

COMMERCIAL IN CONFIDENCE

JACK-based Holonic Control of a Gift Box Packing Cell

Internal Report of the Centre for Distributed Automation and
Control

Martyn Fletcher, Andrew Lucas
Agent Oriented Software Limited
Mill Lane
Cambridge CB2 1RX, United Kingdom

Dennis Jarvis, Jacquie Jarvis
Agent Oriented Software Pty. Ltd.
Adelaide Applications Centre,
15 Kiama Avenue, West Lakes Shore,
South Australia 5020, Australia

Ralph Rönquist
Agent Oriented Software Pty. Ltd.
156 Pelham Street, Carlton,
Victoria 3053, Australia

Duncan McFarlane, James Brusey, Alan Thorne
Institute for Manufacturing
University of Cambridge
Mill Lane
Cambridge CB2 1RX, United Kingdom

May 2003

Executive Summary

The Gift Box Packing Cell has been constructed at Cambridge University's Institute for Manufacturing in order to demonstrate JACK-based holonic control and illustrate how a packing environment can be responsive to change. The cell, with its industrial strength hardware, enables a customer to select three Gillette™ personal grooming products (razor, shaving gel, foam and deodorant) and also how they are to be packed into one of two box types. The development work on this cell is being done by the IfM's Centre for Distributed Automation and Control with contributions from the Auto-ID Centre and Agent Oriented Software Ltd (AOS). Holonic control is a methodology based on the application of autonomous cooperative building blocks that can be put together into a loose organization to manage a factory. These systems are particularly well suited to responsive manufacturing settings where orders, production processes and resources regularly change. In this report, we will show how the holons used to control activities within this cell were designed and how these holons were encoded using the JACK Intelligent Agents™ platform.

This report describes the features and implementation of this holonic packing cell to support a plethora of behaviours needed in responsive factories, including unloading of goods coming into the factory, storage of work-in-progress (WIP), sorting and retrieving of stored goods, mass-customised packing batches of gift boxes based on customer choice and priority, performing quality control of completed gift boxes, unpacking unneeded boxes, handling failure of redundant resources, and dealing with rush orders. The report highlights the conceptual framework for achieving such responsiveness by using holons.

This framework can be seen as a measure of operational performance in the face of disruptions and is mainly influenced by the way holons act, interact and affect the physical manufacturing environment. The main principle for building these behaviours is founded on the independence of orders from resources, coupled with the 'plug and play' approach for incorporating new hardware and the potential for using sophisticated resource allocation and fault tolerance strategies. The JACK Intelligent Agents™ platform was successfully used to build the cell's holonic control system, and this implementation is also discussed here. Throughout this report we attempt to answer the following questions:

- How can agent-based holons be best integrated with manufacturing hardware to provide responsiveness in terms of: (i) offering an increased range of manufacturing operations, (ii) being coupled together coherently at run time, and (iii) compensating for unpredicted failures?
- What types of algorithms are needed by the holons for: (i) scheduling and planning; (ii) routing of goods through and beyond the factory; and (iii) resource allocation, etc?
- How can holons be dynamically introduced, cooperated with and removed?
- How are quantitative measurements to be made on how the holons perform, both at both theoretical and implementation levels? We have commenced work in this topic, albeit from a qualitative focus, via a spidergram-based framework [10].
- What are the functional interfaces between holons and legacy systems within the factory?

Developing holonic control system demonstrators, like the packing cell, is valuable exercise because it provides us with a wealth of opportunities to test out new holonic models and ideas within the scope of a mini-factory environment where industrialists and academics can witness the benefits of responsive manufacturing. It is envisaged that this report will act a springboard for our future work in exploring methods, tools and philosophies associated with the design decisions that must be taken when building factories in the future founded on the holonic vision.

Executive Summary	2
1. Introduction.....	4
1.1. Problem Description	4
1.2. Description of the Physical System	4
1.3. Goals of the Control System.....	6
1.4. Performance Objectives for the Development.....	7
2. Approach – Methodology	8
2.1. Summary of the Key Steps taken in the Design	8
2.2. Identification of Key Designs Made	8
2.2.1. The Cambridge Design	8
2.2.2. Adelaide Design.....	11
2.2.3. Implemented Design	12
3. Approach – Conceptual Overview.....	14
3.1. Identification of Holons	14
3.2. Functionality of Holons	16
3.3. Strategy for Interaction between Holons	17
3.4. Solution Approach to the Control Goals.....	17
4. Approach – Detailed Design.....	19
4.1. Holon Architecture.....	19
4.1.1. Description of Holons	19
4.1.2. Inter-holon Architecture.....	20
4.1.2.1. Interaction between Order Holon and Robot Holon	20
4.1.2.2. Interaction between Order Holon and Docking Station Holon.....	21
4.1.2.3. Interaction between Gate Holon and Reader Holon	21
4.2. Holon Interaction Mechanisms	21
4.2.1. Introducing Batch Orders.....	22
4.2.2. Shuttle Navigation	23
4.2.3. Packing a Box	24
5. Evaluation	26
5.1. Evaluation of Performance of Controlled Operations	27
5.2. Evaluation of Software/Control System	28
5.3. Evaluation of Methodology Used	30
6. Conclusion and Ways Forward.....	31
6.1. Summary and Critique	31
6.2. Lessons Learnt and Future Work.....	32
References.....	34

1. Introduction

Many modern manufacturing systems are highly automated and are now requiring decentralised ‘smart’ architectures to control their hardware and manage the flow of materials / knowledge, in order to be more *responsive* to change. This responsiveness is needed to satisfy an ever increasing consumer need for goods that satisfy their unique requirements which can be delivered to market both quickly and economically. A key route to achieve mass-customisation (a major factor in responsiveness) with distributed control is to apply the *holonic* paradigm [14], and one manufacturing process that exhibits a high potential for responsiveness is packaging. Therefore this report presents some of the main features of such a system – the Holonic Packing Cell demonstrator that has been built at Cambridge University’s Institute for Manufacturing. It must be emphasised that this cell is constructed from state-of-the-art industrial strength facilities to demonstrate a spectrum of responsive manufacturing ideas – it is not built from Lego bricks.

1.1. Problem Description

Our goal is to design, develop and integrate a gift box packing cell to demonstrate how a holonic system offers more responsiveness. The system is characterised by:

- **Business Problems:** There are three problems to be investigated using the packing cell. First, how to pack together consumable goods (in this case Gillette™ personal grooming products like razors, deodorants, shaving gels and shaving foams) in a flexible way to meet specific and variable customer/retail needs. Second, how to efficiently and economically use intelligent storage, transport and packing resources [32] where uncertainty, failures and changes are commonplace [1]. Third, how to get products to decide how best to make, manage and deliver themselves so that they are cost effective and satisfy the buyers’ requirements.
- **Solution:** Holons are introduced onto the shop-floor to model both *resources* (e.g. conveyor shuttles, a robot, storage units) and *products* (orders for gift boxes). Every holon uses its own intelligence and interaction with others to determine how best to use facilities and get the orders packed effectively. The system enables the customer to select any three of four types of grooming product, and pack them into one of two box styles. This solution is now more responsive to changes in the requirements of customers’ orders than conventional packing lines could offer.
- **Benefits:** The demand from customers for unique combinations of products to be packaged together has led to an examination of new control system technologies that will be responsive to retail requirements. Holons better utilise factory resources, make products more efficiently and so increase the responsiveness of manufacturing businesses to changing market needs. Rush orders can buy packing services to get themselves made quicker, partially-packed orders can be broken down so that urgent orders can be made, and resources can create schedules to ensure they are used optimally.

1.2. Description of the Physical System

A physical ‘pick and place’ packing system has been constructed to demonstrate how holons can be used to support responsiveness in manufacturing. The packing cell contains a storage unit to hold the Gillette items using a first-come-first-served discipline. The cell also has a Fanuc M6i anthropomorphic robot to perform material handling jobs (e.g. pack items from storage into boxes, unload items from a shuttle into storage, sort the storage zone and unpack items from boxes into storage), a shuttle conveyor to transport items and boxes around the cell. The layout of the cell has three Montrack™ 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. Photographs of the physical system are given in Figures 1, 2 and 3, with a schematic layout of the cell presented in Figure 4.

Sensors and actuators within the cell (apart from the Fanuc robot) are connected to an Omron SYSMAC CV500-V1 Programmable Controller, which in turn is attached to a Personal Computer via Ethernet, in order to support a blackboard. This blackboard maintains a copy of the data registers held by the controller relating to the status of sensors and actuator parameters that the control system can read from and write to

in order to get the physical hardware to perform desired actions. The aim of the packing cell, as a whole, is to pack items into boxes in a flexible way to meet specific retail needs that vary over time.



Figure 1. Photograph of the Packing Cell.



Figure 2. The Cambridge Holonic Packing Cell



Figure 3. A demonstration to business people of how the holonic cell packs Gillette items into gift boxes after an order has been issued over the Internet.

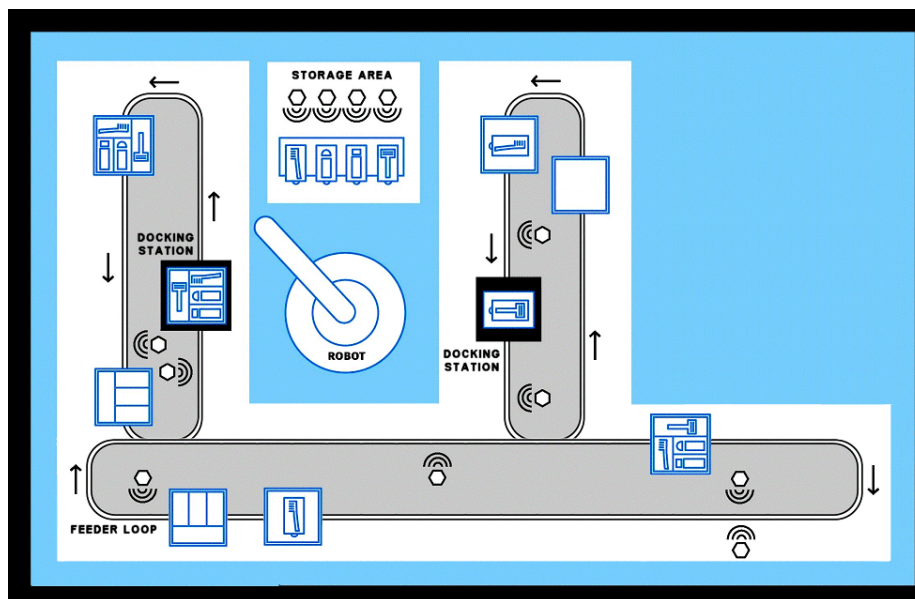


Figure 4. Layout of the packing cell.

1.3. Goals of the Control System

There are three goals for the control system. The first goal is to have a distributed strategy for making maximum use of the degrees of freedom associated with the aforementioned physical hardware. In this light we intend to populate the control system with holons [25]. Holonic systems are a new breed of Intelligent Manufacturing System (IMS) [18] geared towards high degrees of responsiveness [35], and can be seen to exhibit features like:

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

As defined by Marik *et al.* [23], “Holons are autonomous cooperative units which can be considered as elementary building blocks of manufacturing systems with decentralised control. A holon usually contains:

- a hardware/equipment part able to physically influence the real world (e.g. a device, tool or other manufacturing unit, transportation unit or storage facility etc), and
- a control part responsible for both the hardware/equipment part control and communications with the rest of the holonic community.”

Interested readers are referred to the work of Chirn [7] and the papers edited by van Leeuwen [38] for an overview of the classic holonic principles and the current state of the art in applying these principles to a range of manufacturing control environments. The control element of the holon can be implemented using an *intelligent software agent*. Agent systems [28, 29, 30, 39] represent the next major leap in functionality for computer systems. Agents are autonomous software entities that are aware of their situation and can reason about their behaviour to accomplish design objectives. They experience their environment (the larger system they ‘live’ in) through sensors and act through effectors. They are reactive, responding in a timely manner to environmental change that they can sense, and are also proactive, working towards their goals. Agents can dynamically form themselves into teams, as humans do, and play different roles and collaborate in order to achieve their objectives. Intelligent agents have been successfully deployed into a number of real-world domains where decentralization, complexity and uncertainty are common.

The second goal is to have the holons act and interact to demonstrate some responsive manufacturing scenarios, such as:

- *Batch Orders*: Introduce orders for batches of gift boxes so individual order holons manage their packing (e.g. acquire a suitable empty box, shuttle and items). The order holons also negotiate with resource holons (including docking stations and the robot) to achieve processing goals.
- *Unpacking*: Unpack completed or unnecessary boxes so urgent orders can be satisfied quickly.
- *Reconfigure Docking Station Processing*: Disable a docking station (i.e. mimic a failure) and so cause the holons to reconfigure themselves to process work elsewhere. The holons can also handle rush orders that must be packed quickly through changing the schedule of operations on a docking station.
- *Storage Handling*: Handle storage of item in a reactive manner by bringing new shipments into the cell, hold them in storage chutes and handle them correctly.

The third goal for the control system is to demonstrate that holonic control can be integrated with the Auto ID infrastructure into open-loop control so the user has the ability to dynamically change how orders are made and managed [2]. Auto ID will also aid the holons in ensuring that their operations are robust even when raw materials supply is random and varying. Auto ID systems (www.autoidcenter.org) are used to read/write information to tags via high frequency radio waves. These electronic tags can replace barcodes and uniquely identify goods, thereby enabling decisions to be made by the order holons representing them on an item-specific level and removing the line-of-sight requirement when reading.

These goals highlight the potential for the control system to be readily responsive. We define responsiveness in this context as ‘the capability of the system, both in part and as a whole, to conduct business as usual in an environment where change is frequent and unpredictable’.

1.4. Performance Objectives for the Development

The key performance objective of the development is that the shop-floor needs to migrate away from a rigid centralised control philosophy to an open approach where hardware and knowledge resources can be easily and quickly added, removed or reconfigured. The predetermination of what products are to be made in a given manufacturing cell must also be relaxed – orders should arrive, identify what type of resources are needed and how to compose the product, and then control the machinery that is scattered across the factory as needed to make the goods while satisfying all the business criteria imposed on the organisation. This creates a range of complex decision-making problems that cannot be easily solved by conventional approaches due to the high degree of dynamics.

2. Approach – Methodology

2.1. Summary of the Key Steps taken in the Design

The key steps in the design include:

- Creating a theoretical design (known here also as the Cambridge Design).
- Refining the theoretical design into a design that can be implemented using the selected holon development tool and modifying the design to satisfy the required functional and performance objectives. Here, this design is called the Adelaide Design.
- Refining the design into an Implementation Design that can be constructed within existing constraints.
- Testing and debugging the design on the physical hardware in the IfM's automation laboratory
- Running a demonstration of the system's functionality to illustrate responsive behaviour by the holons.

2.2. Identification of Key Designs Made

The key designs made during the lifetime of the project are:

- The Cambridge Design by McFalane [26].
- The Adelaide Design by Jarvis, Jarvis, Rönnquist and Fletcher.
- The Implemented Design by Brusey and Fletcher.

2.2.1. The Cambridge Design

The Cambridge Design is based on the view that the theoretical, or high-level, design of a system specifies the manufacturing objectives without defining how those objectives are met. The framework for this specification is illustrated in Figure 5.

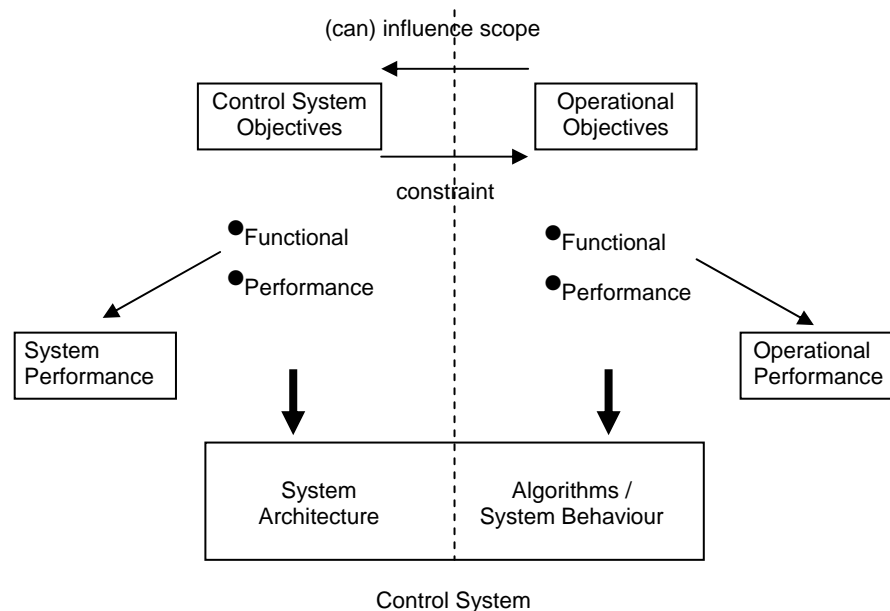


Figure 5. Objectives Framework.

As the diagram shows, the overall objectives for the packing cell 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 as they appear from the outside. The operational objectives begin with a specification of what the inputs are and what is produced but also

includes aspects to do with the system's global *responsiveness*, such as how flexible it is in terms of handling different types of raw material, whether delivery and removal facilities can be interchanged, ability to reconfigure operations to meet different demands, or the ability to handle growth. There is interplay between the two sets of objectives, as control system objectives may constrain what can be achieved operationally. Conversely, an expansion of the operational objectives may introduce additional requirements upon the holons in the control system. 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). Both sets can be divided into *functional* and *performance* objectives. For example, as part of the control system objectives, there may be a functional requirement for a holon to manipulate a gift box in a particular way. In addition, there tend to be performance objectives that are specified in terms of specific measures, such as how many boxes can be manipulated (packed) per minute. Control system and operational objectives lead to specifying the system architecture and overall system behaviour, respectively. Objectives for holons in the Cambridge Design's control system are:

- The delivery of materials will be through being placed on shuttles and are transported around the system. This has an associated shuttle loading holon. Similarly, finished products leave the system by being removed from shuttles. Material handling should be provided by robot holons, loading / unloading station holons, and gate holons. These holons will manage the physical resources to manipulate and move raw materials and products.
- Product management is assigned to a product holon. This holon will track each product through the system to ensure that it is manufactured correctly.
- Material storage will be provided by stack holons that manage the storage of Gillette items within stacks. Since raw materials can also be stored on shuttles, the shuttles correspond to "virtual stack holons" that can be called upon when stack levels become low.

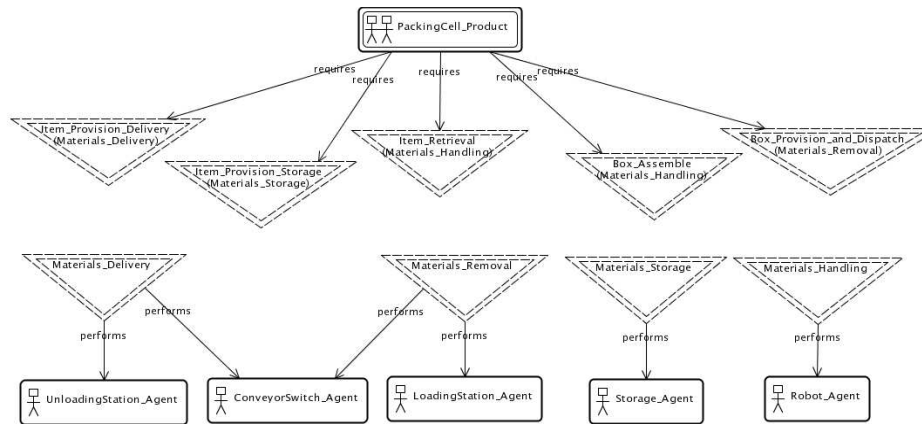


Figure 6: Holons in the Control System of the Cambridge Design.

From the operational perspective, the main functional 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 should be applied to ensure that the recipe has been completed and that the product has been put together correctly. To allow the system to be 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 boxes can be disassembled to allow higher priority orders to be filled. Figure 6 summarises the holons within the control system of the packing system's theoretical design. 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. Additional functional objectives of the cell's operational system are:

- *Remote Ordering.* The user must have the ability to make and change orders for boxes over the Internet. The cell must have the ability to pack batches of boxes that fulfil these orders from multiple customers, with each customer maybe having several box configurations in their order. When an order arrives, order holons are generated and they seek and allocate their required material resources, such as empty boxes and items to be packed into the boxes. They also need to allocate appropriate hardware resources by negotiating with the associated holons. It would be nice to have facilities for the customer to guarantee when an order must be completed by (i.e. an order schedule and fixed deadline) and the maximum cost an order will incur. Also the cell should guarantee the behaviour of its resources to meet the order.
- *Order Intervention.* To increase responsiveness, the system must allow customers to make late changes to their orders at runtime. 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 or alter priority, start/end times and cost values. If the order is removed then it would be nice to have the facility to unpack and repack boxes to satisfy other orders' requirements. In the extreme case, partly-created low priority products (boxes) must be disassembled to allow higher priority orders to be filled.
- *Fault tolerance.* The system must have the ability to allocate resources (e.g. shuttles and boxes) to orders at runtime. The system must be able to have both docking stations hold shuttles and take on the roles of loading items into boxes or unloading items and placing into storage. Facilities must also be available for the shuttles to take different routes around the track so they can get to the correct docking station even if there is some blockage. Facilities are needed to prioritise routes by the gates so that high-priority shuttles are passed through the gates first. Mechanisms are also needed to manage resources' load profiles, raw materials and work-in-progress. It would be nice to have facilities to make special offers to customers when there is a surplus of materials or resources' capacity. Having the shuttles move in both directions around the track would increase route choice.
- *Recipes.* The system must have the ability to describe the elements that constitute the product that has been ordered and how it is to be assembled. An order's packing recipe is at a higher level of abstraction away from the instructions of specific machines in the cell. A quality control phase can then ensure that the recipe has been completed and that the product has been put together correctly. The recipe for making a product should be represented using PML. It would be nice to have simulations for multiple cells (each attached to an agent) that bid for the work of packing an order, and when the physical cell is awarded the job then it packs the box. Also having one generic recipe mapped onto one concrete recipe per cell would mean that a product could be made at any packing establishment. Keeping the current state of a recipe's progress, as the box is packed, in a PML file would improve the interoperability of the system.

The performance objectives of the cell's operational system are:

- *Real-time Operations.* The key focus of optimising the demonstration's speed is in terms of the robot. Once a shuttle arrives at a docking station for packing, we do not want the robot to 'stop and think' about the next move it must perform. In the current demonstration, the delay between arrival and packing the first item is approximately seven seconds, and between subsequent item packs is about 6 seconds. We want this delay to be significantly reduced.
- *Robustness.* The system should demonstrate good tolerance to faults in the cell's hardware. This is best illustrated with the dual docking stations which can either share workload evenly or take on all the work if the other fails. The single robot made itself a potential single point of failure. Also once a shuttle fails and blocks the track, no other shuttle can pass it and there is no way to transfer boxes between shuttles. Hence we aim to be robust to docking station failure but not to other faults in the cell. Having identified these issues, we note that the robotic hardware was very reliable.
- *Throughput.* The cell must maximise its resource utilisation, and minimise idle time, to ensure the highest volume of boxes are packed each hour. However in responsive manufacturing more decisions need to be made by holons than in mass production and so inevitably the cell's throughput will not be as high as it would have been if it was not responsive, e.g. packing 1 type of box with 3 fixed items.

There were several constraints restricting how the system's objectives could be realised including: team.

- *Time*. The deadline for completing an initial version of the system was November 2002 and the final version by February 2003 to coincide with remote demonstrations at the Auto-ID consortium's board meetings where orders could be placed from the USA and be manufactured in Cambridge.
- *Manpower*. There were limited personnel to undertake the project. In terms of advisory input, AOS staff (AL, RR, DJ and JJ) were only able to contribute limited resources to the project. With respect to technical work, MF was only able to spend 50% of effort throughout the project's duration (June 2002 to February 2003) and JB only arrived in October 2002 but was able to commit 100% effort.
- *Learning Curves*. At the project start, both MF and JB had limited experience of implementing solutions with JACK agents, though both have knowledge of holon manufacturing, agent principles and Java development. To get up to speed with using JACK in a simple way, there is a relatively gentle learning curve, yet to use all its features and apply these features using best practices takes a significant learning period. This 'ramp up' was estimated at 6 months and would have bitten into project time.
- *Development Cycle and new hardware*. The development was impacted due to the necessary installation/testing of both the new hardware to operate the main Montrack loop and the low-level control of the gate resources for the loop. This introduction and commissioning was expected to take six weeks from the project start date. There was also no software available to readily simulate the cell's operations and to develop the holonic control system without having the complete hardware system operational. A simulation could have been developed using either JACKSim or other tools, but this would have taken some time. Also there is no software available to assist in developing the interfaces to the hardware. It was expected that *virtual machines* to mimic the protocols used by the hardware would be available in September 2002. All this software was delivered on time.
- *JACK*. The JACK Intelligent Agents™ platform (at the start of the project) lacked two critical features that resulted in the time to develop the holonic control being significantly increased. Firstly support for building multi-agent systems using JACK Teams™ within the JACK Development Environment (JDE) was not sufficiently mature to be part of the main JACK release. The JACK constructs to build team organisations (namely teams, roles, named roles and teamplans) were still being refined and the documentation on how to use these constructs was not finished. Mature team-based functionality and documentation was expected in version 4.0 in October 2002. Secondly the techniques to debug agent applications was, and remains today, very poor. Mechanisms to halt the execution of an agent, inspect belief sets and variables, and step through the code line-by-line did not exist. Some facilities were expected to be delivered in version 4.0.
- *Existing Hardware*. The peculiarities of the physical hardware and the protocols that must be used to send instructions to the hardware impose a restriction on the functionality of integrating the cell's equipment with holons. It also meant that opportunities for making the holons behave in a robust and fail-safe manner were significantly reduced by not having enough data to make good decisions.
- *Blackboard*. The agents could only interact with the hardware through a legacy piece of software called the "blackboard". The blackboard (written in Visual Basic) communicated with an Omron™ PLC on the shop-floor to issue commands and receive sensor data to and from the hardware. The blackboard would also be used as a gateway to facilitate interaction between the agent and the robot. The blackboard was inflexible, a bottleneck and a key point of failure.
- *Auto-ID*. The Auto-ID's readers and Savant™ software system were unable to guarantee that when a tag passes a reader then its electronic product code reading will always be observed. Also there was no guarantee that a tag will be read by only one reader at a time, or that once a tag is read then the Savant will inform the holons within some time frame.

2.2.2. Adelaide Design

Here we provide an overview of the revised holonic design produced in Adelaide [12]. The operational objectives of the Cambridge Design remain intact. However the control system of the Cambridge Design was changed because the AOS staff with experience of implementing holonic agents indicated that it was unworkable, overly simplistic and could not be readily encoded using JACK. Revisions were also suggested in order to incorporate ideas from previous research into part-oriented control done jointly by AOS and Cambridge. Modifications were also adopted to better reflect the approaches developed during the international Holonic Manufacturing Systems (HMS) project [16], namely to explicitly depict holons' functionality and seamless connectivity among the machine, cell, factory and enterprise levels.

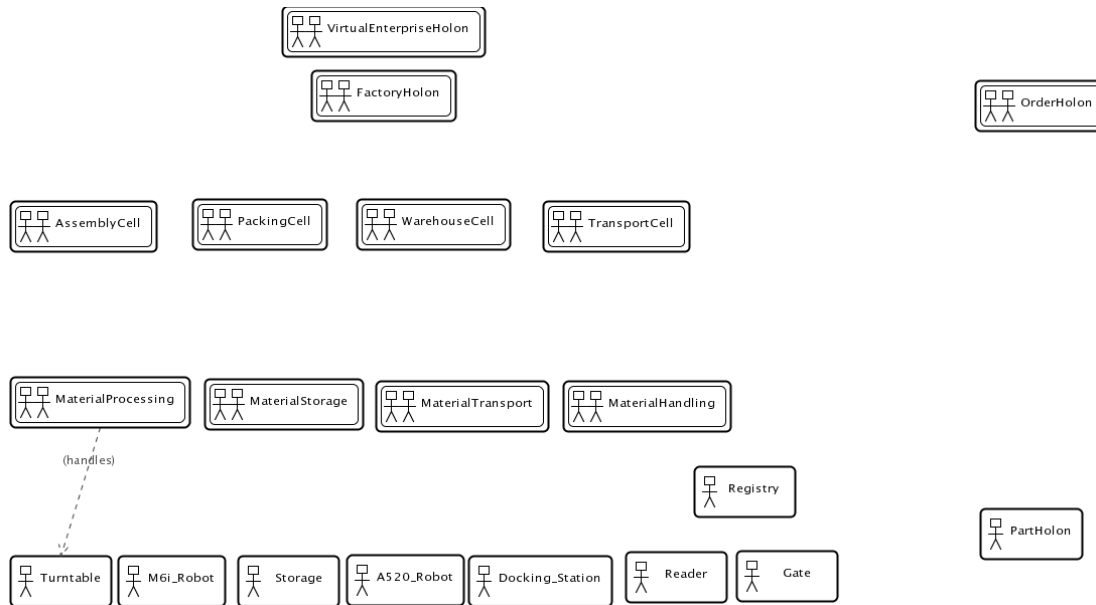


Figure 7: The Adelaide Design.

Within the Adelaide Design (see Figure 7) there is a separation of resource holons (left side) from the order management holons (right side). On the order side, there is a part holon for every physical part (items and boxes) to monitor and control it as it progresses through the cell, and an order holon to manage the manufacture of box orders. On the resource side, there are four distinct echelons:

- At the lowest level, there are holons (modelled as JACK agents) to represent the robots, gates, docking stations and so on, together with a registry. This registry is used by the other agents to register their services and request services from colleagues.
- The second layer is populated with holons (modelled as JACK teams) that model the four basic material management properties in any factory, namely transport, handling, processing and storage. Each team draws upon the functionality of the subordinate agents, e.g. the materials handling holon could use the roles offered by the Fanuc M6i and A520 robots as well as the docking stations to hold and manipulate items.
- The third level has holons to depict different types of manufacturing cells found in a factory, e.g. cells for packing goods, transporting goods or storage them in a warehouse. Again the teams use the roles offered by the material management teams to achieve their goals. For example our packing cell would make use of the roles offered by the material transport, storage and handling teams to bring items into the cell, deposit them into storage stacks and subsequently pack them into boxes when orders arrive.
- The uppermost level contains holons to represent the factory (pulling together the various cells) and the virtual enterprise that the factory contributes to as part of the business' supply chain.

The part holons interact directly with the lowest level resource agents while the order holons interact with the uppermost resource holons. When an order is placed, the factory decomposes how that order is made and assigns work to the cells, etc. down the hierarchy until the order is fully made. In Figure 7, only the teams have been identified to clarify the structure. This team-based design, although not completed with all the roles, plans, events etc. needed to realise the holonic vision, captures the key elements of the Cambridge design and also provides a reasonable migration strategy to the design that might be implemented using JACK [17]. This design was not implemented because of the external constraints listed above.

2.2.3. Implemented Design

The details of the holonic packing cell system design that was actually implemented by Brusey and Fletcher during late 2002 and early 2003 are described below. We are able to highlight the changes that have occurred from the initial Cambridge and Adelaide designs. These differences can be expressed in terms of the constraints highlighted above to help the readers understand where the major restrictions lie:

- *Time*. There was insufficient time to put in place all the software entities (teams, roles, events etc) associated with the Adelaide Design and to complete all of the functional objectives. Hence the authors adopted a dual approach. First it was essential to get some sort of demonstration working remotely by the deadline. So a simplified system architecture was adopted with autonomous agents rather than teams because the overhead in coding agents and agent-to-agent interactions was significantly less than using the teams approach. The overhead can be seen in terms of the number and complexity of JACK entities to code a simple request-response interaction: using agents, we need two agents, two plans and two events. In teams, we need two teams, two capabilities, two teamplans, one event and one role, all of which must be correctly integrated to work. Second, we decided to prioritise the functional objectives that are to be encoded by the deadline. As the conceptual model of recipes remains somewhat ill-defined, we concentrated on the remote ordering, order intervention and fault tolerance aspects, and selected only the portions of these objectives that would have maximum impact on the people observing the demonstration.
- *Manpower*. This limitation meant that the authors had to do all the coding and testing. This was a further argument for prioritising the functional objectives to be tackled.
- *Learning Curves*. Both authors decided to reduce the learning curve by using JACK as a platform for autonomous agents and to encode the agent plans using Java and object oriented programming styles, rather than ‘pure’ agent-oriented styles. This was because the time to learn the correct application of BDI programming or using teams would impinge too much on the time available.
- *Development Cycle and new hardware*. No major impact.
- *JACK*. The lack of mature teams functionality and documents meant that the learning curve was steepened. Limited debugging facilities meant that it would take a significant amount of time to create a problem again using the hardware and a certain configuration of orders, and run a particular ‘buggy’ plan and then analyse a log of data printed to a file. Due to the authors’ inexperience in coding with JACK and the fact that the Auto-ID software used to support the holons in their decision-making was being developed by another person who needed to do their own testing and integration work, there was a relatively high percentage of bugs to fix. This meant that fewer functional objectives, than expected, were coded and tested to a state where we were sure that they worked.
- *Existing Hardware*. Because there is little opportunity to change the holon-hardware interface to make it more robust, the control is not very reliable. Hence we have not met the performance objective to the extent we would have liked. This constraint did not have a major impact on the functional objectives that we were able to develop within the project.
- *Blackboard*. The lack of facilities in the blackboard to support a subscription type of model and the race conditions that it can create had some impact on the ‘real-time operations’ performance objective, making the system somewhat grubby. Holons, as clients to the blackboard, must regularly send a UDP/IP message to the blackboard requesting the value of a specific data register that mirrors the status of some hardware. The blackboard would then respond with this value. This polling is performed approximately every 0.1 seconds, adding to network load and can slow response times.
- *Auto-ID*. The inefficiencies and inadequacies of the Auto-ID system’s readers, Savant software, PML server and virtual warehouse were worked on by other people in the Cambridge Auto-ID Centre. This work has significantly aided the authors in our building of the holonic control system.

We now describe the implemented holonic control system.

3. Approach – Conceptual Overview

Figure 8 illustrates an overview of the conceptual approach that was actually implemented using holons. The idea of a *holon* was initially suggested by Koestler [21] to describe how many natural and manmade organisations displayed characteristics where every element of the organisation is simultaneously an independent system (i.e. it has autonomy) as well as being part of some larger society (i.e. it must cooperate with other holons to solve its problems). The term holon is composed of the Greek word *holos* to mean whole, with an ending *on*, as in proton, to indicate that the entity is a particle.

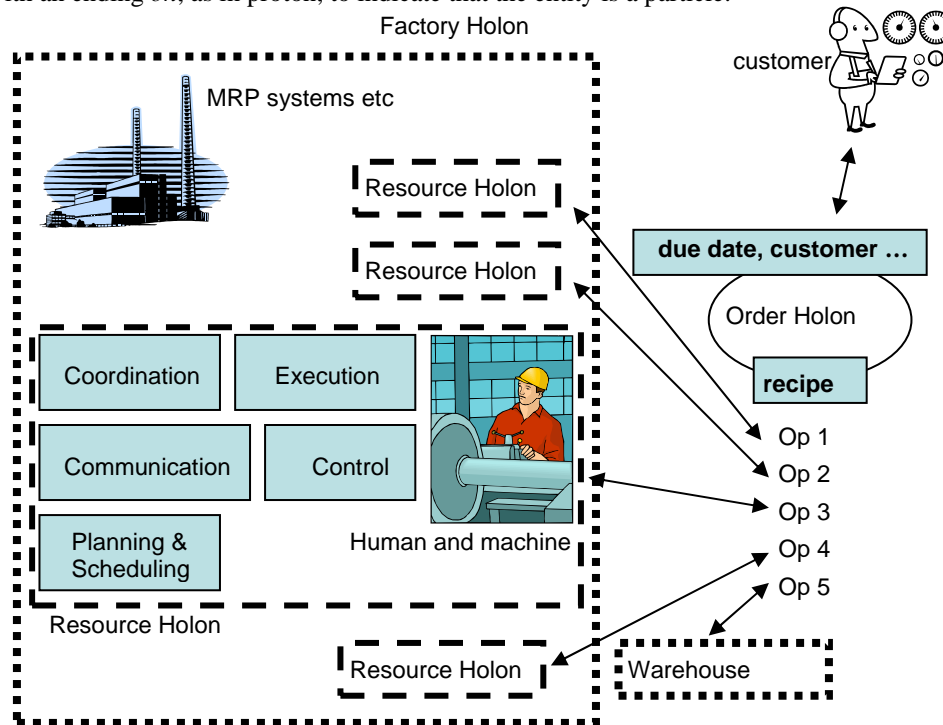


Figure 8: Conceptual Overview of the Implemented Holonic Design.

The conceptual approach is based on populating the control system with holons that can act and interact to achieve the objectives of the operational system as a whole and the control system. How the approach satisfies these objectives, at a conceptual level, is explained in this section.

3.1. Identification of Holons

For our purposes, we identify a holon [5] as containing either physical hardware or some information services coupled with an intelligent agent [3, 31] that can provide both autonomous actions and support cooperative interactions. The agent elements of holons have been encoded using JACK Intelligent Agents™ platform. In the context of the conceptual holonic approach, there are two classes of holons:

- *Resource holons*, which are self-contained system components which can perform operations on work-in-progress, such as fabrication, transport and testing. Beside the visible physical part, a resource holon also contains an invisible control part to perform its operations, make decisions and communicate with the aid of its plan library and knowledge base. For the packing cell, there are the following *resource holon* classes, with the number of instances shown in brackets: Robot (1), Docking Station (2), Gate (2), Reader (7), Track (1), Box Manager (1), Storage (1), and Production Manager (1).
- *Order holons*, which also contain a physical part and a control part. The physical part may include raw materials, finished goods, pallets and fixtures. Meanwhile the control part contains functions to manage routing, process selection, decision-making and facilities to maintain production information. Within the cell, there are *order holons*, with one being spawned for each gift box, in order to orchestrate how that box must be manufactured via collaboration with the various resource holons.

Figure 9 illustrates this taxonomy of identified holons, showing how there are n order holons in the system and various classes of resource holons (with the number of instances depicted by the integer after the \$).

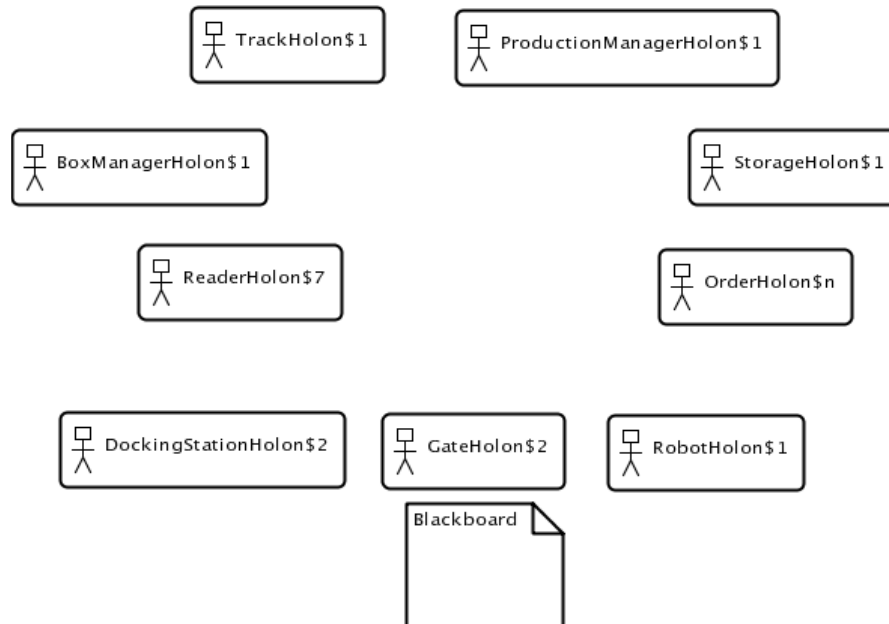


Figure 9. A taxonomy of holons in the control system.

We arrived at this taxonomy by answering the question that is frequently asked of holonic or multi-agent system designs, namely how can the holon or agent types be identified. The answer in this case is that we first identified the types of separable physical hardware in the packing cell that must be controlled. Each of these entities (robot, docking station and gate) has its own independent goals and hence can make a variety of decisions. Therefore these entities should be modelled as resource holons. Only these robot, gate and docking station holons interact directly with the hardware through the legacy blackboard. We can also identify that there may be other pieces of hardware that could be dynamically plugged into a physical system that must be controlled. This hardware might process sensed information or make decisions upon their actions in an autonomous manner. These should also be treated as resource holons, and here this style of holon includes the Auto ID readers.

A trickier question is whether the entities in the control system that do not have an immediate one to one mapping with hardware are also to be modelled as holons. The PROSA architecture [37] uses staff holons to represent entities that do not neatly fit into the heterarchical view of the control system. We are faced with a similar challenge with our packing cell: there are several knowledge serving or processing entities that are neither resource holons nor order holons, yet provide a valuable set of services to the other holons. For example the track holon has the objective of monitoring where each of the shuttles is on the track loops in order to prevent too many shuttles being allowed into a given zone on the Montrack. In the case of the cell, these entities are labelled as (knowledge serving) resource holons, for the want of a better term, and like the use of staff holons in PROSA point to a big problem with the resource/order holon architecture. Resolving this problem is a major topic of our future research.

The objective of the order holon is to ensure that the gift box gets itself made in a timely manner and so satisfies the criteria imposed by quality control. Order holons are spawned upon creation of a gift box purchase request, and have a “recipe” describing how that product is to be made, stored and delivered. The order holon negotiates with the resource holons (who offer manufacturing services) to achieve the goals within its recipe. We consider that all these autonomous cooperative holons interact to accomplish their goals via the structured exchange of messages.

3.2. Functionality of Holons

The application of this holonic approach to manufacturing is envisioned to develop and operate the next generation of shop-floor control systems where every decision-making element (holon) is a building block, with 'plug and play' capabilities. The functionality of the holons in the packing cell's control system is used to encompass intelligent machine control, planning and scheduling of resources, and integration with other factory information systems. The holons also could, in the future, be used to integrate with supply chain management systems and Enterprise Resource Planning systems to manage the entire spectrum of functionality in the enterprise. Of particular interest to this report, holons provide a number of functionalities to support responsive manufacturing:

- Holons can draw upon agent approaches like negotiation, auctions or market economies, and can dynamically construct multi-holon organisations called *holarchies*. Holarchies use these approaches: (i) to join holons together, (ii) to connect holons and humans, and (iii) to link holons with legacy hardware/software. These collaborations allow holons to synchronise their tasks, knowledge and resource use. Intelligent user interfaces can be constructed to support 'smart' dialogues with people of varying authority and knowledge of the system. Moreover coalition formation is a powerful technique to 'glue' together holons into virtual societies geared towards assisting the holonic factory effectively contribute to the management of concurrent supply chains, outsourcing and so on. Real-time signalling between a holon and physical sensors / actuators can be applied to best react to shop-floor changes.
- The *intelligence* of holons is exhibited by how they modify their behaviour to satisfy changing design and organisational objectives, change their decision making processes and operate differently when their environment alters. Holons may use artificial intelligence techniques, e.g. reinforcement learning, to acquire new behaviours and may utilise these behaviours in both reactive and deliberative ways. Such modification is essential due to the open nature of a manufacturing business that is striving to be responsive. For example, the business must be able to make new products and introduce novel production processes on the fly.

The aforementioned holon types have been constructed using JACK Intelligent Agents™ platform, and the details of this implementation are presented in the next section. JACK was selected as the holon development environment over platforms like JADE or Grasshopper for several reasons including it has a rich set of graphical tools for agent design, plan editing and tracing, and it provides a native, lightweight communications infrastructure for situations where high performance is required. Moreover JACK supports a Belief/Desire/Intention (BDI) execution model. Therefore this means that the developer is not left on their own to develop all the software needed to build and run rational agents. This is beneficial for holon development because the BDI model has a very potential for reusing plans from within a plan library and has the possibility of sharing knowledge efficiently among agents using belief synthesis. Furthermore the use of agent intentions provides a basis to attain stability in the face of change by efficiently using the agent's manufacturing and data processing resources. Interaction between many low-level holons can result in a complex system behaviour which is difficult to understand, predict and control.

The well-defined BDI model, and solid implementations of this model like JACK, helps tackle this complexity via using software engineering principles and flexible interaction strategies that can be easily comprehended, visualised and monitored at runtime. JACK agents execute their reactive and deliberative behaviours through pre-compiled plans. Which plan to execute is determined according to internal events that they post to themselves, external events (messages) that they receive from other agents, and knowledge that they have about themselves and the world. If a plan fails, other plans can be executed by the agent until the goal is achieved. The holons also have sufficient functionality to interact with the external world. The interfaces that these JACK agents have to the external world are implemented in Java as JACK *views*, which enable the agent-based holons to interact with:

- A blackboard (in turn connected to a PLC that handles communication with hardware) to exchange sensor data and commands with shop-floor machinery [13].
- The Auto ID system (called the Savant), via Simple Object Access Protocol messages to gather data on which shuttle has just passed a reader and so forth.
- A server, also via SOAP, which provides knowledge on orders being placed etc.

3.3. Strategy for Interaction between Holons

The strategy for encouraging interaction among holons is to make use of various ‘styles of interaction’ which may include:

- Order holons and resource holons collaborate to ensure boxes are packed speedily to meet the customers’ specifications. This involves the order holon having the goal of packing the box and then to achieve this goal, selecting one or more appropriate plans (mimicking a manufacturing recipe). These plans contain sequences of manufacturing operations that are then bound to by suitable resource holons that are capable of supplying these operations. The order holon is also responsible for negotiating with these resource holons to secure necessary services and track the product’s progress through the factory.
- Resource holons interact among themselves to compute their schedules based on maximising the payoff associated with executing their services, and they also co-operate to compensate for unexpected failures [34] by re-distributing workload. The resource holons also interact to acquire knowledge (both internal to the cell’s control system and with the external AutoID systems) that individual holons do not possess.
- Order holons talk to each other to coordinate the buying and selling of goods (namely items and boxes) in order to ensure that customer batches are packaged effectively and shipped in a timely fashion.

The various holonic agents interact with each other through the exchange of messages (which can be viewed as offered services). Therefore, the interaction mechanisms encapsulate 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.

Timeouts are sometimes used to ensure an agent is not deadlocked waiting for a message that might never arrive due to the agent sending the message being disabled, or there being a problem with the hardware or the Auto ID systems. 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 and so forth throughout the control system’s lifecycle.

3.4. Solution Approach to the Control Goals

The key goal of the packing cell’s control system is to be responsive in order to support mass-customisation. Clearly this desired responsive behaviour can be viewed from two perspectives. For the client, the holons provide the support to pack boxes to meet their requirements and do so promptly within a budget that they set. For the manufacturing business, the holons help to maximise the cell’s resource utilisation and hence make a profit. For the adoption of holons within the cell’s control system to be considered a success, we need to show that the added responsiveness that the system offers is achieved in a cost effective and practical manner.

There is little benefit in having a flexible factory if the algorithms that holons must run to make products are so slow that the throughput is merely a fraction of what it would be if a mass production model was in place. A key focus of our research is to quantify how well our holonic solution satisfies the goals of the control system that we outlined in the previous sections. Therefore we need some quantitative evaluation criteria, such as performance that can be measured to assess how well the system is operating. Yet many criteria cannot be easily measured on some scale and so qualitative evaluations may have to suffice. Both qualitative and quantitative measures are incorporated within our evaluation framework (see section 5) to help judge the relative merits of our holonic solution approach. At present in our holonic packing cell, the customer sets a single price per box for the entire batch. Yet in subsequent phases of the cell’s implementation, it is assumed that there is some measure of financial cost associated with both the time

taken for certain boxes to be shipped and the value of items in those boxes. This measure of cost is linked to the client's willingness to pay more for quick delivery of varying amounts of high-value or scarce goods.

The control system was distributed over a number of different computers. One computer was used for the Savant system, which filters Radio Frequency Identification (RFID) tag reads [15], another was used for the PML server and virtual warehouse, while a third was used for executing the JACK agents. In addition, a separate computer was used for the Blackboard System (BBS), and yet another used as a bridge to the Fanuc M6i robot. The intent of this level of distribution is an attempt to avoid any single component dominating and hogging resources. However a cost of this design is that it introduces communication overheads and may cause the system to be affected by other network traffic.

Certainly, the BBS was originally designed to be accessed only by software running on the same computer. The BBS operates by polling the PLC regularly and mirroring the state of a set of registers in the PLC. In addition, changes made to the blackboard's copy (by the agents) are sent back to the PLC. The bridge to the robot operates by polling some of the blackboard's registers. A command is then sent to the robot by setting some registers to say what is to be done, setting another register to indicate that a command is available, and then waiting for another register to show that the robot has become busy and finally idle again. This reflects that the robotic operation has successfully completed and so the Gillette item has been picked out of one location and placed into another.

The use of blackboard registers as a conduit to manage both the Fanuc robot and PLC has the advantage that it provides a consistent interface to the combined system. One disadvantage is that it requires a lot of communication and can cause significant delays. Although this approach was reasonably reliable, we would have preferred to have an agent communicate directly with the robot, and to send commands using agent messages over DCI. The register model may cause subtle bugs if the control system is slow and misses the fact that the robot has become busy. We did not encounter this problem with the robot as each operation takes some time, but we did have a problem with control of the gates.

The Montrack gates are controlled by the PLC. The gates are preceded by a pneumatic stop-dog that signals to the shuttle to stop. When a command is sent via the BBS to set the gate to a particular direction, the PLC sets the gate position and then releases the shuttle using the stop-dog. It monitors the shuttle's progress and will only let one shuttle through at a time. This tends to make the operation of the gate quite robust. However, as noted previously, the use of a blackboard and the over-reliance on polling can cause a race condition. This problem was avoided by ensuring that communication of an action to the gate was performed reasonably atomically. However, as with the communication to the robot, we feel that a better long term solution would be to associate an agent with the PLC [20]. Note that it is necessary to have an agent for the PLC as well as for the gate holon, as there may be a many-to-many relationship between holons and PLCs.

4. Approach – Detailed Design

4.1. Holon Architecture

4.1.1. Description of Holons

Details of the design that was implemented are displayed using JACK's design tool. Here we can represent a holon by an agent (with man icon), capability (square box), plan (round box), event (like an envelope), belief relations (cylinder). See Figure 10. 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.

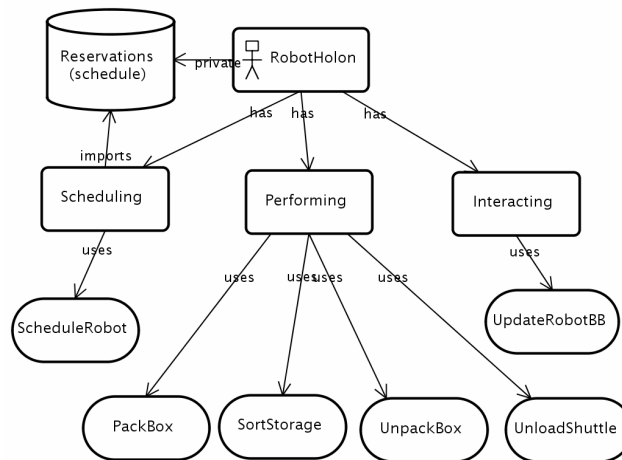


Figure 10. JACK Design of Robot Holon.

Consider an internal operation of the robot – the decision as to what item to pick up and where to place it. 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);
```

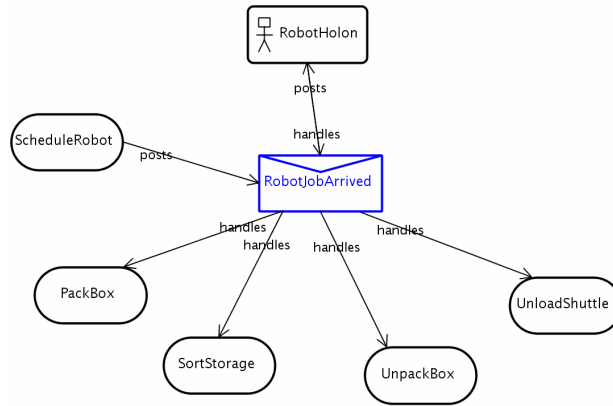


Figure 11. Robot Holon's Performance Interactions.

Note that rja is the identifier of an instance of the RobotJobArrived event that these plans handle. Figure 11 illustrates this event/plan relationship. The remaining events handled and sent by the RobotHolon have been removed from the figure for clarity.

4.1.2. Inter-holon Architecture

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. Here we cite three examples of the inter-holon architecture, namely order / robot, order / docking station and gate / reader. Explanation of the architectures and interfaces for the other holon types has been omitted from this report for economy.

4.1.2.1. Interaction between Order Holon and Robot Holon

Figure 12 shows an example of such interaction between the robot and order holons.

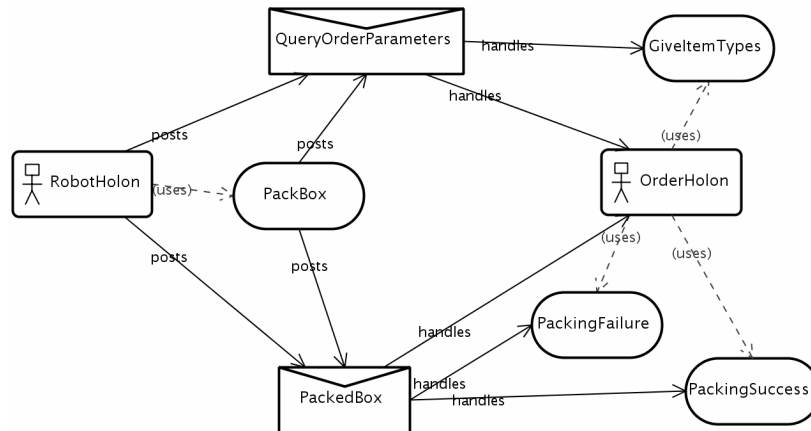


Figure 12. Robot-to-Order Holon Interactions.

The interaction architecture of the holons is geared towards having negotiations that match demand and supply through a simple protocol: order holons in need of a particular service distribute requests to pre-defined instances of the resource agent class that can provide that desired service. 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 information on when that job could be scheduled for execution. This information is used by the order holon to select resources that offer the earliest (or cheapest) execution. In classic agent design, this would be viewed as awarding a contract.

4.1.2.2. Interaction between Order Holon and Docking Station Holon

A second example of the inter-holon architecture is the interaction between an order holon and the docking station holons. 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.

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

Table 1. Initial Reservations in Docking Station.

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.

4.1.2.3. Interaction between Gate Holon and Reader Holon

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 software systems called the Savant and PML Server. A reader holon is regularly polling the server for the Electronic Product Code (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.

4.2. Holon Interaction Mechanisms

There are six responsive manufacturing scenarios that have been implemented that demand interaction between the holons. These are:

- Introduce batch orders, getting suitable empty boxes to satisfy the order, and allocate the jobs to docking stations.
- A shuttle (holding an empty/full box or items) navigating its way along the track to its destination.

- Packing various box types to meet specified configurations.
- Docking station failure and introduction of a rush order.
- Having insufficient raw materials to complete packing a box.
- Unpacking a box that is no longer needed by a customer.

We will discuss the holon interactions within the first three scenarios in detail because they highlight the types of cooperation made by the agent-based holons.

4.2.1. Introducing Batch Orders

Orders are entered via a web interface that allows the user to select the quantity, price, and configuration (conceptually, price is used as a way of managing the priority). In fact, a single order may contain several different configurations of boxes and items. Orders are managed by the PML server and stored as XML in a database. Periodically, the JACK system (ProductionManager holon) polls for new orders. When it receives a response indicating that a new order has been entered, it extracts from the PML server information about the individual order, such as the customer, price, and quantity. This process is shown in Figure 7 as a UML interaction diagram. The method used is referred to as `stripgetxqlpc`, as it involves performing an XQL query, keyed on EPC, and stripping out the required data from the XML. The XML Query Language (XQL) is used to query and change XML data with semantics similar to how the Structured Query Language (SQL) manipulates relational databases.

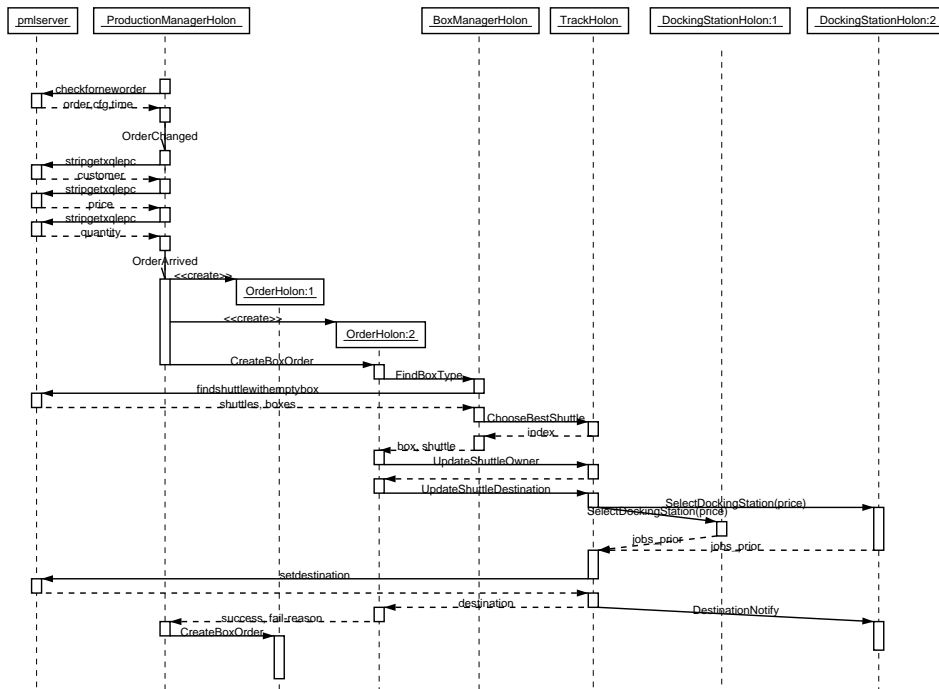


Figure 13: UML Interaction Diagram for Introducing Batch Orders.

For each box to be built, an OrderHolon is spawned. Each OrderHolon, in priority order, is then sent a message telling it to interact with the PML server to retrieve a recipe for manufacturing this box. The recipe determines the configuration of the box and Gillette items to be put into it. At this stage, the OrderHolon simply tries to find an empty box (of the correct type) on a shuttle, and does not attempt to find the items that would be needed to fill the box. This simplification was required because an early design decision was taken not to track individual items, so it was not possible to know if two orders were vying for a given item. In retrospect, tracking individual items and perhaps having an agent [24] associated with each one may be worthwhile.

Information about the location of each box is held by the PML server (virtual warehouse), whereas the TrackHolon has information about where the shuttles are. Therefore, the choice of shuttle is made by first

obtaining a full list of shuttles with boxes on them and their associated box identities, and then sending that to the TrackHolon. The holon assigns a heuristic, corresponding to how close a shuttle is to the docking stations, for each shuttle to make a choice. Once a box is chosen, it is allocated to the order. The next step is to select a docking station where the packing will occur. As there are two docking stations it makes sense to evenly distribute the workload between both. Several approaches were tried, however the approach decided upon is quite simple and just involves asking each how many jobs, of higher priority, would be processed before this box could be packed. The docking station that can pack the box soonest is selected. This algorithm is quite simple but not particularly optimal. For example, if a high priority order comes into the system, it may be allocated to a docking station that already has a lot of jobs.

When the docking station has been selected, the destination of the shuttle is updated. This update causes a notification to be sent to any agents that are interested in that destination, one of which will be the docking station itself. At this stage, the docking station updates its list of jobs. We found this type of Observer / Observable notification more reliable than attempting to manually ensure that the docking station's list of jobs is kept up-to-date when the destination of a shuttle is changed. This is particularly critical when handling a situation where a docking station is disabled and all shuttles queued on it must be rerouted.

The last step in the process of introducing new orders is to report success or failure. At the moment, if a failure occurs when creating an order, the ProductionManagerHolon does not attempt to retry the packing process when the situation changes, e.g. new items have arrived. Indeed, it is interesting to speculate what aspects of the environment the holon would need to monitor to know when to do this. Another possibility is to periodically retry.

4.2.2. Shuttle Navigation

The UML interaction diagram for shuttle navigation is shown in Figure 8. In the current system, a Reader holon polls the PML server every 0.5 seconds for shuttle RFID events. Obviously a more efficient system would be for the PML server to send a message when a shuttle event occurs but the infrastructure in the original system did not allow for this. Actually, this polling mechanism has been a source of a number of difficulties when trying to diagnose bugs. It could be argued that the extra time to put in place the infrastructure would have saved much time, however this was not apparent early on when the original system appeared to be working correctly.

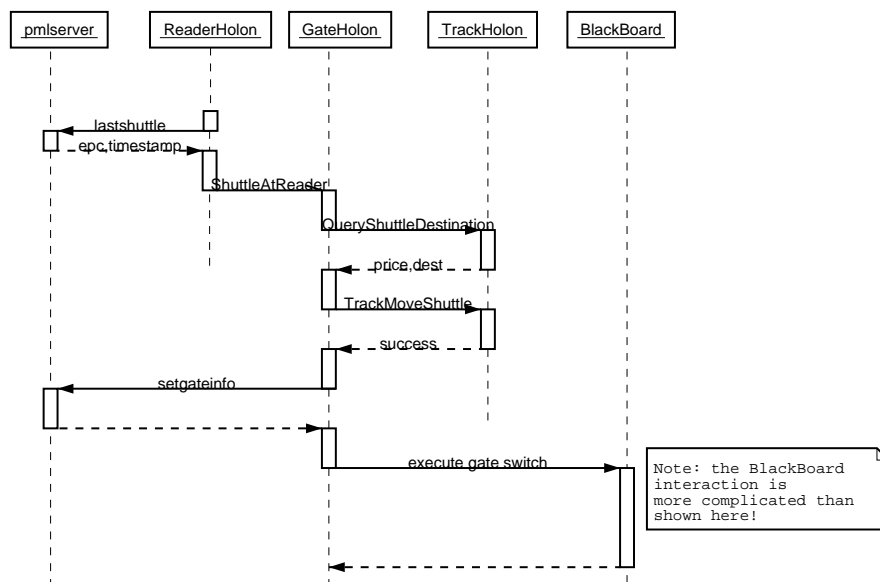


Figure 14: UML Interaction Diagram for Shuttle Navigation.

When a new shuttle arrival event is detected by the Reader holon, it sends a message to the Gate holon or Docking Station holon that the physical reader precedes on the track. In the case of a Gate holon, the first

step is to ask the Track holon where the shuttle is going to. A more agentified design might involve an individual agent for each shuttle having the goal of getting to the desired destination and using different route plans to achieve this goal. One advantage of the approach used here is that there is a limit to the space in each area of track and this is modelled by the TrackHolon. This is used to ensure that gates are not jammed up with shuttles that cannot pass through completely. The routing approach used by the GateHolon is quite simple. Each GateHolon maintains a list of zones that are accessible by the possible switch directions and then it tries to match the destination of the shuttle with a zone in its list that would move the shuttle closer towards the destination. The final stage is to send a switch command to the BBS to activate the movement of the physical gate.

4.2.3. Packing a Box

When an order has been placed, and received into the system, and once the shuttle has navigated to the correct docking station, the box can be packed. As with shuttle navigation, the start of this process is based on the recognition of a shuttle arrival event. The first step is to tell the PLC code controlling the docking station to wait for the shuttle. When it senses the arrival of a shuttle, it locks it in place using a pneumatic piston. This provides a reliable way of ensuring the position of the shuttle (to within 0.1 mm), and due to the nature of the shuttle tops, the position of all items and spaces on top of the shuttle. Once the shuttle has arrived and is locked in, the docking station holon notifies the robot holon.

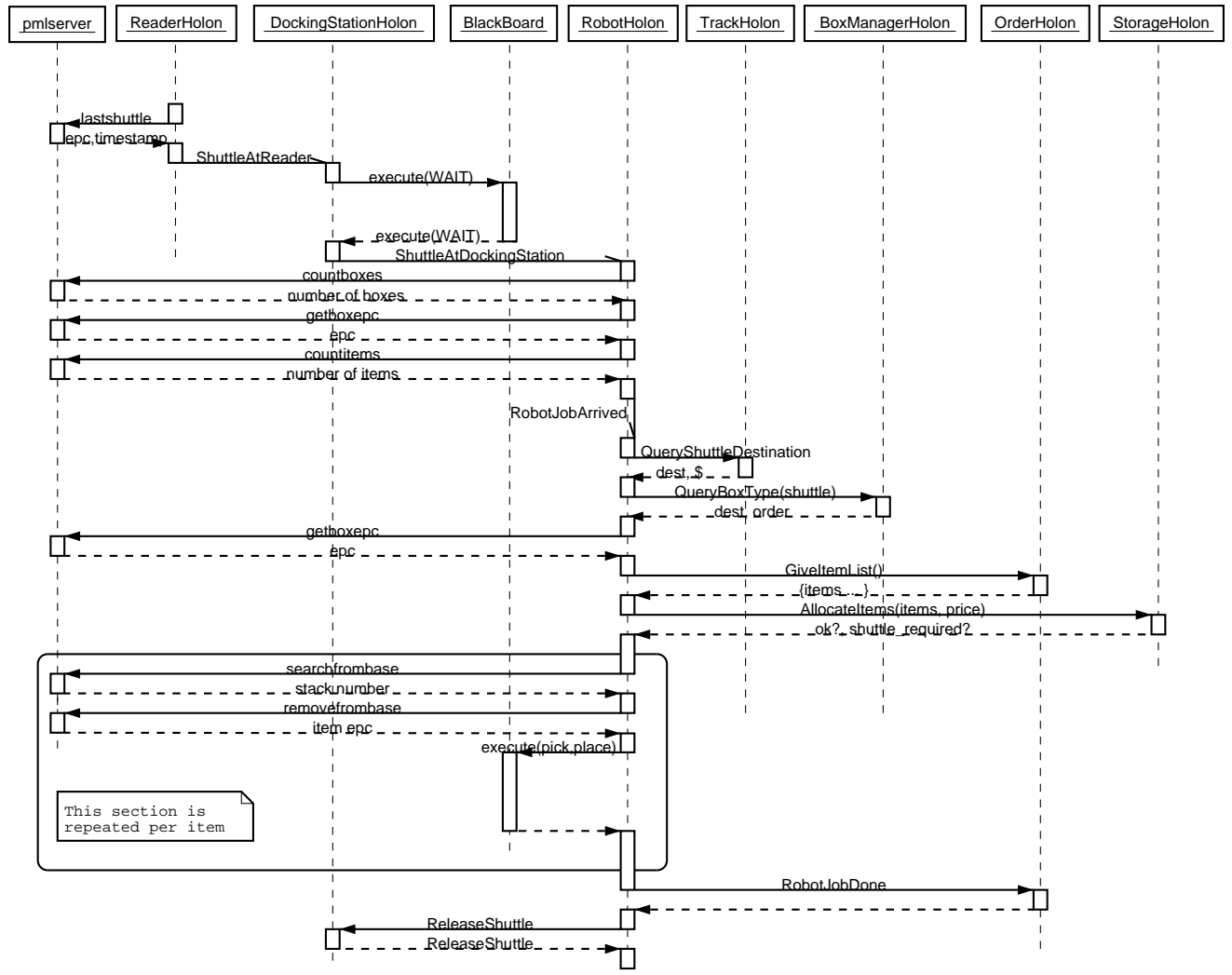


Figure 15: UML Interaction Diagram for Packing a Box.

The robot holon takes different actions depending on whether the shuttle has a box and whether the box is full or not. If there is a box, and the box is empty, it is assumed that this box is ready to be packed. The packing operation currently happens on an ‘all-or-nothing’ basis, i.e. it packs all three items into the box, or none of the items will be packed and the shuttle is released for processing later. One reason for this is that the RFID sensors cannot tell us where the items are in a box, so half-packed boxes are difficult to deal with. Of course, it might be possible to keep track of this information, for example if a packing operation was interrupted. Since packing is all or nothing, all items must be found within the storage stacks at the start.

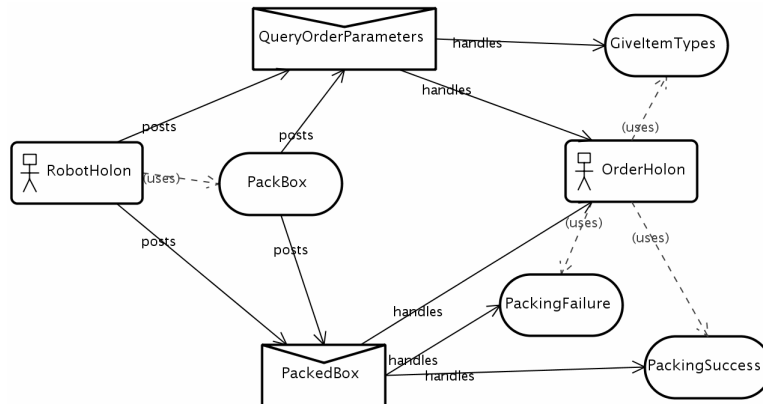


Figure 16: Interactions among Robot and Order Holons.

The first step is to ask the order holon for the list of item types to be packed into the box. This interaction and the confirmation of packing (see later) are illustrated using the JACK design diagram in Figure 9. This “bill of materials” is sent to the storage holon to see if it can be satisfied. The storage holon does not assign items as yet and this is possibly a limitation of the design. In addition, it was found to be simpler and more robust, but perhaps less efficient, to wait until starting the process of packing a box before detecting and responding to a material shortage. The storage holon flags whether or not the “bill of materials” is completely satisfied by the items available in the stack or on shuttle carriers circulating in the cell.

If items are available, but only on shuttle carriers, these shuttles are sent to an appropriate docking station. As soon as all items are available, the process of building the gift box begins. For each item type, the first step is to find a particular instance of that type in the stack. When the correct stack is found, items are removed from the bottom of that stack until an item of the right type is found. Note that the item type can be sensed as it is the first part of the EPC (electronic product code) and is registered by RFID readers that sit at the base of the stack. Also note that the item is logically removed from the base (by telling the PML server) as well as physically removed by sending a message to the robot via the BBS. This process continues until all three items are packed into the box.

The final step in the process is to send a completion message to the OrderHolon, giving the EPCs of the specific items packed into the box (this information is in turn given to the PML Server to track items’ movement). The robot also sends a message to the docking station to release the shuttle (and this involves sending a release message to the BBS). This packing box interaction protocol is shown in Figure 16.

5. Evaluation

Using a methodology to introduce holons into the packing cell's control system may have important responsiveness benefits in terms of reducing development costs, increasing flexibility and providing a higher echelon of robustness in both the resulting overall system and the control system. Yet there may be drawbacks in comparison to traditional approaches, for example the performance of the resulting system may not be as good as a system dedicated to low-variety high-volume production. Therefore a framework to evaluate and compare holonic systems is clearly needed. To evaluate how well a holonic system operates in a scientific and repeatable manner, this framework needs to be characterised by the following guidelines:

- Develop manufacturing systems that handle a dynamically changing environment without having to centralise all the control.
- Keep knowledge on what the product (e.g. a packed box) should 'look like' separate from the machine instructions used to achieve these features (i.e. the actions of the docking station, robot and so forth to pack items into the box). This improves the potential for allocating a packing job to several heterogeneous cells.

The evaluation framework's central proposition is the creation of a uniform model for assessing and judging the relative merits of holonic systems' design and operation. This goal is underpinned by two inter-related factors:

- The need to overcome the fragmented perspectives of experts who are working on various aspects of the holonic system research spectrum. Lacking a single vision of how to assess a holonic system (i.e. not having a level playing field) can distort experimental practice and may introduce duplications of effort, which could continue to hamper the quick deployment of holons into today's industry.
- The design and operational inefficiencies in many holonic concepts have given rise to a range of problems and can adversely impact the quality of the resultant manufacturing control system. These problems mean that it is often unclear how the merits of one concept in one system relate to other ideas or to the same principle as adopted in other systems.

To satisfy these goals a unified evaluation framework is required. Our proposed evaluation framework uses a *spidergram*. The spidergram to evaluate our holonic packing cell is shown in Figure 17.

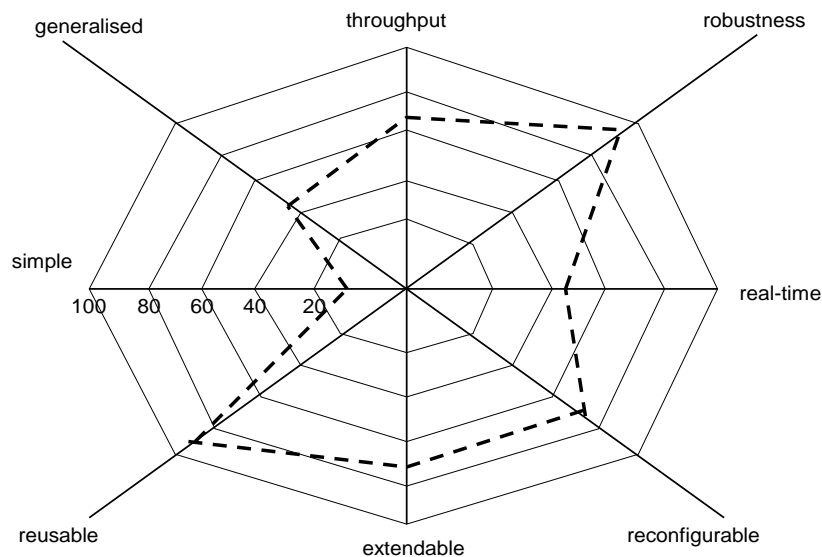


Figure 17. Spidergram to illustrate the evaluation of a holonic system from multiple viewpoints.

This graphical representation helps us to assess the design and operation of one or more holonic systems. The inspiration for this paper is Miles and Baldwin's article [27] where quantitative and qualitative criteria,

such as clear-up rates and crime prevention activities, of 50 UK Police Forces were mapped onto spidergrams to display and contrast their relative performance. A good overview of how to represent complex data using spidergrams is in [22]. Our holonic system evaluation spidergram shows percentages along each edge so that a perfect holonic system would score 100% based on some quantification. A spidergram has been selected as a suitable presentation format because it has facilities, based on relative performance of observable measures, to:

- (i) contrast different facets of a single system, and
- (ii) compare multiple implemented holonic systems.

The authors of this report reviewed several schemes to evaluate holonic systems including reviews of a person's employment or approaches similar to how a reviewer may assess the merits of an EU project proposal. These techniques have their merits including being to measure advances in a person's employment skills or estimate a project's outcomes respectively. Progress in these areas is gauged by experienced people carrying out specific tests, manipulating the data using weights and thresholds to form a single score, and judging this result against established benchmarks. They also have disadvantages, e.g. an assessment's stakeholder can misinterpret a single figure because they do not view all information in the proper context.

In terms of our packing cell, stakeholders such as production managers, academics and decision-makers in manufacturing businesses may fail to take into account factors such as how the system is very good in some respects but is poor in three other areas. They would also fail to take into account the reasons why the results were obtained such as the holons' diverse architectures and capacity for intelligent reasoning. Hence we argue that a balanced judgement cannot be made using such solitary scores. Therefore the authors selected spidergrams as a suitable scheme to evaluate a holonic system, noting that spidergrams could be used in conjunction with other graphical and statistical models depending on the nature of the evaluation to be undertaken. One of the merits of using a spidergram is that by computing the area occupied by a holonic system's spidergram, the diagram provides an overall system score. This number can then be used to rank various holonic designs or implemented systems, and carry out competitor analysis. For example, the spidergram in Figure 3 shows an evaluation of our holonic packing cell with good robustness and reusability but poor simplicity.

The clockwise sequence of metrics is not fixed and can be arranged to graphically illustrate the features of holonic system to reflect the evaluator's opinion. This lack of restriction can, of course, be viewed as a weakness or strength of the framework. For instance, arranging a spidergram's metrics to alternatively display high and low score gives the impression of a 'star' that may be visually less criticised than a cam-shaped spidergram when the metrics are re-sequenced. The number of metrics can be increased and the percentages altered to reflect a revised quantitative assessment strategy and so the evaluation framework is sufficiently flexible to cope with any holonic system and any set of benchmarks. The metrics to measure a holonic system can be categorised into three evaluation groups:

- (i) *performance of controlled operations* (throughput and robustness);
- (ii) *applicability of software / control system* (real-time, reconfigurable, extendable); and
- (iii) *methodology used* (reusable, simple, generalised).

The groups are discussed in subsequent sections of this document. There are no strong rules for placing metrics into particular groups, nor are there rigid guidelines for arranging groups in any given sequence clockwise around the spidergram. This lack of guidance and being deficient in stringent techniques to generate a quantitative value for how well a holonic system performs in some metric are key drawbacks to the evaluation framework. These are topics for further research.

5.1. Evaluation of Performance of Controlled Operations

This group measures the performance of the controlled operations in the packing cell with regard to the overall throughput of goods in some time period, and how robust the holons are to changes of how, where and when packing operations are performed. This group is geared towards satisfying the commercial goals of the manufacturing business. Yet a customer-oriented perspective is of equal benefit because we might want to evaluate other metrics depending on the customer, production environment and so on. For example,

the holons may need to reconfigure themselves in order to minimise the average lateness so that the highest degree of customer satisfaction is achieved. Here we concentrate on robustness because a system that is reliable is of key importance to factory managers, since a major factor impinging on manufacturing volume is the mean time between failures and the limited opportunities a control system has to recover. A primary goal of holonics is to alleviate this problem.

To best ensure efficiency in the case of failures, holons may need to perform some degree of contingency planning. This is a complex task. It includes taking into account task deadlines, the inter-connectivity between subtasks, availability of other resource holons that could adopt the role of executing that task, and other criteria imposed by the order holon (e.g. overall completion cost for the task, quality of service requirements etc.). One option is for the cell, via all of the holons interacting among themselves, to generate and execute a high-level control policy that determines the best course of action for every resource holon to achieve the cell's production goals in terms of optimising business criteria and resource allocation parameters. Yet the resource holons in our current cell do not generate or maintain such meta-level policies due to the communicational overheads. Hence we argue that individual holons should reliably execute their autonomous actions so global reliability emerges.

A key advantage of using JACK is that the agent-based holons are constructed using a solid Belief Desire Intention (BDI) execution engine. This helps model the holon's desires in terms of goals, binds these goals to plans at runtime, and can easily handle failure of agents' plans by re-issuing the goal and binding a different plan to it. Let us consider a theoretical example: the order holon's recipe to pack a given box specifies that the box should be processed on a docking station within a fixed deadline. If the nominated docking station fails then the station holon informs the order holon. The order holon then creates a set of alternative stations and times that satisfy this high-level goal. The best candidates are determined using some statistical evaluation of the recipe's criteria, e.g. candidate docking stations offering early time slots may be preferred. In terms of our evaluation framework, we argue that a holonic system is awarded points if it satisfies certain robustness criteria and loses points if it performs badly.

For instance, if a customer-oriented approach is used then the system deserves up to 30% higher than system solely using a business-oriented approach. If contingency planning and a solid BDI model are used during implementation then the system should be rewarded up to 20% and 25% respectively. If re-assignment of work is done using a dynamic set of candidates then this should be reflected with an assessment of up to 25%. This list of ways to convert qualitatively arguments into percentages is not exhaustive, and the percentages are open to revision.

5.2. Evaluation of Software/Control System

The applicability of a software / control system group examines how well holons handle real-time decision-making as well as their agent-based deliberation and collaboration techniques. Real-time decision-making is critical in dynamic environments where it is not appropriate to perform a fixed sequence of predetermined actions. Therefore a system's ability to make real-time decisions based on sensing the environmental state is an important factor in whether or not holons can successfully be applied in a particular manufacturing situation.

The group also measures how reconfigurable the holons are [4], in terms of having their control structures and decision-making processes [6] changed at runtime and being re-deployed into different operational circumstances with limited 'swap over' time. Again the extent of the support for run-time reconfiguration is a key factor in determining the range of applications that the system can be put to. The third metric in this group is how extendable the holons. Extensibility relates to how well holons cope with new knowledge and new autonomous skills, making new products with the existing hardware and cooperating with new holons. Since, an extendable holon system can be readily expanded to cope with additional robots and other manufacturing resources, extensibility tends to lead to systems with greater applicability. The focus is on extensibility, because this metric is critical if holons are to be introduced seamlessly into factories and interact effectively with legacy systems such as Manufacturing Execution Systems (MES). It should be clearly noted that the approach described below is used just to illustrate one way to get applicability from the holonic system.

In the Cambridge packing cell, neither rigid nor hierarchical inter-holon organisations are defined. Each holon begins with some knowledge concerning which other holons (peers) it should talk in order to achieve some task. This talking is implemented using message events that are passed between agents. An arguably better approach is to replace holons implemented with autonomous agents with an implementation using JACK Teams™ [19] because team-oriented computing provides a rich set of constructs and would support multi-holon cooperation through holarchies. A paper explaining the relationship between JACK Teams™ and holonic principles is being written [11]. With JACK Teams™, interaction among holons is via built-in constructs (roles, named roles, teamplans and teams). These constructs are used to encapsulate the exchange of task requests and knowledge, and provide a clean interface between holons without having to pre-define which holon instance takes on a particular job at execution time. This is an extensible approach that allows reconfiguration at runtime. For example, it can be used to ensure effective and fair allocations of resource holons' valuable commodities (e.g. time slots) across the order holons that need them.

Another argument in favour of using roles as a means of increasing extensibility is that social relationships among holons built upon roles provide a conceptual framework in which each holon either:

- (i) plays its role as a resource producer or resource consumer, or
- (ii) occupies a well-defined position in the society.

For example, the Fanuc M6i robot holon may take on the role of material handler inside the cell, yet this role may later be filled by a Fanuc A520 or Mitsubishi RV-1A/2AJ robot – the identified roles are static but the specific holon occupying the role is dynamic. Moreover, a production manager holon or a track holon occupies a critical position in the organisation to orchestrate the sequence order holons are spawned or monitor where shuttles are currently located. These knowledge server types of holon roles can be taken only by a specific class of software holon, but they can be re-started at any time if failure occurs. Such a role-based approach allows us to:

- *Re-structure and revise holon interactions.* This is within the control system, and between the control system and existing business information systems.
- *Re-assign capabilities.* Encapsulate knowledge and functionality within different holons without having the stop execution, edit, re-compile and run the software.

The role that a holon adopts determines how extensible its is in terms of:

- *Skills.* Skills represent the functions, services and knowledge processing needed by the holon to assume the desired role. Each resource holon is characterised by the set of abilities that are needed to manage that hardware. For example, an extendible docking station holon should have the facilities to: (i) determine if and when it should process a shuttle, (ii) interact with the hardware to grab and release shuttles, and (iii) cooperate with order holons to allocate timeslots in the schedule and inform them once jobs are completed. Extendible order holons are characterised by the range of skills they need to get themselves manufactured. For example, cooperating with resource holons (according to a production recipe) and cooperating with other orders to buy and sell their partially-completed boxes so urgent orders can be delivered on time. By designing the resource and order holons' skills in a sufficiently general way, the system is made more extensible.
- *Responsibilities.* When a holon assumes a role at execution time, the holon becomes responsible for the successful completion of the tasks associated with that role. For example, when the Fanuc M6i robot holon accepts the role of material handler it becomes obliged to pack boxes, unpack boxes, or sort the storage chutes until that role gets re-assigned. One mechanism to ensure that a responsibility is successfully completed is to issue request messages to the resource holons for each goal in the order holon's recipe. If a resource holon becomes unable to complete its responsibilities during execution of the job, it informs the order holon, which is then required to find another resource that can offer the affected role and negotiate with that holon to maintain the processing.
- *Knowledge Sharing.* Data is requested and provided between order and resource holons so that the resource holon assuming a role can fulfil its responsibilities. This exchange is achieved, over Ethernet with guaranteed delivery, by attaching strictly-typed data and objects onto the messages that are passed among holons.

When comparing two holonic systems, then we would expect the role-based system to achieve better scores along the extendibility metric. Therefore the packing cell deserves a relatively high extendibility score (at least 50%) because it uses simple resource request/response messages within roles and it enable resource holons to dynamic adopt their responsibilities. The packing cell is not idle because it does not let new resource holons be added dynamically to fill each role, or let the system expand at runtime without the need for re-engineering.

5.3. Evaluation of Methodology Used

This group evaluates the methodology utilised to develop the packing cell in terms of how reusable, simple, and generalised this philosophy and its methods are. Simplicity in a methodology results in implemented holons that have both simple interactions [8] and autonomous decision-making. To achieve such simplicity, the methodology supports the decomposition of the control system into well-defined units and provides design guidelines. These guidelines are often the first concern in modern software engineering models, like object-oriented analysis, and the development of holonic systems is no exception. However, the main difference is the dynamism that a holonic system exhibits. So the methodology's guidelines should compensate for this dynamic behaviour, and be easy to comprehend. We propose that a suitable methodology will use a spiral approach with each cycle having the subsequent phases and models:

- *System Design Model.* An anthropomorphic model of the holonic system's requirements and purpose is established. This leads to identifying holon types, their private actions, interactions and the interfaces to the legacy factory and business systems. Developing this model involves creating a functional description of the system using UML concepts, and exploring each holon's responsibilities and collaboration metaphors through role-specific scenarios.
- *Holonic Society Model.* An infrastructure for the interactions and dependencies among holons is then constructed via two steps. First, holon roles are described using class diagrams to isolate distinct roles adopted by holon types, the tasks involved when a holon takes on that role and the communication metaphors used to accomplish the role. Second, the conversation protocols, grammar and pragmatic knowledge structures used by holons are designed.
- *Implementation Model.* Holons are encoded as a solution architecture using the JACK Agent Language constructs (i.e. agents, capabilities, events, plans, and belief structures). Source Java code is produced and executed on a suitable platform using the Java virtual machine and the JACK runtime libraries.
- *Test and Refinement Model.* Individual holon, multi-holon interactions in the society and overall holonic system behaviours are verified against requirements. If problem solving is not validated then refinements are identified for next cycle.

By applying this methodology, we make the following proposition: a simple methodology constructs an efficient holon society so that each holon acts and interacts in a comprehensible manner, and that can be easily changed or reconfigured. The methodology used to build the packing cell did not satisfy this proposition and so it deserves a relatively low score on the simplicity metric. We now make some conclusions about the holonic system that we built and postulate future work in this topic.

6. Conclusion and Ways Forward

6.1. Summary and Critique

The report has explained how the JACK-based Holonic Control of a Gift Box Packing Cell in Cambridge was designed and constructed. The cell brings together industrial-strength equipment (that can be found in real manufacturing businesses) with the next generation of control system philosophy. In this new model, autonomous intelligent building blocks of a factory control system (holons) come together, at runtime, to form social organisations through which coordination can occur to meet the challenges imposed by the requirements of a responsive manufacturing environment. These challenges relate to achieving a novel collection of functional and performance objectives within the scope of existing factory infrastructures. The cell has been used as a test bed to experiment with such holon-oriented responsive behaviour. Table 2 is a league table of what targets were set for the current phase of our work, in terms of the operational system's four functional objectives, versus what has been achieved and how much effort is anticipated to complete (E is Easy, M is Moderate and D is difficult in the current approach).

Objective	Target	Achieve	Effort
Remote ordering	Internet ordering	Yes	
	Pack batches of boxes	Yes	
	Priorities on boxes	Yes	
	Deadlines on orders	No	E
	Use product holons	Yes	
	Resource allocation via negotiation	Partial	M
	Guarantee of delivery	No	D
	Guarantee resource contributions to order	No	M
Order intervention	Change order priority, configuration, time	No	M
	Remove order – unpack box and reuse	Yes	
	Disassemble partly-created low priority box to fill higher box	No	M
Fault tolerance	Allocate resources / materials at runtime	Yes	
	Load and unload at both docking stations	Yes	
	Shuttles route themselves to docking stations	Yes	
	Prioritisation of shuttles at gates	No	E
	Manage resource load profiles, raw materials and work-in-progress	No	M
	Make special offers	No	E
Recipes	Bi-directional shuttles	No	D
	Separation from machine instructions	No	M
	In PML format	No	E
	Multi-cell simulations	No	E
	Generic / concrete recipe mappings	No	D
	PML Recipe progress	No	E

Table 2: Planned versus Realised Functional Objectives.

If we were to analyse the methodology that we used to build the packing cell's holonic control system:

- *Positive.* The control system is highly distributed and a number of specialist holons have been constructed in order to clearly reflect the decentralised physical structure and functional diversity of the cell. Each holon uses an agent to provide a suitable degree of intelligence, e.g. in terms of filtering data, deciding how and when to disseminate data and select a course of action from multiple options.
- *Negative.* There is a fudge of reactive and simple deliberative agent approaches embedded in how the holons work. An example of reactive behaviour is when a box is at a docking station for packing and the items it needs are unavailable and so the items are searched for in the cell to satisfy the order with the box having to wait until the replenishment arrives. By contrast, the docking stations use a simple

reservation policy and deliberate over which shuttle be processed next. In terms of the flexibility and capability of the cell, there is a significant lack of reasoning by the agent-based holons in determining their actions. With respect to the ease of implementation, the system is difficult to debug and has a heavy reliance on polling. The system is slow because there are Internet latencies and access problems to get data from the Auto-ID systems into the control system. These deficiencies result in the overall control philosophy being somewhat incoherent and difficult to comprehend.

If we are to honestly critique the control system, then it is only fair to observe that it 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 system are:

- *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 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.

6.2. Lessons Learnt and Future Work

In the form of a post-mortem, there are several lessons that we have learned by undertaking this research that should be of interest to other designers when deploying holonic systems. These include:

- *Ownership of data.* An important design consideration is to establish who (as in which holon, agent or software entity) owns the various parts of the world model. Hopefully, the metaphor used to divide the system into holonic agents is sufficiently intuitive that the information ownership is obvious. However we found a number of thorny issues in this area. For example, if there is a shuttle holon, should it know where it is on the track? If so, calculating whether there is enough room for another shuttle to fit on a section of track may require interacting with many different holons. The approach that we used was to avoid having separate holons and instead to pool the beliefs about shuttle positions into a single “track” holon. An additional reason for not having separate shuttle agents is that the shuttles are not directly controllable *per se*. Instead, the gates and docking stations control their flow [33]. On the other hand, an advantage of using separate agents for each shuttle is that it is a more natural and intuitive design, that allows the designer to clearly identify that a shuttle might know where it is, and perhaps what physical objects it carries, but otherwise be ignorant of the position of other shuttles. Determining if it is possible for a shuttle to move into an area of track requires the shuttle holons to either (a) register with a central mechanism allowing them to be universally queried to count the number of shuttles in a zone, or (b) maintain separate “zone” agents that monitor the utilisation of areas of track.

- *Avoiding Race Conditions.* The fact that agents communicate via messages rather than shared memory helps to avoid most forms of race condition that might arise in a multi-threaded program. Nevertheless, race conditions are still possible, simply because a single agent may execute multiple plan instances simultaneously. In addition, if two agents maintain information about the same thing, it is important that any updates are performed to both atomically. For example, the original design meant that when a shuttle was being sent to a docking station, both the TrackHolon and the docking station holon needed to be informed. In some subtle cases, the shuttle would be rerouted to a different docking station. In this case, it was important that both copies of the information were kept up to date. Our final design fixed this problem by having the docking station monitor the destination according to the TrackHolon. JACK provides some help with race conditions by ensuring that each Java statement is atomic, and also by providing a Semaphore class that can be used to provide mutual exclusion. However care is still required to ensure that data stored separately is logically independent, or if it is dependent, it is useful to have automatic (and atomic) mechanisms to keep the two sets of data consistent. Race conditions were also discovered when communicating with the blackboard system. These were largely resolved by using mutual exclusion for any plans that talked to them.

Another critical factor for future discussion is how to judge whether the implementation is a success and what metrics should be used to evaluate future holonic system developments. We suggest the key metric is reuse so that the holons developed for this cell could be plugged into another physical environment making a different product and they just work together without significant re-design or re-configuration. To construct the next holonic system, we will improve the following elements of the methodology:

- We will use a simulation to experiment with holons' control mechanisms before integrating agent-based holons [9] with the hardware and include them in the holonic supply chain [36].
- We will use a uniform simple protocol, such as the Contract Net, to create the interactions between every pair of holons because having one interaction protocol per holon pair has resulted in complex dependencies that are difficult to debug.
- We will try to undertake the holons' development with a clear research objective in mind for each function objective, e.g. we will formulate a hypothesis concerning how holons should operate, construct an experimental model to test the hypothesis, write code within the scope of model, run tests, analyse results and make conclusions. This will enable us to claim, with some justification, that holons make businesses more responsive to changes in products, processes and automated equipment.

The code written for this control system is logged under Cambridge's Concurrent Versions System (CVS) in directory mlpc-auto1.eng.cam.ac.uk/home/cvs.

References

1. Agile Manufacturing Benchmarking Consortium, www.ambbenchmarking.org/, (2003)
2. Ashton, K.: Auto-ID Center: The Big Picture, Proc. of Sun / Mass e-commerce Adoption Forum, http://www.autoidcenter.com/media/Mass_e-commerce.pdf, (2003)
3. Barata, J., Camarinha-Matos, L.M.: Implementing a Contract-based Multi-agent Approach for Shop Floor Agility, Proc. of 3rd International workshop on Industrial Applications of Holonic and Multi-Agent Systems, IEEE, Aix en Provence, France, (2002)
4. Brennan, R.W., Fletcher, M., Norrie, D.H.: An Agent-based Approach to Reconfiguration of Real-Time Distributed Processes, IEEE Transactions on Robotics and Automation, volume 18, number 4, (2002)
5. Bussmann, S., Sieverding, J.: Holonic Control of an Engine Assembly Plant – An Industrial Evaluation, Proc. of IEEE Conference on Systems, Man and Cybernetics, Tucson, USA, (2001)
6. Bussmann, S., Jennings, N.R., Wooldridge, M.: Re-use of Interaction Protocols for Decision-Oriented Applications, Proc. of 3rd International workshop on Agent-Oriented Software Engineering, Bologna, Italy, (2002)
7. Chirn, J.-L., McFarlane, D.C.: A Holonic Component-Based Approach to Reconfigurable Manufacturing Control Architecture, Proc. of 1st International workshop on Industrial Applications of Holonic and Multi-Agent Systems, IEEE, Greenwich, UK, (2000)
8. Deen, S.M.: A Cooperation Framework for Holonic Interactions in Manufacturing, Proc. of 2nd International Conference on Cooperating Knowledge Based Systems, Keele UK, (1994)
9. Flake S., Greiger C.H., Lehrenfield G., Muller W., Paelke V.: Agent-based Modelling for Holonic Manufacturing Systems with Fuzzy Control, Proc. of 18th International Conference of North American Fuzzy Information Society, New York, USA, (1999)
10. Fletcher, M., Brusey, J., McFarlane, D.C., Thorne, A., Jarvis D.H., Lucas, A.: Evaluating a Holonic Packing Cell, Proc. of the 1st International Conference on Applications of Holonic and Multi-Agent Systems, Prague, Czech Republic, (2003)
11. Fletcher, M.: The Relationship between JACK Teams™ and Holonics (in progress), technical report of Agent Oriented Software and Cambridge University, (2003)
12. Fletcher M., Jarvis J., Jarvis D., Rönnquist R.: Personal Communication, Adelaide, (August 2002)
13. Gaudreau, P.: Marriage of PLCs and Industrial PCs: Trends in Machine Control, <http://www.inova-computers.de/web/article004.html> (2003)
14. Gerber A., Kammenhuber N., Klusch M., CASA: A Distributed Holonic Multiagent Architecture for Timber Production, Proc. of 1st International Conference on Autonomous Agents and Multiagent Systems, Bologna, Italy, (2002)
15. Heyman, D.: RFID in Action: Killer Applications for Profit, Efficiency & Economy, Proc. of 1st RFID in Action Event, London, UK, (2003)
16. Holonic Manufacturing Systems (HMS) project is an international industry-driven project addressing systematisation and standardisation, research, pre-competitive development, deployment and support of architectures and technologies for open, distributed, intelligent, autonomous and cooperating (holonic) systems on a global basis. Details at http://www.ims.org/projects/project_info/hms.html (2003)
17. Howden N., Rönnquist R., Hodgson A., Lucas A.: JACK Intelligent Agents – Summary of an Agent Infrastructure, Proc. of 5th International Conference on Autonomous Agents, Montreal, Canada, (2001)
18. IMS is an industry-led international research and development program established to develop the next generation of manufacturing and processing technologies, <http://www.ims.org/> (2003)
19. Jarvis, J.: JACK Intelligent Agents™ JACK Teams Manual, <http://www.agent-software.com/shared/demosNdocs/teamsguide/html/> (2003)
20. Kalia, D., Khargonekar, P.P.: Formal Verification for Analysis and Design of Logic Controllers for Reconfigurable Machining Systems, IEEE transactions on Robotics and Automation, volume 18 number 4, (2002)
21. Koestler A.: The Ghost in the Machine, Arkana, (1967)

22. Lyman, T.: Adaptive Analysis of Locally Complex Systems in a Globally Complex World, *Conservation Ecology*, volume 3 issue 2, <http://www.consecol.org/vol3/iss2/art13>, (1999)
23. Marik, V., Fletcher, M., Pechoucek, M.: Holons & Agents: Recent Developments and Mutual Impacts, in *Multi-Agent Systems and Applications*, Springer Verlag, LNAI 2322, (2002)
24. Martin, D., Cheyer, A., Moran, D.: The Open Agent Architecture – A Framework for Building Distributed Software Systems, *Applied Artificial Intelligence*, volume 13, number 1, (1999)
25. McFarlane D.C., Bussmann S.: Developments in Holonic Production Planning and Control, *int. journal of Production Planning and Control*, volume 11, number 6, (2000)
26. McFarlane D.: Personal Communication, Cambridge, (July 2002)
27. Miles, A., Baldwin, T.: Spidergram to Check on Police Forces, *The Times newspaper*, <http://www.timesonline.co.uk/article/0,,2-352275,00.html>, (July 10 2002)
28. Müller J.P., Bauer B., Berger M.: Software Agents for Electronic Business: Opportunities and Challenges, in *Multi-Agent Systems / Applications II*, Springer LNAI 2322, (2002)
29. Parunak, H.: A Practitioner's Review of Industrial Agent Applications, *Autonomous Agents and Multi-Agent Systems*, volume 3, number 4, (2000)
30. Payne, T.R., Paolucci, M., Singh, R., Sycara, K.: Communicating Agents in Open Multi Agent Systems, *Proc. of NASA workshop on Radical Agent Concepts*, Greenbelt USA, (2002)
31. Rana, O., Kotz, D.: Special Issue on High Performance Agent Systems in Concurrency and Computation: Practice and Experience, volume 13 number 1, John Wiley & Sons, (2001)
32. Sathern., E.: Envisioning Agile Packaging – Unilever envisions new packaging machinery that treats change as business as usual by focussing on a 'holonic' approach, www.packworld.com/articles/Features/15421.html, (2002)
33. Schaeffer, C.: Holonic Production and Material Flow in Industry, *Proc. of International Symposium on Holonic Manufacturing Systems*, Kitakyushu, Japan, <http://hms.ifw.uni-hannover.de> (2000)
34. Seki, T., Hasegawa, T.: IDPS Operating System for Constructing Fault-Tolerant Systems, *Information Processing Society of Japan Magazine*, volume 36 number 08-018, (2003)
35. Tichy, P., Slechta, P., Maturana, F., Balasubramanian, S.: Industrial MAS for Planning and Control, *Proc. of 2nd International Conference on Industrial Applications of Holonic and Multi-Agent Systems*, Munich, Germany, (2001)
36. Ulieru, M. (ed.): workshop on Holonic Enterprises at the 3rd International Symposium on Multi-Agent Systems, Large Complex Systems, and E-Businesses, Erfurt Germany, (2002)
37. Van Brussel, H., Valckenaers, P., Bongaerts, L., Peters, P.: PROSA: A Reference Architecture for Holonic Manufacturing Systems, *Computers in Industry*, volume 31 number 3, (1998)
38. van Leeuwen, E.H. (ed.): track on Multi-Agent and Holonic Manufacturing Systems at the 5th IFIP International Conference on Information Technology for Balanced Automation Systems in Manufacturing and Services, Cancun, Mexico, (2002)
39. Wooldridge, M.: *An Introduction to Multi-Agent Systems*, John Wiley & Sons, (2002)