The divergence of theoretical and actual designs in a holonic packing cell

Fletcher, M , McFarlane, D. , Lucas, A. , Brusey, J. and Thorne, A.

THE DIVERGENCE OF THEORETICAL AND ACTUAL DESIGNS IN A HOLONIC PACKING CELL

Martyn Fletcher*, Duncan McFarlane†, Andrew Lucas*, James Brusey†, Alan Thorne†

Agent Oriented Software Limited, Mill Lane, Cambridge CB2 1RX, UK.*

Institute for Manufacturing, University of Cambridge, Mill Lane, Cambridge CB2 1RX, UK†.

Abstract: The aim of this paper is to take the reader through the decisions made in designing a holonic system architecture to support customised packing of gift boxes with personal grooming (Gillette) products. The paper highlights issues that cause a divergence between a theoretical model of a holonic system and the approach that was actually encoded.

Keywords: Architectures, Agents, Intelligent control, Packages, Knowledge tool

1. INTRODUCTION

Holonic systems are a new breed of IMS geared towards mass customisation and high degrees of agility (Bussmann, *et al*, 2001). Holonic systems can be seen to exhibited features like:

- Runtime 'plug and play' operation of equipment.
- Independence between hardware/data resources from specification of how products are made.
- Dynamic revision of processes and recipes.

There are several design frameworks for building such holonic systems, e.g. the Holonic Component-Based Architecture or PROSA (Van Brussel, *et al* 1998). An example test bed, being commissioned in Cambridge University's Institute for Manufacturing, is a pick and place cell for packing unique combinations of Gillette grooming items into boxes to satisfy customer orders. A key objective is to develop a holonic control solution for this cell. An interesting research topic is to examine reasons why a theoretical HMS design model may diverge from the actual code that is constructed during the project.

The paper highlights some issues within this research topic, including the physical infrastructure and entities in the cell, a theoretical model for constructing a holonic system with these entities, and the actual design that was encoded. The paper also narrates some lessons learned about what holonic features can be easily developed/shown, and what functional gaps are introduced due to the divergence. The paper concludes with guidelines to build future HMSs that minimise these deficiencies.

2. PHYSICAL PACKING SYSTEM

A physical 'pick and place' packing system has been constructed to demonstrate holonic and agile manufacturing behaviours. In this packing cell,

holons are introduced to model both *shop-floor resources* (e.g. conveyor shuttles, a robot, storage units) and *products* (gift boxes and Gillette items). Every holon uses its own intelligence and interaction with others to determine how best to use facilities and pack orders effectively. This system (see Figure 1) enables the customer to select any three of four grooming products, pack them into one of two box styles, and change how the order should be packed, all on the fly. When contrasted against conventional packing systems, this solution has a strong commercial future (Sathern, 2002) because it is intended to be more flexible and robust to changes in order requirements.

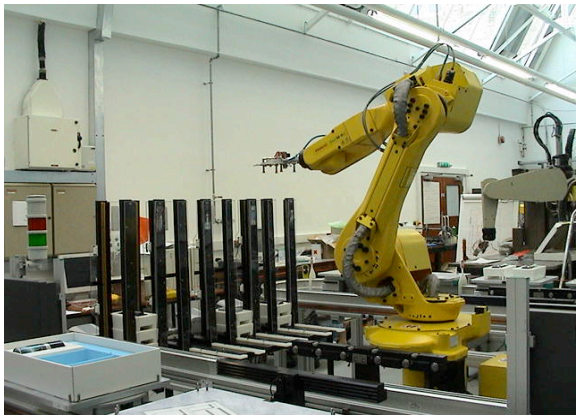

Fig. 1. Photograph of the Packing Cell.

The packing cell comprises a storage unit to hold items, a Fanuc M6I robot to move items throughout the cell, a shuttle conveyor to transport batches of raw materials (items), and the boxes into which items are to be placed around the cell. The Montech track provides added agility by being able to control individual shuttles and dynamically assign transport tasks to them. The system layout is given in Figure 2. The layout of the cell has three conveyor loops and independently controlled gates that allow shuttles to navigate around the track. There are also two docking stations (nominally called loading and unloading) where shuttles are held so the robot can pick and place items into carried boxes. Both stations can perform loading and unloading interchangeably.

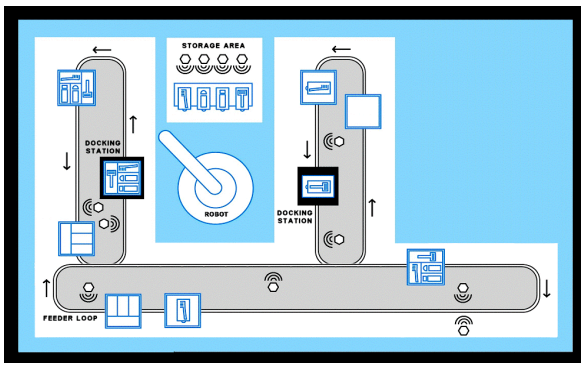

Fig. 2. Layout of the Packing Cell.

As a result of these features, the cell is able to demonstrate many scenarios associated with agile manufacturing (Arai, *et al*, 2001), such as the introduction of rush orders. The theoretical design is responsible for ensuring these agile behaviours are achieved within the scope of the aforementioned physical hardware.

3. THE THEORETICAL DESIGN

The theoretical or high-level design of a system specifies the objectives of the manufacturing system without defining how those objectives are met. The framework for this specification is illustrated in Figure 3.

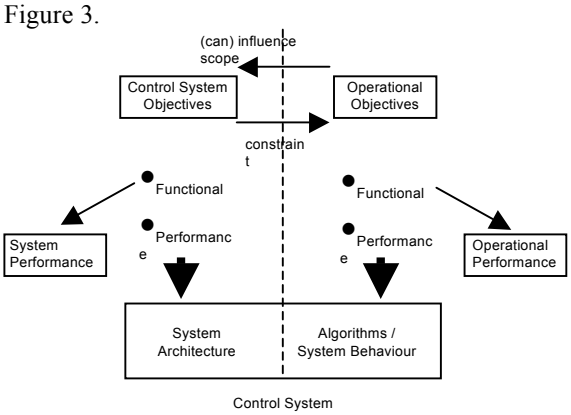

Fig. 3. Objectives Framework.

As the diagram shows, the overall objectives for a system can be split into *control system* and *operational* objectives. Control system objectives relate to the functionality of individual holons, while the operational objectives relate to the behaviour of the whole system. There is an interplay between the two, as control system objectives may constrain what can be achieved operationally. Conversely, an expansion of operational objectives may introduce additional requirements for the control system.

Both sets of objectives can be further divided into *function* and *performance* objectives. For example, as part of the control system objectives, there may be a functional requirement for a holon to manipulate a product in a particular way. In addition, there tend to be performance objectives that are specified in terms of specific measures, such as how many products can be manipulated per minute.

In a typical manufacturing system, control system objectives include such things as the ability to reliably deliver raw materials into the system (materials delivery), to manage parts during production (product management), to allow movement of materials and parts around the system (material handling), to store or buffer materials or parts (material storage), and to extract out finished products (material removal).

In contrast, operational objectives define how the system appears from the outside. This begins with a specification of what the inputs are and what is produced but also includes aspects to do with the

system's *agility*, such as how flexible it is in terms of handling different types of raw material types, whether delivery and removal facilities can be interchanged, the ability to reconfigure to meet different demands, or the ability to handle growth. The control system and operational objectives lead to a specification of the system architecture and overall system behaviour, respectively.

In the following section, we demonstrate how this framework applies to the Cambridge packing cell.

3.1. Control System Objectives

The first issue in the design of the control system is the delivery of materials. Materials enter the system by being placed on shuttles and transported around the system. This has an associated shuttle loading holon. Similarly, finished products leave the system by being removed from shuttles.

Product management is assigned to a product holon. This holon tracks each product through the system and ensures that it is manufactured correctly. Material handling is provided by robot holons, loading / unloading station holons, and gate holons. These holons manage the physical resources to manipulate and move raw materials and products. Material storage is provided by stack holons that manage the storage stacks. Since raw materials can also be stored on shuttles, shuttles correspond to "virtual stack holons" that can be called upon when stack levels become low.

3.2. Operational Objectives

From the operational perspective, the main objective is to fulfil orders. Although there need not be a holon to manage each order, the system must be able to associate an order with a set of product types and quantities. Furthermore, each product type has an associated *recipe*. This describes the parts that constitute the product and how they are assembled.

When an order arrives, product holons are generated and they seek and allocate their required material resources, such as empty boxes and parts to be packed into the boxes. They also need to allocate appropriate hardware resources by negotiating with the associated holons. The product holons then apply their recipe to create the product. A quality control phase ensures that the recipe has been completed and that the product has been put together correctly.

To allow the system to be flexible and responsive, product holons have an associated priority so that it is possible to give preferential passage to parts associated with high priority products. In the extreme case, partly-created low priority products can be disassembled to allow higher priority orders to be filled. To increase the agility of the system, an additional design objective is to allow late changes to orders. This might be simply to change the product type or to manipulate some other aspect of the recipe, or it may be to increase or decrease the quantity.

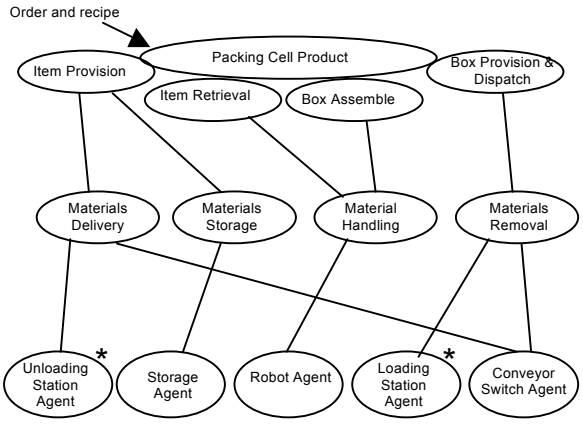

Fig. 4. Theoretical Design.

Martyn, as I'm in the plane, I don't have the original sketched holon architecture in front of me, is fig.4 this? It's not clear in the figure which are holons and which are agents, the terms seem interchangeable, looking at the figure.
Figure 4 summarises the theoretical design of the packing system. Note that unloading and loading holons provide similar functionality and could potentially be merged to provide a single holon to manage both operations. This seems a sensible simplification since the same resource is typically used for both operations.

4. THE ACTUAL DESIGN

The actual (implemented) design is a realisation of the above theoretical design, constricted by:

- The facilities offered by a specific holon development and execution environment.
- The peculiarities of the physical hardware.
- The lack of robust operations from its low-level support software that the holons rely upon. For instance, the inability to guarantee electronic product code (EPC) readings using RF tags with automatic identification (Auto ID) systems.
- The inefficiency of the blackboard system (Chirn *et al*, 2000) that interfaces to the PLC and robot.

The design that was ultimately arrived at, as a consequence of these liabilities, is described here in terms of holon architecture and interactions.
A diagram here would help, to highlight the departure from the planned (i.e. Martyn/Melbourne) architecture, to the Martyn/Cambridge interim, and finally to the Martyn/James final, implemented system. Also, there is no description of why the teamed architecture was set aside, to be replaced by a ma one.

As this requires some explanation, you may not wish to put it all into the public paper, but into the annexe.

4.1. *Holon Architecture*

The holon types that have been implemented in the control system (with the number of instances) are: Order Holon (an instance is spawned for each box ordered), Production Manager Holon (1), Gate Holon (2), Reader Holon (7), Docking Station Holon (2), Robot Holon (1), Storage Holon (1), Box Manager Holon (1), and Track Holon (1).

Each of these holon types has been encoded as an agent using the JACK Intelligent Agents™ environment from Agent Oriented Software Limited. We focus now on the features of the robot holon because it is one of the more complex resource holons, and its processing typifies the interactions an autonomous holon has within itself and with the external world. It can be observed that the robot holon supports the material handling functional objective via:

- Scheduling jobs based on reward.
- Performing various pick and place operations in order to pack boxes, unpacking boxes, sorting the storage area and unloading items into storage.
- Interacting with the physical robot.

Each of these is modelled as a JACK *capability* (containing appropriate events, plans and belief structures) within the robot. It is intended that an additional capability for managing faults will be incorporated soon. Using JACK's design tool, we can represent these agent (with man icon), capability (square box), plan (round box), event (like an envelope), belief relations (cylinder). See Figure 5.

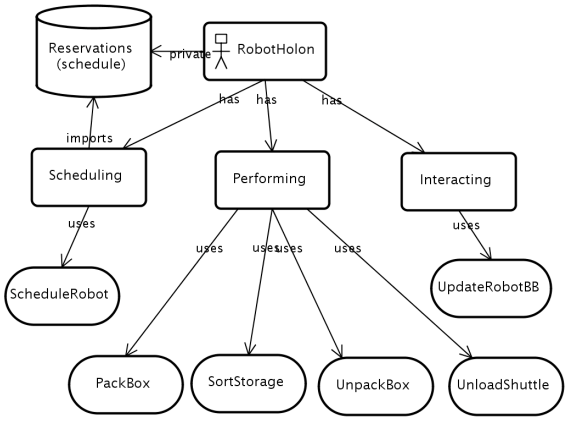

Fig. 5. JACK Design of Robot Holon.

Consider an internal operation of the robot – the decision as to what pick and place action to perform. The ScheduleRobot plan posts a RobotJobArrived event to itself, indicating that the robot should now execute a job (i.e. the one with the highest reward). Four plans in the Performing capability can handle this event. The choice as to which one will handle the event initially is determined at runtime using the context of these plans. Here the context uses the number of items and boxes on a shuttle. For example, the context of the UnpackBox plan is

(rja.numBoxes == 1 && rja.numItems == 3);

While for the UnloadShuttle plan, it is

(rja.numBoxes == 0 && rja.numItems == 2);

While for the SortStorage plan, it is

(rja.numBoxes == 0 && rja.numItems == 0);

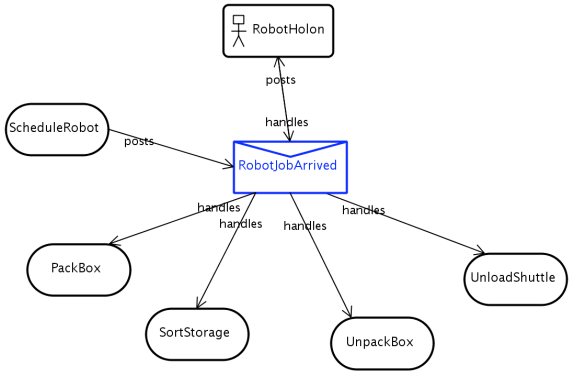

Fig. 6. Robot Holon's Performance Interactions.

Note that rja is the identifier of an instance of the RobotJobArrived event that these plans handle. Figure 6 illustrates this event/plan relationship. The remaining events handled and sent by the RobotHolon have been removed from the figure for clarity. The interaction that an agent-based holon has with other holons is orchestrated through the exchange of (a)synchronous messages. Again the JACK design tool can be used to plot these interactions. Figure 7 shows an example of such interaction between the robot and order holons.

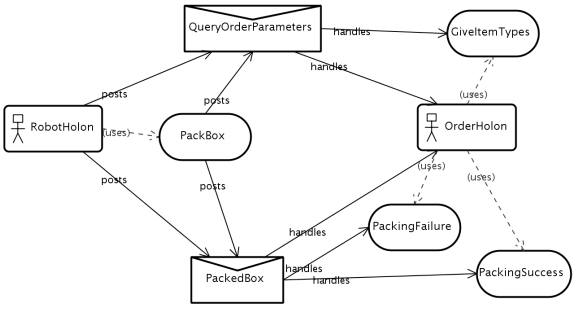

Fig. 7. Robot-to-Order Holon Interactions.

Explanation of the architectures and interfaces for the other holon types has been omitted for economy.

4.2. *Holon Interactions*

The various holonic agents interact with each other through the exchange of messages (which can be viewed as offered services), coupled with a cost framework. Therefore, the mechanism encapsulates the intricate modelling of the physical system, how hardware is controlled as well as the holon's private decision-making processes inside each agent. To coordinate the actions and decisions of agents, events are sent either synchronously or asynchronously. No agent is forced to process or reply to a message because each agent is autonomous but, generally speaking, good co-operation is encouraged by the agents responding to information requests with truthful data, and by executing appropriate plans to achieve the goals associated with incoming events.

Negotiation between the order holon and resource holons, as well as between resource holons (for fault tolerance), is managed solely in terms of price. The customer sets the price per box, when the batch order is placed via a e-manufacturing web page, and this price is used throughout the manufacturing process for sequencing the spawning of order agents, 'buying' empty boxes/items, using a shuttle, reserving a slot in the docking station's schedule, utilising the robot's time etc.

Hence demand and supply is matched in the course of the negotiation through a simple protocol: order holons in need of a particular service distribute requests to pre-defined instances of the resource agent class. The resource holons in turn evaluate the request in terms of their schedule and their status, and issue replies (bids) back to the order holon with the cost of the job. These costs are used by the order holon to select resources that are the cheapest.

In classic agent design, this would be viewed as awarding a contract. For example, having already secured the services of a shuttle holding the correct type of empty box, consider the negotiation between an order holon and the docking station holons to agree where that order's empty box should be packed. The protocol proceeds as follows. The order holon announces to the loading docking station holon that it wants a job done and that the reward for undertaking this job is the price given by the customer. The docking station considers the job, by determining whether it can physically handle the shuttle and box. If the station is faulty, it returns a bid of infinite value. If not, it determines the slot into which the job would be placed. For instance, the existing schedule is shown in Table 1.

Table 1. Initial Reservations in Docking Station

Slot	Shuttle EPC	Reward
1	B00000000C000200A00D1704	$20
2	B00000000C000200A00D2645	$12
3	B00000000C000200A00D3199	$7

If a job worth $15 is requested, then the station could make an offer of Slot 2. Upon receiving this response, the order holon issues the same request to the unloading docking station holon, which will give a similar 'best slot' reply. The order holon then evaluates both bids and selects the one with the earliest slot. The order holon informs the chosen station that it needs to book the named slot for its shuttle at the given price. The docking station inserts the tuple into its private reservation belief set and so demotes other jobs whose reward was lower.

The order holons associated with these demotions are *not* informed of the sequence change; only the order holon of the newly inserted tuple is informed to confirm the deal. The order holon then informs the track holon to set the destination of the available shuttle to the selected docking station. Beyond responsiveness and the ability to cope with failed stations, this protocol has the merit that workload becomes evenly distributed across stations over time. This is an example of interaction between order and resource holons. Now consider an example *resource to resource* interaction. When a shuttle approaches a switch gate, it first passes a RF tag reader. The EPC of the tag attached to the shuttle is gained and processed by the Auto ID (Kambil, *et al*, 2002) software systems called the Savant and PML Server. A reader holon is regularly polling the server for the EPC of the last shuttle to enter the reader's range.

Upon arrival, the reader holon informs the appropriate gate holon of the shuttle's presence. Each gate has two inputs and two outputs, and so the gate can decide which of two waiting shuttles to let through. This decision is currently made on a FIFO basis. The gate holon interacts with the track holon to determine the destination of the inbound shuttle and to determine if there is available space into which the shuttle can move into. The track holon maintains a model of which shuttle is in what zone using a set of queues. The queue is added to when a shuttle enters a zone and popped when it departs. The track holon also keeps a model of the maximum number of shuttles that can be present in each zone (this is fixed at start-up and is tightly coupled with the track's configuration).

Hence the track holon is able to decide if another shuttle can enter a given zone based on the shuttle's intended destination. If space is available then the gate holon is informed and so it can interact with the blackboard to set the hardware's input/output choice and thus let the shuttle through.

5. DISCUSSION

This discussion describes the key holonic features the implemented cell exhibits, gaps in agile behaviour that the implementation did not achieve, and some pointers towards how the design divergence could be rectified in future systems. Some of the agile and intelligent manufacturing scenarios that the packing cell does demonstrate are:

- Introducing batch orders so individual products (i.e. boxes) manage their own resource allocation. This allocation includes acquiring a suitable tray, shuttle and items to accomplish the goal of packing that box. Also included is negotiating with the docking stations to reserve a processing slot.
- Handling of rush orders that must be packed earliest. Their introduction influences both the docking stations' schedules and when other orders are to be packed.
- Unpacking completed boxes so that urgent orders can be satisfied quickly.
- Interacting shuttles and gates to ensure routing of shuttles around the track is both shortest time and gives highest priority to urgent shuttles.
- Docking stations cooperating to evenly distribute their workload at runtime.
- Failing a docking station to demonstrate fault tolerance and re-assignment of jobs.
- Organising a storage unit when the robot is not busy so frequently used items are located at the head of each chute.

These scenarios are sufficient to illustrate some of the merits associated with agile manufacturing as identified in section 3. However the system that was built has limited holonic features. It does not cope with the dynamic introduction of new hardware in a 'plug and produce' manner. Neither does it have the potential to manufacture a range of products – it is limited both by the physical system and by the control system to the packing of two styles of gift box. Furthermore, at present there is no order to order interaction because boxes or items are not exchanged between order holons.

In other words, orders are packed on a FIFO basis, except where there are rush orders that can be packed using available boxes and items without having to compete for scarce resources – these orders are packed before lower value orders. Other key deficiencies in the actual design are:
This list is a hybrid of implementation (reliance on polling), functional performance (slowness), and qualitative (lack of reasoning). I suggest jumping up a level of abstraction – group them in terms shuch as performance, flexibility/capability, ease of implementation. This will help readers to understand where the major limitations lie, i.e., slow, but flexible, or flexible and hard to implement/debug.

- *Reliance on polling*. Holons must continuously poll other computer systems due to there being insufficient mechanisms for pushing data into the JACK agents. (this point isn't clear, is this a limitation in JACK, or elsewhere?) Seven reader holon each poll the PML server every 0.5 seconds to determine if a shuttle is present. The Production Manager holon also polls the server every second to discover new orders, while each resource holon polls the blackboard (up to ten times a second) in handshaking to spot changes in PLC registers.
- *Slowness*. The robot holon takes a significant time to decide its next pick, making the system look clumsy. This delay is, in part, created by the preparation of Simple Object Access Protocol messages by the PML server when giving data.
- *Poor interfaces*. The gate, station and robot holons are forced to use rigid instructions to get the hardware to work. Also there are no reports back from the blackboard of faults. So methods to identify and manage failures are very limited.
- *Lack of reasoning*. No intelligence is coded in the agents to deliberate about unlikely tag reads or strange box/item/shuttle aggregations.
- *Poor debugging*. The mechanics to test and debug agent code are very limited. Agents are distributed complex units of software whose behaviour changes based on their situational awareness. Hence better tools are needed, and these are currently under development by AOS .

We are striving to overcome these limitations within the scope of the physical hardware in order to satisfy the entire set of functional objectives attached to the theoretical design. But we also have to keep an eye on the performance objectives and so we are adopting a spiral model for future developments.

The phases of the spiral are: select next most important functional objective or limiting factor to work on, set appropriate performance objective measurements and targets, design scenario to evaluate system, run experiments, and finally analyse results. In this way we intend to increase the power and performance of the actual design until it grows to a sufficient scale to meet the desired attributes of the theoretical design. Note if we adopt a software engineering perspective and there exists a mismatch between what we want to build and what we were able to build then we have OW! Choose another way to express this. because the design should reflect all the requirements. Another perspective is that we are dealing with a migration problem – we want a pure holonic solution (the theoretical design), but there are constraints. Therefore what we built was a pragmatic holonic solution (actual design), and to get to where we really want to be we need to do more research.

6. CONCLUSION

The paper presented a theoretical design for a holonic packing cell. The main activities of the cell's control system, where agility is possible, include:

- Locating and providing Gillette items in the cell.
- Retrieving items from storage and putting them into the cell.
- Providing box into cell and dispatch out of cell.
- Assembling gift box order using retrieved items.

The paper outlined an implementation to illustrate how a deviation between theoretical design and coding can introduce gaps in holonic behaviour. The divergence is often due to the lack of solid guidelines

or a practical methodology in order to apply the chosen design framework. The development of such guidelines is the focus of our future research.

I suggest a three-column table, with the list of holonic features aimed for at the start of the Packing Cell development in the left column. The centre column would indicate (via a √ or x), as to what has been implemented. The RH column would contain the level of difficulty in implementing the remaining unimplemented features, e.g E(easy), S (substantial work/re-design required), and I (impossible without re-design/new hardware, etc.).

REFERENCES

Arai, T., Aiyama, Y., Sugi, M., Ota, J. (2001). Holonic Assembly System with Plug and Produce, Computers in Industry, vol 46 no 3.

Bussmann, S., Sieverding, J., (2001). Holonic Control of an Engine Assembly Plant – An Industrial Evaluation, proc. of IEEE conf. on Systems, Man and Cybernetics, Tucson, USA.

Chirn, J.L, McFarlane, D.C., (2000). A Holonic Component-Based Approach to Reconfigurable Manufacturing Control Architecture, proc. of 1st int. conf. on Industrial Applications of Holonic and Multi-Agent Systems, Greenwich, UK.

Kambil, A, Brooks, J., (2002). Auto ID across the Value Chain: From Dramatic Potential to Greater Efficiency, www.autoidcenter.org

Sathern,. E. (2002). Envisioning Agile Packaging – Unilver envisions new packaging machinery that treats change as business as usual by focussing on a 'holonic' approach, www.packworld.com/ articles/Features/15421.html

Van Brussel, H., *et al* (1998). Reference Architecture for Holonic Manufacturing Systems: PROSA, Computers in Industry, vol 31 no3.