 48 Simple V-Man FSM

By integrating the *V-Man* FSM with the automation component FSM, the interaction between the operator and the machine can be simulated. This interaction for Semi-Automatic machines is a particular area of concern for predicting the overall cycle time and operator loading (i.e. the percentage of the cycle time the operator is working) of the automation system as there are occasions where the operator is waiting for the machine and when the machine is waiting for the operator.

4.4.9.1 Ergonomic features

The CCE tool provide a limited set of ergonomic features:

1. **Ergozone** which defines the recommended working envelope of the operator.

2. **Anthropomorphic profiles** which allow three different sizes of human to be modelled. This enables the design of machine to be tailored to the size of operator or the process may dictate the size of operator required.

In order to comply with Ford's requirements, the Ergozone is taken from data within their standard practices (see Figure 49). Current tools such as Technomatix's Process Simulate Human [69] are more sophisticated in this area including a skeletal loading based upon the biomechanical models from the University of Michigan [68]. These models will be part of future work for the *V-Man* Models.

The Ergozone shown below is the safe "all day" working zone for an operator. This is currently used with a full size station mock-up and an operator following the written assembly procedure in front of the productivity team. As the operator progresses

through the procedure a measuring stick is used to check that the operators arms are within the zone and their posture is acceptable, the station may then be adjusted to suite. Whilst this works well for manual station design it is difficult to achieve this with semi-automatic machines cost effectively, as the automatic part of the machine needs to be tested and safe before the manual part may be tried out.

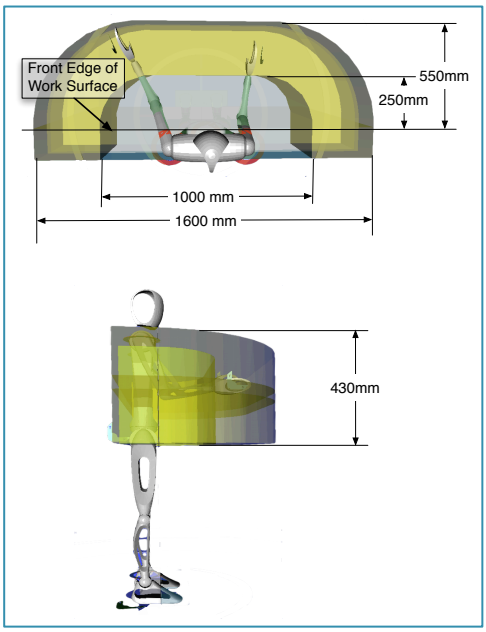


Figure 49 Ergo-zone taken from Ford standard practices.

This Ergozone has been implemented in the *V-Man* module. It consists of a translucent area attached to the users torso that may be used to visually ensure *V-Man's* hands remain inside the designated area.

4.4.9.2 Creating a V-Man Simulation

The *V-Man* editing screen is show in Figure 50. An FSM *View* is provided on the left hand side and selecting a dynamic state will display: (1) the corresponding sequence of moves *V-Man* will take and (2) set the position of the *V-Man* to the previous static state's position.

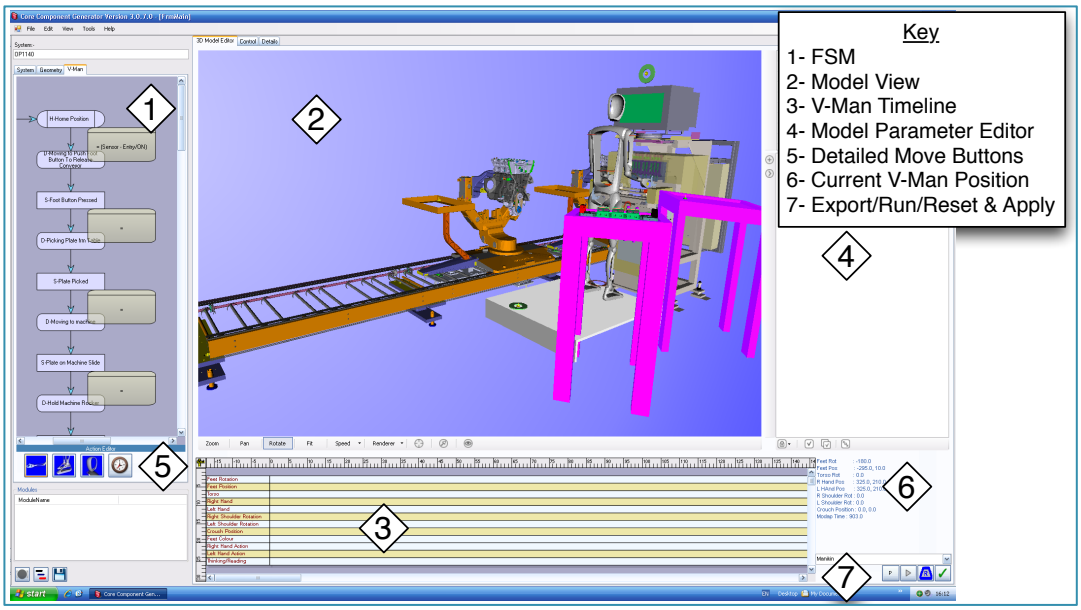


Figure 50 V-Man Module Screen Layouts.

To add a move to the sequence, the *V-Man* simulation is executed to ensure that *V-Man* will assume the correct final position of the sequence. The *V-Man* may then be manipulated to the correct position. Clicking on the V-Man timeline window will generate the move based upon what part has been moved and their move distances. Times for operations are based upon information defined by MODAPTS (i.e. the time is recorded in Mods). This procedure is repeated until the full sequence for the state is complete.

Detailed moves can be added manually by choosing one of the *Detailed Move* buttons to reveal the actions (shown in Figure 51) for the hand, foot and head operations.

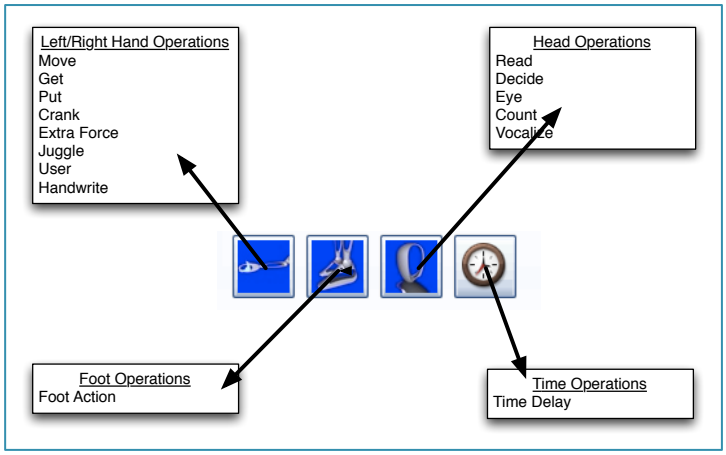


Figure 51 V-Man detailed actions

Each button brings up a list of actions. Selecting an action then displays a list of valid times (in Mods). Once selected the action can be added to the "timeline". These detailed actions cannot be viewed as part of the V-Man operation. Instead the relevant body part will change to green during the simulation to indicate it is doing something.

Time delays may also be input into the "timeline", which are not part of MODAPTS to make the simulation more lifelike. For example if the operator picks up a tool and waits for the machine to complete its action, then a time delay may be added from when the operator grasps the tool to when they actually raise it to the workpiece. Time delays must be used carefully so that they do not affect the cycle time unless specifically required to.

Finally there are "0" time actions (i.e. G0 and P0). These relate to putting and getting under "low conscious control" i.e. automatic response requiring little muscular control, no visual control and no mental control. For example a bank teller counting small change on the counter in which a finger touches the coin and slides it towards the other hand. The MODAPTS code will be represented as M1G0 M1P0 (M1 to move finger to coin G0 touch it then M1 slide coin and P0 release it). These actions proceed by selecting a move group and adding a detailed move from the list. The "0" time actions are displayed as a red ellipse on the timeline.

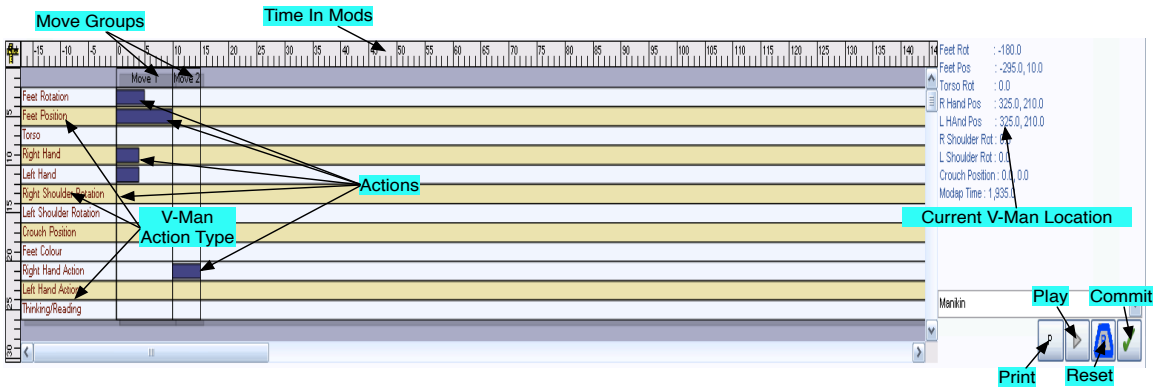


Figure 52 V-Man Timeline

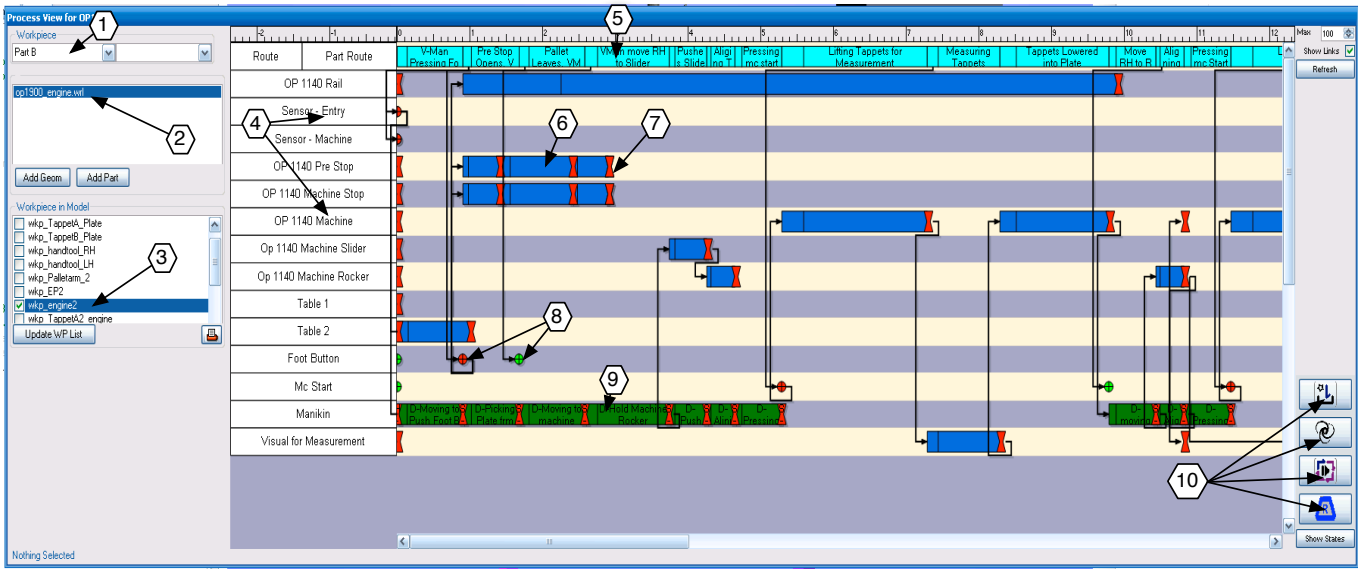
The *V-Man* timeline (see Figure 52) displays all the human movements and times. Each row within the timeline relates to a part of the body such as feet position / rotation, left / right hand and left / right hand actions. Each move group can have a list of actions that occur concurrently. These groups are displayed sequentially on the

timeline. On the right hand side are is a display of the current position of the chosen *V-Man*, and the buttons to print, play, reset and commit the sequence to the *V-Man* component.

Once committed to the automation system, the times and positions for the following states are adjusted to ensure that the overall *V-Man* sequence is consistent and correct i.e. if the finish position of the edited state changes, this affects the start position of the next state and may therefor affect the time for the *V-Man* sequence. As the times of the *V-Man* states are an integral part of the automation system, accurate cycle times may be calculated.

4.4.10 Simulation and Validation Module

The *Simulation and Validation* module calculates and displays the behaviour of the machine based upon its input stimuli (i.e. (1) from sensors triggered by an external source or based upon the workpiece routing (see 4.4.8) and (2) from the interlocks between components (i.e when an pallet arrives at a station it will trigger a sensor. If this sensor has been interlocked with a "tag reader" such that when the sensor is triggered it will read the tag and set some output, the logical sequence could be that when the tag is read AND the downstream path is clear, the pallet stop will lower to allow the pallet to pass. This simple high-level description of the system behaviour is mapped out in the automation system *Process Chart* (see below).



1 - Workpiece Selector
2 - Geometry for Selected workpiece
3 - Model Geometry to replace
4 - Component Rows
5 - Workpiece routing state
6 - Actuator dynamic state
7 - Actuator static state
8 - Sensor state
9 - V-Man state
10 - Buttons (Insert part, Cycle Start, Single step, reset)

Figure 53 Process chart example

In the example provided in Figure 53, each automation line consists of an *Actuator*, *Sensor* or *Manikin* (5), The times at which components change state it is marked upon the chart (6,7,8,9). *Sensors* are indicated with a coloured circle (8) whilst the static and dynamic states of *Actuators* / *Manikins* indicated by a narrow hour glass (7) and a process bar respectively (6). The top component of the chart is the workpiece route (5). The time for each dynamic state is dictated by the behaviour of the machine. For example in Figure 53 it can be see that the "Mc Start" component is linked to the "OP1140 Machine". This means that when the "Mc Sart" button is pressed the "OP1140 Machine" will start.

Process Charts also display the actions that are linked to a state change which are displayed by either clicking on the state: (1) to highlight what triggered it or (2) to display all component interactions for the simulation (see example inFigure 53).

The output of the Process Chart is based upon using knowledge inference techniques [71]. There are two types of inference commonly applied to the system. "Backward chaining" may be used to analyse what may have produced an outcome, whilst "forward chaining" may be used to predict an outcome based up input and rules. The timing chart uses a "forward chaining" strategy to predict that machine's behaviour.

The "forward chaining" method adopted allows real time update of the timing chart so that the effects of any edits can be seen in the actuation of the components and ultimately the cycle time. In order to determine whether the system will enter "livelock" or "deadlock" states, a timeout is used (i.e. if the nominal cycle time is expected to be 30 seconds and the calculated time runs beyond 50 seconds then a livelock situation is inferred, and if the workpiece route end is not reached then a "deadlock" may have occurred).

The method used to calculate the cycle progress chart is listed in the following sequence:

1. Each component (*Sensors, Actuators, Virtual components, Process Logic* and *Manikins*) is set to its initial state and the simulation time is set to 0.

2. The simulation time is checked to ensure it has not extended beyond the time allowed for cycle to complete. If it has the *Process Chart* creation is stopped.

3. The first state in the workpiece route is selected, which sets the state of its selected sensors, effectively simulating the insertion of a part.

4. Any component, which has interlocks that permit a change of state, will change state accordingly. These components are then put onto a "state expiry list" (i.e. dynamic states), sorted by their expiry time.

5. The workpiece route is then tested to see if its interlocks permit it to move to then next state.

6. Any components on the state expiry list that are set to expire are set to the states following static state as indicated by the FSM.

7. The workpiece route is then tested again to see if it may change state. If so its state will be changed.

8. The workpiece route is checked again to see if the expiring states allow it to move to the next state.

9. Finally a test is made to see if the workpiece route has reached its terminal state. If not the simulation time is set to the next expiry time of a dynamic state and the process continues from step 2 above.

For each stage of the above process, every state change will be displayed on the *Process Chart*, so the user may see the start and end time of every operation and an entry is made in the "simulation run collection".

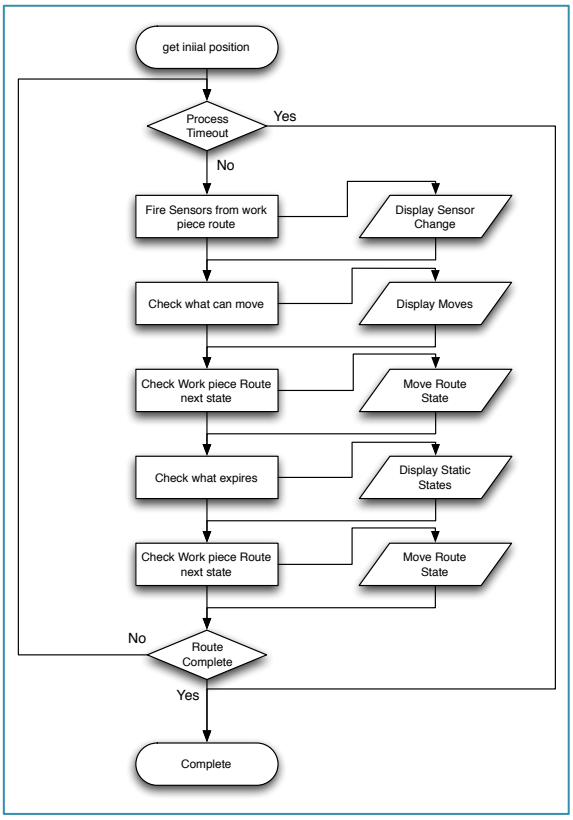


Figure 54 Flow chart of the *Process Chart* creation

The process outline in Figure 54 is called each time a component interlock or workpiece route is changed, so the user immediately sees the effects of the change in the process. Pressing "Insert Part" on the simulation *Process Chart* sets all the components to their initial position and when "Cycle Start" is pressed the process is started. Every 20 ms, the time is incremented and the "simulation run collection" is checked to see if a state change should have occurred. If a state change is required an event is raised so that any *View* integrated with the simulation engine can display this change. Note: the "simulation run collection" is a static list of state changes used by alternative *Views*.

4.4.11 Installation and Runtime System

In order to realise the benefits of creating a component based automation system using the CCE toolkit a system runtime is required. Originally the target platform for the CCE was LonWorks from Echelon, which is a proprietary system for building distributed applications that communicate over a robust network [43,53]. As LonWorks is not accepted by the automotive industry, a second runtime system was based on a centralised Schnieder PLC with distributed IO. As part of the SOCRADES project an experimental system was built using Schnieder Field Terminal Block (FTBs) [6] programmed using the C language and communicating over industrial Ethernet using HTTP and Web Services to describe the interface to the components.

The runtime architecture for the proposed component based automation system has been the topic of several theses [2,105,106,108]. The author has played a key role in the development of the "broadcaster" and the "marshaller" (shown in Figure 55) that control the distribution of messages and invocation of component functionality that are central to the operation of the runtime system. However, only the concepts behind broadcasting and marshalling operation are discussed in this thesis since the main focus is on the design, implementation and evaluation of the author's CCE toolkit.

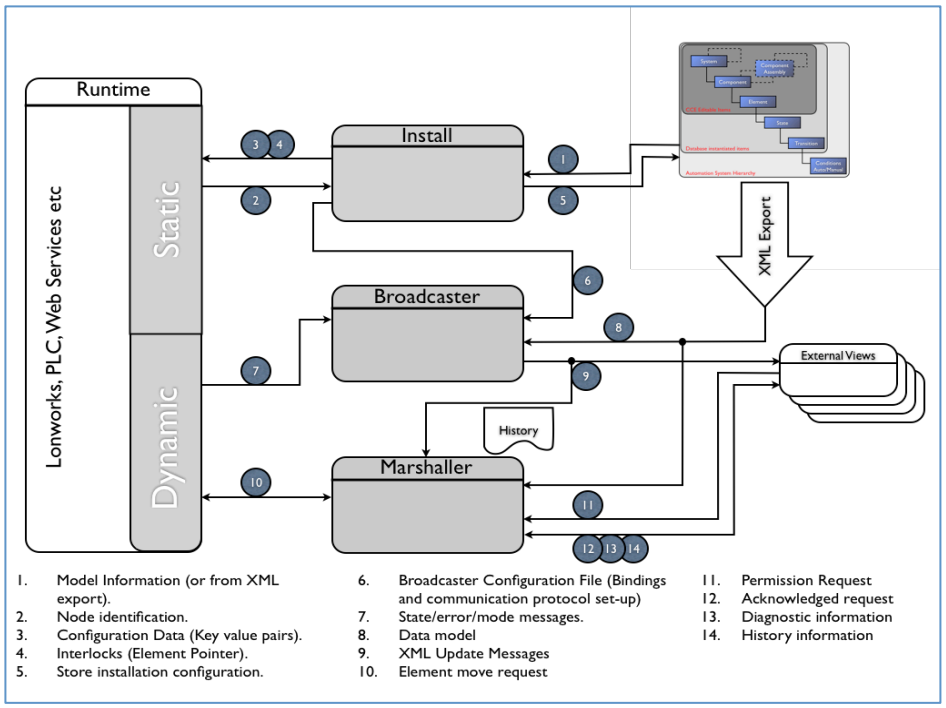


Figure 55 Installation and runtime schematic

When the system ready to be installed on the target platform, the CCE may export the complete XML definition of the machine for a 3rd party tool to interpret or a plugin module may use the "integrated model" directly for the install (1). The "install" program firstly identifies the "runtime components" (this may be a component on a PLC or a node on a network and is totally implementation specific (2)). The configuration data held by the component are then downloaded to the "runtime component" as standard key value pairs (3). These configuration data must be checked for conformance before download since the CCE Toolkit does not currently validate values in terms of type and range. Once all the components have been identified, the interlocks may be downloaded (4). It is vital that all of the components are identified as the interlocks need to reference a valid runtime component. The interlocks consist of a pointer to the state output of other components within the system. Partial (or hybrid) installations may be achieved in the runtime architecture. In this case the CCE tools are used to "emulate" "virtual" components that directly communicate with the "real" components.

This installation configuration will then be stored in the CCE to allow the recreation / reconfiguration of the system without the need to identify the components again (5). The install program configures the parameters (6) required to run the broadcaster and marshaller (8). The machine specific XML file is downloaded to the broadcaster and the marshaller to configure them for the chosen automation system.

4.4.11.1 Broadcaster

The *Broadcaster* is a stand-alone component that accepts and broadcasts state change information from many sources (e.g. the runtime system or a simulation). It has two roles. Firstly it holds a definition of the automation system (XML Definition) as well as the current state of all the components and the system (7). Secondly it broadcasts information and events using real-time data communication mechanisms to client *Views* (9). Corkill [71] identifies these as a blackboard (i.e. current state) and a knowledge source (i.e. the definition of the automation system and system updates).

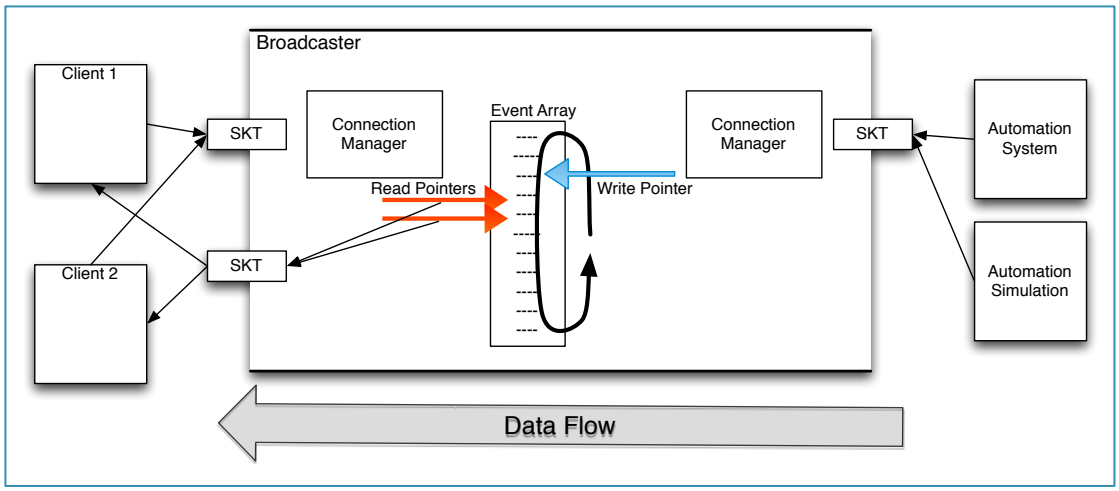


Figure 56 Broadcaster Schematic

The *Broadcaster* exposes a socket (SKT) listening for connection requests (see Figure 56). When a connection request is received from an event source, a new thread is created which effectively subscribes to the updates from the connected automation system and /or simulation. As the events are received from the sources the *Broadcaster*:

1. Validates the message.
2. Updates the current state of the machine (blackboard).
3. Increments to the FIFO (First In First Out) circular buffer write pointer to the next array location, starting again at 0 after reaching the buffer length.
4. Writes the update message to the array at the write pointer position.

When a client connects to the *Broadcaster* requesting subscription to the automation system updates, The connection manager creates a new socket dedicated to the client and: (1) creates a new read pointer, (2) sends the automation system definition to the client, (3) sends the current state of the automation system (i.e. the blackboard) and (4) propagates automation system changes. In order to propagate system changes the read pointer is checked periodically to ensure it has the same value as the write pointer, so when the write pointer is updated the read pointer increments and sends the update until the read pointer is again equal to write pointer.

4.4.11.2 Marshaller

The *Marshaller* is an internet-based dynamic web server implemented using Java server pages, Serlvets, Java Beans and Web Services. Its role is to provide controlled asynchronous access to both static and dynamic automation system information. The *Marshaller* is responsible for:

1. Connecting to the *Broadcaster* to receive events from the automation system.
2. Maintaining a history of events such as errors and machine downtime.
3. Providing authentication and security from the clients.
4. Marshalling information and commands from the clients to the run time components
5. Provide static information about the automation system

The *Marshaller* must support all requests from one or more end user HMI clients which can be implemented as either HTML pages using HTTP or the client can act as a web service entity and use direct HTTP interactions with *Marshaller* server applications. The clients request web pages from the web server, which may be either static or 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 clients request the dynamic scripts.

More complex end user HMI requirements require interaction with the server beyond the downloading of the basic web page (i.e. when an end user HMI requests a list of machine commands from a web page). The command list is send to the web server using a HTML request mechanism. The web server receives the request and must invoke an 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.

4.5 New Knowledge.

There are four main areas of new knowledge introduced in this chapter:

The design and implementation of non-vendor specific components that describe their behaviour in a simplistic way, yet may be used to describe complex systems that integrate mechanical behaviour and error/ diagnostic information (4.4.5).

The integration of a virtual human into an automation system, such that its behaviour interacts directly with the automation system logic through the use of an FSM (4.4.9).

The definition of a lifecycle automation toolkit such that each part of the toolkit (e.g. component builder, system builder or the simulation engine) is a View of the core "integrated model", to allow multidiscipline teams to collaborate and communicate via a common understanding of the automation system behaviour (4.4.1, 4.4.3).

The integration of a workpiece route into the automation system development process based upon engineering knowledge as well as a process engineers experience, to allow machine behaviour to be modelled without the complex use of virtual sensors (e.g. ray beam sensors) in the virtual model (4.4.8).

4.6 Summary

The design philosophy used when developing the tools, the implementation of the structural aspects of the toolkit (e.g. the *System Shell* and the "integrated model") and the development of the modules created to build libraries of reusable components from which automation systems may be configured have been outlined in this chapter.

The toolkit is described in terms of:

- the design philosophy detailed in the *System Shell*
- how software engineering techniques have been applied to the problem
- the database structure adopted for storing the common model along with the multiuser aspects of the toolset

- the toolset functions in terms of component building (including the workflow required to extract 3D information from existing CAD and adding kinematic information)

- building reusable modules from prebuilt components and developing automation systems out of component / modules (including adding interlocks to describe the automation system behaviour)

- workpiece routing to examine / simulate the machine's behaviour as workpieces are inserted into the system

- the integration of the human in to the machine behaviour (V-Man)

- the simulation engine showing the dynamic update of the *Process Chart* using workpiece routing and machine behaviour

- the runtime architecture and installation procedure, along with the *Broadcaster* / *Marshaller* and their possible use for remote diagnostics / control.

There are many aspects to the toolkit each of which have initiated additional research areas (such as the development of the real time 3D model and the business analysis of the automation system design process).

Specifically the objective of this chapter was to address the following research questions:

1. *Is it possible to support a multidiscipline team working to develop an automation system whilst providing a common understanding of the machine layout, performance and operation?*

   The research discussed in this chapter has lead to the development of a well-defined, simple yet effective method of describing automation system that teams of engineers may use throughout the system lifecycle. Specifically the development of the extensible shell and integrated data model provide the core of the system from which different views can be presented to allow the common understanding. It should be noted that the tools do no attempt to

replace current CAD systems but compliment them by allow multidisciplinary teams of engineers to gain the benefit of 3D visualisations directly linked to the automation system behaviour and ultimately its control.

2. *How can the toolkit support the development of "reusable components" that are not vendor specific in terms of hardware and software?*

   The development of components that are described solely in terms of their state behaviour and parameters provides the mechanism of generalising the development of the automation system whilst allowing the specifics to be addressed (in a common manner) in the implementation of the vendor specific runtime system.

3. *How can human behaviour be integrated with machine behaviour, without adding so much complexity to the system that it becomes either specialist or unwieldy?*

   The use of FSMs to describe the operator's sequence of operation creates a flexible mechanism of connecting operator actions directly with the machine behaviour. This linked with reverse kinematics of the human model / and manual move entry that conforms to the industry standard MODAPTS methodology provides a quick, simple, well-defined and accurate method of building automation systems with minimal training and hardware requirements.

Chapter 5 **Case Studies**

This chapter contains two case studies. Each one evaluates the use of the CCE Toolkit for different goals. The case studies are qualitative in nature, in that they validate the engineering approach, and that the component based engineering tools proposed in this thesis can functionally meet the requirements set out in Chapter 2.

There are two case studies detailed in this thesis. The first case study allows the engineering tools to be evaluated when creating and reconfiguring a component based automation system; the second allows the evaluation of an automation system with the V-Man integrated with its behaviour, as part of a real machine evaluation process.

The first case study "Festo Test Rig" uses automation system hardware designed for training and education, to ensure the validity of the controls system output obtained from the CCE tools/ approach. Further to this it assesses the agility of the approach by inserting a new station into the automation system. This not only checks the reconfiguration of the system, but shows how a component breakdown can simplify the understanding and alteration of a complex system.

The Second case study evaluates the CCE Tools / and a lightweight viewer for use as a virtual simulation tool, which is currently in use at Ford Motor Company as a virtual engineering collaboration tool for power train automation system development. This includes the process of "data harvesting" machine definition from 3D CAD, building the virtual models and presenting them at the collaboration meeting (referred to as a Virtual Build Event) and finally a review of the predicted simulation performance of the machine as compared to the installed "real" version of the machine.

5.1 Case Study – Festo Test Rig

5.1.1 Introduction

In order to evaluate the capability of the CCE Tools to simulate/ and control an automation system, a representative system is required. In order to appraise the use of CCE tools, the runtime system has to be in a controlled environment so that experiments can be made and repeated. The Festo Didactic Test Rig (Festo Test Rig)[51] was chosen for this purpose, as it a laboratory based system that is representative of a real automation system used in the automotive industry and captures many of the issues involved in building them. The Festo Rig is also used to test new control systems by Ford and train their MSc course students in automation and control. This rig provides a realistic automation problem so that effective testing and evaluation of the CCE runtime system may be achieved. There are a myriad of ways that can be used to build a machine, from using tools such as Delmia Automation to build the model and potentially generate the code to run the rig, or a more traditional approach of creating a CAD model of the system, generating a process chart to describe its operation and subsequently building the control code to run it. As a result this case study addresses the qualitative aspects of the building of automation systems and not the quantitative aspects. Future work should include a quantitative study of the benefits of using the CCE tools over both traditional and alternative state of the art methods of building automation systems.

The case study will address the following questions.

5.1.2 Research Questions

The Festo Test Rig is used to define the answer to the following research questions.

5.1.2.1 Research Question 1

Can the component-based toolkit express the functionality of the components of the automation system in such a way that the behaviour of the overall system will meet or exceed current practices? See Chapter 3.

5.1.2.2 Research Question 2

What are the benefits of using the proposed methods to create an automation system over current practices?

5.1.2.3 Research Question 3

How does using the component-based toolkit aid in the reconfiguration a machine over current practices?

5.1.3 Festo Test Rig Description

This test rig is shown in Figure 57 and reflects the functionalities of assembly machines typical of line installations at Ford, Jaguar, and other companies supported by their principal machine builder, Krause. The basic operation of the test rig is to move and process a workpiece from one end of the machine to the other by performing a number of operations such as picking, moving, drilling and gauging. An important characteristic of the system is that several operations have to occur simultaneously.

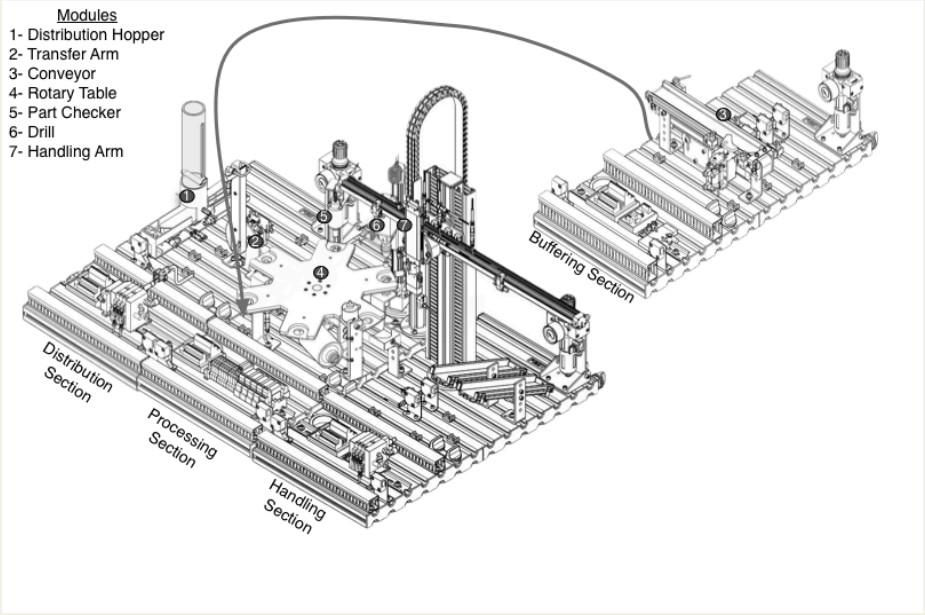


Figure 57 Festo Test Rig, showing the initial system and the insertion of the buffering station to replicate a typical engineering change to a system.

The rig has been implemented using three target alternative control platforms, 1) a centralised Schneider PLC with distributed I/O, 2) distributed Schneider Field Terminal Blocks with Web Services, and 3) a Siemens PLC ustilising Profinet-based distributed IO. Below is an example of the Festo Test rig configured with the Schneider PLC and distributed I/O.

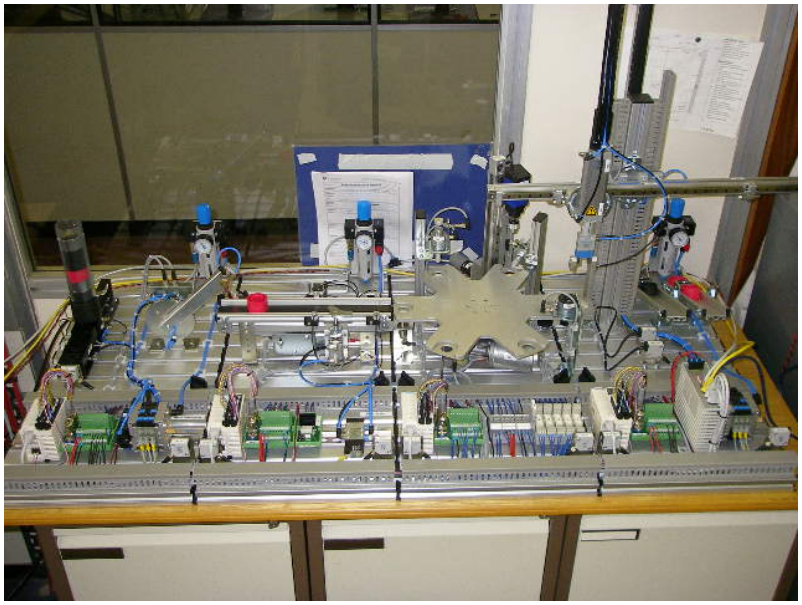


Figure 58 Festo Test Rig implemented on Schneider PLC with Distributed I/O (shown is the complete 4 station version)

The current definition of a component in section 2.4.1, but for clarity this definition will be outlined again. This thesis considers a component as *a unit of composition that has a contractually specified interface and explicit dependencies only, which can be deployed independently and is subjected to compositions to build automation systems, by providing a service, validation, error definition / reporting, and design/maintenance information.*

This definition of a component has been implemented as the encapsulation of the following aspects:-

1. Functional aspects; Controls interface (to allow the configuration of the component to meet specific requirements (E.g. travel speed or acceleration)), Control (internal software implementation of the component operation (E.g. the PLC code that is used to control the physical component) and Control behaviour interface (E.g. the external interface to allow the component to

interact with the other components in the automation system). For this we expose a state behaviour of the machine.

2. Knowledge; manufacturer details, best practice, diagnostics, error handling, lifecycle management and configuration data.

Note: A component may not have a physical representation (such as a timer or sequence lock). However, they are still stored as reusable components within the component library.

These components are then built into modules as detailed in Table 9. A module is defined in this thesis as *a set of standardized independent components that can be used to construct a more complex structure.* A module will comprise of one or more components and encapsulate the physical, functional and knowledge aspects of the contained components, along with the details of their assembly.

The predefined modules and components have then been built into a system.

5.1.4 Festo Rig Component Breakdown.

The Festo Test rig is broken down into modules as displayed in Figure 57 and Table 9, these seven modules, along with graphical items such as the base plates and frames form the basis of the automation system. The table below indicates the requirements in terms of the modules, components, their I/O requirements and the implementation details for the test rig. Each alternate module is shaded to highlight the comprising components, the module numbers in Table 9 relate to the numbers shown in Figure 57.

Module No.	Module Name	Component Name	Digital Inputs	Digital Outputs	Implementation Details
1	Distribution Hopper	Eject Cylinder	2	1	Actuator – 2 Position 4 State (Translation)
		Magazine		1	Sensor – 2 State
		Xfer Ready		1	Sensor – 2 State
2	Transfer Arm	Arm	2	2	Actuator – 2 Position 4 State (Rotation)
		Gripper		2	Actuator – 3 Position 3 States (Colour)
		Vacuum Sensor	1		Sensor - 2 Position
3	Conveyor	Conveyor Drive		1	Actuator – 2 Position 2 State (Colour / Translation)
		Separator	2	1	Actuator – 2 Position 5 State (Rotation)
		Workpiece Available Sensor	1		Sensor – 2 State
		Separator Sensor	1		Sensor – 2 State
		Conveyor End Sensor	1		Sensor – 2 State
		Separator Sequence Lock			Virtual – 2 Position 3 State
4	Rotary Table	Indexing Rotary Table		1	Actuator – 2 Position 2 State (Rotation)
		Ejector		1	Actuator – 2 Position 2 State (Rotation)

		Workpiece Available Sensor	1		Sensor – 2 State
		Workpiece at Checking Unit			Sensor – 2 State
		Workpiece at Drilling Unit			Sensor – 2 State
		Checking Unit Cycle Complete			Virtual – 3 position 3 state
		Drilling Unit Cycle Complete			Virtual – 3 position 4 state
5	Part Checker	Checker	1	1	Actuator – 3 Position 5 state (Translation)
6	Drilling	Drill Unit Axis	2	2	Actuator – 2 Position 5 State (Translation)
		Drill Spindle		1	Actuator – 2 Position 2 State (Colour)
		Workpiece Clamp		1	Actuator – 2 Position 2 State (Translation)
7	Handling Arm	Delivery Arm	3	2	Actuator - 3 Position 6 State (Translation)
		Gripper Extend		1	Actuator – 2 Position 5 State (Translation)
		Gripper		1	Actuator – 2 Position 2 State (Translation)
		Workpiece is Not Black	1		Sensor – 2 Position 2 State
		Workpiece Receptacle	1		Sensor – 2 Position 2 State

Table 9 Module/ Component breakdown for the Festo Test Rig.

As can be seen from the table there is are a total of 23 digital inputs and 18 digital outputs spread over 28 Components (14 Actuators, 11 Sensors and 3 Virtual). Whilst

this is a modest size of automation system, most of the common issues regarding the building of systems can be tested.

The definition of the complete system is given in Appendix 1, but a description of two modules of the rig are included here, i.e. the Distribution Hopper and the Handling Arm. Examples of two virtual components (Time Delay and Sequence Lock) are also provided below.

The Distribution Hopper was chosen as a simple module, that is easily described, that may be used to describe the process of building and interlocking components. The Handling Arm was used to describe how a more complex FSM might be built using the tools. The complexity of the Handling Arm arises from the fact the two of its three static positions are configurable, and that the arm may be configured to move from any position to any other position. The virtual components were chosen as they represent frequently used examples of such virtual components.

5.1.5 Scenarios

Two scenarios have been selected to illustrate the answer the research questions; firstly an automation system has to be created using the tools to define its behaviour (**Stage 1**). This involves building the graphical library for the rig in VRML, assembling the graphical items to create components, adding kinematics to animate their behaviour, and creating the FSM's that will define the operation of the component in the assembly (as displayed in Figure 59). Modules are then built that contain all components and associated geometry to describe a reusable section of an automation machine (For **Stage 1** the modules Distribution/ Processing and Handling sections were built (see Figure 57 and Table 9)).

These modules and components are then built into an automation system, where the component interlocks are added to prevent components from colliding and to allow the manipulation of the workpiece through the machine. To exercise the behaviour of the machine, a workpiece route is added, that describes the flow of a workpiece as it progresses through the machine.

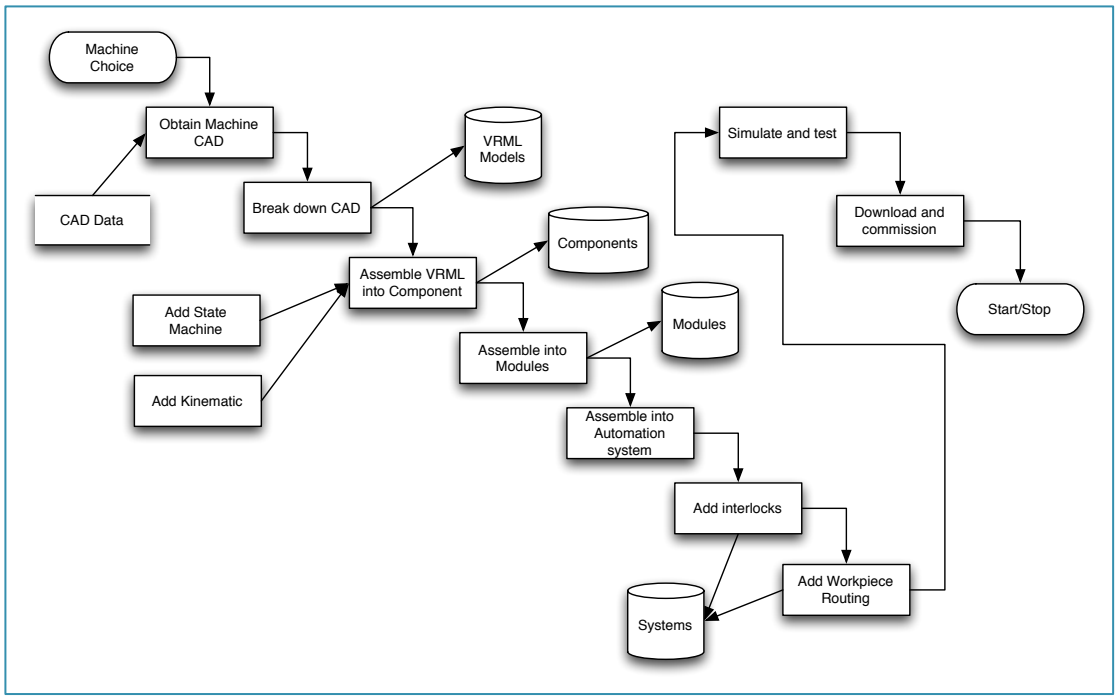


Figure 59 Automation system creation flow chart

Once the simulation and validation of the automation system is complete the software may be exported and downloaded to the target runtime system (for example a Schneider Electric PLC running the Unity programming environment). The runtime system is then commissioned and run on the Festo Rig, and the operation and timing is compared with that of the simulated machine.

Stage two is used to evaluate the agility of the system to react to unforeseen changes, by adding an additional module to the system created at stage 1 to insert a buffer section between the distribution and processing sections. This will allow evaluation of the capability of the tools to support implement changes for an automation system. The components for the new module were created and built into the buffer station module in the same way as in Stage 1. This module was tested as a stand-alone system before being integrated into the automation system built in Stage 1.

The integration involved identifying the Stage 1 components that the new module would impact on, i.e., the Conveyor and the Rotary Table. The interlocks were then removed from these components and the geometry was altered.

5.1.6 Stage 1 Building an Automation System.

The creation of two representative modules is described in this section, a simple module with a fixed two-position actuator and two sensors, and the other a configurable module with relatively complex behaviour. The integration of these and other modules to build a complete automation system is described, along with the workpiece routing creation to provide a real-time simulation.

5.1.6.1 Distribution HopperModule

The distribution hopper functions is a workpiece feeder (module 1 in Figure 57 and displayed in Figure 60). This component has a tube (bin) that holds a number of workpieces ready to be processed. There are two sensors one indicating that a workpiece is available in the bin (Bin Sensor) and the other indicating that there is a workpiece ready for transfer from the component (Xfer Ready). Finally there is a single two-position (four state) actuator that moves a workpiece from the bin location to the ready for transfer location.

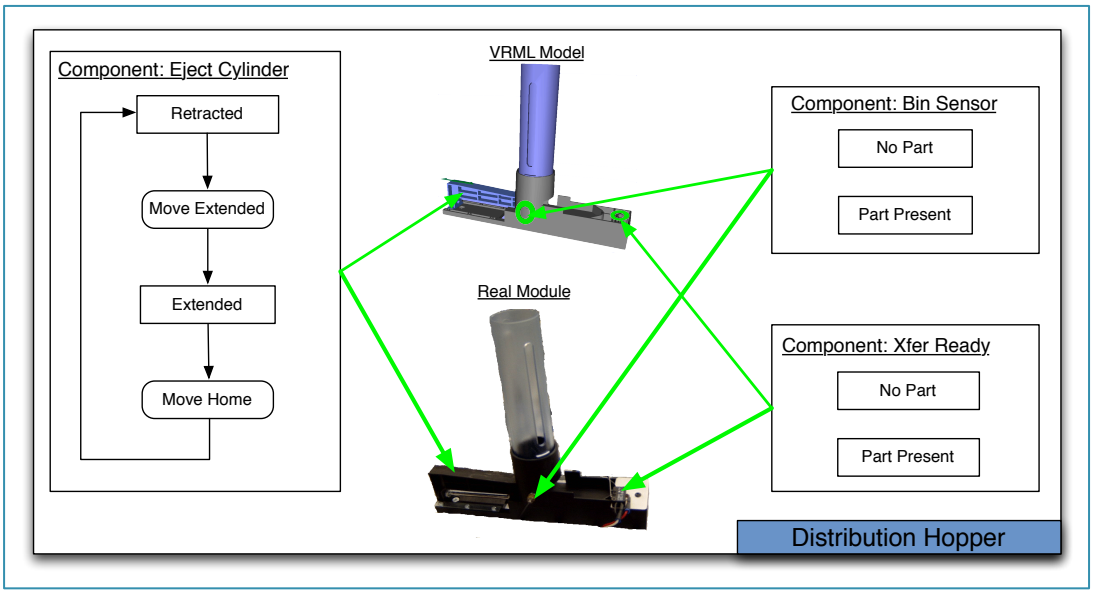


Figure 60 Module Distribution Hopper

5.1.6.2 Component Creation

To create the Distribution Hopper Module three components have to be created as shown in Figure 60 (Eject Cylinder and the part present sensors, Bin Sensor and Xfer Ready). To illustrate the process of creating a component, the generation of the "Eject

Cylinder" is described below. The same procedure is used to create all types of components.

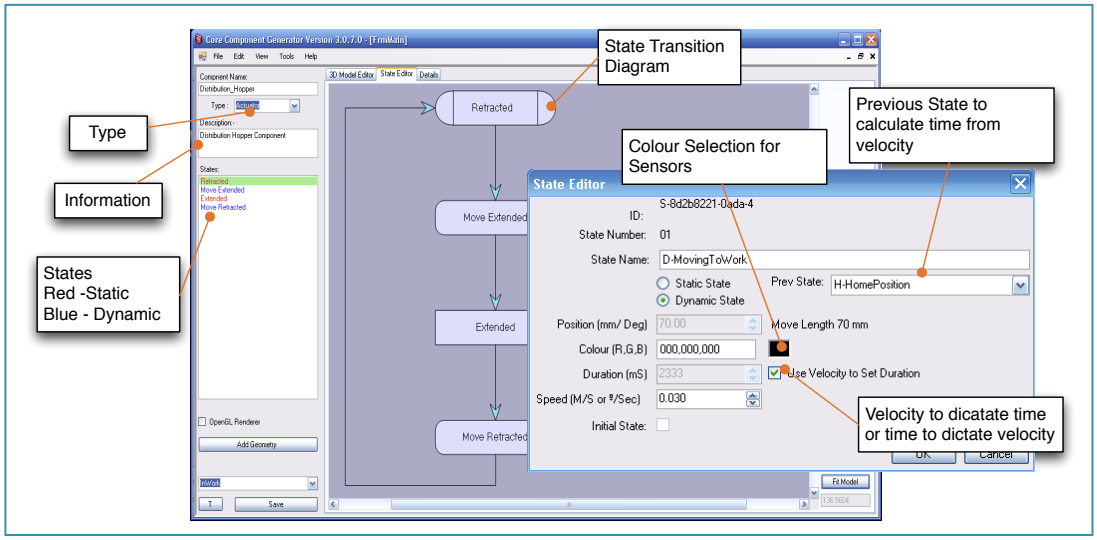


Figure 61 Create Finite State Machine screen layout showing state configuration parameters and component information.

The diagram in Figure 61 shows a screen shot of the CCE component FSM creation process. Each component has a name and description, and any sensor, actuator or manikin also has a FSM associated with it. States and transitions can be progressively created. Details of each state are defined including:

1. Type (i.e., dynamic or static).
2. Position (if it is a static state).
3. State colours (if the component represents is behaviour using colour). Such components are generally sensors or items that have little or no visible movement, where the colour change is used to indicate a state change.
4. Duration in milliseconds.
5. Speed in meters per second.
6. Initial state.

The Eject Cylinder has two static states, Retracted with a position of 0mm, and Extended with a position of 90mm. The state shown in Figure 61 is a dynamic state,

i.e., the eject cylinder moves from one static state to another, and the velocity or time for the move can be defined. Finally there is an initial state indicator, there must be only one initial state, and it must be a static state.

As described previously the behaviour of the component is completely described using state behaviour, and associated parameters and error definitions, from this definition a number of views of the automation system may be created. The main view required is a virtual simulation of the component, which is described below.

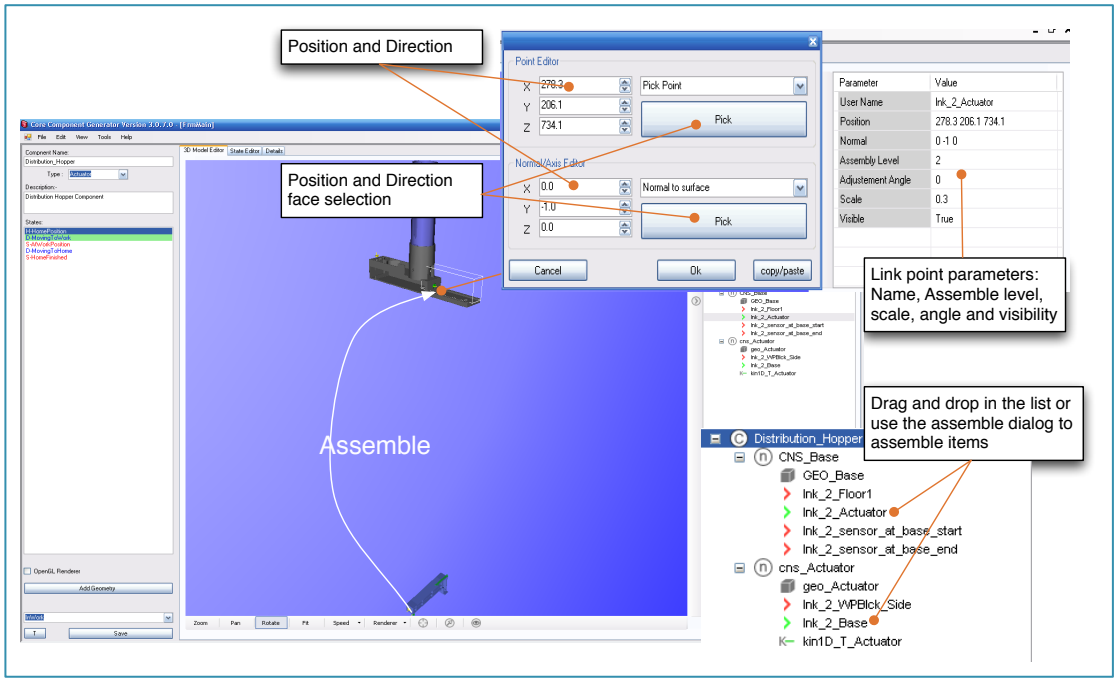


Figure 62 Component assembly from Exported CAD VRML models

Figure 62 shows the two constituent modelled parts of the "Eject Cylinder" (the "Base" and the "Actuator"). The lower left of the diagram shows the tree view of the model with the two constructs ("CNS_Base" and "CNS_Actuator") and their geometry. Each of these has a number of link points assigned to them, the properties of which are shown in the top right of Figure 62.

Once defined the link points may the then be used to assemble the components by dragging the child construct's link point onto the parent's link point to build a parent child relationship between the two constructs and define the relative location of the

child construct to its parent. i.e. defines the location of the "Actuator" relative to the "Base". This assembly and hierarchy can be seen in Figure 63 below.

In order to animate the models for simulation purposes kinematics are created. A kinematic is associated with a construct within the VRML model. This kinematic may be either a rotation or translation, dictated by the kinematic type, the type shown is a translation. Associated with a kinematic is the kinematic handle, which allows the user to "try out" the movement by clicking and dragging the handle to move the geometry. The location of the handle and the direction of the move may be defined using the dialog shown below (similar to the link point dialog box).

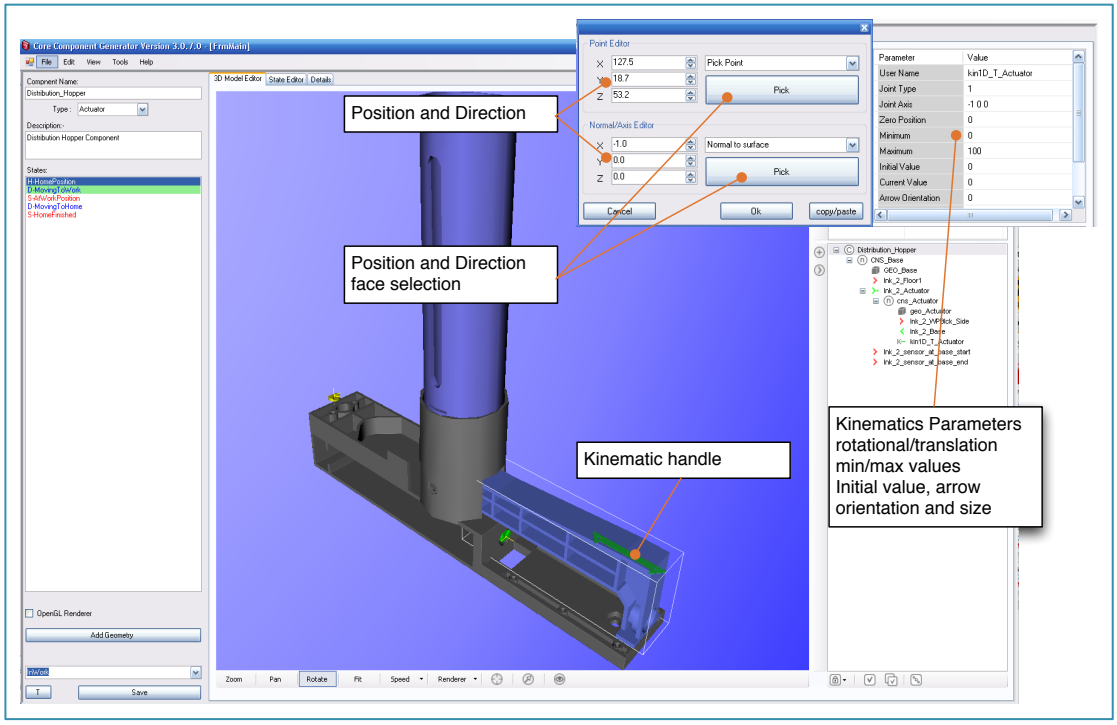


Figure 63 3D Kinematic creation.

Once defined the behaviour of the component may be exercised by clicking any of the valid states on the left of the window, to ensure it is correct operation.

The Eject Cylinder component has two embedded sensors used to sense when the actuator has reached either of its end positions. Such embedded sensors are not exposed to other components but just used for internal control; however, they can be used by the component to indicate that an error has occurred. For example if the Eject

Cylinder actuator takes more than a specified time to complete a movement then a Timeout Error is raised or if both of the embedded sensors are in the on-state simultaneously then this would indicate a sensor fault, as the actuator cannot be in two positions at once, and a Pairs Check Error is raised. This error handling is embedded into the component so that, when errors occur, they can be handled by the runtime system and propagated to the HMI and other relevant views as appropriate. Runtime errors are entered into a list associated with the component.

5.1.6.3 Module Creation.

Once the components are created for the assembly they can either be built into modules or directly into systems. A module can be defined as *a logical grouping of components that achieve a defined goal.* The distribution hopper comprises three components:-

1. The Eject Cylinder.
2. The Bin Sensor.
3. The Xfer Ready Senor.

Figure 64 below shows the assembled Distribution Hopper, created in the Module Editor (assembled using "component-level" link points described earlier). The left of the screen shows the component list and this includes a workpiece that will appear in the routing logic described in 5.1.6.5.

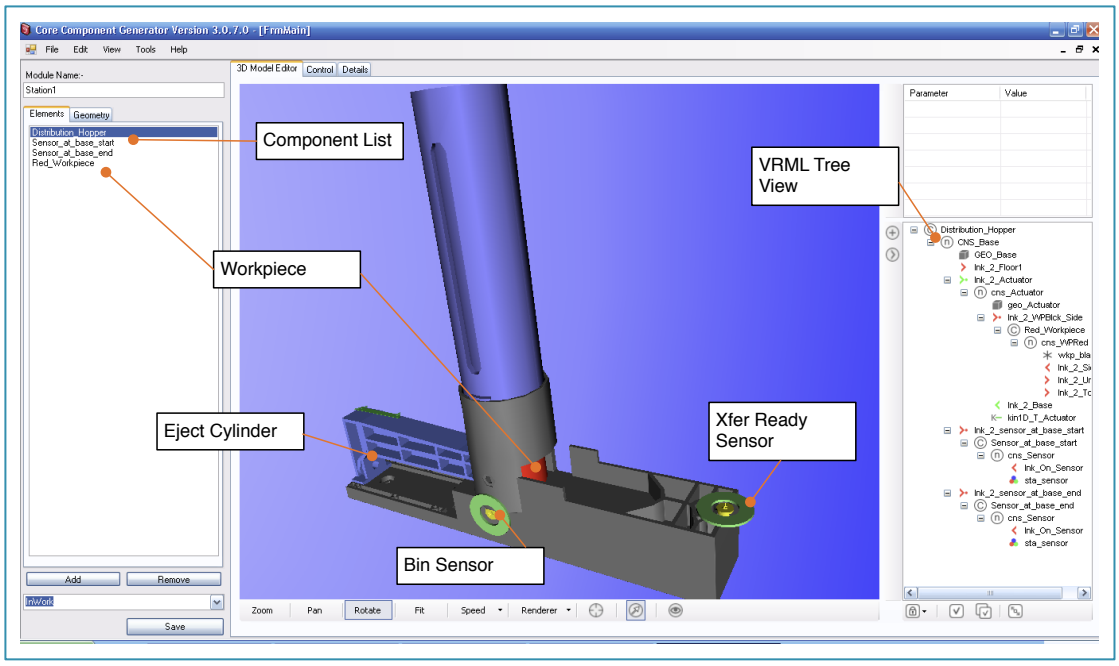


Figure 64 CCE screenshot illustrating the creation of a module from components

Each module created is categorised and stored in the library, with associated documentation, for inclusion into future automation systems as needed.

5.1.6.4 Automation System Creation

The module described above may be built into a stand-alone automation system, and its behaviour, defined using the interlocks shown in Figure 10, simulated using the System Editor. The Eject cylinder's initial position is Retracted as shown in the figure, it is interlocked with the Bin Sensor and the Xfer Ready sensor such that if there is a part present in the bin and there is no part at the Xfer Ready position then it will change to the Move Extended state. Once at the Extended state the Eject Cylinder will wait until the Xfer Ready sensor indicates there is no part present then the cylinder will change to the Move Retracted state.

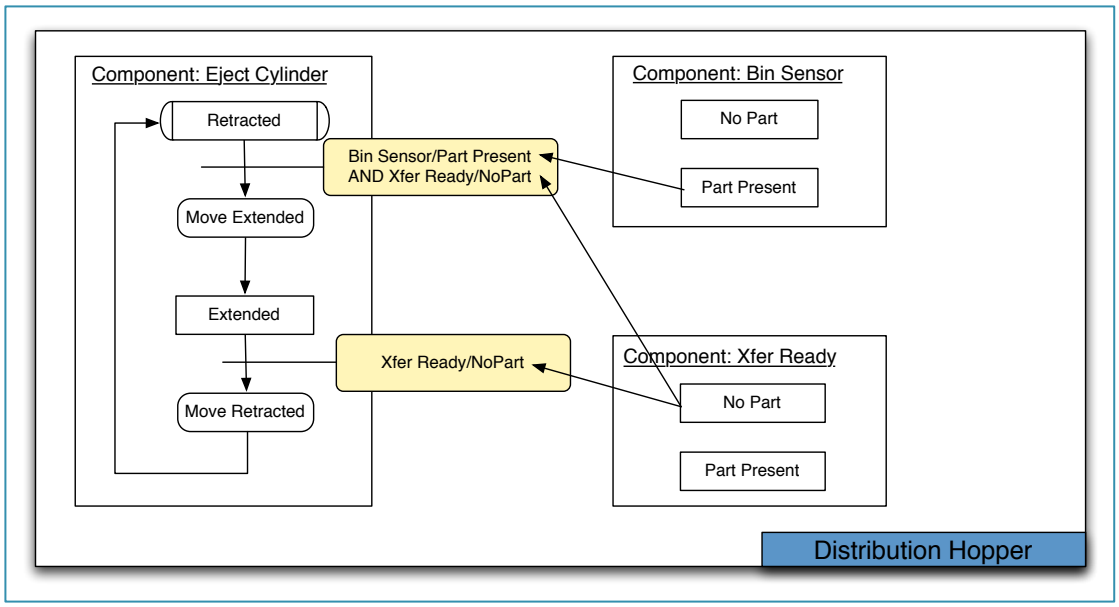


Figure 65 Distribution Hopper Stand alone Interlocks – Component Builder (Ref Chapter 4.4.5)

To create the interlocks in Figure 65 the Distribution Hopper module has to be added to the new system from the module library. Adding this module adds all the components of the virtual assembly. Figure 66 illustrates how the interlocks for any given component can be viewed or edited. Using a drag and drop approach OR and AND conditions can be progressively added associated with each transition.

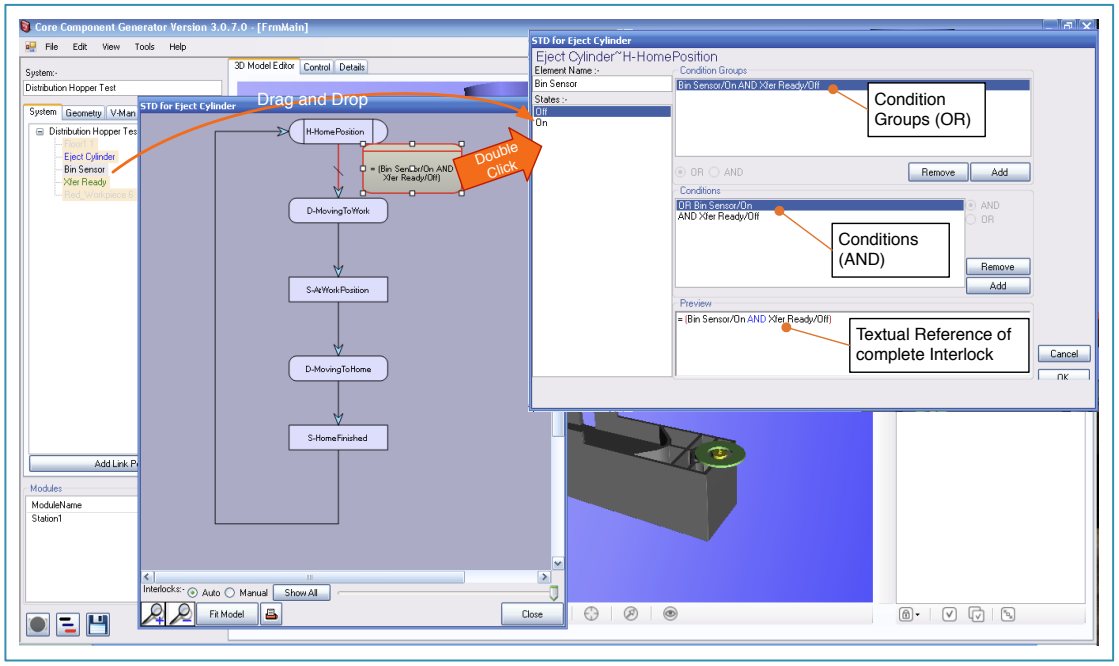


Figure 66 Using drag and drop to add component interlocks that define the automation system operation.

Once all the interlocks have been created the logic may be exercised using the routing logic and simulation engine, described below.

5.1.6.5 Routing Logic and Simulation Engine.

The routing logic defines how the workpiece interacts with the control system as it progresses through the automation system (as described in Chapter 3 Workpiece Path Editor Module). Figure 67 below shows the workpiece routing for the Distribution Hopper test system created as described above.

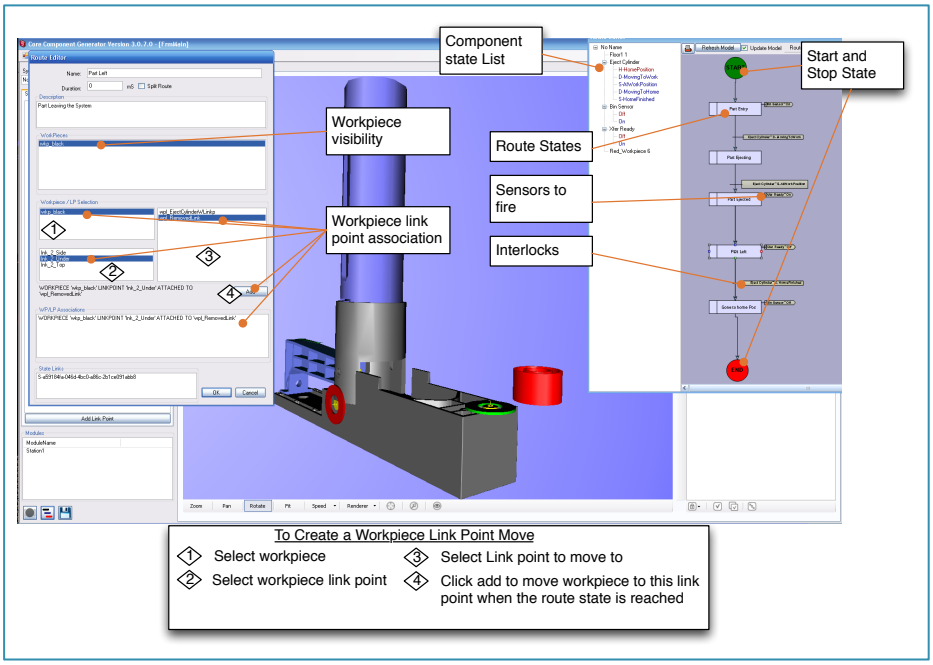


Figure 67 Create workpiece routing

A system may have many routes associated with it. As shown in Figure 67, the green and red circles define the start and stop states in an example route. Drawing boxes in the editor creates the route states and interconnecting them with lines as shown in the figure dictates the flow.

Each state of the route can "fire" sensors when the route state is reached. Dragging the component states from the list and dropping them onto the interconnecting lines between the route states creates an interlock. Workpiece visibility and link point assignment can be defined at each state of the Workpiece routing as required.

E.g., in the Festo rig the workpiece is pushed out from the Distribution hopper and the transfer arm moves over to grab it and at that point the workpiece is transferred from the Distribution Hopper Eject Cylinder to the Transfer Arm Gripper.

5.1.6.6 Handling Arm

The Handling displayed in Figure 68 arm comprises of 5 components. The Delivery Arm a three-position (6 State) gantry arm, a vertical two-position (5 state) Gripper Extend actuator providing vertical movement. On the end of the Gripper Extend actuator there is a workpiece gripper with an integrated colour sensor, that indicates if the work piece is black, or not.

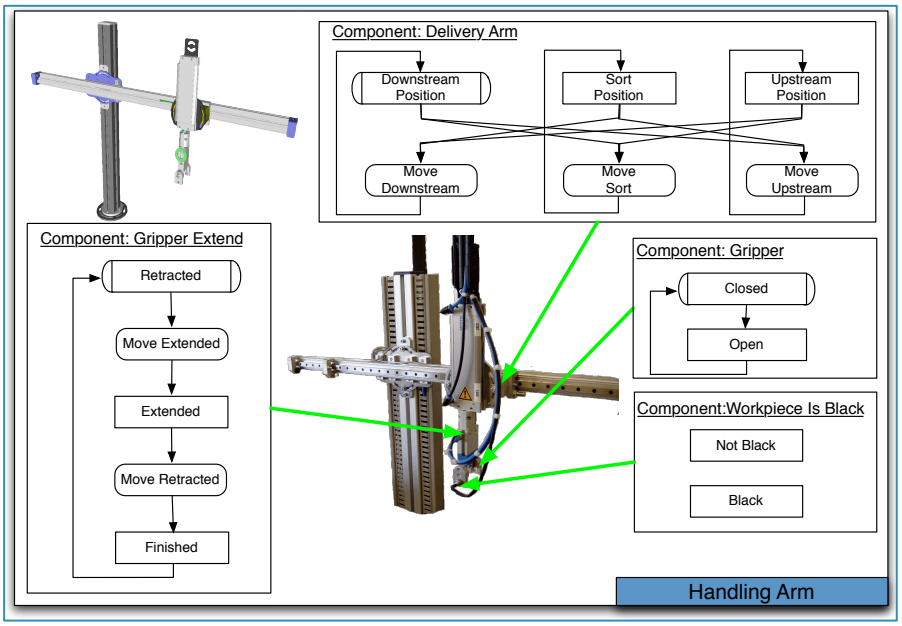


Figure 68 Handling Arm.

The Gripper Extend component is very similar to the eject cylinder on the distribution hopper, apart from the geometry, distance moved and the Finished state to indicate that it has completed a complete cycle. The Finished state is required to allow the Gripper to be interlocked with the gantry such that when the Delivery Arm is in the upstream position (left most position) and the Gripper Extend is in the Retracted position the Finished state will indicate that the Gripper Extend has cycled so it should not cycle again until the Delivery Arm is moved. This Finished state has been introduced to support the best practices of the machine builder. In particular, the Finished state has been introduced to many components to simplify the interlocking. By standardising on the state definition the same run-time code from the Eject

Cylinder can be reused to create the run-time Gripper Extend component, increasing code reuse to aid the creation of more robust components.

The gripper component is a spring-return actuator and moves so rapidly that a dynamic state from closed to open was considered to be of little practical use. Note that the gripper is normally in the closed state; this is a recognised safety feature in that if the power to the system fails then the gripper will remain closed..

Lastly there is an emissivity sensor called "Workpiece is Black" that detects the amount of light emitted from the surface of a workpiece. This sensor is interlocked with the Delivery Arm so that parts may be sorted, based on their colour, and placed at the appropriate exit point.

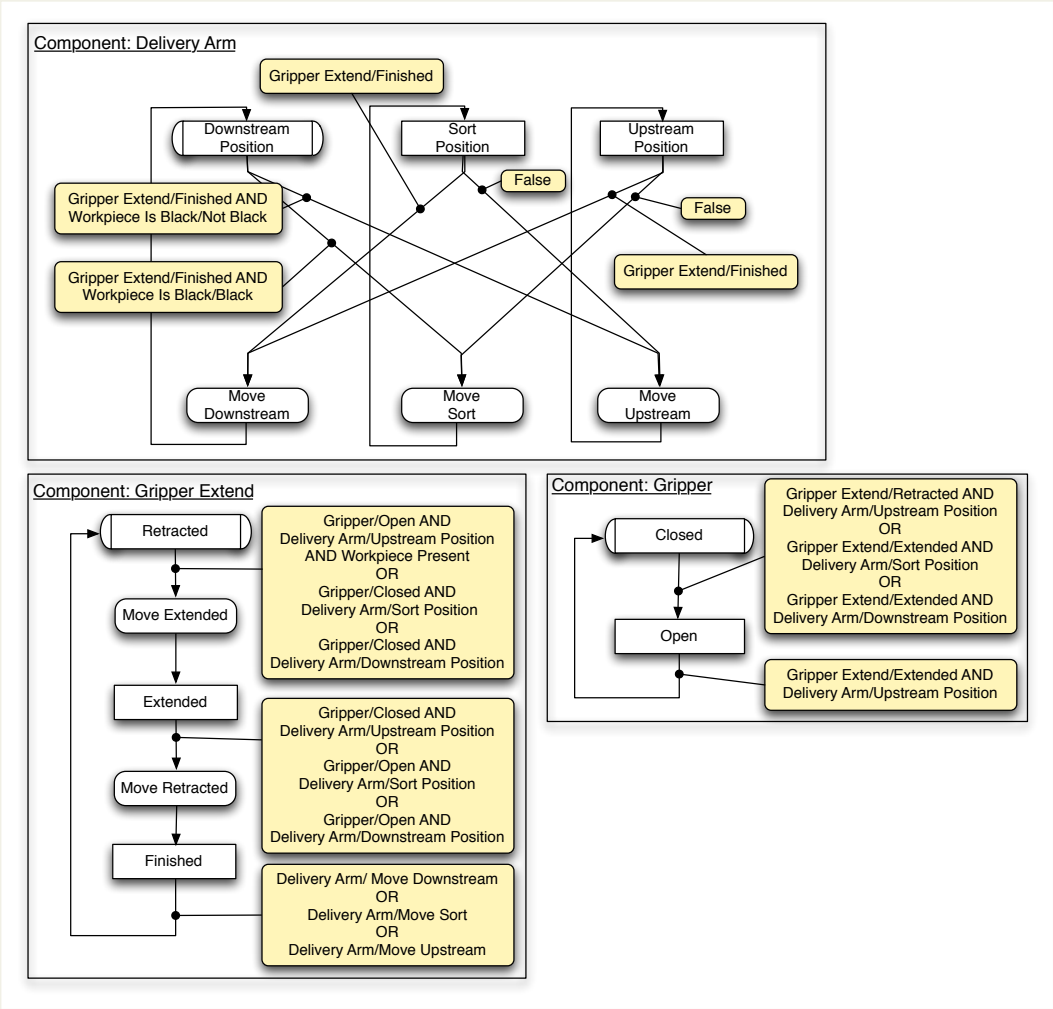


Figure 69 Handling Arm component FSM's and Interlocks

The state diagrams for the actuators in the Handling Arm components are included in Figure 69. The interlocks shown are the Automatic interlocks that provide sequence to the Arm behaviour.

As previously explained the Gripper starts at its closed state and will only open if the Delivery Arm is upstream and the Gripper Extend is in the retracted position (waiting for a workpiece). When a Workpiece arrives the Gripper Extend will then move to Extended, once extended the Gripper will close (grabbing the workpiece) and the Gripper Extend will retract and wait at the finished state.

Note: the Gripper Extend will be in the Finished state until the Delivery Arm has moved. The Delivery arm will move to the Sort Position or the Downstream Position only if the Gripper Extend is in the Finished state.

When the Delivery Arm reaches either the Sort or Downstream Position and the Gripper is closed, then the Gripper Extend will again move to the Extended Position. This time as the Delivery Arm is in the Sort/Downstream Position so when the Gripper Extend reached Extended the Gripper will open (releasing the workpiece). When the Gripper is Open the Gripper Extend will Move to Retracted and wait at the finished state for the Delivery Arm to Move Upstream allowing the Gripper Extend to move to the Retracted state and the cycle will start again.

Describing this behaviour in a simple generic way, using FSM's, encapsulates the complex behaviour of the module. The resulting process definition and visualisation of the system may be easily understood by a wide range of engineers, without in-depth knowledge of PLC programming, whilst the underlying code may be written by experienced control engineers to ensure the run-time components are optimised and robust.

5.1.7 Stage 1 - Automation System Deployment.

The first stage of system build involves the addition of components as per section 5.1.6.4. The component interlocks are added and the workpiece routing is also added as described in section 5.1.6.5, thus providing the control and sequence logic along with the stimuli to exercise it. The system is then simulated, driven from the defined workpiece routing, to ensure the desired operation is realised.

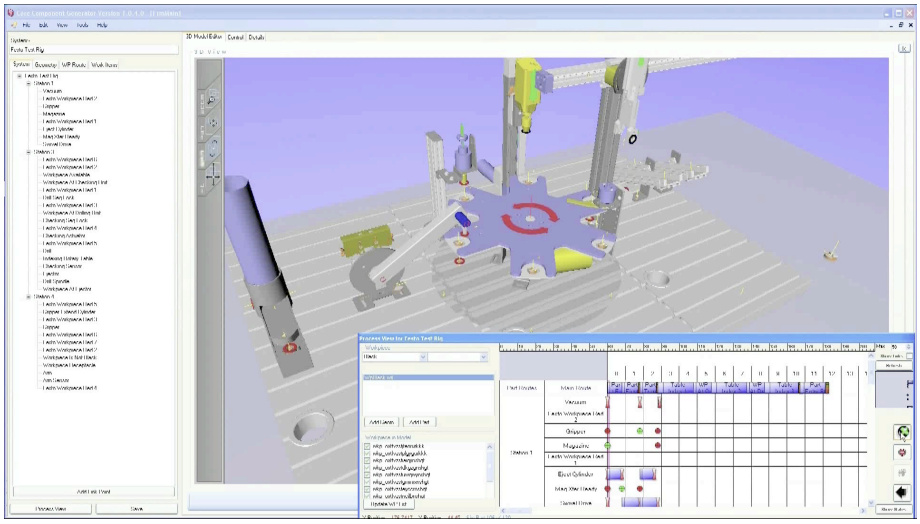


Figure 70 Automation system simulation with component process chart

Once the automation system has been is built and validated, the logic definition can be downloaded to the runtime hardware. To achieve this components (written to conform to the CCE component interface) are added to the target PLC code, the I/O on the PLC is configured to the components so they can communicate with the physical sensors, and then the configured components are interlocked and managed by matching the CCE components to the real-time components on the PLC.

This download is currently done manually by a separate procedure, by taking the XML definition of the system and linking the CCE and real components by name. In the future this procedure will become integrated as part of the CCE tools. The automation system configuration will be stored as part of its definition, ensuring the real-time components are complete before downloading the interlocks. If changes are made to the components used then mapping of the new/changed components is necessary.

5.1.8 Stage 2 – Reconfiguration of an Existing Automation System.

This section of the case study aims to evaluate the effectiveness of the design tools to implement changes to an automation system. The example change is to add a module (Buffer Module) to the assembly.

To complete this change the user must:

1. Create and test the new components/ modules as per stage 1.

2. Add the new module to the automation system built in stage 1
3. Remove interlocks between module 1 and module 3
4. Change the physical component assembly to insert the new component
5. Create new interlocks for module 2 so that it interacts with modules 1 and 3
6. Alter the workpiece routing by inserting the routing for the new module.
7. Simulate the system to exercise the logic and prove its behaviour
8. Download the new configuration; change the hardware to include the sensors and actuators for the new component.

To add the new created module the control system, workpiece routing and 3D model are required to be edited (item 3 in the above list). To remove the control logic that links module 1 to module 3, the interlocks that bind them should be identified and removed. Looking at the control view diagram (Figure 71) we can identify that the Transfer Arm component and Rotary Table component need to be investigated to unlink them.

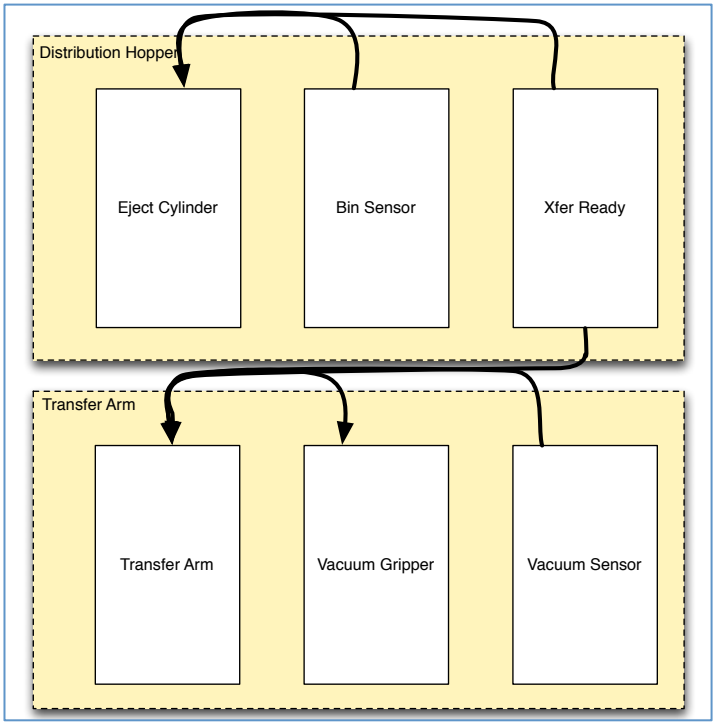


Figure 71 Component Control View; show the interlock links between components.

Then selecting the Transfer Arm we can delete the reference to the Rotary Table sensor from the interlock. This mechanism is a lot simpler than editing the traditional PLC code, where adding a new module requires the detailed coding style, error and diagnostics handling methods to be evaluated and understood before the engineer can even think of making such a change. Although structured programming techniques used on PLCs, such as STEPS and EDDIE, can help improve the consistency of such coding changes, a lot depends on the skill and experience of the controls engineer working on the software [32,33].

Next the virtual model should be edited; moving stations 3 and 4 to allow station 2 to be inserted. This is achieved by dragging the component from the 3D model view onto the route of the model tree.

The new module may then be added and assembled into the 3D model as described earlier. The interlocks for the new module must now be added, this is relatively simple as the engineer is only required to understand the how the components of the new module interact with components within the same locale. Also adding the new component automatically adds the associated maintenance and diagnostic information encapsulated in the.

Again this is different to traditional PLC programming where the scope of the direct interaction of actuators and sensors is potentially with respect to whole program. Unless the whole program is clearly understood then this can lead to unforeseen problems.

In order to simulate the automation system behaviour the workpiece routeing has to be edited, by inserting new states into the diagram, adding interlocks, and changing of the relevant sensor states. The system may then be simulated to ensure correct behaviour before it is implemented on the target platform.

Note: the component code must be written to conform to the CCE interface definition, containing maintenance, error and diagnostic information.

5.1.9 Summary

This case study has described the process of creating a library of components, with their state based behaviour, virtual models and kinematic behaviour. The process of assembling these predefined components into a digital model of the Festo Test Rig is described along with the addition of interlocking the components with other components to define the overall system behaviour.

The addition of the behaviour of a workpiece passing through the system is also described, which in turn is used to exercise the automation system logic and validate its behaviour. A brief description of how the automation system software is downloaded to the runtime platform is shown. Finally the process of integrating a new module is described, outlining the fact that the behaviour may be modified using only component interaction without a requirement to understand the complete underlying software application as in the traditional approach.

The results of the case study are outlined below.

5.1.10 Research Question Answers

The case study has been developed such that the research questions highlighted in 5.1.2 can be addressed:-

5.1.10.1 Research Question Answer 1

Can the component-based toolkit express the functionality of the components of the automation system in such a way that the behaviour of the overall system will meet the required functionality?

The development of the Festo Test Rig has illustrated that real-time control at the process level may be achieved using the CCE tools. Hard real-time code if required may be encapsulated into the component state behaviour (2.4). Hard real-time control, such as drive synchronisation should always be encapsulated by a component as the communication has to be deterministic. The definition of a control system using interlocked FSM's is, with the addition of virtual components such as process level control, timers and sequence locks, capable of describing all the functionality of an automotive powertrain assembly automation system.

The automation system designed and built using the CCE toolset describes the behaviour of the Festo Test Rig, although there was some additional work required to build virtual components that enabled the state of a component to be managed, i.e., The rotary table should only rotate when the checker and the drill have completed their operation, however the drill and checker should only operate if a workpiece has been detected at their position. To resolve this issue a virtual component was created to describe this behaviour, see below Figure 72

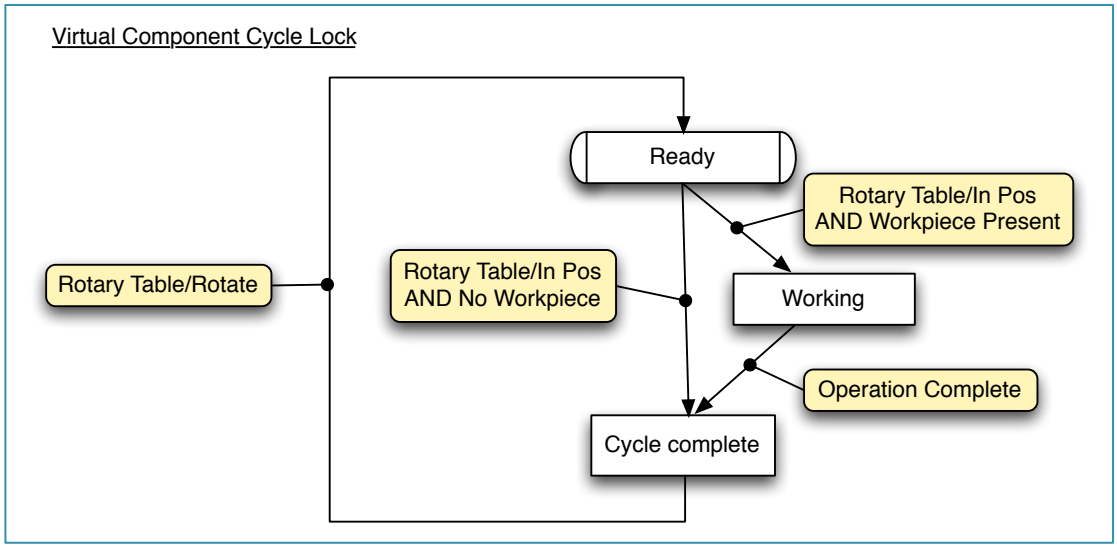


Figure 72 Virtual Component Cycle Lock

As can be seen in Figure 72 the cycle lock starts at the ready state, if a workpiece is not present and the table is ”In Position” it moves to the working state and waits for the component it is linked to, to complete its cycle (e.g. waiting for the drill to reach “depth” then return to its home position. If no workpiece is present the lock component will move to cycle complete. When the cycle lock is at the “Cycle Complete” state the table will be allowed to rotate (and the Cycle Lock will return to its “Ready” state).

This virtual component has proved to be very useful in the creation of automation systems that require a component to complete a sequence of operations before the automation system can progress.

5.1.10.2 Research Question Answer 2

What are the benefits of using the proposed methods to create an automation system over current practices?

There has been much research into component-based systems [CB Systems Papers] (from software to automation systems), and the benefits of reuse have been cited as a major motivation for this research. However for a company to truly gain the benefits of using component-based systems they must carefully manage their libraries of components to prevent the proliferation of component variants. For example a recent component survey within Ford identified 14 different pallet stops that do exactly the same thing. Either the end user can accept and manage the different variants of components and be flexible in their implementation and use, or they can control the library judiciously to ensure that new components are only added if there is a good reason for creating one. The latter will require a careful categorisation and definitions of components/ modules, with requests for new components being used to define and produce a library of flexible components to meet the goals of the automation system whilst maintaining an optimum library size.

Using traditional techniques the control system was created working from station 1 and the distribution hopper defining the behaviour and then the error diagnostics. As each item is built it is linked to the inputs and outputs to control it, i.e. as the "Eject Cylinder" cannot extend until the "Xfer Ready" sensor is logic 0. Whilst the control engineers are creating the program they are required to understand the behaviour of the complete machine, so that they can build the control system accordingly. The control system is therefore built up in a very efficient way, generally using structured programming methods (e.g. STEPs) and experience. The sequence of operations for the machine will have already been defined in general terms by the process engineer, typically using a process timing chart. The process-level control is then added above this so that sequence of operation can be defined, error and diagnostic information is added as the code is developed. The development of the code is typically done once the hardware has been defined, and the machine is well on the way to being built. This phase of the machine lifecycle is critical since making late changes will typically involve expensive code modifications.

Using the component-based approach each part of the system may be developed simultaneously in isolation, ensuring all the safety and diagnostics are built in, and the behaviour of the physical model is created at the same time. This has proved to have several benefits when creating a machine:

1. The definition of components can be made at an early stage allowing well-defined component-control software to be designed and built early in the machine lifecycle, whilst allowing the components that may be subject to change to be built as the design becomes stable.

2. As components can be built simultaneously the overall development period may be reduced.

3. The initial time for developing components may be higher, in that the components must be identified, categorised and built such that they are flexible enough to meet the requirements of many automation systems [106].

4. Once completed, the components may be quickly built into automation systems where their interaction can be defined and simulated. This system verification is vital in the production of a robust automation system.

5. The real performance and operation of the components can be fed back to the component design thus defining a more accurate model that will produce more accurate simulations when they are used to build future machines, thus reducing risk.

6. The component defined becomes an "asset" that, used correctly, can enable the development of new systems very quickly, rather than redesigning software from scratch each time an automation system is built.

7. The system definition is *vendor neutral.* This allows the automation system defined to be deployed on a different target control platforms.

Use of the CCE component-based approach does however present some challenges:

1. The management of the component/module library: This is a new role within the automation system development process.

2. Library ownership: It is believed that initially the library ownership will be by the end-user, as they have most to gain from the reuse of designs. However, being able to offer a library of proven component for building automation systems that can quickly be used to demonstrate machine behaviour is also advantageous to the machine builder.

3. Up-Front costs of building a library: There will be substantial initial investment involved in the development and management of such a library. However, once built, it will become a valuable asset, which could be highly beneficial in reducing the costs and risks associated with future machines.

5.1.10.3 Research Question Answer 3

How does using the component-based toolkit aid in the reconfiguration a machine over current practices?

Work has been done to quantify the gains of using component based systems over current practices, although the author believes any such work has to be taken in the context that this is still new in the area of automation systems, and the engineers are not proficient in their use [105, 106]. The result is that engineers are likely to be more efficient in creating machines using their current practices and be resistant to building and using components. Also when changes are made to the system they can be made without the generation of low-level code so it is probable that different engineers will be involved in the activity (e.g. if the change can be made using existing components then the process engineers may be able to make the change with little involvement from the controls engineers).

This makes the comparison between the proposed and current system difficult at best to quantify in time or monetary terms. However being able to communicate the change more effectively and simulate and test the system behaviour without the need to follow the traditional waterfall design approach will reduce the risk of change.

The main benefits are of using a component-based toolkit for reconfiguration are:

1. Error definitions built into the component, and handled in a structured way so that errors are reported consistently, and without omissions. The error reporting mechanism handles all errors without having to write additional code. Although, as before, it has been acknowledged that the upfront development and management of these components will make the initial automation system more costly the subsequent savings will be significant, and each component may be worked on in parallel which can reduce the overall time for the automation system development.

2. Mechanical/electrical and process engineers will be able to define and implement changes with the controls engineers being focussed on the more complex and less mundane creation of hard real-time components that require their specialist knowledge. So it is likely that the new engineering roles will emerge that spans the current departmental boundaries.

3. Components are only interlocked with a minimal number of other components (typically between three and six in the test cases studied) making changes to the system is very simple as you can quickly identify the effected components, make and test the change offline (including standalone testing of the new module/components), and then configure the run-time system with the new components and download the new configuration. Using traditional methods the impact of the change has to be carefully analysed to ensure that it does not adversely affect the current control system. E.g. adding the buffer module will require a control engineer to understand the way in which the program is written, with regard its programming style, structure, error reporting, diagnostics and methods of interlocking. The new module will then have to be created and interlocked with the machines I/O. Thus any change to the control has the potential to disrupt the behaviour of the overall automation system.

4. It is possible to check the aspects of the real automation system behaviour with the virtual model, using the broadcaster to create a hybrid system. Allowing partial control system testing before the hardware is complete.

5.2 Virtual Build Event

5.2.1 Introduction

From the user requirements studied in Chapter 3 it can be seen that there is an immediate requirement to view the powertrain assembly machine operation in the early stages of development, and that can usefully be shared between a team of multidisciplinary people including safety, logistics, plant layout, controls, and productivity engineers. This activity will also involve people from the supply chain to such as the machine builder and the material handling manufacturer.

At the time of writing the end-user convenes a "Build Event" for design approval. This generally requires the collection of all available data for the automation system, including machine CAD, a prototype engine CAD built to the level of the assembly to be assessed, 2D and 3D layout drawings and floor plans, Modapts productivity charts for any manual operations and any specific tooling required for the given engine assembly process stage. The stakeholders and vendors then use a combination of a meeting room and conferencing tools to discuss and analyse the automation system design in order to gain design approval as described in section 5.2.4.

One of the key goals of the tools is to provide models that may be used throughout the lifecycle of the automation system, so the CCE tools were used to develop models that could be used by every engineer to evaluate automation systems and aid in the design review and sign-off process. This have enabled a new type of meeting known as a **Virtual Build Event,** where the output from the CCE Tool is used as a collaboration and visualisation aid between the both the internal stakeholders and the external supply-chain partners.

This case study will discuss the current practice and how the outputs from the CCE tools are now being used to improve it.

5.2.2 Research Questions.

5.2.2.1 Research Question 1

How can the tools be used in a multidisciplinary team to improve on the current process of the build events?

5.2.2.2 Research Question 2

Can the representation of the virtual manikin provide enough detail allow analysis of the automation system operation?

5.2.2.3 Research Question 3

Does the integration of the Virtual manikin accurately represent the behaviour automation system and it's interaction with the virtual manikin?

5.2.3 Chosen Automation System for the Case Study.

For the purpose of this case study the Block Load machine was chosen, as it provides a reasonably complex machine that requires the multiple interactions between the virtual man and the automation system control. Also this machine was the first machine to be demonstrated at the end-user (Ford) to evaluate the effectiveness of the proposed virtual build events using the CCE Viewer. This automation system is shown below in Figure 73 (the photographs were taken during commissioning of the machine in Köln Germany).

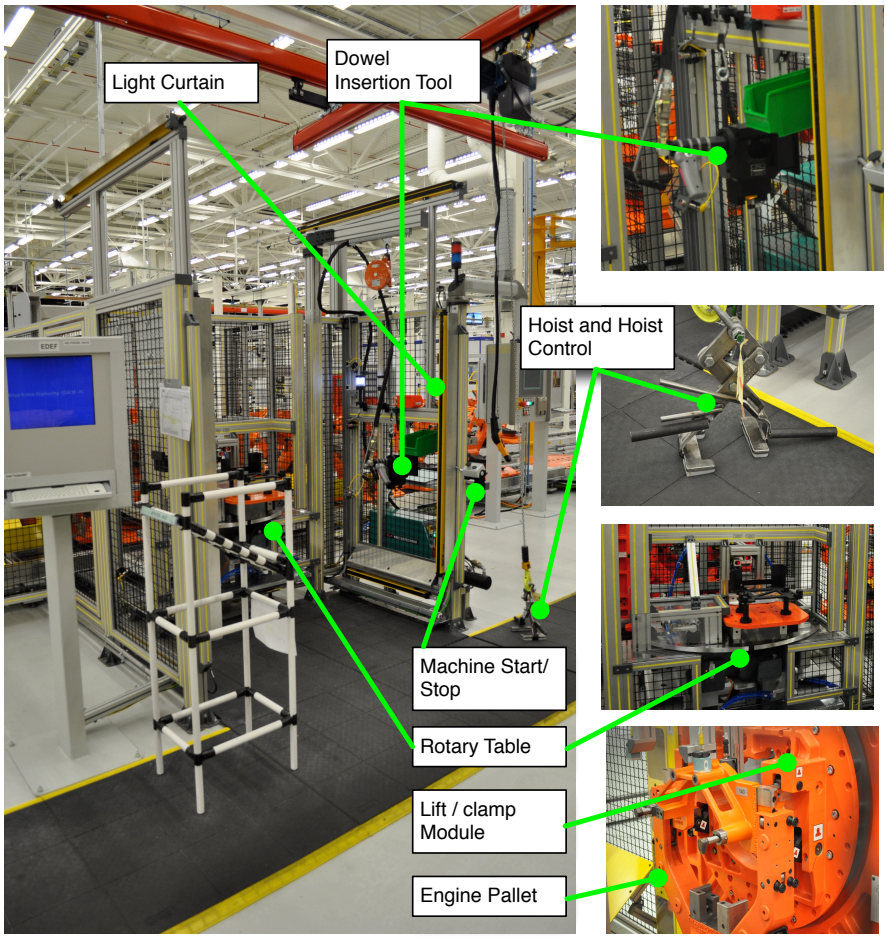


Figure 73 Picture of the OP60 Block Load machine during commissioning

The simulation created using the CCE Toolkit is shown in Figure 74, with each major component to be modelled identified and labelled.

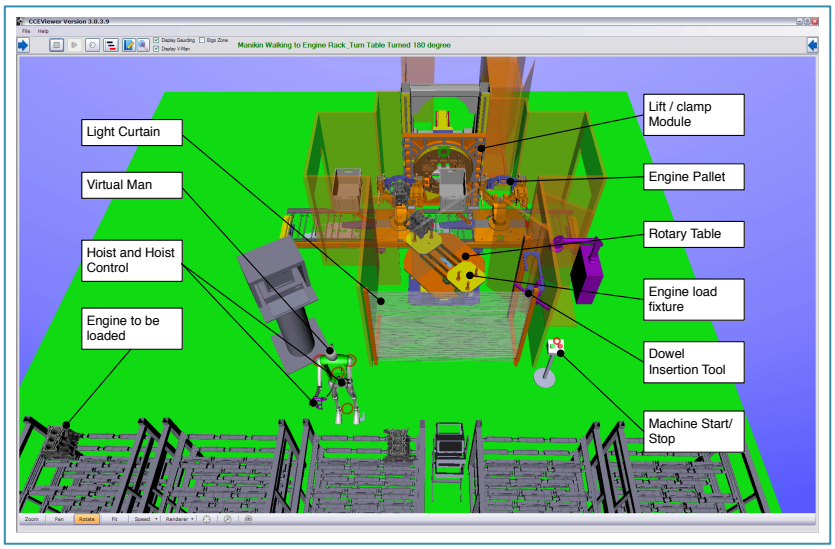


Figure 74 OP60 Virtual Machine Layout highlighting the major components to be modelled

The automation system comprises:-

1. Lift clam module: to clamp the engine mounting plate and place it in the correct position.
2. Engine pallet: onto which the block will be loaded.
3. Rotary Table: to rotate the loaded block for assembly to the pallet.
4. Engine Load fixture: jig onto which the operator can easily place the block.
5. Dowel Insertion Tool: hand tool to insert location dowels in the block.
6. Machine start control: two-handed machine start button, to start the automatic machine sequence.
7. Light Curtain: detects the present of a person or object in the automation system, if there is the automation system machine sequence is stopped.
8. Virtual Man: digital representation of the operator.
9. Hoist and controller: the hoist for moving to the block. This also contains a block hook, which is a simple hook designed to be attached to the block to lift it without damage.
10. Block to be loaded into the machine

5.2.4 The Previous Design Review Process

During the design of the automation system there are several phases of design review and sign-off during program approval and development completion see section 3.6. These include concept ready and design approval and production ready. During these phases there are many engineers involved in the design of the automation system and supporting infrastructures (from plant layout to controls engineers). In order to increase the efficiency of the final automation system design and reduce the errors due to miscomprehension a series of slow-build events are held.

There are three types of build events, all of which use an existing product (i.e. a new engine) built to the correct build-level that the engineers can handle discuss and evaluate methods of assembly to the next build level. These build events are described as follows.

5.2.4.1 Build Event 1 – Virtual Build Event

This type of event is generally used for automatic stations. A team of engineers, representing all aspects of the automation system design (usually between 20 and 30) are assembled into a room. In the room is the partially assembled engine for examination, plus projection facilities used to display:-

1. The build stage of the automation system.
2. CAD of the proposed automation system (if available), or CAD / pictures from an existing automation system that completes a similar task.
3. Plant layout drawings showing automation system location and access (if available.
4. CAD for the product.
5. Any other information that can be obtained from existing automation systems.

The event manager then runs through the assembly of the next stage of the engine build, using the real or rapid-prototyped product. The engine orientation, build order and accessibility for tools and workpieces are discussed. Then the type of automation

system is discussed using CAD from the machine builder for this or similar machines, in terms of:-

1. Assembly mechanisms, i.e., whether fully automatic, semi automatic or manual assembly is required.

2. Mechanical, are there any collisions with the workpiece or other parts of the automation system. If so can this be rectified or does there need to be a more in-depth study.

3. Productivity: can the operator realistically work for a full day comfortably and efficiently?

4. Does the machine meet the desired cycle time?

5. Automation system layout, product delivery /orientation.

6. Safety, e.g. access and guarding.

7. Workpiece delivery, loading workpieces into the automation system.

8. Controls, i.e., is the automation system behaviour achievable using the chosen hardware.

During this event notes are taken on comments and actions, so that if required changes could be made to the system these changes are then checked and the machine is approved as "concept ready" when it is ratified by all parties.

5.2.4.2 Build Event 2 – Slow Build

This type of event is for manual stations, and manual aspects of semi-automatic stations by using a physical "mock up" of the automation system. The product (mounted on a pallet), conveyor system, workpiece racks and all the equipment used to assemble the product are built into a model of the proposed automation system. The team of engineers then use the proposed design documentation to work through and evaluate the process, making improvements to it as they go.

The process is evaluated and signed off as in 5.2.4.1, but as these are manual stations more emphasis is placed on the productivity and safety aspects of the automation system.

5.2.4.3 Build Event 3 – In depth CAD/ Ergonomic analysis.

For particularly problematic stations an in-depth CAD/Ergonomic study is commissioned. Here the Station CAD is used or created to build a model of the station, and its operation is simulated using the functionality of the CAD tool. If the station is semi automatic or manual then an ergonomics tool such as JAK is used to simulate the human within the machine. This solution is only used where serious problems are envisaged, such as handling or collision problems, as the generation of the model is time consuming and requires a specialist CAD engineer to develop the model.

Once complete the solution has to run on an expensive CAD machine and manipulated by an experienced CAD engineer, in order to get the best results.

5.2.4.4 Previous Design Review Process Analysis

These type of events are invaluable in the automation system lifecycle, bringing the multidisciplinary team together provides a unique opportunity for the engineers to express concerns with the automation system design, bring experience (or "lessons learned") from other systems, allow engineers to discuss and understand the issues and solutions and also bring experience from disparate disciplines together. However, there are many problems with it:-

1. The organisation of personnel is difficult as it is a multidisciplinary team.

2. The meeting are costly in terms of man-hours and travel time, especially if the supply chain companies are involved.

3. Attendees often only start to learn about the automation system once they are in the meeting, so a lot of time is wasted whilst they familiarise themselves with the problem and resources.

4. When changes to the automation system are made, the results are generally not reviewed until the next meeting.

5. Full appreciation of the automation system is sometimes missed due to the variety and lack of integration of the information.

6. Once a "mock-up" has been dismantled it cannot be quickly checked after the event. Although the final process is often videoed, it often does not show every aspect of the machine, so is of limited use.

7. The complex CAD models built in 5.2.4.3, once complete are costly to maintain with automation system changes so they quickly go out of date and become redundant.

5.2.5 The Proposed /Current Process

As the CCE tool use a "lightweight" model built from relatively simple components it was proposed to use the generated models to see if they could improve the current design review process.

The process involved:-

1. Data harvesting.
    a. Retrieving the CAD and simplifying the geometry
    b. Breaking the CAD down into geometry for the components
    c. Exporting the Geometry as VRML for use by the CCE Tools
2. Building Components
3. Building Modules
4. Building a System
5. Add adding sequence interlocks

6. Adding workpiece routing information
7. Adding Manikin Moves
8. Add model release notes
9. Building a distributable file
10. Share the model with the engineers at least two days before the VBE
11. VBE evaluation notes and resultant actions from the VBE.
12. If required make changes to the model and arrange a new VBE

5.2.5.1 Data Harvesting (1)

As part of the research within the group a number of researchers were given authorization to approach the machine builders and other suppliers to gain CAD of the proposed machines. They obtained preliminary Modapts worksheets for the definition of the operators' procedure, and cycle times for the predicted machine operation. This information was used to identify the components of the automation system. They worked with the machine builders to help them export CAD component geometry in a suitable format for use in the CCE tools (i.e. removing parts and geometry superfluous for simulation).

The output of this process was a data pack that includes VRML files that show the product at the required build stage, VRML files for the components split into the static and dynamic parts, preliminary timing charts for Modapts and the machine sequence and any other information gleaned during this process.

5.2.5.2 Automation system build (2-7)

The automation system was then built using the tools to create the component, modules and the system. For details see (Case Study 1 and Chapter 3). In addition to this the Virtual Manikin (V-Man) is added (see Chapter 3), with the nominal state behaviour outlining its basic. Each static state defines the manikin fixed position e.g., at engine rack, each dynamic state defines the tasks required to fulfil the requirements of the state, e.g. pick up workpiece, insert workpiece to engine and press start button.

The V-Man module as shown in Figure 75 below shows the behaviour for a specific state (highlighted in yellow), which describes the actions required to move from the releasing of the hoist controller and hook to grasping the dowel insertion tool.

The superimposed V-Man (holding the hoist) shows the initial position for the manikin as taken from the previous state, from here the user will add a sequence of moves as described below:-

1. The user will position the manikin using the V-Man using the controls attached to the V-Man limbs. Clicking the mouse whilst holding the control key in the V-Man timeline (at the bottom of Figure 75) enters the move into the sequence. Here the manikin is shown turning and walking a short distance to the dowel tool whilst raising his arms ready to grasp the tool. This is displayed in the V-Man timeline by the first 4 dark boxes. Note these actions are done simultaneously.

2. Once at this position the V-man arms are positioned to grab the dowel gun. This can be considered as the final movement, reaching for the gun after getting into position. This is displayed by the following 2 parallel bars in the V-Man timeline.

3. Now the manikin has to grasp the tool in its right hand. As this move only involves the hand (which is too detailed for the simplistic V-Man), a manual move has to be entered. The user selects the hand action button (from the left hand side of the screen, which brings up a "pick-list" of hand actions) when the grasp action is chosen a list of valid times for this action is displayed. The single bar on the V-Man timeline shows this.

4. Finally the tool needs to be released from its hook, which is achieved by moving the V-Man arms to the release position and clicking the mouse whilst holding the control key in the timeline window. This move is shown by the final 2 parallel bars in the timeline.

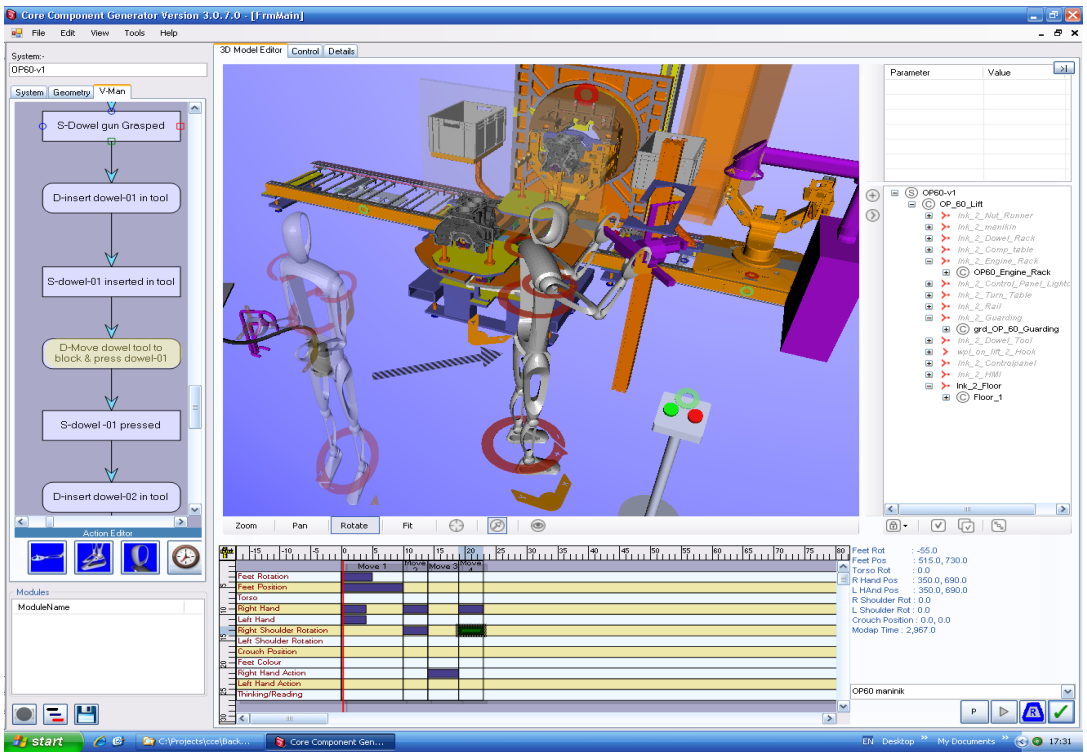


Figure 75 V-Man Editor

The generated Modapts code for this move is (W5 W10 M4 M4)(M4 M4)(P5)(M4 M4).

The manual moves include foot press, get, put and juggle for the hand, read, write, speak and think for the head. Time delays can also be input into the timeline.

The worst-case scenario was evaluated to ensure that the operator could complete all tasks within the required cycle time. The OP60 process timing chart snippet, shown below, displays the interaction between the workpiece, the automation system and the operator. The top row is the work piece routing (i.e. how the workpiece fires the sensors as it moves through the automation system. The bottom row is the manikin states, which are interlocked with the automation system behaviour to provide the cycle time based up the machine and human operation.

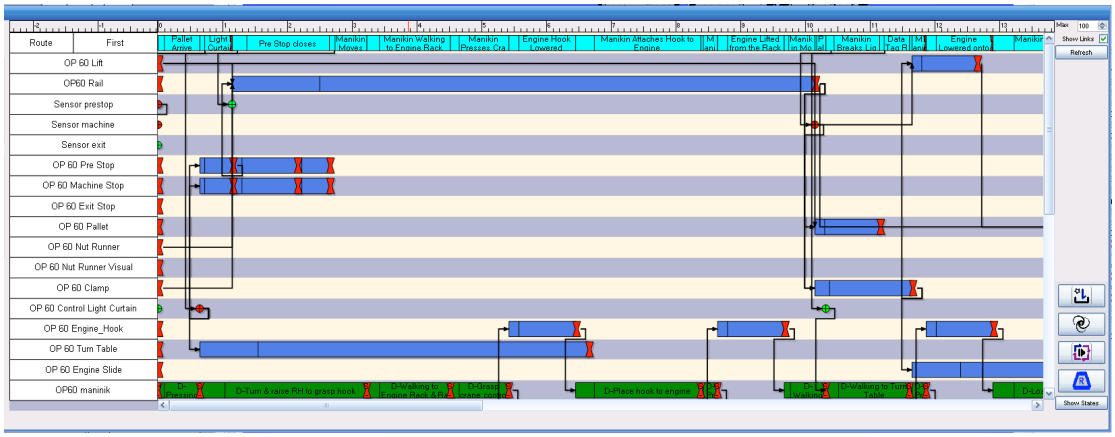


Figure 76 Process timing chart snippet

In order to display the handling of tools and workpieces the manikin has link points on each hand, which when set to the parent of the workpiece or tool link point results in the item being moved with the hand, as if the manikin had grabbed the item (Workpiece routing is described in 4.4.8). The action of grabbing the item is not modelled, as the level of detail required is too great and the animation of this is superfluous for this type of simulation. Instead the Modapts codes for this action are added manually e.g. (M2, G1 (M2 = hand movement, G1 = simple grasp get)).

During the creation of the model questions regarding the stations operation, are raised, these are either resolved by talking to the engineers or noted for discussion in the release notes detailed below. Once complete the model will be released with a "Release Note" document as described below.

5.2.5.3 Release Notes (8)

The creation of a 3D model of a process will always create issues that need to be discussed or rectified either in the machine or process design. Information regarding the operation and utilisation of the machine is also recorded.

The release notes are created for every machine and include the following information (each of which is discussed in detail for OP60, and the full report is shown in Appendix 1):-

1. A history of the model. A description of the process that model represents, and a change history detailing the changes made at each version of the released model.

2. Simulation Report. List of the anomalies and errors highlighted during the creation of the simulation

3. Summary. Highlight the statistics of the process such as cycle time, operator utilisation and complexity rating.

4. The machine builders cycle timing diagram, from which the actuator operating times and the machine cycle are taken.

5. The productivity Modapts data sheet. This is the productivity teams initial definition of the process, using Modapts to describe the actions of the operator.

6. The resultant productivity chart automatically generated by the tools.

5.2.5.3.1 Model History

The model history lists the scenarios for the machine and, generally these will be normal cycle times (i.e. normal operation of the machine and operator), and the worst case scenario (i.e. if the operator has to restock a container, or get a workpiece from the furthest position).

For OP60 two scenarios have been modelled these are the best-case scenario and worst-case scenario. The best-case scenario is where the operator collects the workpiece from the closest rack; the worst case is the collection of the workpieces from the furthest rack.

5.2.5.3.2 Simulation Report

The simulation report details all the anomalies noted during the creation of the model. For OP 60 these include the following.

Saftey: The dowel insertion tool, when stored in the holder, clashes with the guarding and the turntable slider extends beyond the table, which may cause injury to the operator.

Machine assembly: The original CAD for the Nut runner that attaches the workpiece to the pallet clashed with the pallet (this was subsequently changes before the release of this model).

Machine Operation: It was noted the time for clamping the pallet is different to the exact same process on a different machine.

Missing/ Non-released CAD: Colouring these components purple in the model identifies these. They are also listed here for completeness. In this version of the model the engine block racks, engine hook and the dowel tool is under review. Below is the original Engine hook, the dimensions of which were taken from a prototype hook.

The CAD for the operator HMI and start buttons were also unavailable so assistance was sought from the productivity team and a best guess was made as to the type and location of the start buttons, with the CAD for the HMI being taken from an existing line.

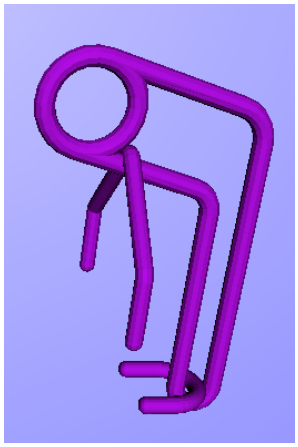


Figure 77 Example of Non-released CAD - Original Engine Hook Design taken from prototype.

Finally the dowel insertion tool was under review at the time of the creation of the model.

Ergonomics: Two items were noted here. Firstly when the operator is attaching the hook to the block at the top of the block rack he is twisting his shoulders outside of the ergo zone which may cause fatigue, secondly the position of the dowel gun is too high (making the operator reach) as well as being potentially difficult to remove from the holder due to its close proximity to the guarding.

5.2.5.4 Share Model and Run Virtual Build Event

In order to share a model all the data acquired to build the automation system must be collated and packaged in a suitable format for distribution. The output file of the CCE tools for simulation is a ".ccZ" (Core Component Zip) file which contains the 3D model, the definitions of the components used by the Automation system, an XML file that describes the structure and configuration of the components (including interlock data), and any other documents that have been attached to the system or components (e.g. release notes and data sheets). The ccZ file is effectively a "Zip" file that contains all the files required in the correct file structure required completely described the automation system. It is intended to accompany the automation system and be updated during its lifecycle; as such the CCE tools all the import of the ccZ file so that the tools may be used as a stand alone maintenance and diagnostic tool.

For the purpose of the Virtual Build Event the ccZ file is used as a distribution tool so that all the invited engineers may simply open the file in the CCE Viewer tool on their desktop for simulation and e

5.2.6 Results of the Simulation as Compared to Real-World Results

The Block Load machine OP60, along with other stations modelled, is (at the time of writing) under commissioning on site in Koln Germany. This has provided the opportunity to compare the results of the study against the real world results.

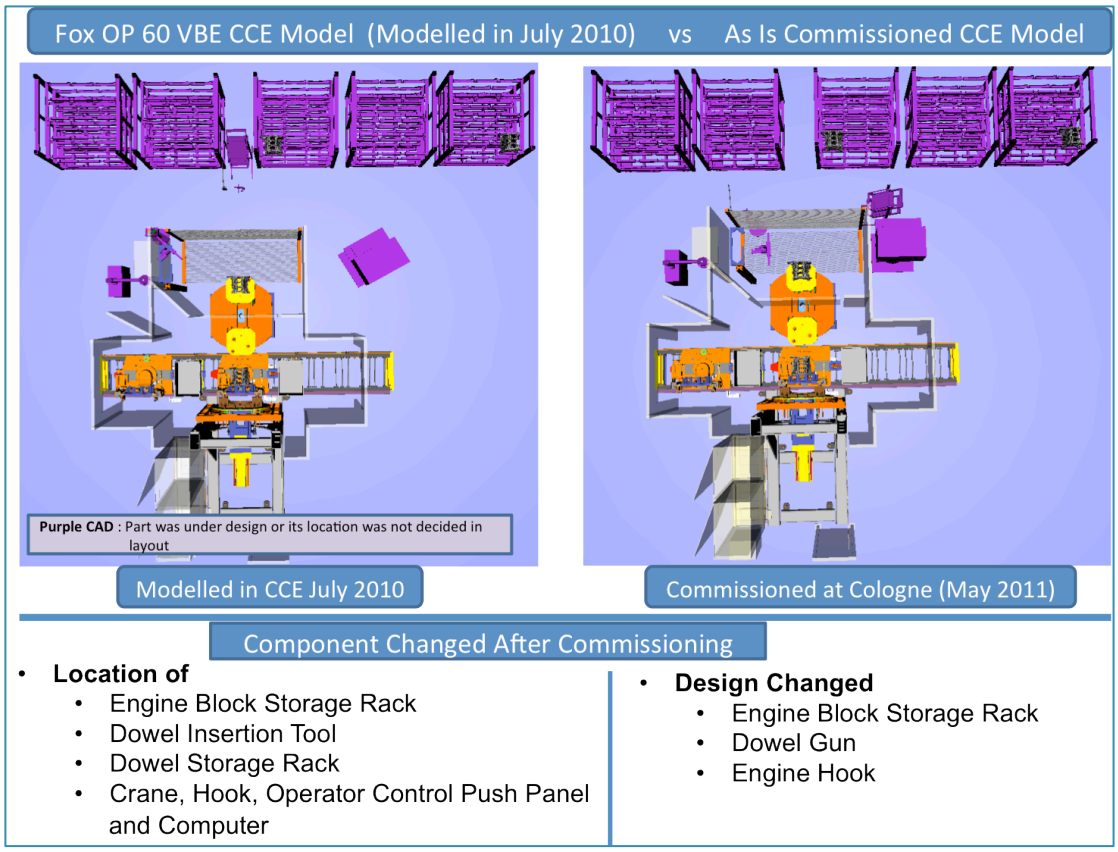


Figure 78 Changes made from the original OP60 Machine to the delivered machine.

Figure 78 above shows the changes from the original model, completed July 2010, to the commissioned station in May 2011. There are four item location changes the most critical being the change of the dowel insertion tool (which may be due to the concerns raised in the release notes) and the crane, hook and operator start buttons (which were all unknowns in the July 2010). There are also three design changes made to the engine block rack, the dowel gun and the engine hook. The most significant change was to the engine block racks. Here three engine blocks could be loaded before a separator had to be removed and placed in a rack adjacent to the block rack, also to get the next row of three blocks the rack has to be turned by 180 degrees.

These changes resulted in a marked increase in cycle times between the actual and delivered machine/operator sequences, and consequently the machine running over-cycle by 136% as shown below.

	CCE (July 2010)	Actual (May 2011)
Cycle Time	39.9 sec	55.1 sec
Operator Loading	98.5%	136%
Target Cycle Time = 40.5 sec		

Table 10 Comparison between modelled and commissioned results.

It may be seen in the table below that differences between the modelled and actual times exist.

Action 1: Difference of 3.1 seconds is due to the relocation of the start buttons and the racks, as well as the redesigned engine hook.

Action 2: Difference of 0.8 seconds is due to the assumption made for the tool operating time and the location of the dowels

Action 3: Difference 0.5 seconds, due to the relocation of the dowel tool holder and the start buttons.

Action 4: Difference 3.6 seconds, Process missing from original specification

Action 5: Difference 14.4 seconds, Process missing from original specification

Action No.	Process	CCE	Actual Commissioned
1	Load Block from Rack to Turntable	23.6 sec	20.5 sec
2	Assemble Two Dowels	12.4 sec	13.2 Sec
3	Grasp hoist, walk out off light curtain, start turntable	3.9 sec	3.4 sec
4	Remove one separator and place into empty rack		3.6 sec
5	Back fill three separators and rotate rack		14.4 sec
	Total	39.9 sec	55.1 Sec
	Operator Loading	98.5 %	136 %

Table 11 Analysis of the modelled and real world timing

There were some lessons to be learned from the study such in that for the original design block rack CAD was taken from previous programs, the decision to change the design caused significant errors in the new machine cycle time. Once the optimised rack has been approved, any changes to it in future programs must provide significant payback times, else they will be rejected as the change may provide uncertainty and risk for the process.

From the knowledge gained by the study the missing actions in Table 11 (Action 4 and Action 5) and alterations were made to the machine to reflect the actual machine.

5.2.7 Summary.

This case study has shown how the tools may be used to create and distribute a functional animated model that may be used stand-alone or as part of a collaborative virtual build event. It allows engineers from different disciplines to examine, discuss and communicate ideas and problems with the automation system layout and design before production.

Described in this chapter are details of the complete process of creating an automation system from data harvesting from external CAD, building reusable components and building and simulating the complete system.

5.2.8 Research Questions Answers.

5.2.8.1 Research Question Answer 1

How can the tools be used in a multidisciplinary team to improve on the current process of the build events?

The tools proved effective for the new process of virtual build events for several reasons: -

1. The tools allow the engineers to evaluate the machine before the event so that time is not wasted discussing the machine operation during the event. This capability also enables the engineers to evaluate the design and hence pose more informed questions about the station operation or design aspects such as safety and material handling at the event.

2. The shared understanding of the automation system is greatly enhanced between all the disciplines across the team, rather than being mainly confined to the core automation system development team.

3. Discussions on machine operation can now be made by playing the operation through over and over turning on and off geometry to closely examine different aspects of the automation system behaviour.

4. The Modapts codes defining the operator's behaviour can be evaluated with respect to the machine behaviour, which is currently not available with current practices.

There are however some problems found in the process, in that the material handling in the current procedure is often left as a variable that is fixed during the commissioning phase of the machine. These "last minute changes" are often made to the station layout to accommodate the operator's requirements.

However, by modelling the system lessons can and have been learned, and the benefits of a model that accurately depicts the current behaviour will benefit future programs, requiring any future changes to be proved before they are made, and this should result in more standardised material handling processes being utilised.

A total of 15 automation systems have been modelled so far, and due to the perceived benefits evident to Ford, future powertrain lines will be modelled.

5.2.8.2 Research Question Answer 2

Can the representation of the virtual manikin provide enough detail allow analysis of the automation system operation?

The virtual manikin developed for the CCE tools provide basic ergonomic behaviour, with the more complex hand/head operations being described using Modapts codes only. This has resulted in simple operator simulation models being produced. However, these models have proved very effective at conveying concepts, problems, timing, and interaction between the human and the machine, which can be readily displayed, analysed and evaluated.

For example during the evaluation of the tools it became obvious that an important metric to be taken from the simulations is the operator fatigue rating. For example if an operator lifts a workpiece, walks three steps and lowers it onto an assembly, then the energy that the operator uses may be calculated, and hence a fatigue rating can be calculated. This capability is being integrated into the toolset.

The tool does have limitations in that it does not accurately model complex moves in the way that a tool like JACK does [69, 90]. The models are lightweight and

shareable requiring a lot less effort to create, and in some cases can expose areas where a full ergonomic study is subsequently required. The overall result of this research is likely to be that more machines will be now modelled using tools like the CCE so that lessons may not only "be documented" but also the lessons may be built into the dynamic simulation models so that future programs may be built from working, optimised systems.

5.2.8.3 Research Question Answer 3

Does the integration of the Virtual manikin accurately represent the behaviour automation system and it's interaction with the virtual manikin?

As can be seen from the results of the study the automation system behaviour is very close to behaviour of the real machine, errors can be attributed to minor deviations in the actuators performance, and possibly control time (i.e. time taken for signals to be processed). Also the manikins timing was very close to that of the operators, so that overall there is approximately a 10% increase from predicted to actual timings. As the machine is in the commissioning stage it is likely that the actual times will reduce as the operator becomes more familiar with the task. It is also likely that as the system becomes accepted then more detailed analysis of the process will be made by the productivity team so that the virtual prototype will more closely resemble the final solution.

Chapter 6 **Evaluation**

6.1 **Introduction**

In this chapter the results from the two case studies are outlined with respect to the initial requirements defined in chapter 3. The requirements are taken from work derived from modelling of the business process used at the machine builder and end user for the implementation of a power train assembly line, as well as interviews with process, controls, productivity and commissioning engineers within these companies.

As part of the evaluation of the CCE tools, engineers used the tools for their own evaluation and were asked to comment on usability, features, bugs and the applicability of the general approach taken by this research.

The evaluation has been carried out, in the main, from the perspective of the functionality provided to users. Each requirement is listed against the evaluation section where it is covered; some requirements are covered in multiple sections. The evaluation criteria are as follows:

1. Integration of human in the machine control system
2. Development of an integrated model.
3. Develop a vendor neutral toolkit.
4. Investment in design and re-use.
5. Increased automation system agility.
6. Improved maintenance.

This evaluation process has taken place iteratively over a three year period and has proved invaluable in defining how the continuing development of the tools should be focussed in order to effectively address “real world” issues.

6.2 Requirements Checklist and (See Chapter 3)

Below are the requirements table defined in chapter 3, it has been included here for easy reference for the evaluation.

No.	Description
1	To be able to respond and react quickly and accommodate changes throughout the lifecycle.
2	A high degree of reuse is desired throughout the lifecycle of machine design and build. This includes reuse of automation hardware, control software, engineering knowledge and best practices acquired from previous projects.
3	A common data representation is required to support the various phases of the lifecycle. A consistent data representation would help to reduce repetitive work of interpretation and translation of design specification. This common representation seems all the more important when the partners in the projects have differences in (i) geographical location, (ii) levels of experience and understanding, and (iii) cultural and language backgrounds.
4	Support integration of a simulated human with the control system.
5	Use simulated human to provide accurate timing of manual and semi automatic station.
6	Provide Modapts/MTM output sheet to describe the behaviour of the simulated human.
7	Visualisation of system behaviour through modelling and simulation prior to installation is desired. This would enable the control engineer to validate the system before the physical assembly.
8	Support for virtual engineering, system try-out and commissioning.
9	There is also a need for an approach that will enable the machine design and the associated control behaviour to be available to all interested parties throughout the lifecycle. It has to be in a format in which the users can easily relate to.
10	A more integrated support for the system diagnostics and maintenance is required.
11	Business process and functional benefits of innovations, new technical architectures and approaches need to be appreciated readily by non-technical managers to ensure commitment, uptake and investment are achieved.
12	Ability to (re) configure machines built from reusable modules. Maximise reuse of machine design. In order to maximise manufacturing agility at minimum time and cost, it is vitally important to be able to reconfigure production machinery easily and quickly.
13	Any system should be easily integrated with higher level enterprise systems.
14	During the lifecycle of the machine the models created shall be visible and supported by the supply chain partners.
15	Provision for integrated production monitoring, for process management.
16	Any solution should be vendor neutral encouraging open systems, only specialising in where required (e.g. control hardware)
17	High level machine configuration capability
18	High level process description
19	Plant layout support
20	Capture “Lessons Learned”
21	Inherent compliance with standards
22	Lifecycle support from engineering tools
23	Support for globally distributed engineering teams

Table 12 List of requirements (Recap)

6.2.1 Integration of Human in the Automation System [4,5,6,]

The integration of the human into the automation system is seen as key requirement for automation system design especially when there are several points in the machine cycle where the machine waits for the operator or the operator waits for the machine. This interaction causes under or over utilization of the operator and/ or excessive cycle times. In locations where manual labour is used to compliment automatic

assembly for price and flexibility reasons many machines are designed with the human integrated into the process.

This integration has been achieved by using a simplified virtual human (without fingers) which can be positioned in the virtual model according to simulate the human behaviour. Modapts being a predictive tool for manual operations provides not only the timing for the actions taken in positioning the virtual human in the model, but also provide a structured way of entering the non-simulated actions such as grasping a component, pressing a button, reading text or making a decision. Each small sequence of operations is embedded into at state, which in turn is embedded into the virtual humans state machine (as a component) that describes their complete operation.

Each time the operator is waiting for the automation system to complete an operation the virtual humans state is interlocked with the machine control. Conversely if the machine is waiting for the operator (i.e. waiting for the operator to press the start button) it is interlocked with the virtual humans state machine. This interlocking allows semi automatic machines to be designed with acceptable accuracy, and allows changes to the operators behaviour and/or the machines behaviour to be reflected in the overall cycle time automatically.

This simplified approach provides a cost effective tool for assessing manual and semi-automatic machines, that may be used in conjunction with more sophisticated tools for more in depth ergonomic studies where required.

Finally the ability to automatically output Modapts data sheets, which are generally created manually, provides real benefits in the design of semi-automatic automation systems, with regard efficiency of design, consistency between models and a good visualisation tool. The generated Modapts sheets are further used to create work descriptions for the shop floor operator, so accuracy of the output will be beneficial in producing acceptable work descriptions.

6.2.2 Vendor Neutral (Req's 2, 3, 7, 8, 9, 16 and 17)

As the CCE tools are focused on the functional and process related design aspects of the automation system, and not specifically on the real-time control, the resultant output is generic and not linked to any specific controls vendor, until installed on a specific platform.

"Vendor neutral" as used in this thesis indicates that any interfaces or outputs of the tools are openly available and free to use, and not linked to specific software or hardware vendors.

The output of the CCE tools is an XML definition of the control system and a VRML model as a digital representation of the automation system. The XML model contains all the information required to build an automation system. It contains a definition of the each component within the system, in terms of its state behaviour, parameters, error definitions and predicted performance as well as the application logic (i.e. the interaction between the components that define the behaviour of the automation system). The component definition is used to define each components interface, which is built into a target specific component or function block. The components defined and built using the CCE tools runtime interface are manually configured to meet the specific real world IO requirements, using an IO Mapping tool. This information in conjunction with interlocking of components may then automatically loaded to the target platform.

This XML definition has been used to configure the behaviour of the component based control system implemented on both 1) a distributed web service based platform [108] and 2) standard Schneider Electric and Siemens PLCs. The components built to the defined CCE runtime interface are currently manually configured to meet the specific IO requirements, using an IO Mapping tool, and then the interaction between the components is automatically downloaded to the target platform.

6.2.3 Integrated Model (1, 2, 3, 8, 10, 11, 12, 17, 22, 23)

As shown in Figure 24 the integrated model is at the core of the automation system design. This contains all the information required to describe, simulate and install the automation system. For example, the associated 3D model, installation, HMI and Modapts are all views of this same model. This model-based approach makes the design of the automation system extensible by progressively adding to the models data structure.

The integrated model supports the automation system lifecycle as, unlike approaches from current commercial software which focus is on 3D design (requiring a large investment in both technology and training) that may support the semi-automatic generation of PLC based software, the CCE methodology focuses firmly on building automation systems from newly created or existing well defined components, with incorporated 3D models taken from existing CAD using a lightweight modelling approach [109]. The integrated model and lightweight CAD enable the same tools may be used on the shop floor to maintain and modify existing automation systems, with or without 3D visualisation support, eliminating the need to return to the design office to effect the change and then regenerate the code for deployment.

The integrated model also offers business benefits in that new automation systems may be quickly "mocked up" and demonstrated to customers or non-technical managers to show the benefits of new automation system methods or technologies. Also the automation system simulations and documentation will be available any user wishing to install the viewer, such that, with little or no training, non-technical users may rapidly gain an appreciation of the automation system.

6.2.4 Investment In Design - Design Reuse (1, 2, 9, 12, 14, 17, 18, 23)

The component-based approach taken by the original research and supported by the development of the CCE toolkit provides an ideal platform for design reuse. The definition of components has been greatly simplified by representing their behaviour using finite state machines and parameterisation of the configuration described in 4.4.5. This simplification and parameterisation greatly enhances the integration of components, as the system builder only has to consider the component state and does

not have to understand the context, measurement units, and conversion of data in order to create or edit an automaton system.

The component approach also allows control engineers to concentrate on areas where their expertise is really required (such as drive synchronisation and complex control loops), and not on the higher-level process control, which may now be handled by process engineers. Using the CCE tools translation from the process engineer's requirements to the control engineer's implementation is largely implicit, so that the traditional work involved and assocated sources of error are either greatly reduced or eliminated.

Also by making the tools vendor independent the design reuse is enhanced as the same system and/or component designs may be used on hardware from different vendors, even if the components are taken from designs implemented on competing hardware platforms.

6.2.5 Increased Automation System Agility (1, 2, 12, 14)

As defined in section 2.3the current manufacturing systems are moving from monolithic centralised control systems to a more modular and distributed solutions. New research into manufacturing systems is increasingly proposing distributed control architectures and modular system designs (section 2.3.3). The CCE tools capitalise on these trends by offering a component based approach that can be equally well applied to current centralised PLCs as well as distributed control solutions. The encapsulation of component behaviour increases agility by allowing users to prepare for and react to change of an unpredictable nature by readily modifying system behaviour or composition, by the reduction of interdependencies of software components, and by the creation of well defined boundaries of functionality.

6.2.6 Improved Maintenance (Req's 1, 3, 9, 12, 13, 14, 17, 22, 23)

The encapsulation of error, diagnostic and maintenance information into components allows for structured error reporting and access to diagnostic information (Section 2.4.1). The automation system model is used to dynamically generate a rich web based HMI that may incorporate a 3D animated model (Section 4.4.11). The model may be used to convey the machine position to a maintenance engineer on or off site

(IT infrastructure permitting). The maintenance engineer can identify the location of the component at error and information regarding the error.

Using the CCE Tools and the CCE Viewer the maintenance team may create and validate automation system changes before downloading them to the real hardware, reducing the risk and system downtime when making changes to the automation system.

6.3 User Comments from External Users who have Evaluated the CCE Tools

The CCE Tools have been demonstrated to a small selection of End Users at FORD and Krause and other Interested Engineers. These engineers have used the system to create demonstration models of automation systems. The comments below are based on the users experience of using the tools.

- The biggest problem by far is that there is no “Undo” button.
- Using VRML is a good way of using existing CAD
- There is a logical progression in building an automation system
- Good visualisation of the process that give accurate cycle times.
- There is no automatic clash detection.
- The modelling tool is dimensionless, which makes link point placement difficult to assign accurately.
- Each component is in a different model space, so when a new component is inserted into the system it may not be visible as it is so far away with respect the existing components.
- There is no ability to add ad hoc components at the system level.
- Error messages are unhelpful and there are no help screens.

The general feedback is that whilst the CCE Toolkit has bugs and problems (such as no undo) the general approach used is valid for the domain and it is simple to create realistic simulations with accurate cycle times with very little training.

6.4 Summary

The implementation of the CCE toolkit development through this research has allowed the benefits of component-based engineering within the automotive powertrain domain to be realised and used by the collaborating companies involved. The CCE Tools address all aspects of the user requirements identified in Chapter 3 as indicated in Table 12, and major time and risk savings can be achieved using this approach over the traditional CAD and control system development methodologies, especially when the complete lifecycle of the automation system is considered.

Chapter 7 Conclusions and Future Work

7.1 Conclusions

The aim of this research was to build upon and support the continuing research of Prof. Harrison and Prof. West and the team of research associates and Ph.D. students who support them in the Wolfson School of Mechanical and Manufacturing Engineering at Loughborough University.

The approach taken by the research was to evaluate the target end users' requirements [25,63,105,106] and integrate these requirements with a component-based approach [2,43] and support the development of diagnostics systems [63,106] and novel HMI's [64,110].

The research objectives identified in Section 1.4 were listed as:

- The development of an Integrated Model. A model that contains the required information to simulate and deploy an automation system.
- The integration of the human into the machine control system. Allow the Virtual human to interact with the behaviour of the automation system, such that if either one changes the results are reflected in the other.
- The support of Model Reusability. Automation systems built from a library of components.
- The development of System Simplicity. The use of FSM's to encapsulate the complex behaviour of a component.
- The support of System Deployment functionality. Allow the deployment of the components such that they are interlocked in exactly the same manner as the simulation, either on centralised PLCs or distributed using technologies such as SOA.
- The development of a Vendor Neutral toolkit. Allow the Deployment to devices from different vendors without requiring any changes to the model.

Section 4.4.3 outlines the design approach taken to build the *integrated model*. The *integrated model* is at the core of the CCE toolkit for the definition of the component-based automation system using FSM's. In addition to this, the ability to include a "virtual" human as part of the integrated model of the automation control system (i.e. *V-Man*) is detailed in section 4.4.9, facilitating simulation of manual and semi-automatic automation systems by the CCE Toolkit.

The automation system definition generated by the CCE tools provides the information to describe the component behaviour without details of the specific target runtime architecture. The instantiation of this on a target runtime extends, but does not alter, the *integrated model*. Therefore the CCE's *integrated model* is non-vendor specific.

The simplicity of automation system design has been maintained throughout the CCE toolkit via link point assembly (sections 4.4.5 to 4.4.7) providing a building block approach to assembling CAD and component state interlocking defining process behaviour (section 4.4.7).

7.2 Novelty and Contributions to Research.

The development of the Engineering toolkit laid out in this thesis has included work from many academics from Loughborough University, working on knowledge elicitation and business process modelling of key members of the supply chain (such as Ford as an end user, ThyssenKrupp Krausse as a machine builder and Schneider Electric as a controls/ component supplier) [25,63,105,106]. This knowledge aided the development of the key requirements for the engineering toolkit.

Figure 79 illustrates the research topics that have been addressed in the building of the Engineering toolkit. The author has contributed to all aspects of the tools development, but has more specifically made the following contributions:

1. Developed the underlying data structures to support the component-based approach.

2. Developed the toolkit architecture as described in chapter 4.

3. Developed the prototype integration between the engineering toolkit and VRML models, this was further developed by Dr Daniel Vera [109]

4. Developed the prototype Broadcaster/Marshaller, which was subject of Dr Vishal Barot doctoral thesis, and supported the work on web-based HMI's [110]

5. The integration of a workpiece route into the automation system development process based upon engineering knowledge as well as a process engineers experience, to allow machine behaviour to be modelled without the complex use of virtual sensors (e.g. ray beam sensors) in the virtual model.

6. Developed the integration of the virtual human (V-Man) with the engineering toolkit, using MODAPTS and FSM's such that the operator's behaviour is integral to the behaviour of the automation systems.

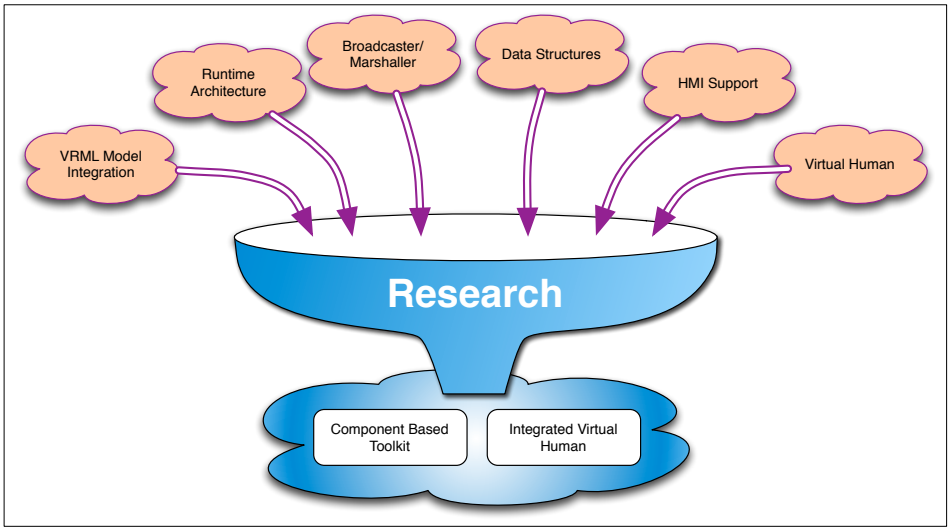


Figure 79 Identification of research topics

However, the main focus of this thesis is the development of the Engineering Toolkit to support component-based automations systems, and the use of "workpiece routing" defined by process engineering knowledge rather than embedded in the a CAD model using FSM's [2], as well as the novel integration of a virtual human.

7.3 Future Work.

The development of the toolkit described in this thesis has addressed the major aspects of the end user requirements. There are some aspects of the toolkit that require further development or functional enhancement. However, it is important that these enhancements are achieved without impacting the toolset's current simplicity and usability, whilst maintaining the core concept of the *integrated model*.

7.3.1 Virtual commissioning

As the CCE toolset provides a mechanism for building operational scenarios using workpiece routing, the ability to simulate the failure of a part sensor may be readily included in the automation system simulation, or actuators may be forced to fail for example with simulated timeout errors. The predicted response of the automation system may thus be observed, and captured, for various operating cases and failure scenarios. This includes checking the correctness of diagnostic information to be given to the machine operator and evaluating the system's ability to recover from the error. As this can process can be simulated rapidly, machine sensors and actuators can be failed at many points through the machine cycle to validate that the errors are reported in the correct format and that the automation system can be recovered from each failure case. This is referred to as *virtual commissioning*, and if implemented correctly, typical commissioning times can be reduced by up to 60% [ref]. This saving is potentially significant, as: 1) the end-user costs involved in having commissioning engineers on site for several months are high and 2) much more significantly, any delays in ramping-up machine operations to production volumes at likely to have a major impact on overall company profitability.

7.3.2 Automation system evaluation factors

There is potential for the inclusion of system evaluation factors, so that the assembly of components may automatically generate useful metrics. The reuse of existing known components may allow the semi automatic generation of evaluation metrics to provide design analysis or system performance metrics. The proposed evaluation factors are detailed below.

Complexity rating: Given that an appropriate component-based complexity model is defined, the toolset could potentially enable the complexity rating of any given machine design under consideration to be assessed. This would allow automaton systems to be automatically graded, highlighting more complex, and perhaps potentially risky system designs during the engineering process.

Fatigue rating: The fatigue rating of a human in the automation system could be assessed by the toolset, given an appropriate fatigue model, e.g., such as the Barnard Model [111], and the capture of the necessary work parameters (e.g., distance moved and load carried) and the degree of repetition within this work via the V-Man simulation.

7.3.3 Energy profile.

There is potential for the toolset's *integrated model* to be extended to include support for the energy profiles of each actuator. Such energy profiles could then be used in conjunction with the overall automation system sequence to predict the machine energy usage. Further to this, the information could be used to optimise the power requirements of the complete line, i.e., using the profiles from each individual automation system the overall line power usage may be derived and optimised.

7.3.4 Runtime evaluation

There is a large body of work yet to be carried out regarding the implementation of techniques to model the installed behaviour of components in a fully distributed real-time environment. This work requires knowledge of the target platform(s) their response times, network topology and network performance information. Once the final implementation is modelled with these parameters the communications responses can be evaluated, validating correct event processing order and therefore system operation.

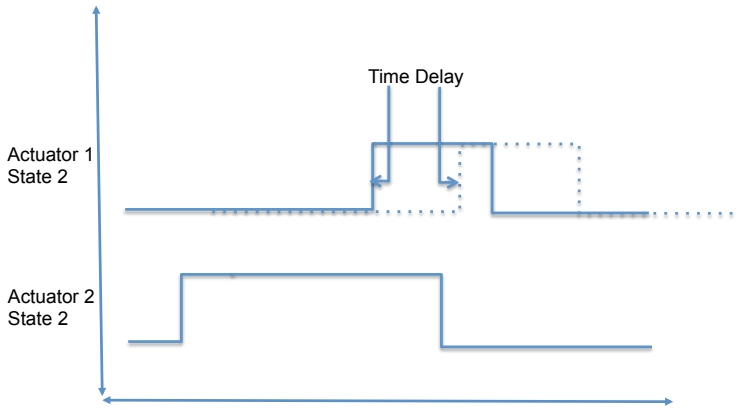


Figure 80 State change network time delay that can affect runtime operation

An example of this is shown in Figure 80. If an actuator is waiting for two conditions prior to movement (indicated by the overlap of Actuator 1 - State 2 and Actuator 2- State 2 in Figure 1) then the first signal could be delayed from the expected time shown. This will result in the control system responding differently from as designed. By introducing implementation specific response times and communication delays, timing errors may be determined prior to installation.

7.3.5 Robotic arms

There is also an immediate requirement to integrate the simulation of robotic cells as part of the CCE toolkit. There are a number of mature, dedicated, robot programming and simulation tools most notably RobCAD, Delmia Automation and Tecnomatix [73,75,76,90,91]. Within the CCE toolkit the focus of this new robotic functionality should be to support rapid initial overall system process planning, rather than the detailed robot path definition supported by the above toolsets.

7.3.6 Future Quantitative Evaluation

To date a largely qualitative evaluation of the toolset capabilities, against end-user requirements, has been carried out. Future work should be carried out to evaluate the approach in a more quantitative manner. This could involve evaluating the times to make a given design change, or the level of understanding necessary to be able to make a given change. The scope of this analysis might cover the generation of HMI code, error and diagnostic information, recovery sequence programming and documentation. In each case the CCE approach should be compared with current best

practices. Such a quantitative evaluation might also study the amount of reuse enabled by using the CCE tools and the related new component-based methodology, compared with traditional approaches.

7.4 Summary

The development of the CCE toolkit has proven the value of the component-based approach taken by the research in the automotive powertrain domain. It has been enlightening to witness the growing end-user acceptance of the approach, from one of mild interest to active involvement. This success has also introduced problems in that each member of the growing list of stakeholders (e.g., mechanical, process, layout, controls, productivity engineers) has identified their own new requirements for toolkit enhancements. This evolvement from the wider stakeholders indicates the validity of the approach, but it is important to evaluate the implementation of these requirements to ensure the CCE Toolkit does not become over complex and detract from the simplicity of the approach.

[1] Fank Lüders, “An Evolutionary Approach to Software Components in Embedded Real-time Systems”, Department of Computer Science and Electronics Mälardalen University 2006

[2] Szer Ming Lee, “A Component Based Distributed Control Paradigm For Manufacturing Automation Systems”, Doctoral Thesis Loughborough University

[3] Definitions from Wikipedia.org

[4] G.H. Lee, “Reconfigurability consideration design of components and manufacturing systems”, Int. Journal of Advanced Manufacturing Technology, vol. 13 pp. 376-386, 1997

[5] François Jammes, Harm Smit, ”Service-Oriented Paradigms in Industrial Automation”, Schnieder Electric Corporate Reseach, France.

[6] RI-MACS Project deliverable 2.2 “Guidelines for multi-agent based distributed control architecture and ontology for manufacturing and reference specifications for specific industrial domains”

[7] “Quality of Service Technical White Paper”, Microsoft Press, September 3, 1999

[8] Gerardo Pardo-Castellote, Ph.D., Stefaan Sonck Thiebaut, Ph.D., Mark Hamilton, Henry Choi, “Real-Time Publish-Subscribe Protocol for IP-Based Real-Time Communication”, Real-Time Innovations, Inc. http://www.rti.com/), September 2001

[9] Gerardo Pardo-Castellote, Stan Schneider, and Mark Hamilton, “NDDS: The Real-Time Publish-Subscribe Middleware,” Proceedings of the IEEE Real-Time Systems Symposium, 1997

[10] A. Gunasekaran “Agile manufacturing: enablers and an imlementation framwork”, int. journal prod. Res.,1998, Vol. 36, no. 5, pp.1223-1247

[11] Anbazhagan Mani, Arun Nagarajan “Understanding quality of service for Web services”, IBM Developerworks technical article Jan 2002

[12] Chris Peltz, “Web Services Orchestration and Choreography”, SOA World Magazine, Jun 17, 2003

[13] Thomas, D.W., West, A.A., Harrison R, McLeod C.S. "A Process Definition Environment For Component Based Manufacturing Machine Control Systems Developed Under The Foresight Programme”, at SAE 2002 World Congress & Exhibition, March 2002

[14] James S. Albus, “RCS: A Reference Model Architecture for Intelligent Control”, Computer IEEE Computer Society, May 1992 (Vol. 25, No. 5) pp. 56-79

[15] Brown A, “From Component Infrastructure to Component-Based Development”, Proceedings of the International Workshop on Component-Based Software Engineering (Kyoto, Japan, 1998).

[16] Vitharana. Risks and challenges of component-based software development. Communications of the ACM (2003) vol. 46 (8) pp. 67-72

[17] John Paul MacDuffie, Kannan Sethuraman Marshall L. Fisher “Product Variety and Manufacturing Perfomance: Evidence from the International Automotive Plant Study”, Management Science/ Vol. 42 No. 3 March 1996

[18] Arian Zwegers, “A study in shop floor control to determine architecting concepts and principles”, Doctoral Thesis Eindhoven University Press, Eindhoven, ISBN 90-386-0699-4.

[19] anon, “The Transaction Manager Resource Manager”, Open Blueprint Transaction Manager, IBM Corp 1996.

[20] F.B. Verndat, “Research agenda for Agile Manufacture”, International journal of Agile Manufacturing Systems, vol. 1, pp 37-40, 1999

[21] R.M. Setcho and N. Lagos “Reconfigurability and Reconfigurable Manufacturing Systems – State-of-the-art Review”, 2nd IEEE International Conference on Industrial Informatics (INDIN’04) Berlin 2004

[22] T.Wagner " An Agent-oriented Approach to industrial automation systems", 3rd International Symposium on Multi-Agent Systems, Large Complex Systems and E-Business MALCEB2002Erfurt/Thuringia, Germany , 2002

[23] Patterson, R., Hardt, D., Neal, R. "Next-generation Manufacturing – A Framework for Action" The Agility Forum, 1997

[24] Hoda A., "Flexible and reconfigurable manufacturing system paradigms", International Journal of Flexible Manufacturing Systems, Volume 17, Number 4, pp 261-276, October, 2005

[25] Monfared R.P., West A.A., Harrison R. Weston R.H, "An implementation of the business process modeling approach in the automotive industry", Proceedings of the I MECH E Part B Journal of Engineering Manufacture, Volume 216, Number 11, 1 November 2002, pp. 1413-1427(15).

[26] Tom Donnelly, David Morris and Tim Donnelly , "Modularisation and supplier parks in the automotive industry", Caen Innovation Marché Entreprise / Caen : IAE Caen Basse-Normandie , 2006

[27] Miller, Thomas Dedenroth; Pedersen, Per Erik Elgård, "Defining Modules, Modularity and Modularization : Evolution of the Concept in a Historical Perspective", Proceedings from Fuglsoe Research Seminar, IPS 1999.

[28] Duncan McFarlane, "Modular Distributed Manufacturing Systems And The Implications For Integrated Control", Proceedings of IEE Colloquium on Choosing the Right Control Structure, London, UK, 1998.

[29] Chun-Che Huang, "Overview of Modular Product Development", Proc. Natl. Sci. Counc. ROC(A) Vol. 24, No. 3, 2000. pp. 149-165

[30] C.Szyperski, "Component Software: Beyond Object-Oriented Programming", ACM Press, Addison-Wesley Professional (December 19, 1997)

[31] H. Kopetz, "Component-based design of large distributed real-time systems", Control Engineering Practice, Volume 6, Issue 1, January 1998, Pages 53-60

[32] anon. , “STEPS (Structured Transfer-machine DEEI Programming System). Release 3.1”, Ford Motor Company 1996.

[33] anon. , “Function block steps STEPS (Structured Transfer-machine DEEI Programming System). Release 3.1”, Ford Motor Company 1996.

[34] Theodore J. Williams and Hong Li, “The Lifecycle of an Enterprise”, Institute for Interdisciplinary Engineering Studies, Perdue University, USA

[35] Quinn, R.D.; Causey, G.C.; Merat, F.L.; Sargent, D.M.;Barendt, N.A.; Newman, W.S.;Valasco, V.B.; Podurski, A.;Ju-Yeon Jo; Sterling, L.S.; Yoohwan Kim, “Design of an agile manufacturing workcell for light mechanical applications”, IEEE International Conference on Robotics and Automation, 1996.

[36] Paul T Kidd, “Agile Manufacturing: Key Issues”, Online Paper, http://www.cheshirehenbury.com

[37] Nada Matta, Benoit Eynard, Lionel Roucoules, Marc Lemercier, “Continuous capitalization of design knowledge”, Workshop on Knowledge Management and Organizational Memories ECAI'02

[38] Joe Schlesselman, Brett Murphy, “Publish-subscribe Approach Aids Military Comms”, Real-Time Innovations, COTS Journal, March 2003.

[39] C. Riquier, N. Ricard, C, Rousset, “DES (Data Exchange System), Publish/subscribe architecture for Robots”, First National Workshop on Control Architectures of Robots – April 2006 Montpellier.

[40] C.S. McLeod, I. A Coutts, R. Harrison, “COMPAG System Specification”, Internal MSI report 2006.

[41] Gamma E., Helm R., Johnson R., Vlissides J.,”Design Patterns Elements of Reusable Object-Oriented Software”, Addison-Wesley ISBN 0-201-63361-2

[42] Jammes F., Harm Smit “SODA Service Oriented Device and Delivery Architectures”, ITEA Full Project Proposal Oct 2005.

[43] Lee S.M., Harrison R., and West A.A. ,"A component-based control system for agile manufacturing", Proc. IMechE Vol. 219 Part B: J. Engineering Manufacture

[44] Handy C., "The Age of Paradox", Boston: Harvard Business School Press 1995 ISBN 0875846432

[45] Maffeis S., Schmidt D.C. ,"Constructing Reliable Distributed Communication Systems with CORBA", IEEE Communications Magazine Vol 14, No. 2. 1997

[46] Dilts D.M., Boyd N.P., Whorms H.H.,"The Evolution of Control Architectures for Automate Manufacturing Systems", Journal of Manufacturing Systems, Vol 10, No. 1, pp 354, 1991

[47] anon, "DPWS Toolkit Ver. 2 User Guide", Schnieder Electric, Apr 2006

[48] Detoni A., Tonchia S., "Manufacturing Flexibility: a literature review", International Journal of Production Research Vol36 No. 6 pp 1587-1617 1998

[49] Yu J., Krishnan K.K., "A Conceptual framework for agent-based agile manufacturing cells", Information Systems Journal, Volume 14, Number 2, April 2004, pp. 93-109

[50] Moore P.R., Yeung W.H.R, "Distributed cell controllers using colour petri-nets and Fieldbus: An Application in flexible assembly", International Journal of Production Research Vol 35 No. 2 pp 237-340 1997

[51] anon, "Festo Didactic MPS® User Manuals", Festo Didactic GmbH & Co

[52] Tennefoss, M.R. "Lonworks Technology Comparison: LONWORKS® Systems versus DeviceNet ®", White Paper, Echelon Corporation.

[53] Anon, "Open System Design Guide", Echelon corporation.

[54] Allan McNaughton,"Tooling Up for Agile Development", whitepaper, Seapine Software promotional material. http://www.technical-insight.com/my_samples/agile.pdf

[55] Edward Yourdon, "Object Oriented System Design an Integrated Approach", Prentice Hall International, ISBN 0-13-176892-1

[56] Colombo A. W., “Service-Oriented Cross-Layer Infrastructure for Smart Eervice-Oriented Cross-Layer Infrastructure for Smart Embedded Device”,

[56] Sperling W, Lutz P, “Designing Platforms for am OSACA Control”, Proceedings of the International Mechanical Engineering Congress and Exposition, (The ASME Winter Annual Meeting) Dalles/USA, November 16-21, 1997

[57] Orfali R, Harkey D, Edwards J., “The Essential Distributed Objects Survival Guide”, John Wiley & Sons, 978-047112993, 1995

[58] Crnkovic I., “Component-based software engineering – new challenges in software development”, in Software Focus: Jon Wiley and Sons, 2001.

[59] Anon. “OPC Standard”, The OPC Foundation, http://opcfoundation.org

[60] Harrison R. and West A.A., “Component based paradigm for the design and implementation of control systems in electronics manufacturing machinery”, Journal of Electronics Manufacturing, Vol.10, pp. 1-17, 2000.

[61] Anon, “Service-Oriented Architecture (SOA) Definition”, http://sericearchitecture.com/webservices/articles/service_oriented_architecture_soa_definition.html

[61] Anon, “A Risk Management Standard”, Institute of Risk Management, http://www.theirm.org/publications/PUpublications.html 2010

[62] ISO Guide 73:2009

[63] Ong. Evaluating the Impact of Adopting a Component-Based Approach within the Automotive Domain. (2005) pp. 1-295

[64] Mellor. A component based approach to human machine interface systems that support agile manufacturing. (2010) pp. 1-242

[65] Shuping…. The research of V model in testing embedded software. Computer Science and Information … (2008)

[66] http://en.wikipedia.org/wiki/V-Model_(software_development)

[67] Johanson. The V-Model. (1998)

[68] Chaffin. Development of computerized human static strength simulation model for job design. Human Factors and Ergonomics in Manufacturing & Service Industries (1997) vol. 7 (4) pp. 305-322

[69] Erlmann. Tecnomatix Process Simulate Human. (2008) pp. 1-2

[70] Karhu. Correcting working postures in industry: A practical method for analysis. Applied Ergonomics (1977) vol. 8 (4) pp. 199-201

[71] Efstathiou,J, Expert Systems in Process Control 0-582-04267-4 Longman

[72] Corkill, D.D., Blackboard systems. AI expert, 1991. 6(9): p. 40-47.

[73] Bernard. Virtual engineering: methods and tools. J. Engineering Manufacture (2005) Vol. 219 Part B

[74] Burdea, G. and Coiffet, P. Virtual Reality Technology, 1993 (John Wiley New York).

[75] Wang. Definition and review of virtual prototyping. Journal of Computing and Information Science in … (2002)

[76] UGS Tecnomatix: The importance of Design Process, Manufacturing Today Magazine, March 2005, no 131.

[77] DELMIA V5 Automation Platform: Merging Digital Manufacturing with Automation, ARC White Paper February 2006, www.3ds.com

[78] Manufacturing talk Editorial Team on 15 September 2006, Software integrates the shop-floor with PLM

[79] V-Sim Incorporated: Virtual Manufacturing and software Emulation: White Paper, © V- Joe Hugan, October 2006, Novi Detroit, Michigan. www.v-sim.com.

[80] Moore et al. Virtual engineering: an integrated approach to agile manufacturing machinery design and control. Mechatronics (2003)

[81] Rychtyckyj. DIMS: Ten Years of AI for Vehicle Assembly Process Planning. … OF THE NATIONAL CONFERENCE ON ARTIFICIAL INELLIGNCE (1999)

[82] Katayama H. A Method of Analysing Trade-off Relaiton and its Application: A Case Study on Formation Management of Physically Challenged Workers. International Journal of Information (2008)

[83] Reddy C. Industrial Engineering and Management. books.google.com (2007)

[84] Macdonald et al. An evaluation of current practice in setting work rates. Report to Worksafe Australia (1999)

[85] http://www.managers-net.com/pmts.html

[86] Modapts Association, Modapts Modular Arrangement of Predetermined Time Standards, ISBN 0-72956-220-9 June 2006.

[87] Rychtyckyj. Ergonomics analysis for vehicle assembly using artificial intelligence. AI Magazine (2005) vol. 26 (3) pp. 41

[88] Rychtyckyj. Intelligent manufacturing applications at Ford Motor Company. Fuzzy Information Processing Society, 2005. NAFIPS 2005. Annual Meeting of the North American (2005) pp. 298 – 302

[89] http://www.sammiecad.com

[90] Erlmann. Tecnomatix Process Simulate Human. (2010) pp. 1-3

[91] DELMIA-Virtual-Ergonomics-brochure. (2008) pp. 1-4

[92] Snook and Ciriello. The design of manual handling tasks: revised tables of maximum acceptable weights and forces. Ergonomics (1991) vol. 34 (9) pp. 1197-1213

[93] Liberty Mutual Tables. (2004) pp. 1-30

[94] Lopes et al. An anthropomorphic robot torso for imitation: design and experiments. Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on (2004) vol. 1 pp. 661-667 vol. 1

[95] Camarinha-Matos and Afsarmanesh R.J Rabelo, Infrastructure developments for agile virtual enterprises. International Journal of Computer integrated Manufacture (2003) vol. 16 pp. 235-254

[96] Ishii. Modularity: a key concept in product life-cycle engineering. Handbook of Life-cycle Engineering (1998)

[97] Gunasekaran and Yusuf. Agile manufacturing: a taxonomy of strategic and technological imperatives. International Journal of Production Research (2002)

[98] Elkins and Huang…. Agile manufacturing systems in the automotive industry. International Journal of Production Economics (2004) vol. 91 (3) pp. 201 – 214

[99] Michel Freyssenet, Koichi Shimizu and Giuseppe Volpato Globalization or Regionalization of the European Car Industry? ISBN 1403905819, 9781403905819

[100] A. K. Sethi, S. P. Sethi, 1990. Flexibility in manufacturing: a survey. The international journal of flexible manufacturing, vol.2, pp.289-328.

[101] B. K. Min, Z. Huang, Z. J. Pasek, D. Yip-Hoi, F. Husted, S. Marker, 2002. Integration of real-time control simulation to a virtual manufacturing environment. Journal of advanced manufacturing systems, vol. 1, no. 1, pp. 67-87.

[102] S. Fujii, T. Kaihara, H. Morita, 2000. A distributed virtual factory in agile manufacturing environment, International Journal of Production Research, vol. 38, no. 17, pp. 4113-4128.

[103] M. Upton, A. McAfee., 1996. The Real virtual Factory, Harvard Business Review, vol. 74, no.4, pp. 123-133. July-august 1996, reprint 96410.

[104] F. B. Vernadat, Enterprise modelling and integration: Principles and application, Published by Chapman & hall London, ISBN 0-412-60550-3.

[105] Haq I, Innovative Configurable and Collaborative Approach to Automation Systems Engineering for Automotive Powertrain Assembly; Doctoral Thesis 2009

[106] M.H. Ong Evaluating the Impact of Adopting a Component-Based Approach within the Automotive Domain: Doctoral Thesis 2004

[107] E. Gamma, R. Helm, R. Johnson and J. Vlissides, Design Patterns Elements of Reusable Object Oriented Software ISBN: 0-201-63361-2

[108] Punnuluk Phaithoonbuathong, Web Service Control of Component-Based Agile Manufacturing Systems. Doctoral Thesis 2009

[109] Daniel Vera, Innovative Approach to the Design and Realisation of a Virtual Prototyping Environment for Manufacturing Systems Engineering, Doctoral Thesis, December 2004

[110] Edward Mellor, A Component based approach to human machine interface systems that support agile manufacturing; Doctoral Thesis

[111] David Strong, MPH, CIH; Astra C. Townley, MS, Practical Field Methods for Assessing Ergonomic Risk Factors, ASSE Professional Development Conference and Exposition, June 22 - 25, 2003 , Denver, Colorado

[112] anon, Ford Internal Document, Ergonomics Health and Safety Global Reference of Ergonomic Assessment Tools (GREAT)

[113] D. Williams, The Birmingham Gun Trade; ISBN 978-0-7524-3237-3

[114] Li-Ling Wang; Hong-Ying Wei, Development of a distributed control system for PLC-based applications; Machine Learning and Cybernetics (ICMLC), 2010 International Conference. Vol 2 p906 – 909

[115] 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

[116] R. J. Rabelo, L. M. Camarainha-Matos, 2000, Agent based brokerage for virtual enterprise creation in the mould industry, E-Business and virtual Enterprise, pp. 281-290.

[117] D. Brandl and B. Consulting, “What is ISA-95? Industrial Best Practices of Manufacturing Information Technologies with ISA-95 Models,” 2008.

[118] B. Scholten, “The Road to Integration: A Guide to Applying the ISA-95 Standard in Manufacturing.”, 2007, ISBN 978-0979234385

[119] Lee, S.M. (2004). A Component-Based Distributed Control Paradigm for Manufacturing Automation System. PhD dissertation, MSI Research Institute, Wolfson School of Mechanical and Manufacturing Engineering, Loughborough University.

[120] Harrison, R., A.A.West., R.H.Weston, and R.P. Monfared, Distributed engineering of manufacturing machines. Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture, 2001. 215(2): p. 217-231.

[121] Wiklander J, Eliasson J, Kruglyak A, Lindgren P, Nordlander J, “Enabling Component-Based Design for Embedded Real-Time Software”, JOURNAL OF COMPUTERS, VOL. 4, NO. 12, DECEMBER 2009

[122] http://www.modapts.org

[123] Tom, Kirkham; Bepperling, Axel; Colombo, Armando Walter; McLeod, Stuart; Harrison, Robert, A Service Enabled Approach to Automation Management; Information Control Problems in Manufacturing, Volume # 13

1) BAROT, V., HARRISON, R. and MCLEOD, C.S., 2010. Distribution of machine information using Blackboard designed component for remote monitoring of reconfigurable manufacturing systems. IN: IEEE 24th International Conference on Advanced Information Networking and Applications Workshops (WAINA), Perth, WA., 20-23 April, pp. 145-151

2) V. Barot, S. McLeod, R. Harrison, and A. A. West, Efficient real-time remote data propagation mechanism for a Component-Based approach to distributed manufacturing; International Journal of Aerospace and Mechanical Engineering 4:3 2010

3) BAROT, V, McLeod C.S....et al., 2009. “Broadcaster”: An architectural description of a prototype supporting real-time remote data propagation in distributed manufacturing. IN: 7th IEEE International Conference on Industrial Informatics, (INDIN 2009), Cardiff, Wales, 23-26 June, pp. 557 - 563.

4) West, A., Harrison, R., Weston, R., Monfared, R. et al., "Distributed Engineering of Automotive Manufacturing Machines under the Foresight Vehicle Programme," SAE Technical Paper 2002-01-0467, 2002, doi:10.4271/2002-01-0467.

5) Thomas, D., West, A., Harrison, R., and McLeod, C., "A Process Definition Environment for Component Based Manufacturing Machine Control Systems Developed Under the Foresight Vehicle Programme," SAE Technical Paper 2002-01-0468, 2002, doi:10.4271/2002-01-0468.

6) Tom, Kirkham; Bepperling, Axel; Colombo, Armando Walter; McLeod, Stuart; Harrison, Robert, A Service Enabled Approach to Automation Management; Information Control Problems in Manufacturing, Volume # 13

7) Phaithoonbuathong, P.; Kirkham, T.; Mcleod, C.S.; Capers, M.; Harrison, R.; Monfared, R.P.; , "Adding factory floor automation to digital ecosystems; tools, technology and transformation.," Digital Ecosystems and Technologies, 2008. DEST 2008. 2nd IEEE International Conference on , vol., no., pp.288-293, 26-29 Feb. 2008 doi: 10.1109/DEST.2008.4635176

8) West, AA, Smith, JD, McLeod, CS (2009) Development and initial evaluation of a smart resistance training system, PROCEEDINGS OF THE INSTITUTION OF MECHANICAL ENGINEERS PART P-JOURNAL OF SPORTS ENGINEERING AND TECHNOLOGY, 223(P1), pp.31-47, ISSN: 1754-3371.

9) West, AA, Smith, JD, McLeod, CS (2009) Development and initial evaluation of a smart resistance training system, Proceedings of the Institution of Mechanical Engineers, Part P: Journal of Sports Engineering and Technology, 223(1), pp.31-47, ISSN: 1754-3371.DOI: 10.1243/17543371JSET23.

10) Edwards, JM, Coutts, IA, McLeod, CS (2000) Support for System Evolution through Separating Business and Technology issues in a Banking System. In Brand, L and Eds, JV (ed) Proceedings International Conference on Software Maintenance (ICSM 2000), San Jose, California, USA, pp.271-276, ISBN: 0-7695-0753-0.

11) Edwards, JM, Millea, T, McLeod, S, Coutts, IA (1997) A Legacy Banking System. In EPSRC SEBPC Legacy System Workshop 1, Durham University, pp.27-34