

The Story of the Holonic Packing Cell

Martyn Fletcher

Agent Oriented Software Limited
Institute for Manufacturing, Mill Lane,
Cambridge CB2 1RX, United Kingdom.
Tel: +44 (0)1223 308000

martyn.fletcher@agent-software.co.uk

James Brusey

Auto-ID Centre
Institute for Manufacturing, Mill Lane,
Cambridge CB2 1RX, United Kingdom.
Tel: +44 (0)1223 765605

jpb54@eng.cam.ac.uk

ABSTRACT

The Holonic Packing Cell is a manufacturing environment that is responsive to change and has been constructed at the Institute for Manufacturing (IfM) at Cambridge University. 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 (AOS). The cell is geared towards experimenting with, and demonstrating the benefits of, holonic control. 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 production settings where orders, production processes and resources regularly change. In our story, we will show how this cell was conceptualized and how holons were encoded using the JACK Intelligent Agents™ platform.

Categories and Subject Descriptors

I.2.11 [Distributed Artificial Intelligence]: Intelligent Agents and Multiagent Systems.

J7 [Computers in Other Systems]: Industrial Control.

General Terms

Algorithms, Design, Experimentation.

Keywords

Holonic Manufacturing System, JACK Intelligent Agents.

1. INTRODUCTION

Nowadays, manufacturing companies are facing a rapidly increasing demand from retailers and consumers for mass-customisation. In other words, they are being asked to provide people with products that have been uniquely made to fit that particular person's configuration or taste. Furthermore, people are not prepared to wait long lead times for delivery. Also the proliferation of the Internet means that manufacturing companies

have to more closely integrate new e-business tools to work with their suppliers [4]. For example, a customer may say "I want, via a website, to order an exclusive mobile phone with a specific coloured face, and incorporate the functionality of GPRS, Bluetooth and have a number of selected Java™ games pre-loaded. This specific configuration is not offered by a standard model. Also I want it delivered to my door in 48 hours".

Today, this responsive manufacturing scenario is not feasible because traditional manufacturing control systems are hierarchical and geared to mass production of low-variety goods. Often a schedule of the manufacturing operations on a given day is developed well in advance. As the schedule is executed, unexpected events, typical of many production, warehousing and supply chain environments, tend to invalidate the schedule and change how operations are managed. These events include introduction of rush orders, occurrence of machine failures, incorporation of new multi-function hardware or goods being returned for re-work. In other words, manufacturing businesses are being pushed, by market forces, to be more responsive in terms of providing mass-customisation of their product families and to react more quickly to consumer demands, often within the scope of their existing factory infrastructures. There are several approaches to solve these business problems, which are often put together under the umbrella term of intelligent manufacturing systems [1]. This paper presents one such solution, namely *holonic manufacturing* [5] [6], where each entity in the factory is concurrently an independent system and part of a larger system.

However a key drawback to the rapid adoption of a new manufacturing philosophy, based on the research findings into holonics, is that there is often no associated adoption strategy. In our case, such a roadmap is needed to deploy the conceptual holonic model into real-world factories where there are heterogeneous sub-systems and hard-coded methods within the controllers that manage the physical machinery [7]. This paper presents our story on the lessons learnt while endeavouring to develop such a strategy via building a practical demonstration of holonic manufacturing technology. This demonstration is in the form of a 'pick and place' robotic packing cell where orders for gift boxes are mass-customised by being packed with Gillette™ personal grooming products to meet specific customer requests.

The paper is structured as follows: Section 2 outlines the key aspects of the new holonic technology and its role in tackling the challenges presented by responsive manufacturing. Section 3 outlines the requirements of a packing cell to demonstrate these holonic characteristics, and narrates an adoption strategy in terms

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

Conference AAMAS'03, July, 2003, Melbourne, Australia.
Copyright 2003 ACM 1-58113-000-0/00/0000...\$5.00.

of converting a conceptual holonic design into a pragmatic design that can be implemented using a suitable technology within certain constraints. Section 4 outlines how we have implemented the holons using the JACK Intelligent Agents™ platform. This technology was selected because agents and holons share a number of common models and principles, and JACK offers several features useful for building open agent systems in complex decentralised real-time environments. The paper is concluded with a summary and an outline of opportunities for further research.

2. HOLONS AND THEIR ROLE IN RESPONSIVE MANUFACTURING

In this section we investigate where and how holonic technology can provide a means to deal with the challenges posed by responsive manufacturing. The analysis is based on the principles noted in Müller *et al.* [10].

2.1 Challenges for Responsive Manufacturing

Conventional factory control systems often adopt a hierarchical architecture that is heavily reliant upon a tree of Programmable Logic Controllers (PLCs) that are configured together to manage machines within a cell and then cells inside the factory. PLCs read data in from electronic sensors on the shop-floor, select appropriate actions using 1960s software techniques like ladder logic, and issue commands back to actuators or to the robots. In some modern factories [2], personal computers are being connected to the PLCs via Ethernet to provide more long-term decision-making such as performing simple fault recognition on some hardware. However the real-time and non real-time aspects of the processing in the control system lacks both cohesion and opportunities to reconfigure how the machinery operates (e.g. take the control system off-line, encode a new strategy using Petri Nets, compile it down into ladder logic, push it onto the PLC, bring back on-line and test). An example of this architecture based on the combination of PCs and PLCs is a packing cell that has been constructed at Cambridge University's Institute for Manufacturing. A packing cell is a group of automated machinery and robots. This system (see Figures 1 and 2) enables the customer to select any three of four types of Gillette personal grooming items (e.g. razor, shaving gel, deodorant or shaving foam), and pack them into gift boxes. There are two classes of box: a 3-in-a-row, and a T-shape, each with a different configuration of slots where items can be placed.

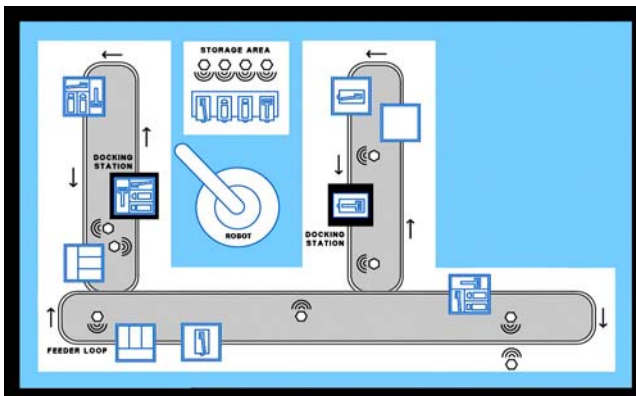


Figure 1: Layout of the Packing Cell.

The system is intended to be responsive to changes in the requirements of customers' orders, and to be robust to hardware alterations. A three-loop Montrack™ conveyor system is used to transport shuttles around the cell. Shuttles hold batches of raw materials (Gillette items) and the boxes into which items are placed. The track (with its independently controlled gates) provides a high degree of responsive behaviour by switching gates to control the flow of shuttles. Shuttles always move forwards along the track unless they touch stop-dogs on the track near the gates and docking stations (which the PLC controls to let the shuttles move when ready) or when their infra-red sensors indicate that another shuttle is in close proximity. There are also two docking stations where shuttles are held so the Fanuc M6i robot can process a shuttle's contents: pack items into boxes, unload raw materials into the item storage zone, and unpack items if the box is no longer required for an order.



Figure 2: Factory Automation Laboratory showing part of the Packing Cell.

Although this is a relatively small scale manufacturing system, the degree of sophistication in the control system to provide the desired functionality is rather complex. In the authors' opinion, as a result of building and debugging such a system, the challenges raised by coding software to handle the complexity needed to give truly responsive behaviour is not adequately supported by today's manufacturing control infrastructures. There are several benefits to manufacturing companies to being more responsive. These include the ability to rapidly react to market forces, the ability to corner niche markets quickly, and the ability to charge more money for the value-added services they attach to each of their produced goods. However there is a downside: at present, substantial investment is required to create an automated factory infrastructure that is responsive to changing needs. So the driving force behind our work is in order to deliver responsiveness into existing factories without incurring the high set-up costs, using demonstrators like the packing cell discussed in this paper. Two major challenges to such 'on the cheap' responsiveness are:

- *Intelligent interaction.* How can autonomous machines, tools, raw materials, knowledge servers, manufacturing cells, and entire enterprises interact intelligently in order to coordinate their activities into a coherent whole when trying to make a rich variety of products and deliver them into the

market? How can the knowledge required to facilitate this interaction be represented, exchanged and processed efficiently? How can we describe how a product is to be manufactured in a resource independent manner, and how can the goals within this description be associated with roles and then be assigned to physical processes on a shop-floor? How can people, legacy hardware and software be integrated effectively? In existing corporate frameworks, huge efforts have been made to establish Manufacturing Resource Planning (MRP) systems and eBusiness solutions that gather, process and provide factory data to enable people and machines to work towards a common goal. Still the problem of pulling all these diverse sub-systems together remains difficult due to their highly dynamic nature.

- *Adaptive manufacturing processes.* How can complex distributed manufacturing processes be modelled in terms of active intelligent software entities with situational awareness that can interoperate with other shop-floor entities and diverse processes to monitor their environment and change their behaviour on demand? This transformation of behaviour must be achieved in real-time and without loss of production capacity in order to make, transport, store and re-use an ever-changing collection of products. How can these processes and the equipment that support these processes cope with such change? What new ideas are needed to implement this new generation of responsive processes based on existing process models and technologies? While several approaches exist to make flexible factories, automating them and letting the shop-floor entities reconfigure themselves to accomplish this flexibility remains a significant problem.

The authors argue that the above challenges to empowering a manufacturing business with high degrees of responsiveness are principally due to the lack of flexibility and competence of the software entities in the factory's control software. These deficiencies lead to situations where the control system cannot handle the inclusion, removal and reconfiguration of machinery, or cannot tolerate unexpected behaviour by an entity or unplanned interactions with other factory sub-systems. To address these challenges we are studying a philosophy based on the notion of holonics where a *holon* models each entity on the shop-floor.

2.2 Characteristics of Holonic Manufacturing

The idea of a *holon* was initially suggested by Koestler [11] 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. 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. In Marik *et al.* [8], a holon is defined as "... autonomous cooperative unit which can be considered as an elementary building block of manufacturing systems with decentralised control". We would like to enhance this definition using Wooldridge's definition of a

rational agent [12] to reflect the strong cohesion between holons and agents. Put simply:

A holon, aware of its situated environment, uses its intelligence, autonomy, cooperation and self-similarity to meet the design and organisational challenges associated with responsive and flexible manufacturing by performing rational reasoning tasks and balancing goal-directed with reactive behaviour.

Holons will be used to encompass intelligent machine control, planning and scheduling of resources, and integration with other factory information and supply chain management systems. In one of the popular holonic systems models, called the Holonic Component Based Architecture [7], there are two classes of holons: *resources* and *orders*, each of which are intelligent and can be composed of other holons. Order holons are spawned upon creation of a 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. Figure 3 shows an architectural framework for holonic manufacturing systems using order and resource holons.

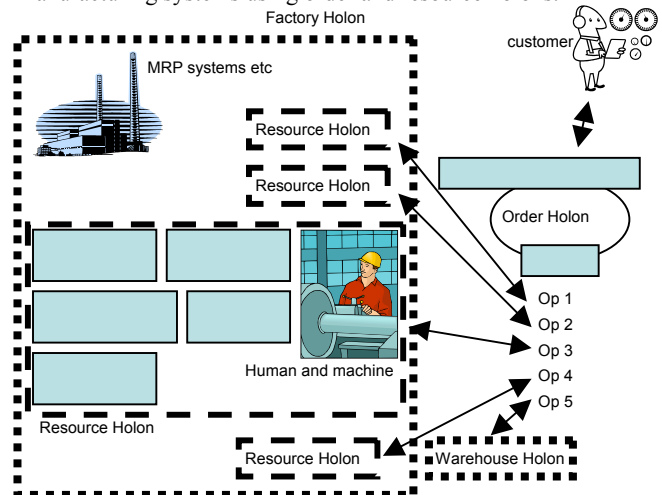


Figure 3: A Holonic Manufacturing System Architecture.

Often special properties are associated with holons; see the articles edited by van Leeuwen [9] for an overview and state-of-the-art review. Of particular interest to this paper, holons provide a number of features to support responsive manufacturing:

- Holons 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 and actuators can be applied to best react to changes upon the shop-floor.

- 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.
- Holon techniques support *decentralisation* at different levels: multi-holon systems are inherently distributed because the hardware, services, processes, knowledge and models of how to control resource holons can be scattered throughout the manufacturing business. Furthermore the recipes on how to make, handle, store etc various products will be resident in multiple order holons that could be strewn across several machines within the factory or even over the web.

Therefore our objective is to move from today's manufacturing systems to the full holonic vision in order to give businesses the competitive advantage and sustainable growth that being responsive provides. However getting companies to throw away their existing automation and control systems to adopt a radical new technology is rather unlikely. Hence we need an adoption strategy and industry-strength environments that can demonstrate the benefits of the holonic vision. The aforementioned packing cell is such a demonstrator. We now highlight how we designed and eventually implemented holonic behaviour within the cell.

3. HOLONIC DESIGN APPROACHES

3.1 The Cambridge Design

Here we outline the holonic system design for the packing cell developed by McFarlane [13]. Note that the majority of the physical hardware for the packing cell was already installed and software based on a conventional control system design was in place. The system had been used to demonstrate the new Auto-ID (automatic identification) technology for tracking and identifying physical objects in the cell (items, boxes etc) using attached RFID tags. Each tag has a unique Electronic Product Code (EPC) and objects have data on how they have been and are to be processed stored in Product Mark-up Language (PML) files.

3.1.1 Objectives for the Holonic System

McFarlane's 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 overall objectives for the 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. This begins with a specification of what the inputs are and what is produced but also includes aspects to do with the system's *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 to meet different demands, or the ability to handle

growth. There is interplay between the two, 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. Both sets can be divided into *functional* and *performance* objectives. The control system and operational objectives lead to specifying the system architecture and overall system behaviour, respectively.

The 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. To allow the system to be flexible and responsive, order holons have an associated priority so that it is possible to give preferential passage to parts associated with high priority products and so important boxes are packed at the earliest opportunity. 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 agility, 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.

In the cell, control system objectives include the ability to reliably deliver raw materials into the system, managing parts during production, moving materials around the cell, storing/buffering materials, and extracting finished products. They influence the construction of the holonic system architecture. These control system objectives lead to the identification of holon types and their interaction infrastructure as illustrated in Figure 4.

3.1.2 Constraints

There were several constraints, imposed by AOS and the client, which restricted how the system's objectives could be realised by the development team. These constraints included:

- *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. In fact by the end of the project, they contributed a total of 65 hours. 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 have 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 them using best practices takes a significant learning period. This 'ramp up' might be 6 months and will bite 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 take 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™ 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 are 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 means that the opportunities for making the holons

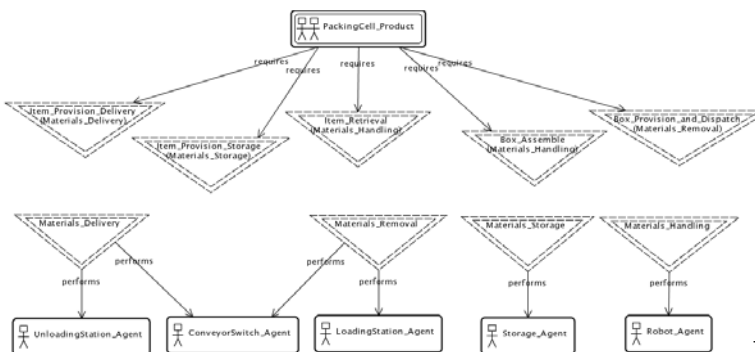


Figure 4: The Cambridge Design.

behave in a robust and fail-safe manner are significantly reduced by not having enough data to make good decisions.

- *Blackboard*. The agents can only interact with the hardware through a legacy piece of software called the “blackboard”. The blackboard (written in Visual Basic) communicates with an Omron™ PLC on the shop-floor to issue commands and receive sensor data to and from the hardware. The blackboard is also used as a gateway to facilitate interaction between the agent and the robot. The blackboard is inflexible, a bottleneck and a key point of failure.
- *Auto-ID*. The Auto-ID’s readers and Savant™ software system are unable to guarantee that when a tag passes a reader then its electronic product code reading will always be observed. Also no guarantee is made 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.

3.2 The Adelaide Design

Here we provide an overview of the revised holonic design produced in Adelaide [14]. 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 principles of Computer Integrated Manufacturing (CIM) and the approaches developed during the international Holonic Manufacturing Systems (HMS) project, namely to explicitly depict holons’ functionality at the machine, cell, factory and enterprise levels.

Within this design (see Figure 5) 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 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 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 depicts 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 the 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 5, 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 [3]. This design was not implemented because of the external constraints listed above.

4. IMPLEMENTED DESIGN

We now present details of the holonic packing cell system design implemented by the authors during late 2002 and early 2003.

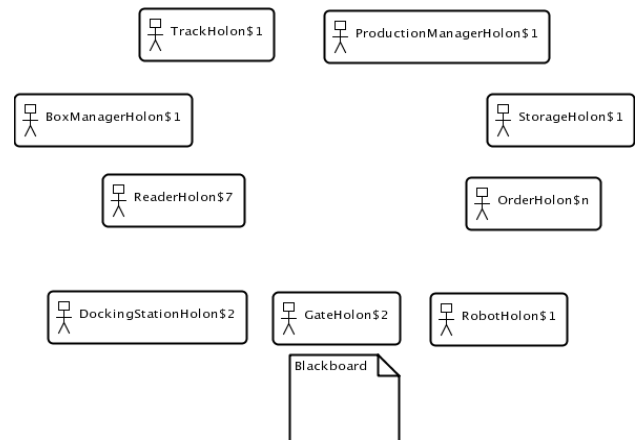


Figure 6: The Implemented Holonic Design.

Figure 6 illustrates the design of the holon architecture that was actually implemented. 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 \$). Only the robot, gate and docking station holons interact with the blackboard. Therefore 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 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, putting additional load on the network 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.

To highlight key features of the implemented model, we outline the system architecture, describe the holonic scenarios that the model supports and discuss some of the lessons learnt.

4.1 System Architecture

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, 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. 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.

4.2 Scenarios

There are six responsive manufacturing scenarios that have been implemented to best illustrate (to academics and industrialists) the above operational system’s functional objectives. 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 first three scenarios in detail because they highlight the types of autonomous decisions and cooperative interactions made by the agent-based holons.

4.2.1 Introduction of 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 `stripgetxqlepc`, 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.

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 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.

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.

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.

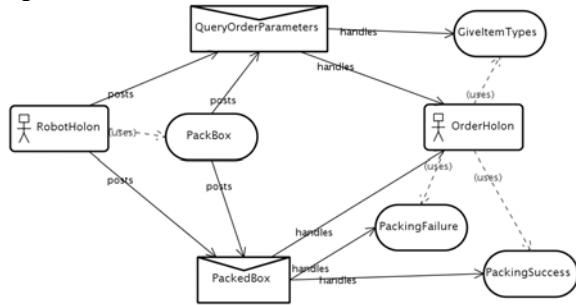


Figure 9: Interactions among Robot and Order Holons.

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 10.

4.3 Discussion

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

- *Ownership of data.* An important design consideration is to establish who (as in which agent or software entity) owns the various parts of the world model. Hopefully, the metaphor used to divide the system into agents is sufficiently intuitive that the ownership of information is obvious. However we found a number of thorny issues in this area. For example, if there is a shuttle agent, 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 agents. The approach that we used was to avoid having separate agents and instead to pool the beliefs about shuttle positions into a single “track” agent. 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. 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 agents 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 is 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 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 the holons just work together without any significant human intervention to re-design or re-configure them.

5. SUMMARY AND FUTURE RESEARCH

The paper explained how the Holonic 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. A robotic packing cell has been used as a test bed to experiment with such holon-oriented responsive behaviour. Table 1 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	Achievement	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
	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
	Bi-directional shuttles	No	D
	Recipes	Separation from machine instructions	No
	In PML format	No	E
	Multi-cell simulations	No	E
	Generic / concrete recipe mappings	No	D

	PML Recipe progress	No	E
--	---------------------	----	---

Table 1: 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 information, 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.

In construction of a new holonic system, we will improve the methodology in the following ways:

- We will use a simulation to experiment with holons' control mechanisms before integrating holons with the hardware.
- 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 do help businesses be more responsive to changes in products, manufacturing processes, and automated equipment.

6. ACKNOWLEDGMENTS

The authors wish to thank Andrew Lucas, Dennis Jarvis, Jacquie Jarvis and Ralph Ronnquist of AOS, and Duncan McFarlane and Alan Thorne of the Auto-ID Centre for their input.

7. REFERENCES

- [1] 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.

- [2] Gaudreau, P. Marriage of PLCs and Industrial PCs: Trends in Machine Control, <http://www.inova-computers.de/web/article004.html> 2003.
- [3] Howden N., Ronnquist R., Hodgson A., Lucas A., JACK Intelligent Agents – Summary of an Agent Infrastructure, 5th int. conf. on Autonomous Agents, Montreal, Canada, 2001.
- [4] Allen R.H., Nidamarthi S., Regalla S., Enhancing Collaboration using an Internet Integrated Workbench, ASME conf. on Design Engineering, Las Vegas, USA, 1999.
- [5] Gerber A., Kammenhuber N., Klusch M., CASA: A Distributed Holonic Multiagent Architecture for Timber Production, 1st int. conf. on Autonomous Agents and Multiagent Systems, Bologna, Italy, 2002.
- [6] Flake S., Greiger C.H., Lehrenfield G., Muller W., Paelke V., Agent-based Modelling for Holonic Manufacturing Systems with Fuzzy Control, 18th int. conf. of North American Fuzzy Information Society, New York, USA, 1999.
- [7] McFarlane D.C., Bussmann S., Developments in Holonic Production Planning and Control, int. journal of Production Planning and Control, vol 11, no 6, 2000.
- [8] Marik V., Fletcher M., Pechoucek M., Holons and Agents: Recent Developments and Mutual Impacts, in Multi-Agent Systems and Applications II, Springer LNAI 2322, 2002.
- [9] van Leeuwen E.H. (editor), track on Multi-Agent and Holonic Manufacturing Systems at the 5th IFIP int. conf. on Information Technology for Balanced Automation Systems in Manufacturing and Services, Cancun, Mexico, 2002.
- [10] 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.
- [11] Koestler A., The Ghost in the Machine, Arkana, 1967.
- [12] Wooldridge M., An Introduction to Multiagent Systems, John Wiley and Sons Ltd, 2002.
- [13] McFarlane D., Personal Communication, Cambridge, July 2002.
- [14] Fletcher M., Jarvis J., Jarvis D., Ronnquist R., Personal Communication, Adelaide, August 2002.

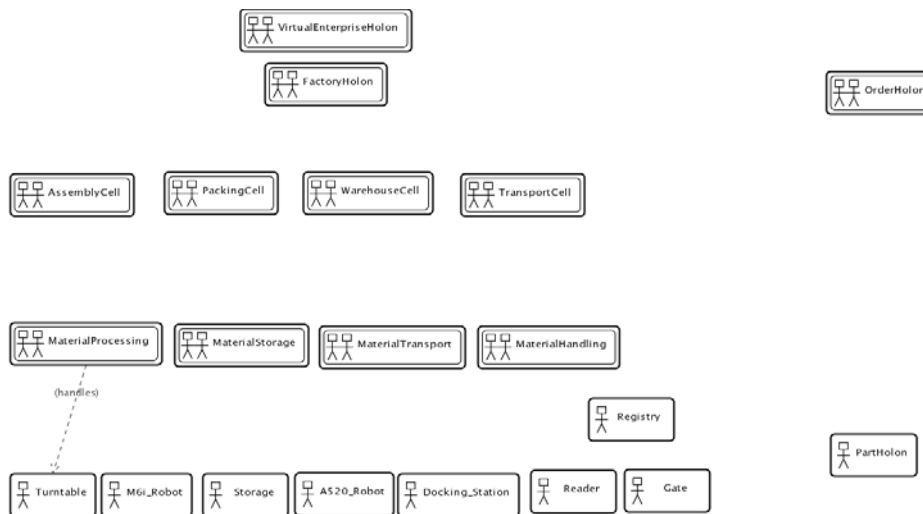


Figure 5: The Adelaide Design.

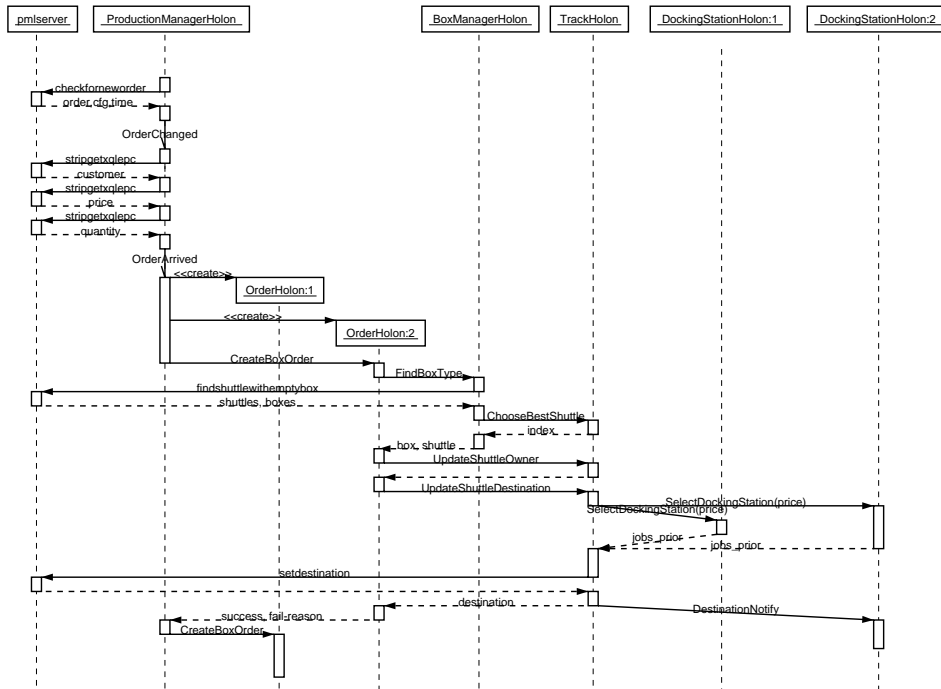


Figure 7: UML Interaction Diagram for Introducing Batch Orders.

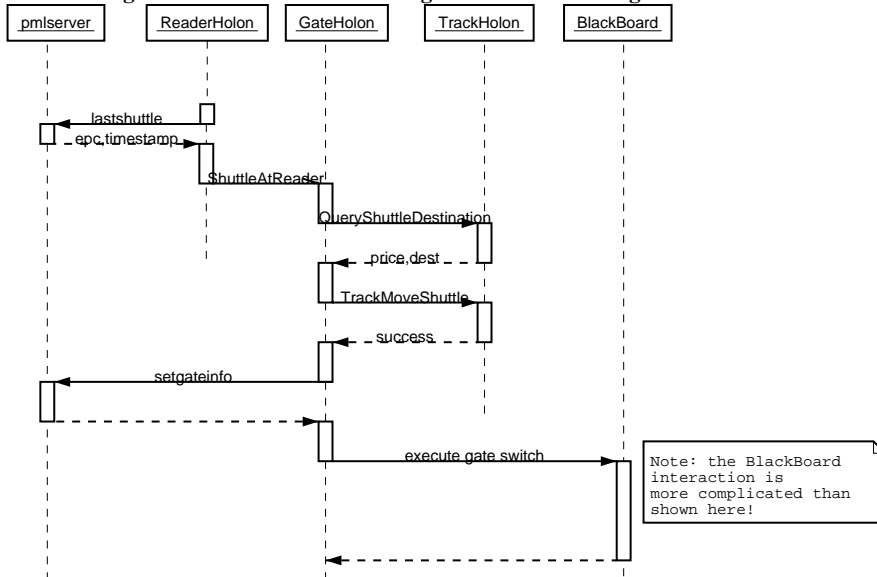


Figure 8: UML Interaction Diagram for Shuttle Navigation.

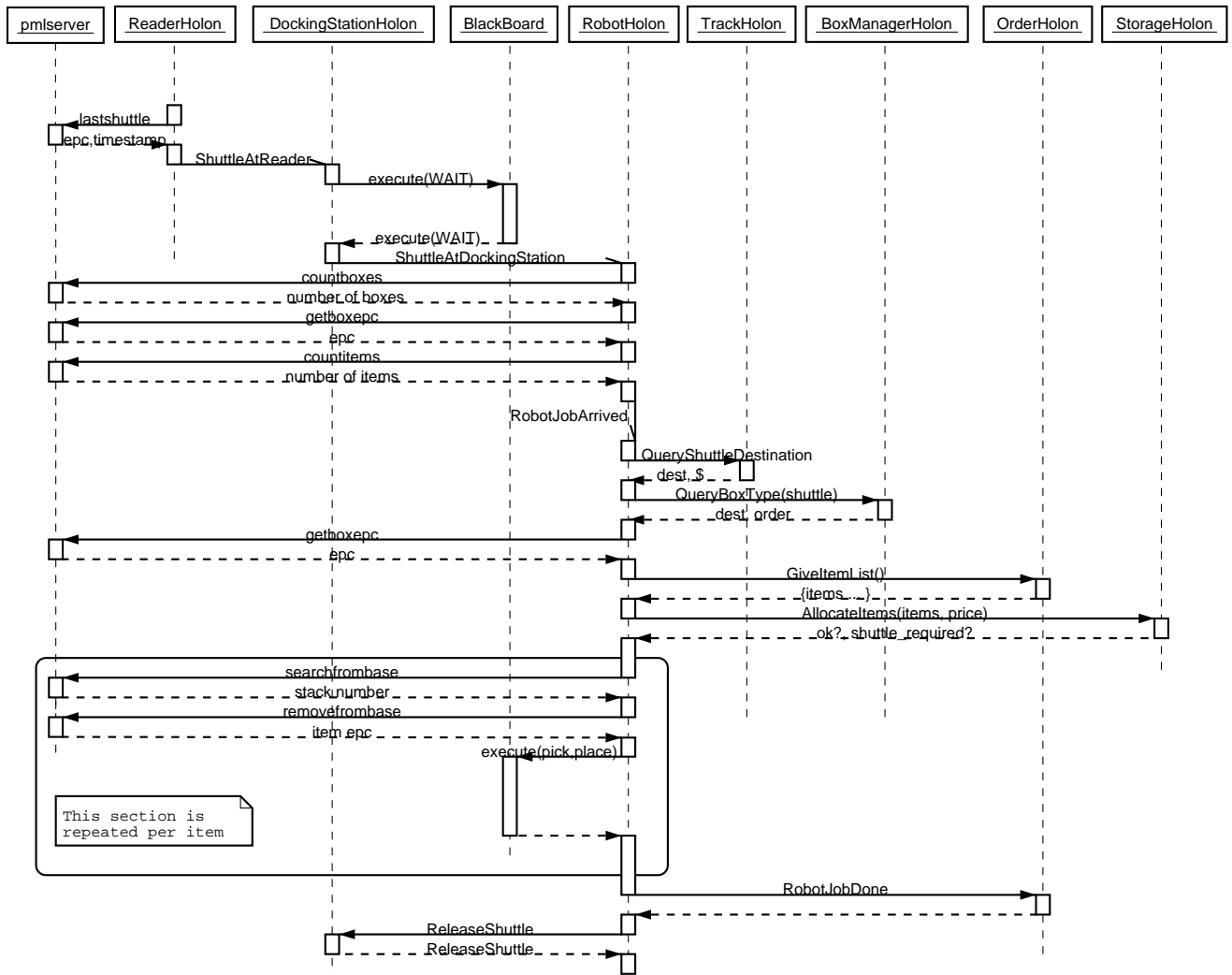


Figure 10: UML Interaction Diagram for Packing a Box.