A component-based control system for agile manufacturing

S M Lee*, **R Harrison, and A A West**
Wolfson School of Mechanical and Manufacturing Engineering, Loughborough University, Loughborough, Leicestershire, UK

The MS was received on 22 September 2003 and was accepted after revision for publication on 16 September 2004.

DOI: 10.1243/095440505X8000

Abstract: Increasing complexity and interdependency in manufacturing enterprises requires an agile manufacturing paradigm. This paper considers the characteristics that a component-based control system needs to embody to support agility in this context. The concept of a component that incorporates automation hardware, software, control services, and manufacturing-related services into a basic modular unit is reported. A component encapsulates low-level, real-time implementation and component-specific knowledge, which are abstracted and exposed as standard interfaces to enable it to be integrated into a manufacturing system as a common, reusable building block. The concept and control of the component is outlined in this paper and the methodology for the implementation of a component-based control system is illustrated. Finally, the implementation of the component-based concept at Johann A. Krause Maschinenfabrik GmbH, a major manufacturer of automotive assembly automation systems, is described.

Keywords: component-based systems, distributed control, agile manufacturing

1 THE NEED FOR AGILE MANUFACTURING

The rapid spread of information technology and economic globalization has directly affected manufacturing industries across the world. The increasing array of available automation and information technology brings with it an explosion of technical knowledge necessary to operate next-generation systems. This in turn drives an increasing complexity and interdependency in manufacturing enterprises as more and more knowledge is required to fulfil customer expectations [**1**]. In order to survive in this environment, a series of measures have to be employed in order to reduce investment risk, respond more flexibly to volume changes, speed-up model turnover, and facilitate equipment upgrading [**2**].

The emergence of an agile manufacturing paradigm is driven by the need of manufacturing enterprises to respond rapidly and flexibly to unpredictable market demands [**3**]. This demands a manufacturing system that can be reconfigured to accommodate changes in the product and process. It also emphasizes the reuse of knowledge and core competencies by suitable alliances to reduce cost and improve efficiency [**4**].

In response to the requirements for agile manufacturing, there is a need for a manufacturing control system that can:

(a) provide the means and capabilities to support flexibility, reconfigurability, and reusability; such system characteristics are important to enable the launch of new product models to be undertaken quickly, and to support the rapid adjustment of manufacturing system capacity to meet changes in market demands;
(b) enable new functions and process technology needs to be rapidly integrated into existing manufacturing systems.

The characteristics of traditional manufacturing control systems often do not readily enable the agility required for modern manufacturing. The control is often implemented in a rigid, hierarchical structure and the control code (normally ladder logic programming) is often complex and difficult to modify and

**Corresponding author: MSI Research Institute, Wolfson School of Mechanical and Manufacturing Engineering, Loughborough University, Loughborough, Leicestershire LE11 3TU, UK; email: s.m.lee@lboro.ac.uk*

reuse as it couples application-level process definition together with input/output (I/O) software of low-level control devices.

Researchers in the Manufacturing System Integration (MSI) Research Institute have been involved in an Engineering and Physical Sciences Research Council (EPSRC)-funded project to investigate a 'component-based paradigm for agile manufacturing' (COMPAG) and its application in automotive engine manufacture [**5**]. Component-based control technology has been identified as the key enabler to this paradigm. This paper will establish a definition of a component in this context, describe the control concept, and outline the implementation of a manufacturing control system using the component-based approach.

2 MODULARITY IN MANUFACTURING

Modularization has been identified as a means of handling the complexities and changes in manufacturing [**6**, **7**]. The concept of modularity can be applied in a number of different domains. According to Miller and Elgård, modularization can be broadly classified into physical, functional, and knowledge [**8**], as shown in Fig. 1.

Under *physical modularization*, physical blocks or modules are used to compose a final product. Modularity in this context has a finite physical boundary. The main characteristic of such modular systems is the design of generic hardware components that can be rearranged to suit different production configurations. A notable example of such a modular manufacturing system is the reconfigurable manufacturing system (RMS) developed by the Engineering Research Centre for Reconfigurable Manufacturing Systems (ERC/RMS) at the University of Michigan [**9–11**]. A reconfigurable robot workcell proposed by Chen *et al.* [**12**] at the Robotics Research Centre, Nanyang Technological University in Singapore, also adopts a similar concept.

Functional modularization links the definition of modules to service units commonly known as 'function blocks'. A manufacturing system utilizing the concept of functional modularity seeks to decompose complex manufacturing tasks into smaller and simpler structural functions, which are stored in function libraries and reused. A component-based control architecture for Holonic manufacturing systems (HMS) proposed by Chirn and McFarlane [**13**] is an example of a manufacturing system that adopts the functional modularity approach. The functional modules are reusable, extensible, and have the 'plug-and-play' capabilities. Function blocks under HMS are called software components, which are independently implemented packages of software services delivered in an encapsulated and replaceable container [**14**].

The Interface for Distributed Automation (IDA) group [**15**] and Component-Based Automation

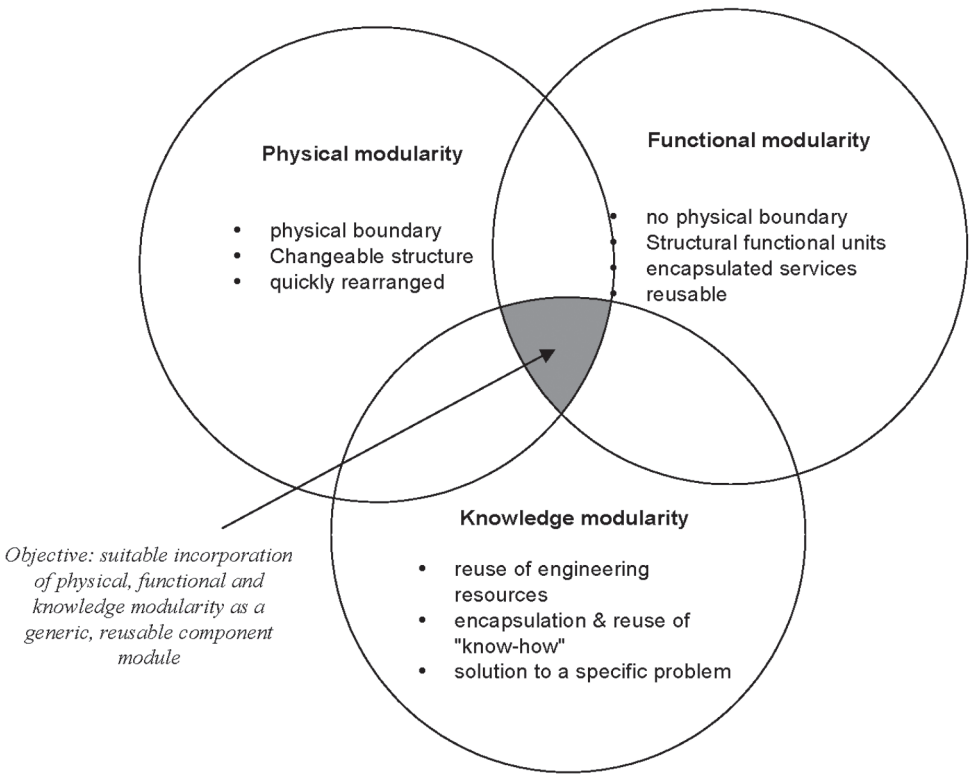

Fig. 1 Types of modularization according to Miller and Elgård [**8**]

advocated by Siemens [**16**] are examples of commercial organizations that have also recently adopted the function block approach in the implementation of their control systems. Both utilize IEC61499 function block diagrams [**17**] to organize control functions into reusable software modules.

Knowledge modularization identifies modules as 'carriers of knowledge' and 'reuse of engineering resources'. According to Miller and Elgård, such modules 'carry knowledge about problem, context and solution' [**8**] that could be reused at different phases of development or in different systems.

The complexity of modern manufacturing systems demands tight integration of physical and software functionalities, as well as engineering know-how, in order to achieve efficient resource and knowledge reuse. Therefore, modularity should not be limited solely to any one of the three domains of modularization described above. To meet the requirements for manufacturing agility, automation hardware, software, control services, and manufacturing know-how, which are common to a manufacturing task, should be grouped together so that they can be manipulated, integrated, and managed as one modular unit. Such a module would represent a suitable intersection of physical, functional, and knowledge modularity, as shown in Fig. 1.

3 DEFINITION OF A COMPONENT

Functional decomposition of manufacturing systems and their evaluation against the drivers of change [**18**] have shown that the granularity of a modular unit has to be set at a low level, i.e. at the device level, in order to address the different agents of change [**19**] throughout the manufacturing life cycle. In the present authors' work, this basic unit of granularity is defined as a 'component'. A component, in this case, encapsulates the required physical and functional attributes, knowledge, and system characteristics of a manufacturing system so that it can be readily integrated with other components to compose any required system [**20**].

Figure 2 illustrates the mapping of the attributes of modularity on to a component. A component consists of three constituents:

1. The *runtime constituent* represents the control aspect of a component. It contains physical elements (automation device, local controller, and interfacing circuits) and control functions that determine the runtime behaviour of the component. It is also a repository for knowledge that is specific to the component, such as manufacturer's data (e.g. manufacturer and product identification), device-related parameters, error diagnostics and management, life-cycle history, and predictive and preventative maintenance services.
2. The *engineering constituent* provides the information about a component that is required for process design and verification. In this constituent, the control behaviour of the component is represented in an abstract form for the user to design a component-based system and to validate it by simulating the system in an engineering environment. The engineering environment provides a platform for system integrators to design and build a manufacturing system based on a library of components. It hosts a suite of software tools to create and edit the system structure and to define the system application processes without dealing with the low-level control and implementation details of the components.
3. The *visualization constituent* includes information about the component required for presentation purposes. It includes computer aided design (CAD) data and a behaviour description to enable the component to be modelled and visualized in a two- or three-dimensional environment.

In the following section, a description is given of the runtime constituent of the component and the implementation of a distributed control system based on the concept of a component. A discussion of the engineering and visualization constituents of the component has been presented in other publications [**20**, **21**]. The word 'component' used in the following sections of the paper therefore refers to the runtime constituent of the component.

4 COMPONENTS IN A RUNTIME ENVIRONMENT

In general, current automation solutions consist of centralized control units connected to sensors and actuators either directly or via a fieldbus. Although intelligent sensors and actuators are now widely used, the control of the process usually remains within a centralized unit that utilizes traditional command–response control methods.

As a result of increasingly powerful microprocessors, decreasing physical size, computing cost, and better communication technologies, control functionalities previously dedicated to central processors can be totally distributed to intelligent automation devices communicating via a distributed network. Apart from local control, these devices are also capable of local decision making, error handling, and local information processing [**22**]. A distributed control system that is composed of such distributed devices does not require a central processor to coordinate control sequences and issue commands.

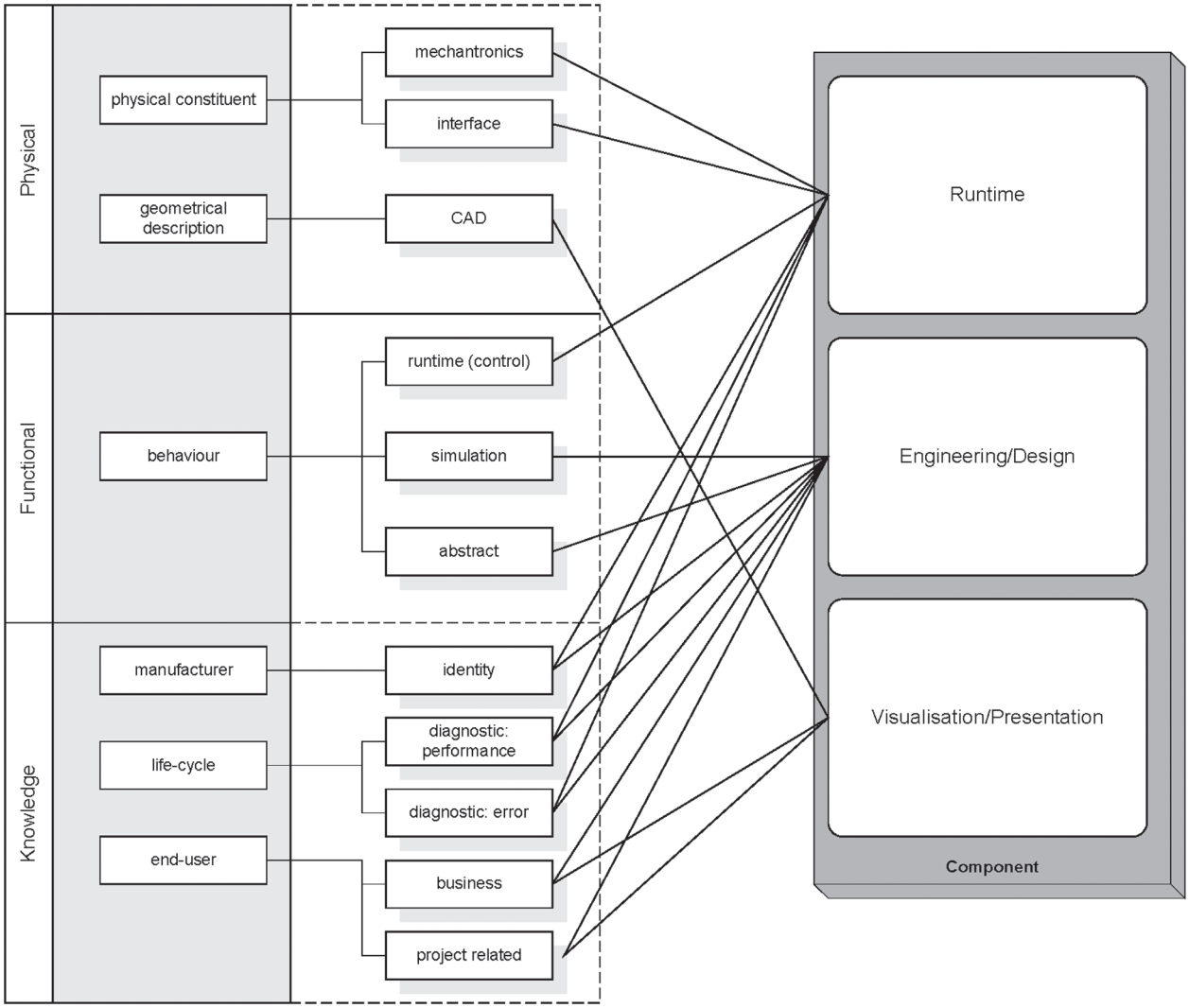

Fig. 2 Constituents of a component

The structure of a distributed control paradigm matches the requirements of modularity and agility established in this research as it marries the required functionalities and knowledge attributes of a component to its physical counterpart. The component-based control system proposed here adopts the distributed control approach. It utilizes an object-based decomposition methodology proposed by Hopkinson [**23**] to resolve a control system into distributed, interacting components. Each component of this component-based control system behaves as an autonomous control unit. It is aware of its states and has local reasoning capability to assess the situation of the system (i.e. is situation aware) to decide when it could execute tasks at any given point in time. The components communicate through a distributed network by sending and receiving state messages to and from other components. Such a component would know *what* operation it should perform, *how* to execute the operation and *when* it should execute without requiring any form of command from another host.

4.1 Physical representation of a component

Figure 3 shows the physical representation of a component. A component consists of a controller, interface electronics and the automation device. The local controller (microprocessor) contains the application for controlling the I/O for the device. It can communicate with other components in the network in a peer-to-peer manner. Data local to the component – such as configuration parameters, diagnostic information, and life cycle history – are stored in the memory of the controller. Such data can be accessed through a management tool via the network for engineering, commissioning, and maintenance purposes. Components contain one or more automation devices. These are actuating and sensing devices that interact and perform work in the manufacturing

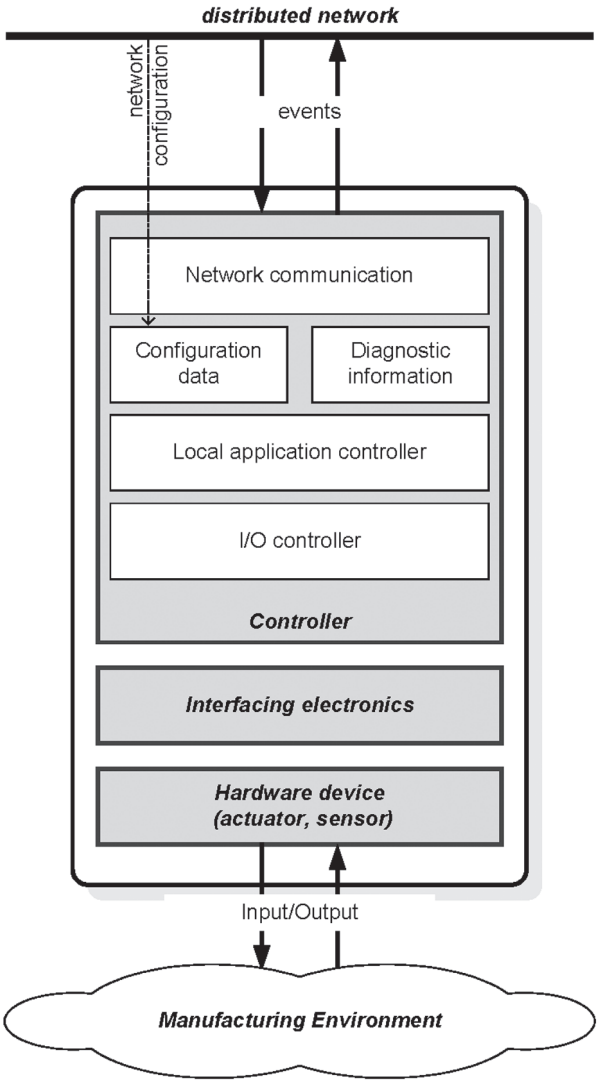

Fig. 3 Physical view of a control component

environment. The interfacing electronics are used to condition and translate the output control signals from the controller to the automation device and the input signals from the device to the controller.

4.2 Control behaviour of a component

Automation devices within a component are grouped into control elements. Each element provides independent, closed-loop control of its associated automation devices. For example, an element for a two-position clamp actuator (Fig. 4) controls the outputs to extend and retract the clamp. Position sensors are installed at the two ends of the stroke of the clamp piston to provide position feedback to the element.

The behaviour of the control elements is represented using finite state machines. Such behaviour can be described by a set of states, with transitions to proceed from one state to the next, and a combination of events known as conditions that need to be satisfied to trigger the transitions. The finite state machine for the element F_{ele} could be represented as follows

$$F_{\text{ele}} = \{X, \alpha, E\}$$

where X is the set of all states in the finite state machine, α is the transition, and E is the set of events that triggers the transitions.

The set of events E that triggers a transition could be either external or internal conditions of the manufacturing system. Internal conditions are events that occur within the boundary of the component, such as a local sensor feedback signal or the expiry of an internal timer. External conditions, on the other hand, are related to the states of the elements of other components in the system. The event E could, therefore, be represented as

$$E = \{E_{\text{e}}, E_{\text{i}}\}$$

where E_{e} represents external conditions of the state of other elements and E_{i} represents internal conditions related to the local feedback.

Figure 5 shows the finite state representation of a clamp element as discussed earlier. It is composed of four states: retracted, extending, extended, and retracting. External conditions need to be satisfied

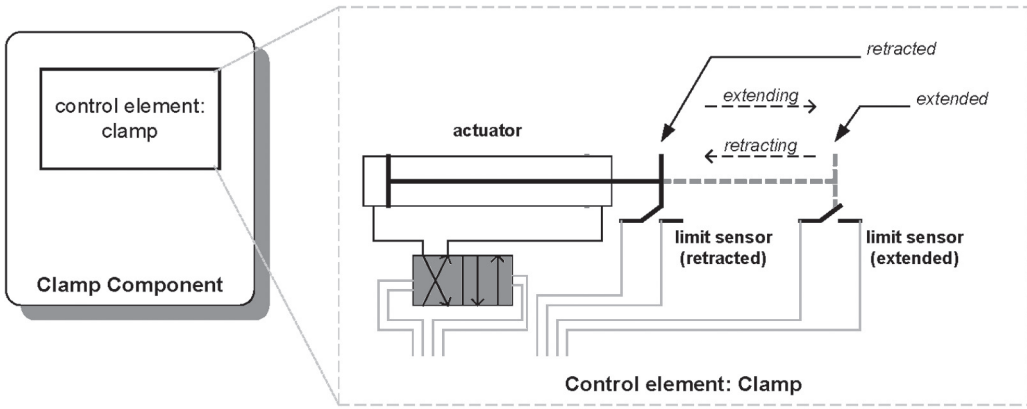

Fig. 4 Control element of a component

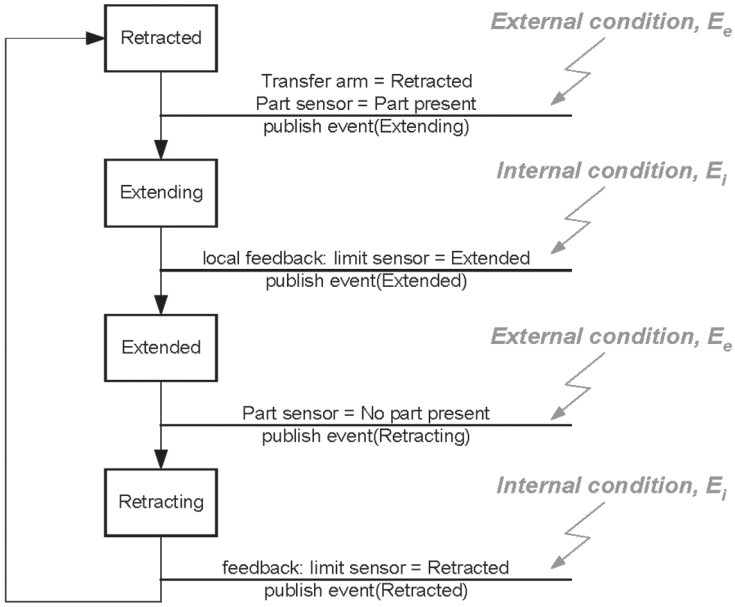

Fig. 5 State-transition behaviour of a clamp element

for the clamp to proceed from the retracted to the extending state, and from the extended to the retracting state. These conditions are associated with the states of other control elements and are external to the clamp element. The condition for the transition will be satisfied if the state of the clamp is at retracted, and the external condition defined for the transition is related to another transfer arm element and part sensor element in the following relationship

$$E_{\text{e,retracted}} = \text{transfer arm is at state retracted } \textbf{AND} \text{ part sensor is at state part present}$$

i.e. if the state of the transfer arm element is at retracted and that of the part sensor element is at part present. The clamp will move from the retracted to the extending state. At the same time, the clamp actuator will extend its piston.

Internal conditions are related to the local position feedback of the limit sensors when the clamp has reached the end of travel. In the case of the clamp element, when the clamp has reached the end of travel while extending its piston (at state extending), the position sensor installed at the end of the stroke is triggered. The input signal will be received by the controller and interpreted as the internal condition for the clamp element to advance to the next (extended) state. The same reasoning applies for the clamp to proceed from the retracting to the retracted state.

This approach enables generic operation specific to the automation device to be pre-programmed and embedded into the component as an element. The control of the device is hidden and its behaviour is represented by the finite state machine. The behaviour of the elements is not specific to any manufacturing process. However, by integrating the components into a system and correctly interlocking the correct external conditions of the control elements to the states of other elements, the system could be configured to execute specific manufacturing operations.

4.3 Engineering of component-based control systems

The component-based control system adopts the object-oriented concepts of encapsulation and abstraction to enable the user to exploit the functionalities of the components without thinking about its implementation complexities. This 'black-box' implementation approach allows changes and enhancements to the manufacturing process to be conducted more rapidly with minimal disruption to the system.

From the logical perspective, a component consists of interface, internal behaviour, and invocation. The component interface defines communication 'plugs' called network interfaces where the component could publish data to the system and subscribe to data published from other components. Figure 6 shows the network interfaces of the component. A component receives network messages through the input interfaces shown on the left of the schematic, and publishes state and status messages through the output interfaces on the right.

The component can operate in three operating modes. In the AUTOMATIC mode, elements in the components will cycle continuously through their states as long as the conditions are satisfied. In the

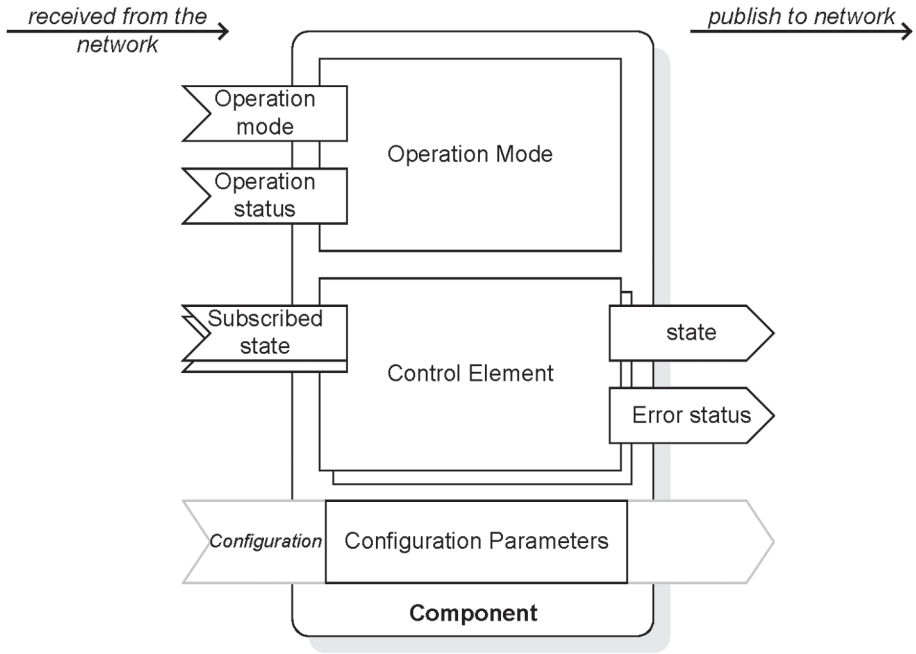

Fig. 6 Network interfaces of a component

STEP mode, when a step event is issued, the element will evaluate the external conditions associated with the current state and proceed through the states if it is satisfied. It will stop at the subsequent state where the next external condition is required and wait for the operator to generate the next step event. In MANUAL mode, the operator has maximum control over the component, with the ability to set the state of the elements to any state as long as the external conditions permit. The operation mode of the component could be set through the input network 'operation mode'. Operating commands such as START, STOP, and RESET are received through the 'operation command' input network. The state and error status of the control element in the component is published to the network through the 'state' output network. The states of other components that are required to invoke the operation of the component are received through the 'subscribed state' network inputs and configured using the 'configuration' input network.

The component internal behaviour defines how the component works. The basic operations of the automation devices may be pre-programmed by the component supplier and encapsulated into the component. The encapsulated behaviour may deal with complex control dynamics that require special knowledge and expertise from the component developer, as well as any functionality that could enhance the capability of the component, e.g. error diagnostics, predictive and preventative maintenance, or life cycle data acquisition. The operations are abstracted into finite state behaviour of control elements as described in section 4.2. Communication through the network is also essential to the correct behaviour of the element. The component handles incoming network messages and evaluates the conditions for controlling the inputs and outputs to the devices. It also publishes state and error status updates to the network.

The invocation of a component determines when it can execute its operations. The condition for invocation of an operation is dependent on the combined states of other components in the system. The invocation condition is not programmed into the component. It is mapped on to the controller so that it can be defined and changed by the component user.

Practical control of a system often requires complicated logical expressions involving various combinations of **AND** and **OR** of various states of the actuator and sensors to interlock various automation devices in the system. Such interdependency relationships are defined via the external conditions of the respective element. The condition E_e can be stored in the controller of the component as configurable data. An example of such condition is

$$\begin{aligned} E_e = \{&(Exit\ Sensor = OFF)\ \textbf{AND} \\ &(Clamp = RETRACTED)\ \textbf{AND} \\ &(Probe = HOME)\ \textbf{AND} \\ &(Gantry = RAISED)\}\ \textbf{AND} \\ &\{(Drill\ Sensor = ON)\ \textbf{OR}\ (Probe\ Sensor = ON) \\ &\textbf{OR}\ (Entry\ Sensor = ON)\} \end{aligned}$$

The engineering of a component from its three perspectives – interface, internal behaviour implementation, and invocation – effectively decouples

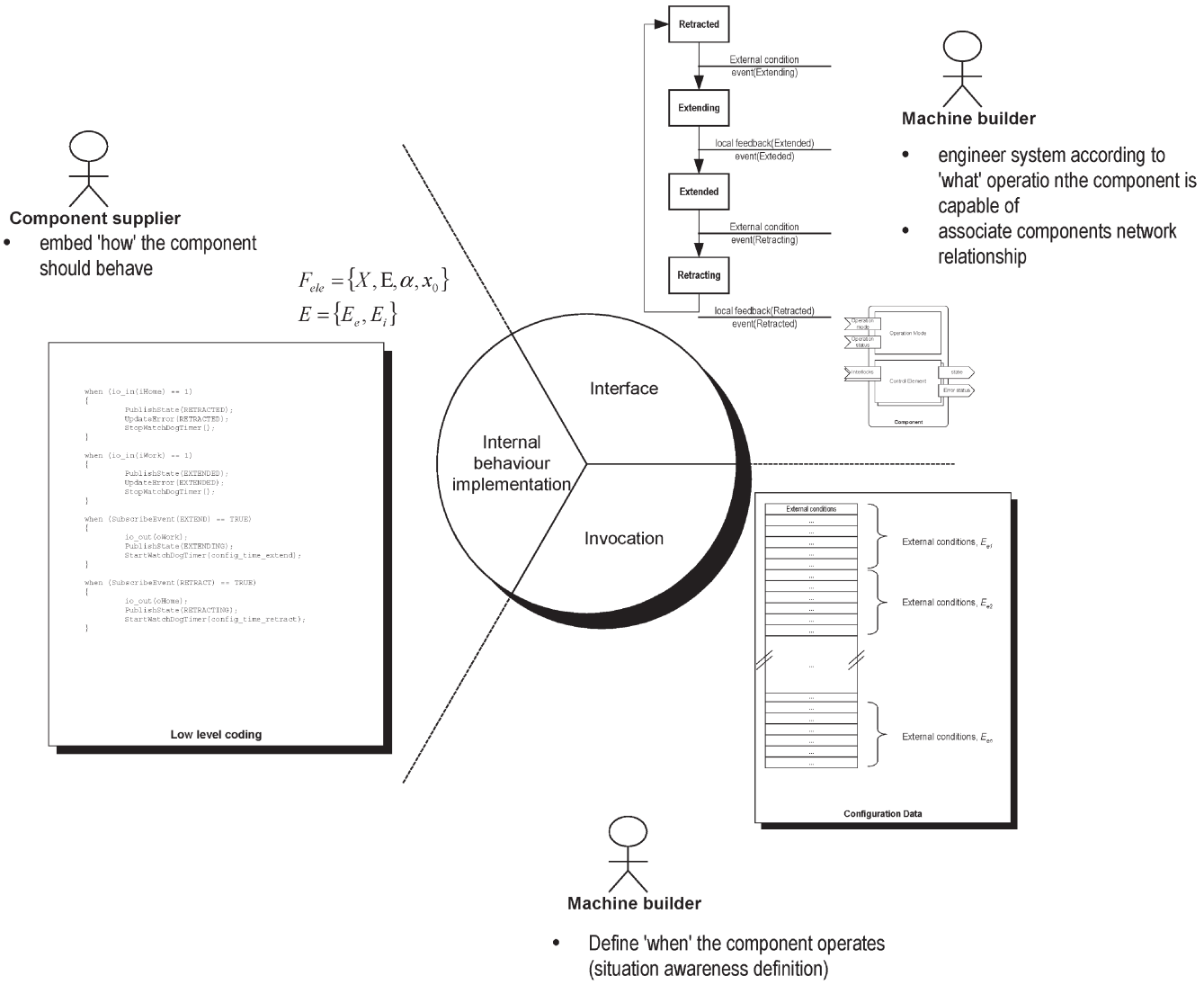

Fig. 7 Actors and their roles in a component-based control system

the component development process with component implementation and system process definition. As shown in Fig. 7, component suppliers can embed the internal behaviour of the component as a standard application in the controller. Additional capabilities, knowledge, and specialization can be encapsulated into the component to fine-tune or parameterize the automation of the component. I/O networks are exposed as network interfaces for components to communicate.

Users of components, typically the machine builders, can compose an automation system by selecting the set of components that can perform the desired manufacturing tasks. The user can then proceed directly to design the system's automation process and sequence defining the interdependencies of the components through specifying the external conditions of the elements in order to ensure the proper coordination of the sensors and actuators in the system. Based on the finite state representation, the external conditions for each element are defined as logical expressions related to the states of other elements. The network interfaces of the components are then inter-connected together and the external conditions are downloaded to the components.

This methodology enables the sequence and interlock logic of the control system to be specified and modified via the interface and invocation parts of the component, without reprogramming the component. Changes in the manufacturing process only involve the alteration to the invocation conditions of the affected component. Since the internal behaviour of the component has already been well tested and verified by the component supplier, the time and effort taken to commission the system are drastically reduced.

It is important that the network required for the component-based control system supports peer-to-peer communication and provides the capabilities of encapsulation and abstraction as discussed earlier. 'Late binding' capabilities are required so that communication paths and external conditions defining the interdependency of the elements can be defined after the components internal functionalities have been implemented. The term 'late binding' is commonly used in component-based software development. Binding is a process of invoking function calls of other software components. Late binding allows the exact property or method of the targeted software components that is being invoked by a software program to be determined at runtime, hence removing the need for such information to be defined and compiled into the program during the development phase. In the context of this report, late binding refers to the process of establishing the intercommunication relationship between the automation components *after* these components have been developed and implemented. Late binding is the process of logically interconnecting the components' output network interfaces (the data producer) to a targeted components' input network interfaces (the data consumer) after the components have been developed. Late binding allows the connection of components that are completely unaware of each other. Such connection information, together with the configuration parameters, will be downloaded to the components using a suitable network management tool. A number of suitable fieldbuses that support locally distributed automation are outlined in references [**24**] to [**26**].

5 ILLUSTRATION: COMPONENT-BASED CONTROL IN AUTOMOTIVE ASSEMBLY

The component-based control concept has been implemented in a test assembly line at Johann A. Krause Maschinenfabrik GmbH (Krause) in Bremen, Germany (Fig. 8), as part of an EPSRC project supported by major industrial partners including the Ford Motor Company UK (end-user for automotive production systems), Krause Assembly System, Cross Hüller Limited, Lamb Technicon Machining System (automotive production machine builder), Parker Hannifin Limited, and Bosch Rexroth Group (component vendors).

Krause develops, designs, and manufactures assembly systems for automotive assemblies and is a major assembly system builder for the Ford Motor Company as well as other automotive manufacturers [**5**]. In order to evaluate thoroughly the component-based implementation from an industrial perspective, the component-based system was subjected to standard commissioning tests conducted by the company's own commissioning team after the systems had been installed. The objective of these commissioning tests is to check the robustness of the machines in a real industrial environment subjected to typical disturbances. They provide practical verification of the ability of the component-based systems to meet the runtime requirements of automotive assembly systems.

The component-based assembly machine implemented at Krause comprises two systems: transport system and assembly system. The transport system delivers pallets around a test loop and sorts them

Fig. 8 Component-based control system in the Krause assembly line

into two lanes, according to the assembly operation to be performed on the pallet. The assembly system picks and places parts on to part holders on the pallet from a rack. The following components were implemented:

(a) five stop units with integrated pallet sensor,
(b) four conveyor drives,
(c) two power supply units,
(d) one diverter unit,
(e) two radio-frequency (RF) tagging units,
(f) two servo drives,
(g) one ultrasonic sensor,
(h) one fixing unit,
(i) one pneumatic gripper,
(j) one part sensor/monitor unit.

5.1 Implementation platform

The assembly system has been implemented using a commercial distributed control network technology called LonWorks™ (Local Operating Networks) developed by Echelon Corporation [**27**]. LonWorks™ [**28–30**] is a control technology that originated outside the industrial automation domain. LonWorks™ was developed by Echelon Corporation and standardized later as ANSI/EIA 709 [**27**]. Although it was not historically tailored for any specific application area, it has been primarily adopted for home and building automation. Unlike most fieldbuses, which are fundamentally based on a master/slave control and communication relationship, LonWorks™ provides a truly distributed, heterarchical architecture that allows devices full autonomy to communicate with one another through a common communication protocol.

A LonWorks™ based control system consists of: (a) a network referred to by Echelon as a LON (Local Operating Network) and (b) a set of control and communication devices, known as neurons, which communicate over the LON network using a peer-to-peer communication protocol known as LonTalk. Neuron-based devices (often referred to as control nodes or control modules) provide network communication, application processing, and I/O functionality. LonWorks™ technology enables the creation of programs for each distributed control node, enabling each node to perform some required application in response to a set of network and/or I/O-based events.

Twelve components were implemented for the transport system and eleven were implemented for the assembly system. Of the twenty-three components, only ten different types of components were required. For instance, the same *STOP* component was utilized for all five *STOP* units.

Details of the implementation and analysis of the result will not be given here, but are available in references [**31**] and [**32**]. Nevertheless, it is important to appreciate that, by adopting the component-based implementation, it is expected that as much as 50 per cent of the implementation effort could be saved. This is achievable due to the ability to utilize proven pre-assembled mechatronic modules (components) as the common building blocks to compose manufacturing automation systems by simply reconfiguring and interlocking them together instead of developing new control programs for each application [**33**].

5.2 System engineering and development

The development of each component underwent an iterative Analysis–Design–Implementation–Test life cycle (ADIT) [**34**], as illustrated in Fig. 9. The functional requirements of assembly automation were captured through interviews and reviews with the machine builder Krause and their component

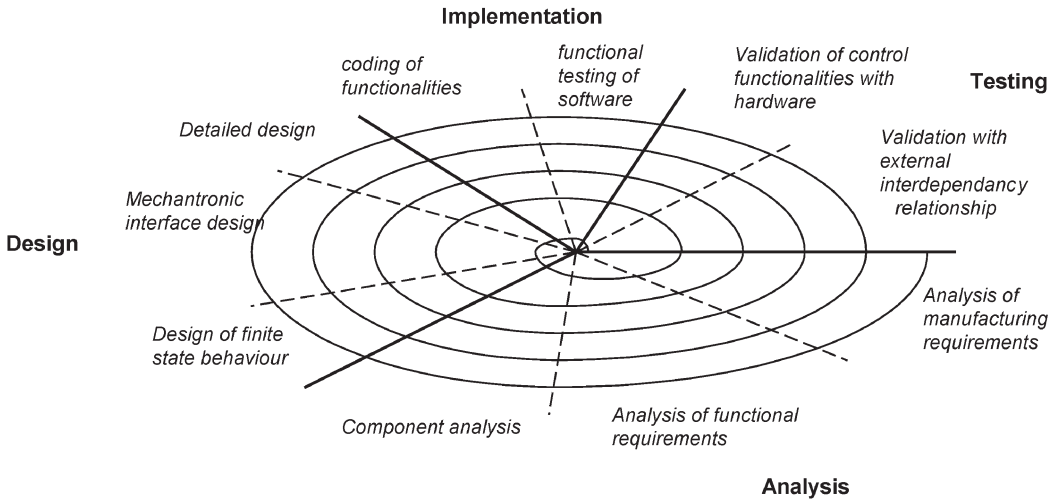

Fig. 9 Component development life cycle

suppliers. The design and implementation of the components were then conducted in-house. The functionality of each component was implemented based on the state-transition requirements of the required control elements. The components were then tested on-site to verify the operation of the automation devices. Validation at this stage of the work was focused on the automation functions of the devices associated with each component. This component-based development life cycle enabled the components to be designed, implemented, and validated concurrently and independently. In contrast, a conventional approach would largely have prevented validation until the complete system had been implemented.

Subsequently, the system was designed and integrated using an engineering software known as Process Definition Environment (PDE), developed separately as part of this research project [**21**]. Figure 10 shows a screen capture of the PDE tool. The PDE enables the component-based system structure to be constructed by graphically adding (drag-and-drop) components from the component library view to the machine hierarchy view. The control behaviours of these components are represented by their control elements and can be viewed as state-transition diagrams in a separate control element view. The interlocks of the control elements are defined by adding the state of other control elements to the conditions of the transition.

These software tools allow a system designer to compose a system from a library of components and then define the interdependency of the control elements to assure their proper coordination. The system was validated in a three-dimensional modelling environment before being implemented in the automation system [**35**, **36**]. The configuration data (containing the external conditions) were downloaded to the components in the runtime environment and the network interfaces of the components were bound logically together to enable the components to send and receive network messages. The control system was able to operate in AUTOMATIC and MANUAL operating modes and provided a realistic test bed for the implementation and evaluation of the component-based control approach.

The component-based approach in engineering and implementation of the assembly automation system has greatly improved the changeability and reusability of the system. As illustrated via the modified classical

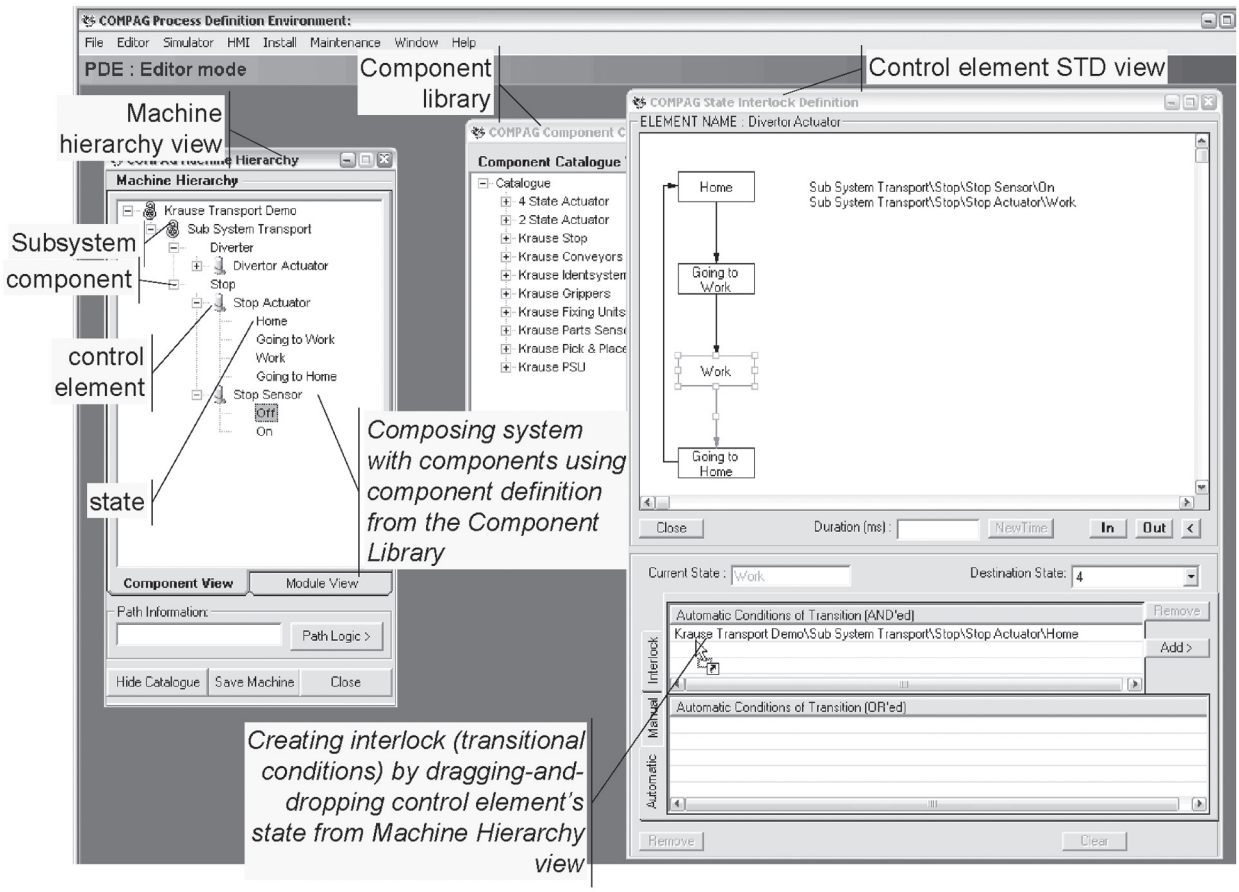

Fig. 10 Application development using the PDE tool

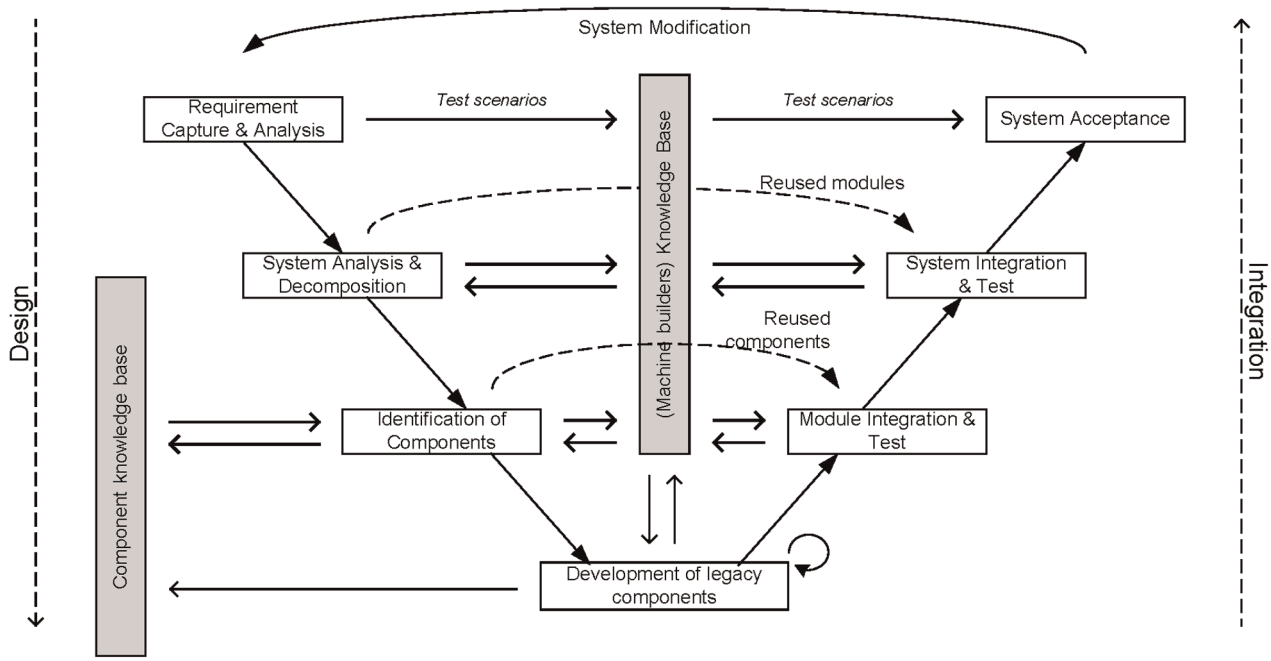

Fig. 11 Component-driven manufacturing system engineering life cycle

'V' diagram of a system engineering cycle in Fig. 11, during the system design process, existing groups of components (modules) that have been developed for earlier systems can be retrieved from the machine builder's knowledge base and reused in the new system. Conversely, new components can be added to the knowledge base for use in future systems. Legacy or new components are developed following the ADIT life cycle. The components are then integrated into the required system. Since the components have already been tested and validated individually, the integration task is significantly simplified and shortened. Hence, during commissioning of the assembly machine at Krause, most of the effort during the system integration and system acceptance phases of the engineering life cycle was devoted to ensuring the correct overall machine behaviour (execution timing and sequence of the automation).

The implementation will be discussed in more detail in a separate publication. Nevertheless, it is important to recognize that the effort taken to implement the control system using the component-based approach was found to be significantly reduced relative to the implementation of a PLC (programmable logic controller)-based system using current best practices [**33**].

6 CONCLUSIONS

A component-based approach has been proposed to address the challenges of agile manufacturing. In this paper, the definition of a component is established by integrating the modularity attributes of physical, functional, and knowledge modularity. The concept of a component is further developed for the control runtime environment. A distributed cooperative control concept described in this paper enables components to behave as autonomous, 'situation-aware' control units. An implementation methodology for the component-based control system is outlined that allows components to be developed and to be reused as 'black boxes' and that allows control systems to be reconfigured instead of reprogrammed. This system has been evaluated as a major vendor of assembly automation equipment and offers significant advantages in terms of system agility over traditional approaches.

ACKNOWLEDGEMENTS

The authors gratefully acknowledge the support of EPSRC, the Ford Motor Company Limited, Cross Hüller Limited, Johann A. Krause Maschinenfabrik GmbH, and the other collaborating companies in carrying out this research.

REFERENCES

1 Next-generation manufacturing: a framework for action, agility forum. In Agility Forum, January 1997.
2 **Camuffo, A.** Rolling out a world car: globalization, outsourcing and modularity in the auto industry. In *IMVP Working Papers*, 1999 (MIT Press, Cambridge, Massachusetts).

3 **Gunasekaran, A.,** and **Yusuf, Y. Y.** Agile manufacturing: a taxonomy of strategic and technological imperatives. *Int. J. Prod. Res.*, 2002, **40**, 1357–1385.

4 **Gunasekaran, A.** Agile manufacturing: enablers and an implementation framework. *Int. J. Prod. Res.*, 1998, **36**, 1223–1247.

5 **Harrison, R.,** and **West, A. A.** Assessment and implementation of a component-based paradigm for agile automation. MSI Research Institute, Loughborough University, EPSRC Innovative Manufacturing Initiative GR/M43586, 1998.

6 **Lee, G. H.** Reconfigurability consideration design of components and manufacturing systems. *Int. J. Advd Mfg Technol.*, 1992, **13**, 376–386.

7 **Garro, O.,** and **Martin, P.** Towards new architecture of machine tools. *Int. J. Prod. Res.*, 1993, **31**, 2403–2414.

8 **Miller, T. D.,** and **Elgård, P.** Defining modules, modularity and modularization – evolution of the concept in a historical perspective. Presented at 13th IPS Research Seminar on *Design for Integration in Manufacturing*, Fuglsoe, Denmark, 1998.

9 **Mehrabi, M. G., Ulsoy, A. G.,** and **Koren, Y.** Reconfigurable manufacturing systems: key to future manufacturing. *J. Intell. Mfg*, 2000, **11**, 403–419.

10 **Mehrabi, M. G., Ulsoy, A. G.,** and **Koren, Y.** Reconfigurable manufacturing systems and their enabling technologies. *Int. J. Mfg Technol. Mmt*, 2000, **1**, 113–130.

11 **Koren, Y., Heisel, U., Jovane, F., Moriwaki, T., Pritschow, G., Ulsoy, G.,** and **Brussel, H. W.** Reconfigurable manufacturing systems. CIRP Keynote paper, Vol. 48, 1999.

12 **Chen, I.-M., Chen, P., Yang, G., Chen, W., Kang, I.-G., Yeo, S. H.,** and **Chen, G.** A component technology based reconfigurable robot workcell. Presented at the Third International Conference on *Mechatronics Technology*, Hsinchu, Taiwan, 1998.

13 **Chirn, J.-L.,** and **McFarlane, D. C.** A holonic component-based approach to reconfigurable manufacturing control architecture. In Proceedings of *HolonMas 00*, London, 2000.

14 **McInnis, K.** Component-based development – the concepts, technology and methodology. Castek Software Factory Inc., USA, 2000.

15 Interface for distributed automation. IDA White Paper Revision 1.0, Modbus-IDA Organization, USA, 2001.

16 **Provis, G.** Point-and-click plant integration. In *Industrial Networking and Open Control*, 2002, Vol. 8. Internet resource http://www.industrialnetworking.co.uk.

17 **Lewis, R. W.** *Modelling Control Systems using IEC 61499 – Applying Function Blocks to Distributed Systems*, 2001 (The Institution of Electrical Engineers, London).

18 **Ong, M. H.** Modularity analysis of transfer line machine in Lamb Technicon. MSI Research Institute, Loughborough University, Loughborough, 2003.

19 **Monfared, R. P., West, A. A., Harrison, R.,** and **Weston, R. H.** An implementation of the business process modelling approach in the automotive industry. *Proc. Instn Mech. Engrs, Part B: J. Engineering Manufacture*, 2002, **216**(B11), 1413–1427.

20 **Harrison, R., West, A. A., Weston, R. H.,** and **Monfared, R. P.** Distributed engineering of manufacturing machines. *Proc. Instn Mech. Engrs, Part B: J. Engineering Manufacture*, 2001, **215**(B2), 217–231.

21 **Thomas, D. W., Harrison, R., West, A. A.,** and **McLeod, C. S.** A process definition environment for component based manufacturing machine control systems developed under the Foresight Vehicle Programme. In Proceedings of the SAE World Congress and Exposition on *Foresight Vehcile Technology: Consumer Driven Design, Manufacturing, Supply Chain and Purchasing*, Detroit, Michigan, 2002 (Society of Automotive Engineers, Warrendale, Pennsylvania).

22 **Lee, S. M.** Industrial automation: a network approach. MEng dissertation, Department of Mechanical and Production Engineering, National University of Singapore, 1999.

23 **Hopkinson, P.** A new approach to the development and maintenance of industrial sequence logic. PhD thesis, Department of Manufacturing Engineering, Loughborough University, Loughborough, 1998.

24 **Neumann, P.** Locally distributed automation – but with which fieldbus system? *Assembly Automn*, 1999, **19**, 308–312.

25 **Ranky, P. G.** Modular fieldbus designs and applications. *Assembly Automn*, 2000, **20**, 40–45.

26 **Schickhuber, G.,** and **McCarthy, O.** Distributed fieldbus and control network systems. *IEEE Computing and Control Engng J.*, February 1997, **8**(1), 21–32.

27 Control network protocol specification. EIA/CEA-709.1-B-2002, Echelon Corporation, 1999.

28 **Hollingum, J.** LonWorks – solutions in silicon. *Assembly Automn*, 1994, **14**, 25–27.

29 **Raji, R. S.** End-to-end solutions with LonWorks control technology. Echelon Corporation, 2001.

30 **Mahalik, N. P.,** and **Moore, P. R.** Fieldbus technology based, distributed control in process industries: a case study with LonWorks Technology. *Integrated Mfg Systems*, 1997, **8**, 231–243.

31 **Lee, S. M.** A component-based distributed control system for manufacturing automation. PhD dissertation, MSI Research Institute, Wolfson School of Mechanical and Manufacturing Engineering, Loughborough University, Loughborough, 2004.

32 **Lee, S. M., Harrison, R.,** and **West, A. A.** A component-based distributed control system for assembly automation. Presented at the 2nd IEEE International Conference on *Industrial Informatics* (*INDIN'04*), Berlin, Germany, 2004.

33 **Ong, M. H.** Evaluating the impact of adopting the component-based system within the automotive domain. PhD dissertation, MSI Research Institute, Wolfson School of Mechanical and Manufacturing Engineering, Loughborough University, Loughborough, 2004.

34 **Douglass, B. P.** ROPES: rapid object-oriented process for embedded systems. I-Logix White Papers, 1999.

35 **Vera, D. A.** A component based approach to the design and implementation of a virtual prototyping environment (for manufacturing systems). PhD dissertation, MSI Research Institute, Wolfson School of Mechanical Manufacturing Engineering, Loughborough University, Loughborough, 2004.

36 **Vera, D. A., West, A. A., Harrison, R.,** and **Thomas, D. W.** Virtual visualisation and prototyping environment for component-based production machinery. Presented at the 31st North American Manufacturing Research Conference, Hamilton, Ontario, Canada, 2003.