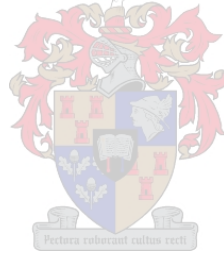


Evaluation of vision-based robot control configurations for reconfigurable assembly

by
Clint Alex Steed

Thesis presented in partial fulfilment of the requirements for the degree of Master of Engineering (Mechatronic) in the Faculty of Engineering at Stellenbosch University



Supervisor: Prof AH Basson

March 2016

Declaration

By submitting this thesis electronically, I declare that the entirety of the work contained therein is my own, original work, that I am the sole author thereof (save to the extent explicitly otherwise stated), that reproduction and publication thereof by Stellenbosch University will not infringe any third party rights and that I have not previously in its entirety or in part submitted it for obtaining any qualification.

Date: March 2016

Copyright © 2016 Stellenbosch University
All rights reserved

Abstract

This thesis considers vision based robot control for a reconfigurable manufacturing cell. Reconfigurable manufacturing systems are aimed at rapidly adapting to fluctuations in market demand, shorter product life cycles and product customization. Computer vision is a promising component of such manufacturing systems, in particular aiding the flexibility to handle a range of products with reduced set-up time and fewer fixtures.

The objective of the research presented here was to develop a simulation-based approach to compare eye-in-hand and fixed camera vision-based control within a reconfigurable assembly cell. These configurations have dissimilar system costs and system performance characteristics. The research used a KUKA six degree of freedom articulated arm robot, a DVT Legend 540 camera and a Cognex ism-1100 camera, as well as the software supplied with the cameras. The simulation was aimed at predicting the throughput rate of multiple eye-in-hand and fixed camera configurations, where configurations include varying the numbers of reconfigurable singulation units, positions of machines, parts being singulated, etc. The throughput rates for different configurations can then be compared to their costs and ease of reconfiguration.

The simulation uses a holonic control system that was designed based on the ADACOR architecture. The thesis describes in detail the holons used, including their hardware, data structures and responsibilities. A multi-agent system was implemented for high-level control and some of the agents communicated with their respective lower-level controllers during validation testing. In the simulation, software (including KUKA Simpro and some custom software) replaced the hardware and low-level controllers. Tests on a number of physical laboratory configurations were used to validate the simulation.

The simulation's application is demonstrated in a number of experiments in which cell configurations and machine performance were varied. For example, for a particular situation simulated, these experiments showed that the eye-in-hand configuration has a competitive cost to performance ratio when tasks have a significant waiting period (pallet exchange time, tasks deployed sparsely, etc.). In most experiments, however the fixed camera configuration performs better. The simulation allowed "hardware in the loop" testing, which also makes it a useful tool for development of the cell.

Uittreksel

Hierdie tesis ondersoek visie-gebaseer robotbeheer vir 'n herkonfigureerbare vervaardigingsel. Herkonfigureerbare stelsels is daarop gemik om vinnig aan te pas by skommeling in die markaanvraag, korter produklewensiklusse en produk-aanpassings. Rekenaarvisie is 'n belowende onderdeel vir sulke stelsels, in besonder vir die buigzaamheid hantering van 'n reeks produkte met verminderde opsteltyd en verminderde gebruik van setmate.

Die doel van navorsing wat hier aangebied word, was om 'n simulatie-gebaseerde benadering te ontwikkel vir die keuse tussen "eye-in-hand-" en vaste-kamera-beheer in 'n herkonfigureerbare samestelling-sel. Hierdie konfigurasies het verskillende eienskappe ten opsigte van stelselkoste en -werkverrigting. Die navorsing het 'n Kuka ses vryheidsgraad geartikuleerde-arm-robot, 'n DVT Legend 540 kamera and 'n Cognex ism-1100 kamera, asook die programmatuur wat met die kameras voorsien word, gebruik. Die simulatie is daarop gemik om die deursetkoers van verskillende "eye-in-hand-" en vaste-kamera-konfigurasies te voorspel, waar konfigurasies verskillende aantal herkonfigureerbare singulasie-eenhede, posisies van masjiene, onderdele wat gesinguleer word, ens. insluit. Die deursetkoerse van verskillende konfigurasies kan dan met hul koste vergelyk word.

Die simulatie gebruik 'n holoniese beheerstelsel, gebaseer op die ADACOR argitektuur. Die tesis beskryf in detail die holons wat gebruik is, insluitend hul hardeware, datastrukture en verantwoordelikhede. 'n Multi-agent-stelsel is geïmplementeer vir die hoë-vlak-beheer en sommige agente het met hul onderskeie laer-vlak-beheerders tydens validasie-toetse gekommunikeer. In die simulatie is die hardeware en lae-vlakbeheerders vervang met programmatuur (insluitend Kuka Sim Pro en sommige doelgemaakte programmatuur). Toetse op 'n aantal fisiese laboratorium-konfigurasies is gebruik om die simulatie te valideer.

Die simulatie se toepassing is gedemonstreer in 'n aantal eksperimente waarin selkonfigurasies en masjienprestasie gewissel is. Byvoorbeeld, vir 'n spesifieke konfigurasie wat gesimuleer is, het die eksperimente getoon dat die "eye-in-hand"-konfigurasie 'n laer koste-tot-prestasie-verhouding het wanneer take 'n beduidende wagtydperk (ruiltyd vir pallette, take yl ontplooi, ens.) het. Die simulatie het "hardeware in die lus" toetsing moontlik gemaak, wat dit ook 'n nuttige hulpmiddel maak tydens die ontwikkeling van die sel.

Acknowledgements

I would like to express my sincere gratitude to my supervisor Prof AH Basson for the continuous support of my research, for his patience, motivation, and wealth of knowledge. His guidance helped me in the time of research and writing of this thesis.

I would also like to thank my fellow students in the Mechatronics, Automation and Designed Research Group for your opinions in many stimulating discussions. I would also like to thank Reynaldo Rodriguez for his technical and assistance.

Finally yet importantly, I would like to thank my family and close friends for supporting me throughout my academic career. It has been a long journey and I would not have come this far was it not for the support of those around me.

Table of contents

	Page
List of figures	x
List of tables	xiii
Abbreviations and terminology	xiv
1. Introduction	1
1.1. Background.....	1
1.2. Objectives	2
1.3. Motivation	2
2. Literature review	3
2.1. Manufacturing systems overview	3
2.1.1. Flexible manufacturing systems.....	3
2.1.2. Reconfigurable manufacturing.....	4
2.2. Reconfigurable control	5
2.2.1. Holonic control paradigm	5
2.2.2. ADACOR architecture	6
2.3. JADE	8
2.3.1. Communication.....	8
2.3.2. Agent management	9
2.3.3. JADE behaviours	9
2.4. Computer vision for control in automation	11
2.4.1. Image processing and lighting	12
2.4.2. Vision based control.....	13
2.4.3. Vision system physical configurations	15
3. Case study.....	17

3.1.	Reconfigurable assembly cell overview	17
3.2.	Feeding station.....	19
3.2.1.	General configuration	19
3.2.2.	Fixed camera configuration	21
3.2.3.	Eye-in-hand configuration	22
3.2.4.	Subsystem control design requirements.....	22
4.	Control architecture	23
4.1.	Reference architecture selection	23
4.2.	Coordinator Holon	23
4.3.	Supervisor Holon	24
4.4.	Task holon	24
4.5.	Subtask Holon	24
4.6.	Operational holons.....	25
4.7.	Operation flow	25
4.8.	Scalability considerations	28
4.9.	Storage of component position coordinates.....	29
4.10.	EIH open loop vs close loop.....	30
5.	Implementation.....	32
5.1.	JADE as the HLC Software platform	32
5.2.	Coordinator holon.....	33
5.2.1.	Architecture.....	33
5.2.1.	Behaviours and lifetime	33
5.3.	Task holon	33
5.3.1.	Architecture.....	33
5.3.2.	Behaviours and lifetime	36

5.3.3.	Task subtask communication	37
5.4.	Supervisor Holon	38
5.4.1.	Architecture	38
5.4.2.	Service list	39
5.4.3.	Behaviours and lifetime	39
5.4.4.	Booking phase	40
5.5.	Subtask holon	42
5.5.1.	Architecture	42
5.5.2.	Behaviours and lifetime	42
5.6.	Common aspects of operational holons	45
5.6.1.	Architecture	45
5.6.2.	Overview of behaviours and lifetime	45
5.6.3.	Service reservation response (booking phase)	47
5.6.4.	Service complete listener	49
5.6.5.	EIH inspection considerations	49
5.7.	Singulation unit (SU) holon	51
5.7.1.	Architecture	51
5.7.2.	Coordinate system calibration	54
5.7.3.	Behaviours and lifetime	55
5.8.	Transport holon	58
5.8.1.	Architecture	58
5.8.2.	Robot control	59
5.8.3.	Gripper control	62
5.8.4.	Behaviours and lifetime	62
5.9.	Quality inspection holon	63

5.9.1.	Architecture.....	63
5.9.2.	Behaviour and lifetime.....	64
6.	Simulation model	66
6.1.	Overview	66
6.2.	Experiment 1: Validation of simulation model	66
6.2.1.	Fixed camera test	66
6.2.1.	EIH system test	67
6.2.2.	Robot movement scaling.....	68
6.2.3.	CPU effect on KRC controller	69
6.3.	Simulation 2: Influence of SU performance characteristics	70
6.4.	Simulation 3: Task scaling	73
6.4.1.	Simulation 3A: Task scaling without implicating transport time.....	73
6.4.2.	Simulation 3B: Task scaling with the implication of transport time	75
6.5.	Assessment of experimental results.....	77
7.	Conclusions	79
8.	References	81
Appendix A	Agent services.....	84
Appendix B	Simulation experiment additional information	85
B 1.	Simulation configuration data	85
B 2.	Saturation calculations.....	86
B 2.1.	FC application	86
B 2.2.	EIH application	87
B 2.3.	Example calculations	88
Appendix C	Conversational behaviours.....	90

C 1.	Conversations of feeding one component	90
C 2.	Nested services	91
Appendix D	Practical measures related to computer vision.....	92
D 1.	EIH Camera Calibration	92
D 2.	Fixed camera viewing angle for component identification	93
D 3.	Environmental considerations for visual inspection.....	93
Appendix E	Further notes	94
E 1.	Agent Class hierarchy.....	94
E 2.	When and how to split a holon	94

List of figures

	Page
Figure 1: Manufacturing cost vs. capacity (Koren & Shpitalni, 2011).....	4
Figure 2: A holon as described by Vrba <i>et al.</i> (2011).....	6
Figure 3: ADACOR holon classes (Leitao & Restivo, 2005).....	7
Figure 4: Conceptual model for an ADACOR holon (Leitao & Restivo, 2005).....	8
Figure 5: Composite behaviour example.	10
Figure 6: Request Interaction Protocol.	11
Figure 7: Contract Net Interaction Protocol.	11
Figure 8: Effect of external lighting on segmentation (Kopparapu, 2006).....	13
Figure 9: Example of open loop visual servoing.	14
Figure 10: Reconfigurable assembly cell developed by RMS research group (Kruger, 2013).	17
Figure 11: Photograph of Pallet with components in the fixtures (Kruger, 2013, p.29).	18
Figure 12: Cell controller holarchy.....	18
Figure 13: Schematic layout of the feeder subsystem (Kruger, 2013, p.22).	19
Figure 14: The operation of a stepped conveyor (Kruger & Basson, 2014).....	20
Figure 15: Three component magazines.	21
Figure 16: Flow diagram of operational sequence.....	27
Figure 17: Holon communication production.....	28
Figure 18: Scalability illustration.	29
Figure 19: Task holon architecture.	35
Figure 20: Pallet agent flow diagram.....	37
Figure 21: Task-Subtask conversation.....	38

Figure 22: Supervisor agent flow diagram.	40
Figure 23: The booking of services.	41
Figure 24: Subtask Agent flow diagram.	43
Figure 25: Execution phase, interaction between subtask and OHs.	44
Figure 26: Operational Holon.	46
Figure 27: Generic Operational Agent flow diagram.	47
Figure 28: Flow diagram of Service Reservation Response behaviour.	48
Figure 29: Flow diagram of Service Complete Listener behaviour.	49
Figure 30: Sub services architecture.	50
Figure 31: Simulated FC and EIH SUs used for experiments.	52
Figure 32: SU holon architecture configurations.	53
Figure 33: Cognex calibration showing SU base.	54
Figure 34: In-sight software Locating components.	55
Figure 35: Component collection coordinate transformation.	55
Figure 36: Fixed Camera Singulation Unit Agent Behaviours.	56
Figure 37: EIH singulation agent behaviours.	57
Figure 38: Transport holon components.	58
Figure 39: Robot communication between Java and KRL program.	60
Figure 40: Transport holon service supplier behaviour.	63
Figure 41: Quality inspection holon architecture.	63
Figure 42: Quality inspection agents service supplier behaviour.	64
Figure 43: Quality inspection of planet.	65
Figure 44: Graphical results from scaling the robot speed during simulation validation experiment.	69
Figure 45: Graph showing simulation C with CPU burden.	70

Figure 46: Layout of holons Simulations 2 and 3.....	71
Figure 47: Throughput rate of system for various SU properties.	72
Figure 48: Throughput rate per unit cost for system Simulation 2.....	73
Figure 49: Feeding time for simulation 3 with no transport time.	74
Figure 50: Throughput rate to cost ratio results simulation 3 with no transport time.	75
Figure 51: Feeding time for simulation 3 with 12-second transport time.	76
Figure 52: Throughput rate to cost ratio results simulation 3 with 12-second transport time.	77

List of tables

	Page
Table 1: Open loop vs. closed loop vision based control.	31
Table 2: Part data class.	36
Table 3: Content of “ServiceList” class.	39
Table 4: Control integer description.	61
Table 5: Hardware configurations fixed camera validation experiment.	67
Table 6: Fixed camera validation experiment results.	67
Table 7: PC hardware specification during experiments	68

Abbreviations and terminology

ACL	-	Agent Communication Language
CFP	-	Call for proposal
EIH	-	Eye-in-hand, referring to the vision sensor being fixed to the end effector of the robot
FC	-	Fixed camera
HLC	-	High level controller
LLC	-	Low level controller
OH	-	Operational holon
PH	-	Product holon
QI	-	Quality inspection
RMS	-	Reconfigurable manufacturing system
SCL	-	Service complete listener
SH	-	Supervisor holon
SRR	-	Service reservation response
TH	-	Task holon
VS	-	Visual servoing

1. Introduction

1.1. Background

In recent decades, the global market has grown to demand high quality lower cost, highly customized products with shorter life cycles (Leitao, 2009). Economic globalisation and turbulent demand has forced manufacturing enterprises to consider alternative production paradigms (Bi et al., 2008). Manufacturing systems today must meet new requirements in terms of flexibility, quality, response and agility if they are to stay in business (Leitao, 2009).

Manufacturing is one of the main wealth generators of the world economy making up approximately 20 % of European Union National Gross Product (Leitao, 2009). Manufacturing is also important to the economic growth of South Africa; however, labour cost in South Africa is high relative to other developing countries (Edwards & Golub, 2004) and recent strikes have yielded higher risks for manufacturing companies that rely on manual labour. South Africa's growth rate of exported goods has been below world growth rates from 1980 to 2000 (Edwards & Golub, 2004). Edwards & Golub (2004) note many developing countries dramatically altered their export composition away from primary products and towards manufactured goods, but this was not the case for South Africa. Since South Africa does not manufacture on the large scale of many other developing countries, automated manufacture using conventional automation approach is in many cases not feasible. A new manufacturing paradigm, called Reconfigurable Manufacturing Systems (RMS) is emerging that may possess characteristics well suited for South Africa's situation.

Computer vision is a versatile and powerful technology that allows computers to evaluate a scene or image and determine from it a number of parameters such as a part position or orientation. The technology is used in many fields, which include artificial intelligence and manufacturing. Vision based control (or visual servoing) is the use of a camera or vision system to control a manipulator, usually a robot. In general, two camera placement configurations are found, namely the (1) **fixed camera (FC)** and (2) **eye-in-hand (EIH)** configurations. The first refers to mounting the camera in a fixed position where the vision sensor and area viewed will not change. The second refers to the camera being placed on the end effector of a robot, in this case the vision sensor moves and the scene changes. Both configurations have advantages and disadvantages. The relative advantages and disadvantages of FC and EIH configurations in the contexts of RMSs are not immediately apparent. This thesis is aimed at helping to provide a basis to choose between FC and EIH configurations.

This research forms part of a bigger project in which reconfigurable manufacturing technologies are being developed by the Mechatronics, Automation and Design Research Group at the University of Stellenbosch. The Research Group uses a

reconfigurable automated assembly cell as the research vehicle. The assembly cell is designed to assemble a family of circuit breakers using a product family of CBI Electric Low Voltage. This thesis will use a substation of the assembly cell as a case study.

1.2. Objectives

The objective of the research is to develop a simulation-based approach to choose between EIH and FC vision-based controls within reconfigurable assembly systems similar to the reconfigurable assembly cell mentioned above. The research was constrained to use the major equipment currently available in the cell, including a six degree of freedom articulated arm robot, a DVT Legend 540 camera and a Cognex ism-1100 camera, as well as the software supplied with the cameras.

The simulation must reasonably be able to predict the throughput rate of alternative EIH and FC station configurations, where configurations include varying numbers of singulation units, positions of machines, parts being singulated, etc.

1.3. Motivation

Vision systems are inherently functionally flexible since they are not limited to a specific part or part geometry. The technology is becoming more prevalent in the manufacturing industry with multiple applications including counting, locating components, navigation and inspection.

FC vision configurations are the preferred choice in industrial applications for a number of reasons, such as that they are less susceptible to changes in environmental variables which include lighting, viewing angles, etc. This makes them more robust and, in some instances, requires less processing time than eye-in-hand systems. Fixed configurations are ideal for fault inspection since they perform repeatable and reliable inspections at a high rate.

EIH vision configurations have the advantage of using only one camera instead of multiple fixed cameras. Vision systems are often expensive pieces of equipment and extra cameras would increase the cost of the subsystem containing the vision system substantially, hence increasing the cost of the final product. In addition, EIH configurations can be used for viewing an object from multiple angles, thereby providing more information that can be used for control or quality assurance.

Both configurations therefore have advantages and disadvantages and this research entails the investigation into each configuration's advantages and disadvantages, and which situations are best suited for each configuration.

2. Literature review

2.1. Manufacturing systems overview

In the 21st century, manufacturing systems have to be able to deal with frequent and unpredictable changes to both product type and production quantities (Koren et al., 1999). These changes could be driven by a number of influences including global competition (Koren et al., 1999), rapid introduction of new products and constantly varying product demands (Koren & Shpitalni, 2011). Therefore, to be competitive, manufacturing systems of the 21st century should be equipped for greater product variety, shorter product life cycles and large fluctuations in demand, while retaining quality and delivery time.

Mass production is known as the production of large quantities of standardized parts. This is synonymous with Henry Ford and the invention of the moving assembly line (Koren & Shpitalni, 2011). Mass production was also made possible by the invention of the lesser-known dedicated manufacturing lines (DML). These lines are designed to produce large quantities of specific parts with little to no flexibility regarding part variation (Koren & Shpitalni, 2011). Each DML utilizes a fixed automation system to manufacture high volumes of a single part at high production rates (Koren et al., 1999); therefore, DMLs have the advantage of lower cost of final products.

Since a DML only manufactures a single product, the line can be considered rigid (or not flexible) and is not scalable since it has a fixed production capacity. An entire line has to be constructed to increase production rates (refer to Figure 1). DMLs are only feasible when the demand for product exceeds supply and the line can operate at full capacity (Koren et al., 1999). Therefore, it is easy to see why dedicated manufacturing systems are not feasible for medium to small product quantities as in the case of South Africa.

Recently customer demand and global competition have led to the development of mass customization (Hu, 2013), where manufacturers design basic product options and customers are allowed to select assembly combinations. Mass customization was made possible by many new technologies, including flexible manufacturing lines and reconfigurable manufacturing systems (Hu, 2013).

2.1.1. Flexible manufacturing systems

With the invention of NC and CNC machines, came the development of flexible manufacturing systems (FMSs) (Koren & Shpitalni, 2011). These systems can produce a variety of parts with changeable volumes (scalable), but have a low throughput rate when compared to dedicated manufacturing systems (refer to Figure 1 below). Flexible manufacturing systems use expensive, general-purpose machinery, and as a result require large capital investment (Koren et al., 1999).

Since the specific application is not known in advance, the system is constructed with a wide range of functionality. This requires large capital investment and results in underutilization within the system. The high capital investment, coupled to a low throughput rate, yields a high cost per product when compared to DMLs. Koren *et al.* (1999) stated FMSs have a low level of acceptance and satisfaction in the manufacturing world, which is not the case today as FMSs are used in the automotive industry.

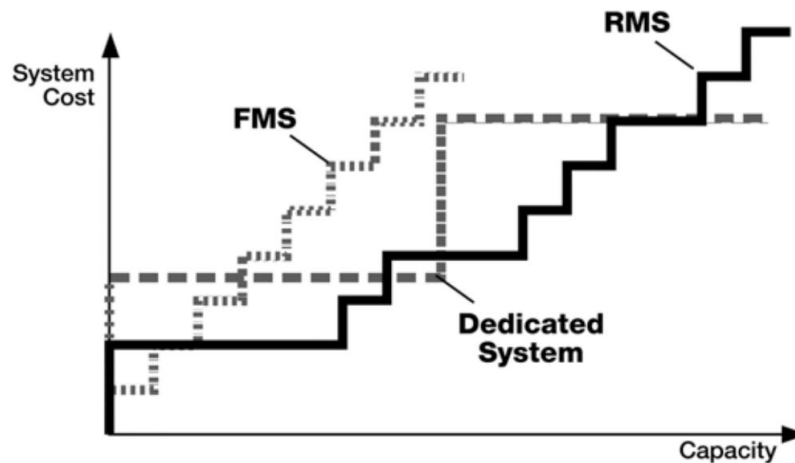


Figure 1: Manufacturing cost vs. capacity (Koren & Shpitalni, 2011).

2.1.2. Reconfigurable manufacturing

RMSs are based on a novel technology that combines the throughput rate of DML with the flexibility of FMS (Koren *et al.*, 1999). The term reconfigurability has many meanings in different contexts but for RMSs, it is defined as the ability to adjust production capacity and functionality through changing system components (Harrison *et al.*, 2007). A RMS also has rapid responsiveness which allows the systems to quickly launch new products and rapidly and cost-effectively react to: product demand changes, product changes, the introduction of new products and non-critical system failures (Koren & Shpitalni, 2011).

Koren & Shpitalni (2011) define the characteristics of a RMS as:

- Scalability - the ability to change production capacity, usually achieved by adding or removing machines/modules for the system.
- Modularity – the compartmentalisation of operational functions into units. These units can be added, removed and rearranged as needed.
- Integrability – allows for rapid integration of modules with respect to mechanical, control, communication and informational systems.
- Diagnosability – the ability to detect the cause of interruptions, errors and product defects.

- Convertibility – the ability to change the function of a machine or module to suit a new process or product.
- Customization – system or machine flexibility (limited to a single product family).

Through these characteristics, RMSs have the advantage of being able to reconfigure and reuse machinery (Harrison et al., 2007) and also adapt or add newly developed technologies and processes through modular design (Borangui et al., 2009). This maximises return on investment, noting that these systems are often initially more expensive (Harrison et al., 2007).

2.2. Reconfigurable control

2.2.1. Holonic control paradigm

New manufacturing systems require new control paradigms that allow for the characteristics of these manufacturing technologies. One proposed control paradigm that is widely accepted in RMSs is holonic-manufacturing systems (HMSs). Holonic manufacturing is based on the concept of a holon, which is defined as an autonomous and co-operative building block of a manufacturing system. Holon comes from the Greek *holos*, meaning whole and *on*, meaning part of or a particle (Koestler, 1967). Therefore, a holon can be a particle in a system, a complete system on its own or both a complete system on its own and particle in a larger system.

HMSs use object-orientated concepts like aggregation and specialisation (Van Brussel et al., 1998). According to Van Brussel *et al.* (1998), HMSs promise to handle both product adaption and high flexibility successfully. The holonic manufacturing architecture also allows for self-configuration, extension, modification, more flexibility and a larger decision space at higher control levels. HMSs do not separate the manufacturing system from the control system, the holons comprise of both. This form of encapsulation reduces the number of interaction between low-level components. Such interactions, according to Van Brussel *et al.* (1998), results in a system that is difficult to understand, control and predict.

A holon is usually separated into a high-level controller (HLC), a low-level controller (LLC) and the control interface between the two (Vrba et al., 2011), as illustrated in Figure 2. The separation of the HLC and LLC is often a requirement in terms of modularity, as holons need to communicate with each other through a standard interface, here, higher-level programming languages are appropriate. Higher-level languages are often not able to communicate directly with hardware; hence, a LLC layer is needed. The LLC can be a generic controller, for example a PLC but may also be an application specific controller or OS, for instance a smart camera or robot controller. It is also worth noting that a holon can contain multiple

LLCs. In addition, a holon may consist of an information-processing component with no physical components and no LLCs.

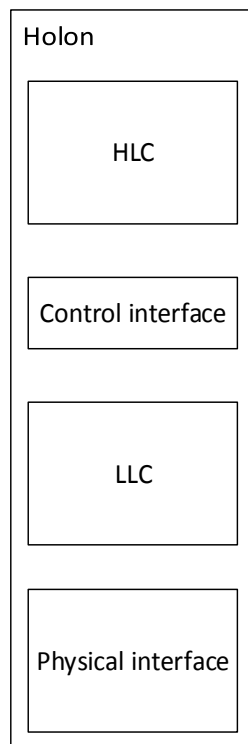


Figure 2: A holon as described by Vrba *et al.* (2011).

2.2.2. ADACOR architecture

Two popular reference architectures for holonic manufacturing systems are PROSA (Van Brussel *et al.*, 1998) and ADACOR (Leitao & Restivo, 2005). These architectures describe a group of generic holons and their respective roles within the system.

ADACOR is a particularly promising reference architecture as it was designed for adaptive and agile manufacturing control and implements a hybrid hierarchal/hierarchical approach that allows for optimisation, which PROSA does not. The focus of the ADACOR architecture is the shop floor with systems characterized by alternative routings, asynchronous and concurrent processes. The ADACOR architecture is described by four manufacturing classes, namely the product holon (PH), task holon (TH), supervisor holon (SH), and operational holon (OH) as illustrated in Figure 3. These holons are modelled as intelligent, autonomous entities with learning and self-organisation capabilities.

A **product holon** (PH) exists for each type of final product. It contains all the information related to the product and can be thought of as the product “recipe”. A

task holon (TH) contains the dynamic information of each production order and is responsible for managing execution. The **supervisor holon** (SH) is responsible for optimization in coordinated holon groups. The **operational holons** (OH) represents a shop floor resource or machine. Further, **staff holons** offer utility services to the other holons.

When the system is running without any disturbances the SH optimises processes at the operational level. This is done by proposing an optimized execution order to the TH. Alternatively, when a disturbance in the system occurs each TH communicates directly with OHs, disregarding the execution order proposed by the SH. This allows for optimised performance when the system is functioning correctly and the ability of the systems to handle errors in an agile manner.

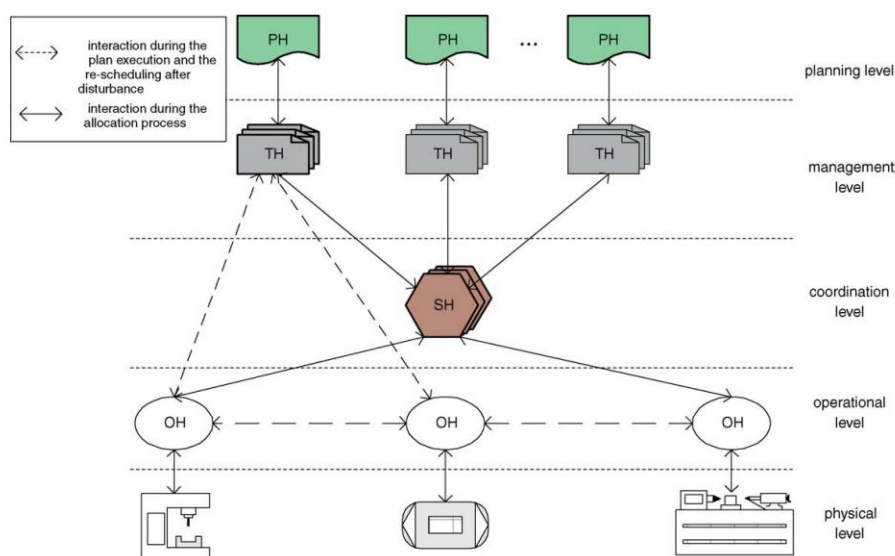


Figure 3: ADACOR holon classes (Leitao & Restivo, 2005).

A generic ADACOR holon is modelled by a logical control device (LCD) and an optional physical resource. Figure 4 illustrates this model. The LCD consists of three main components, namely the communication component (ComC), decision component (DecC) and physical interface component (PIC). The ComC is responsible for inter-holon communication and requires a common vocabulary. The DecC is responsible for decision making, which includes scheduling, planning processes and execution, as well as adaption to emergence of disturbances. The PIC accesses the physical manufacturing resource through communication to the virtual resource. The virtual resource could be running on a distinct platform. When comparing the models of Leitao & Restivo (2005) and Vrba, *et al.* (2011) for the holon, it would seem that the LCD, containing the ComC, DecC and PIC, are contained within the HLC. The virtual resource is then the LLC.

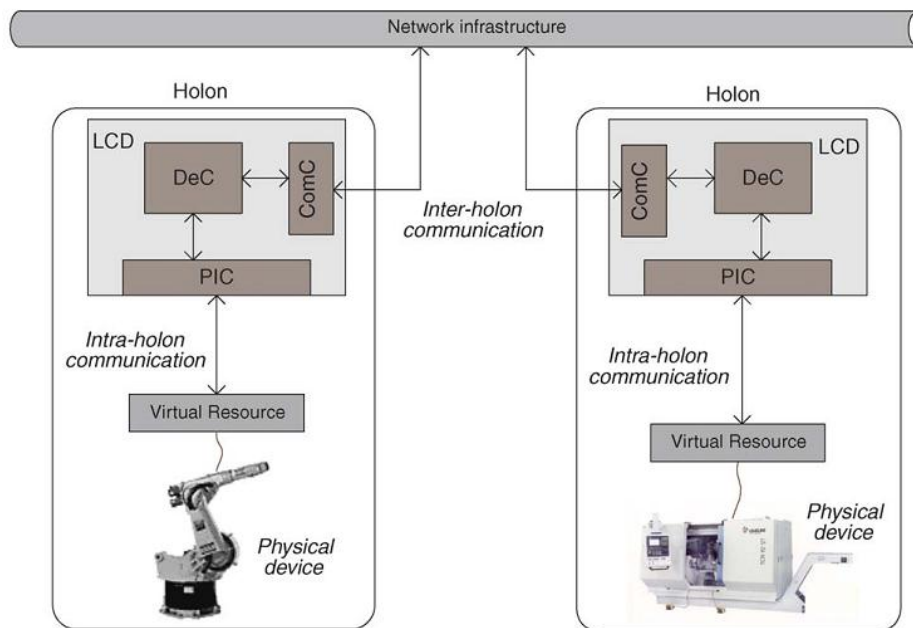


Figure 4: Conceptual model for an ADACOR holon (Leitao & Restivo, 2005).

2.3. JADE

Agent orientated programming (AOP) is a software paradigm which consists of multiple software entities, called agents, which function together as a distributed software system. These agents are characterized by autonomy and the ability to communicate, among other things (Bellifemine et al., 2004, p.1). AOP seems to be the preferred method of implementation for distributed manufacturing systems in an academic environment. It was implemented by Borangiu (2009), Leitao & Restivo (2005) and Vrba, *et al.* (2011).

The controller implementations for EIH and FC configurations influence the reconfigurability of the configuration. Therefore, the aspects of AOP salient to the research presented in this thesis, amongst the wealth of functionality provided by AOP platforms, are considered here.

Java Agent Development Environment (JADE) is software framework implemented in Java. JADE complies with the specifications of the Foundation for Intelligent Physical Agents (FIPA). The specifications describe, among other things, agent **communication**, agent **management** and agent **architecture** (Bellifemine et al., 2004, p.13).

2.3.1. Communication

JADE comes with a built in standardized messaging system in accordance with FIPA standards called FIPA-ACL (Agent Communication Language) or ACL.

ACL messages use performatives to represent actions or communicative acts. These performatives include inform, not understood, agree, failure, etc. ACL messages contain a number of fields, such as sender, performative, receiver, content, etc. The content slot of the message contains a string, which can be used to transport data using ACL messages. JADE ontologies, Java serialization or a XML parser are some options that may be used to embed data within the content slot/string. JADE ACL messages also have tools to ensure messages are not confused, wrongly received or misinterpreted. These tools include conversation IDs and ontologies.

FIPA specifies **interaction protocols** between agents. Two of the interaction protocols available in JADE are the Request Interaction Protocol (RI) and the Contract Net Interaction Protocol (CN). The RI protocol is used by one agent, the initiator, to request a service from another agent, the responder, while CN is used to negotiate obtaining a service from a number of other agents. These protocols are explained further below, with the behaviours used to implement them.

JADE has also been used extensively in implementation of holonic control systems for research purposes (Leitao & Restivo, 2005). Java is well suited for high-level control and communication, because it is an object orientated programming language, but it is limited in terms of low-level control/hardware communication.

2.3.2. Agent management

Agents inhabit an agent platform and can migrate from one JADE agent platform to another. These agent platforms can be running on distinct machines. The JADE message transport system allows for communication between these agent platforms.

JADE also includes the functionality of a directory facilitator agent which acts as a “yellow page directory”, supplying the names of agents offering a specific service. Agents publish their services with the directory facilitator and other agents can request services and will be informed of which agents currently supply these services. This allows for a non-deterministic system where non-critical agents can be added and removed with little consequence.

2.3.3. JADE behaviours

Each JADE agent runs in its own thread. Behaviours represent a task that can be executed by an agent and is implemented using an instance of a class. These behaviours provide a means of managing multiple tasks using one thread, this include communication and processing. Behaviour switching provides performance improvements over Java thread switching and eliminates thread synchronization issues.

Composite behaviours allow for a systematic approach to solving complex tasks. Composite behaviours are manifested in a hierarchical manner with parent and children behaviours, as shown in Figure 5. One parent (A) can have multiple

children (B and C) and a child behaviour can have its own children (D is a grandchild to A, but a child to B). All children share a path to their parent and, hence, root behaviour. This can be used to share data among sub behaviours, for example, data stored in A can be accessed/modified by D.

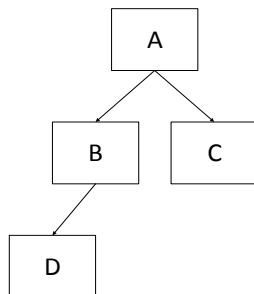


Figure 5: Composite behaviour example.

The two FIPA interaction protocols mentioned above, namely the Request Interaction Protocol (RP) and the Contract Net Interaction Protocol (CNP) are implemented in JADE behaviours.

2.3.3.1. Request interaction Protocol

The RP is a simple interaction, shown in the Figure 6 that allows one-on-one communication between agents. The initiator sends a *request* to the participant or responder to perform an action. The participant may respond with a *refuse* or accept the task, in which case an optional *agree* can be sent. The participant then informs the initiator of the completion of the task, replying with a *failure* if the task has failed or an *inform* if the task was completed successfully. The request interaction behaviours are called “AchieveReInitiator” and “AchieveREResponder” in JADE semantics.

2.3.3.2. Contract Net Protocol

The CNP is used when one agent requires a task performed by one or more other agents. The initiator sends a call for proposal (CFP) to any number of responders. The responders may respond with a *refuse* or *propose* the task, depending on whether or not they can perform the requested task. The proposal may contain a characteristic of the task for example cost or time. This would be used to determine which of the proposals are sufficient and respond appropriately, with either a *reject-proposal* or *accept-proposal*. The respondent(s) then respond with an *inform* if the task has been successfully completed or a *failure* if the task has failed.

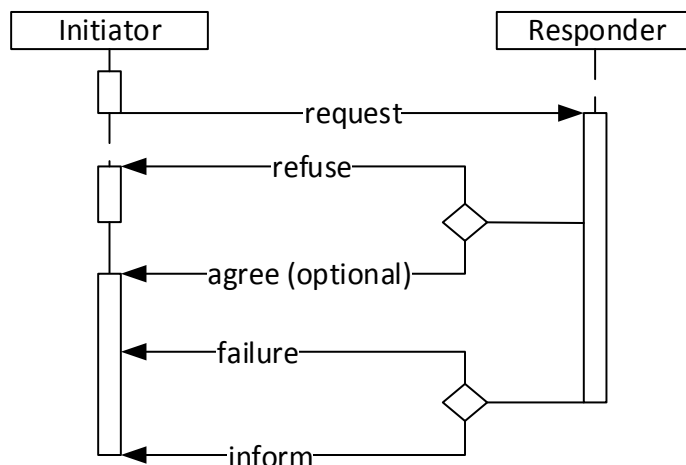


Figure 6: Request Interaction Protocol.

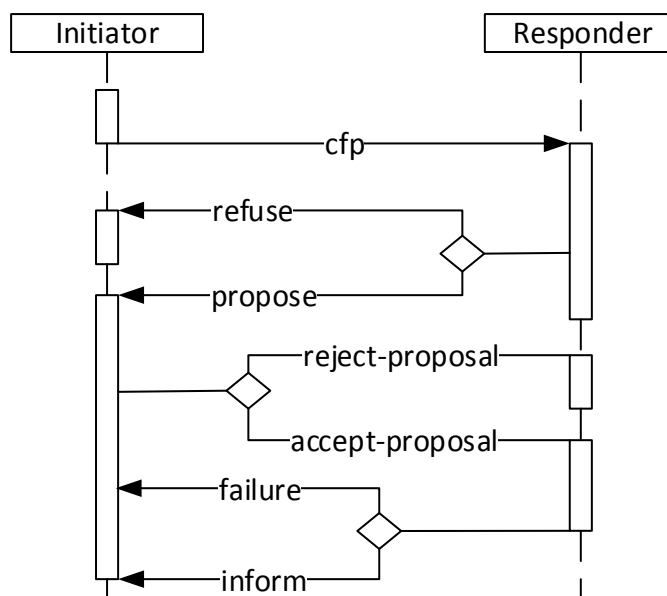


Figure 7: Contract Net Interaction Protocol.

2.4. Computer vision for control in automation

Computer vision is a versatile sensing technology used for many applications from high precision part inspection to autonomous vehicle navigation. Like other sensors, cameras are used to obtain information about the system environment.

A vision system can consist of multiple cameras (stereo vision system). In some cases, for example Kermorgant & Chaumette (2011), additional cameras have been found to improve system performance in terms of robot movement paths and robustness. Much research is being conducted into the development of new

algorithms in vision control that address problems like pose estimation of moving parts and obstacle avoidance of automated robots.

The use of a vision system in assembly often involve many challenges, such as data acquisition, invariant object recognition and coordinate transformation, as well as configuration and integration into the robot environment (Herakovic, 2009, p.505). Robots are largely being used in the manufacturing world to replace manual labour in tasks, which are both repetitive in nature and dangerous to humans (Khan et al., 2009).

Traditional “teach repeat automation” use costly fixtures and have long ramp-up/setup times (Zhang et al., 2011). Parts are usually presented to the robot in a known orientation using fixtures or simple sensors. If this cannot be done, the part is handled by a human operator (Jones & Hage, 1985). A vision system can be used to overcome this problem, calculating the position and orientation of the part relative to robot, making this a highly flexible system, suitable for both fixtureless applications and applications that lack precise consistency (Jones & Hage, 1985). Vision systems could further benefit an automated system by reducing the need for costly fixtures, tooling and material handling mechanisms by providing increased accuracy to robotic manipulators in less constrained environments (Vyawahare & Afzulpurkar, 2006).

2.4.1. Image processing and lighting

Image processing is the extraction of information from a vision sensor. This could be part position, size, tolerance, presence, etc. Image processing may involve both simple low-level tasks and complex algorithms. The most simple and perhaps most well-known low-level image-processing task is segmentation, the process of separating an object from its background (Spong et al., 2006). More complex algorithms like the Cognex Patmax© algorithm can identify objects of variant scales, positions and orientations (Braggins, 2006). According to Braggins (2006) these algorithms are “practically immune to changes in lighting and contrast”, but experience gained in the research presented here cast doubts on this claim.

Lighting plays a large role in vision systems and image processing. Appropriate lighting conditions have been found to improve vision system performance in terms of both accuracy (Zorcolo et al., 2011) and image processing time. A variety of lighting methods are available and the appropriate one depends on the application, considering factors such as surface texture or finish, colour, translucency, geometry, etc. Large vision system suppliers offer off-the-shelf lighting solutions for various applications. Figure 8 illustrates the effect of poor lighting on segmentation: a and b show the results of segmentation for no externally applied lighting, c and d show the results of a poorly placed light source. Other factors that also affect vision system performance are sensor placement, calibration methods and camera lenses.

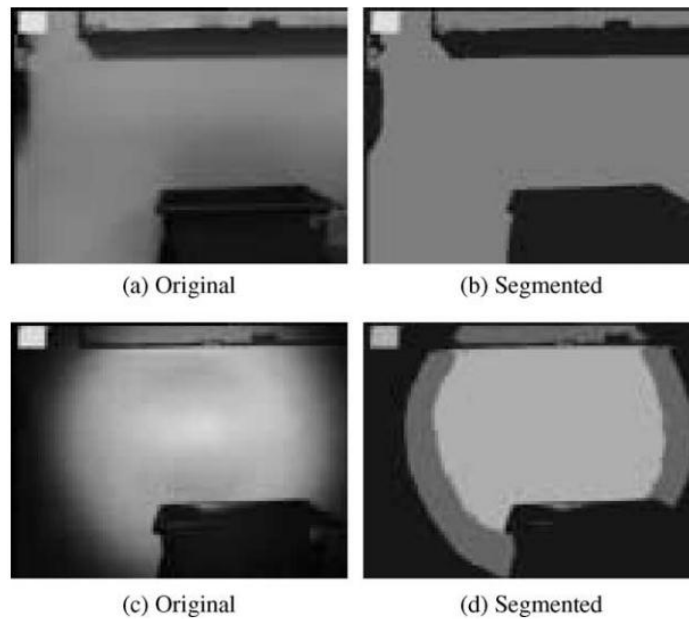


Figure 8: Effect of external lighting on segmentation (Kopparapu, 2006).

2.4.2. Vision based control

Vision based control is motion control, usually that of a manipulator, through information extracted from a vision system. The information obtained about the system environment by the cameras can be used to build up a partial or 3D view of an object (a CAD model) or the system environment. This view can then be used to control motion.

2.4.2.1. Open-loop control

Open loop control is a control method where the vision system and the handling system are used in sequence: after the vision system has acquired the required information, it does not interact with the handling system, and while objects are being handled, the vision system is not employed (Shirai & Inoue, 1973). Open-loop control relaxes real time constraints on image processing (Vyawahare & Afzulpurkar, 2006), allowing the use of a slower image processing system. As a result, it has been the preferred choice of vision-based motion control in the past. Figure 9 illustrates simple open loop visual control.

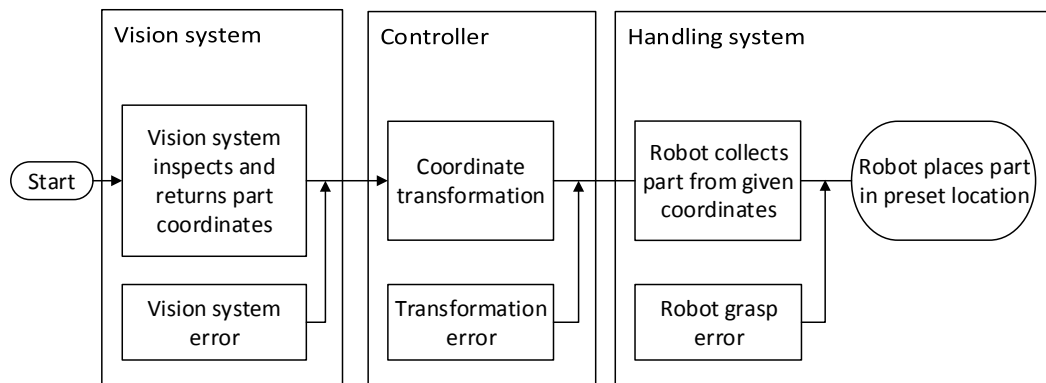


Figure 9: Example of open loop visual servoing.

Operational accuracy depends directly on errors of the visual input device; errors of digitization in the visual system; positioning errors in the handling system; and errors in coordinate transformation (Shirai & Inoue, 1973).

Open-loop control systems require accurate calibration to allow mapping of coordinates between the vision, robot and real world coordinates. In open-loop vision control systems the objects position is usually calculated and then a blind grasp is performed. Therefore, small errors in the object position can cause the grasp to fail (Vyawahare & Afzulpurkar, 2006). Grasping errors are also expected due to imperfections and wear and tear of the gripper (Zhang et al., 2011). These are particularly important considerations in applications with small parts and tight tolerances (Zhang et al., 2011). On the other hand, open-loop systems usually avoid camera blur, since the camera is stationary during image acquisition.

2.4.2.2. Closed loop or feedback control

Visual servoing (VS) is the control of a robot using feedback data extracted from a vision system. Considerable progress has been made since VS began in 1980s, showing steady progress in areas such as control theory, real-time computations, high speed image processing as well as kinematics and dynamics (Gascon & Barraza, 2012).

The “look and move” approach is often used in feedback control. In this approach, the image processing and robot motion are done in sequential repetition. Image processing is often computationally expensive and often result in the vision control loop having longer sampling periods than the motion control (Oda et al., 2009), which could affect system performance. Many methods of tracking delay suppression are being used including: multi-rate control; performing image processing in parallel with robot control; assuming constant velocity and imposing constant velocity (Scaggs, 1993). Feedback vision systems can also be used to track an object with uncertain motion (Scaggs, 1993).

The errors occurring in an open loop controller can be compensated with a visual feedback loop detecting the difference between actual position and desired position

(Shirai & Inoue, 1973). This includes compensation for grasping errors as shown by Zhang *et al.* (2011). Closed-loop control systems have also been shown to be suitable for automated placement of small parts with tight tolerances by Zhang *et al.* (2011). Further, the visual feedback loop can increase flexibility and dexterity of robot tasks (Oda *et al.*, 2009), making them suitable for the tracking of moving objects and force feedback.

Visual servoing does not take advantage of the repeatability of commercial robots (Zhang *et al.*, 2011), but can be used in a non-calibrated environment, as shown by Zhang *et al.* (2011), Piepmeier *et al.* (2002) and Vyawahare & Afzulpurkar (2006), making it suitable for applications where thermal expansion is considerable. Visual servoing requires access to the terminal error between the target and current pose at regular intervals and this can be overly demanding in terms of high speed image processing (Zhang *et al.*, 2011 [2]).

2.4.3. Vision system physical configurations

There are three main configurations in vision systems namely (1) eye-in-hand, (2) fixed camera and (3) hybrid configurations, each having their own advantages and disadvantages.

In a **fixed camera** configuration, the camera is placed in a static position, usually overhead. The advantages of this approach are that, the geometric relationship between the camera and the workspace is constant and can be calibrated offline (Spong *et al.*, 2006). The fixed camera position also allows for an external lighting design as done by Kopparapu (2006) and this can enhance processing times as less processor cycles are used to make vision algorithms more robust. A disadvantage of this configuration is that the manipulator can block the field of view of the camera when moving through the workspace (Spong *et al.*, 2006). The camera, in its fixed position, can hinder manipulator movement. The line-of-sight angles, which occur when the line of sight is not perpendicular to the plane being viewed, cause difficulty for static vision systems although technology like Cognex non-linear calibration claims the ability to accommodate mounting angles up to 45 degrees (Cognex Corporation, 2013). However, experience gained in the research presented here indicates that, for complex part shapes, smaller deviations from perpendicular are required for acceptable accuracy.

In **eye-in-hand** configurations, the camera is fixed to the manipulator. The camera can be fixed above the wrist so that wrist motion does not affect camera motion and the camera can observe motion of the end effector at a fixed resolution (Spong *et al.*, 2006). In this configuration, the manipulator does not occlude the camera's view. A disadvantage of this configuration is that a small movement of the manipulator can drastically change the field of view of the camera. Another disadvantage of this configuration is that the geometry of the workspace also changes as the manipulator moves. A concern for this configuration is the

requirement of an external lighting source in this configuration, as a badly placed external lighting source can do more harm than good as show in Figure 8.

Hybrid systems combine these two configurations, containing both a FC and EIH sensor. These systems benefit and suffer from the respective advantages and disadvantages of both configurations. Hybrid configuration performance has not been tested extensively and therefore cannot be accurately compared to the other configurations at this time.

3. Case study

The reconfigurable automated assembly cell mentioned in the introduction was used as context for the research presented here.

3.1. Reconfigurable assembly cell overview

A reconfigurable assembly cell was developed to assemble a subassembly of a family of circuit breakers for CBI Electric: Low Voltage. The family extending to single, double and triple pole breakers. The assembly cell consists of five main subsystems, namely a storage station, welding station, feeding station and the transport subsystem (or conveyor). These subsystems are illustrated in Figure 10.

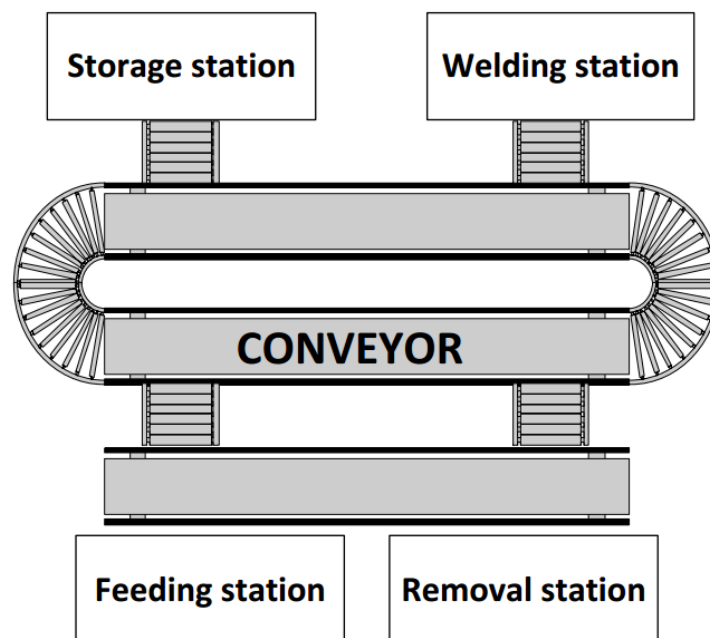


Figure 10: Reconfigurable assembly cell developed by RMS research group (Kruger, 2013).

The transport system consists of a modular conveyor on which pallets move. These pallets have modular fixtures where components are placed. Figure 11 illustrates a pallet with components in the fixtures. Pallets enter the stations, where operations are performed on them before moving to the next station. For example, pallets enter the feeding station for the required components to be placed in the fixtures on the pallet by a pick-n-place robot. Thereafter the pallets can move to the next stage of assembly.

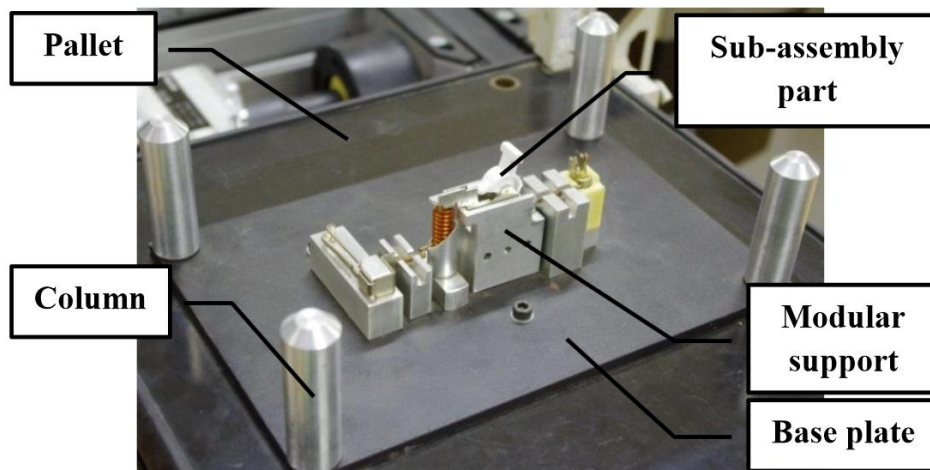


Figure 11: Photograph of Pallet with components in the fixtures (Kruger, 2013, p.29).

The assembly cell control system uses a hierarchical, holonic architecture, where each station or substation can be considered a holon, as illustrated in Figure 12. Note that not all subsystem level controllers are shown and that the refinement of only the feeder controller is shown.

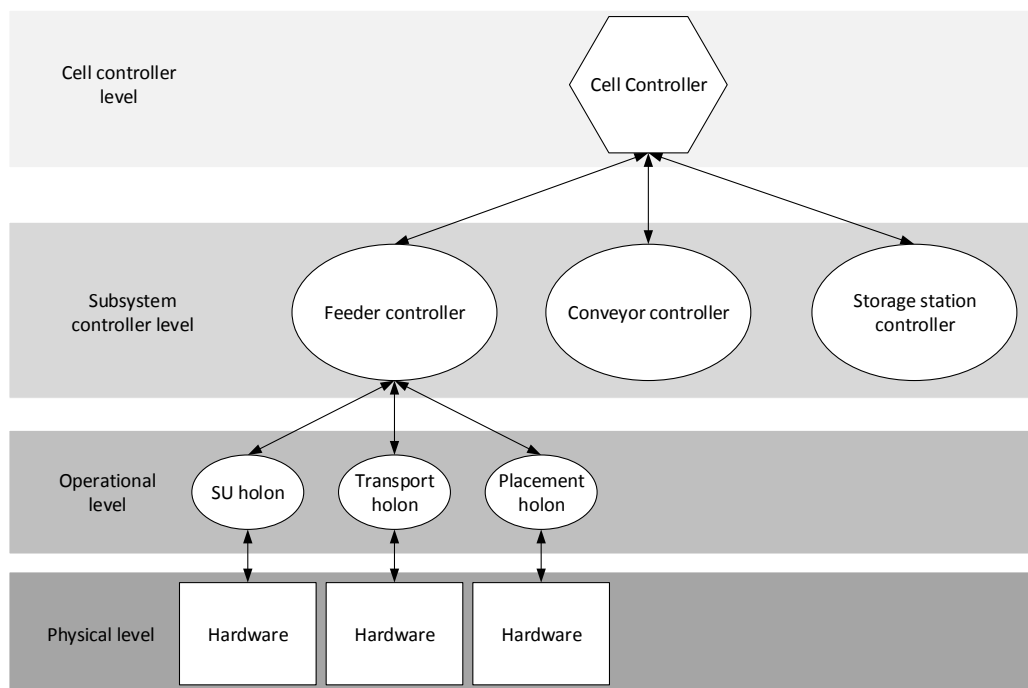


Figure 12: Cell controller holarchy.

3.2. Feeding station

The feeder subsystem was selected for this case study for the research presented in this thesis because it relies on a vision system to identify multiple poses of a part and could reasonably use either a FC or an EIH configuration. The best configuration is expected to depend on a number of parameters encountered in the station.

3.2.1. General configuration

The feeder station is responsible for singulating and feeding components. Figure 13 shows a schematic layout of the feeding station.

Components are supplied in bulk to the singulation units (SUs). These SUs are responsible for singulating components and presenting them in a known pose (position and orientation) to the pick-and-place robot. The pick-n-place robot collects the component and places it in fixtures on the pallet that awaits its arrival on the conveyor. The schematic layout shows a short sector of the conveyor, parallel to the main conveyor circuit, which is dedicated to the feeding station.

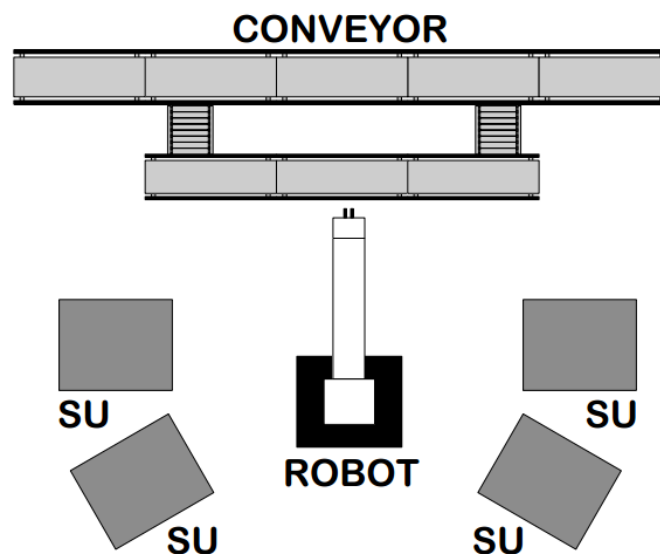


Figure 13: Schematic layout of the feeder subsystem (Kruger, 2013, p.22).

In accordance to the ADACOR holonic architecture, as described in Section 2.2.2, the physical devices of the station are considered part of the holons. Since the devices and operations are inextricably linked, the following paragraphs explain the functioning of the feeding station in terms of the different types of operational holons and their respective functions are as follows.

The **SU holon** is responsible for presenting singulated components to the robot in a known collectable orientation. Different methods of singulation exist, but the methods commonly used in industry, for example vibratory bowl feeders, are not well suited to reconfigurable systems. The Mechatronics, Automation and Design Research Group is therefore developing two SU concepts i.e. one based on a tumbling barrel and the other on a stepped conveyor (Kruger, 2013). These SUs operate by employing a series of controlled falls designed to separate and untangle components.

To illustrate the operation of these SUs, consider the stepped conveyor SU shown in Figure 14. Components are placed in the input bin (bulk bin) and a stepped conveyor scoops up individual components. Parts are transported up the conveyor, as shown by the arrow, and when at the top parts are dropped into a gateway mechanism. The mechanism channels components to be rejected back to the bulk bin or to be presented on the inspection platform. Components on the platform are inspected by the camera to evaluate whether these components are collectable or not. If so, the platform is raised and the component is collected by a robot arm. If not the platform is lowered and the component is sent back to the bulk bin.

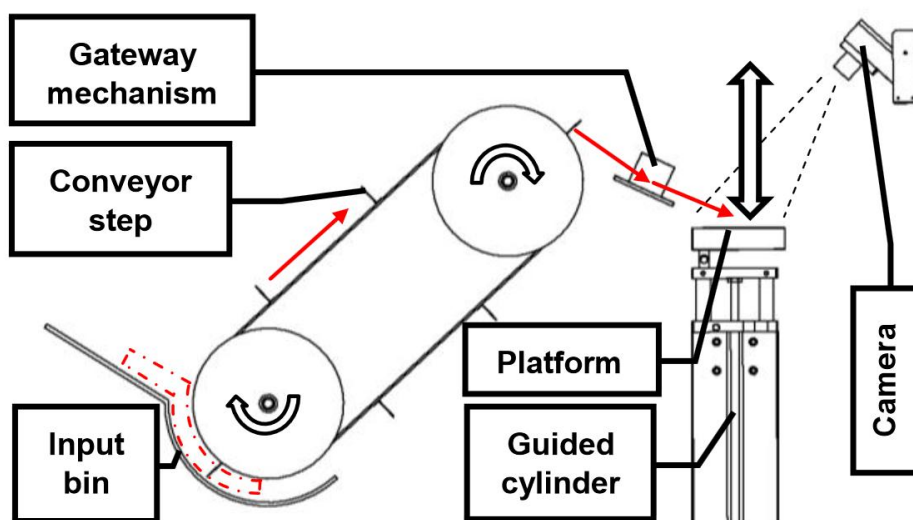


Figure 14: The operation of a stepped conveyor (Kruger & Basson, 2014).

Since components land in a random pose on a collection platform, a vision system is used to identify whether the component can or cannot be collected by the robot in its current pose. If the component pose is such that the robot cannot collect it, the SU returns the component to the bulk bin and it is recirculated. If however, the component can be collected, the vision system determines the location of the component on the collection platform and communicates that information to the control system. The control system then directs the pick-n-place robot to collect the component from the collection platform and place it in a fixture on the conveyors pallet. Through Kruger's (2013) research, it became evident that the probability of

successful singulation and feeding rate are dependent on the type of component and the rate at which components are fed.

Manual labour is also a means of singulating components, in which case components are manually placed in fixtures in component magazines. Since singulation units and part magazines supply the same service in the feeder station, both are considered SU holons, regardless of the method of singulation. Figure 15 illustrates three full component magazines.

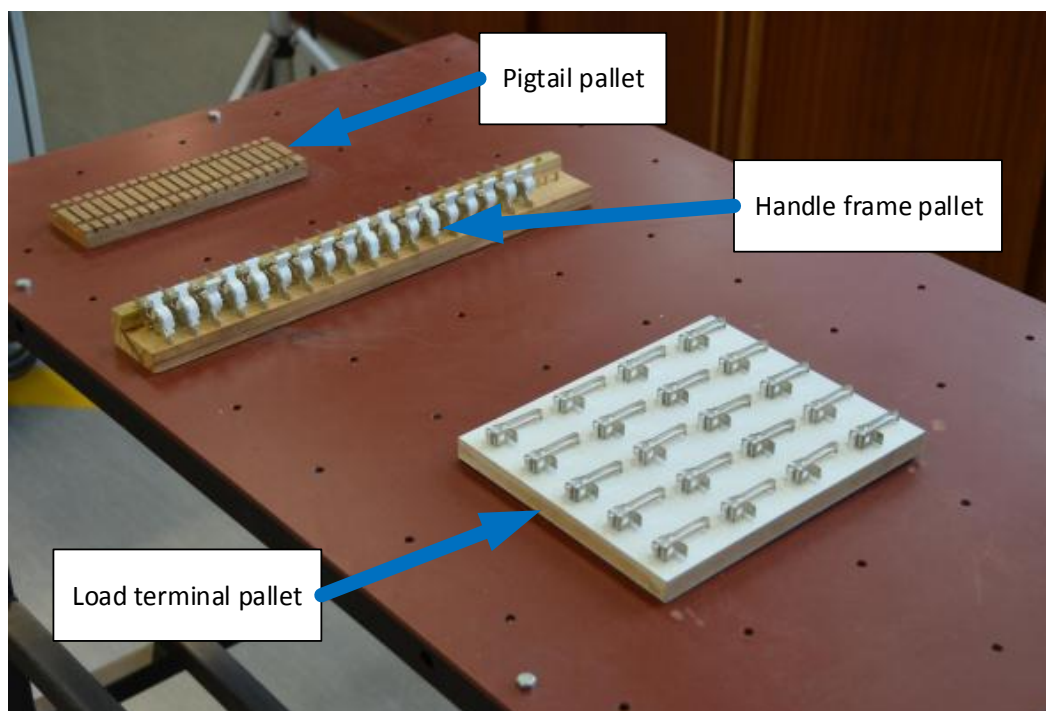


Figure 15: Three component magazines.

The **transport holon** is responsible for transporting components from SUs to the fixtures on the pallet. This service is provided by a “pick-n-place” robot. Since all the parts fed by the feeding station are handled by the single pick-‘n-place and this robot is likely to be the most expensive device in the subsystem, the robot should be the device that determines the subsystems throughput. Ideally, the pick-‘n-place robot should not have to wait for a singulation unit to have a component ready for collection.

3.2.2. Fixed camera configuration

In this configuration, each SU, part magazine and placement station has its own fixed-camera. This implies that each SU requires its own vision system, increasing the cost of each SU substantially. On the other hand, SUs will operate independently

of the pick-‘n-place robot and will be able to singulate and inspect without the use of the robot. This can reduce the time the robot will have to wait between successive feeds. Therefore, this configuration is expected to have a higher throughput rate when singulation units have a low probability of parts landing in collectable orientation combined with a high feeding rate.

3.2.3. Eye-in-hand configuration

In the eye-in-hand configuration, the camera is fixed to the end effector of the robot. Therefore one vision system per pick-‘n-place robot will be required for the feeder station and this reduces the cost of the station. Additionally, the eye-in-hand robot can be used as a remote inspection device.

Since the singulation units require the EIH robot to identify whether parts are in a collectable orientation, the robot must move to each singulation units’ platform to identify whether components are collectable. The eye-in-hand robot may have to inspect multiple times before finding a collectable part. Therefore, this configuration is expected to have a low throughput rate in situations where the probability of successful singulation (SS) is low but be cost effective when a high rate of SS occurs.

3.2.4. Subsystem control design requirements

For this case study, the control of the conveyor system will not be considered. It is assumed that the cell controller or another controller will be responsible for the transport of pallets to and from the feeding station.

The design requirements for the controller of the feeding subsystem that are relevant to the research presented here are:

- The vision system should identify parts and their coordinates to an accuracy that allows the robot to successfully collect and place components in the appropriate fixture.
- The feeding station and its holons should allow for ease of reconfiguration.
- It was expected that the robot would be the limiting resource in both configurations being tested and therefore the robot idle time should be kept low.

4. Control architecture

This chapter describes the control architecture used in the research presented in this thesis, i.e. the control of a vision-aided robot for component feeding in a reconfigurable manufacturing system.

4.1. Reference architecture selection

ADACOR was selected as the reference architecture, because it was developed for concurrent and asynchronous processes making it a good choice for this system, as singulation units operate in this manner (the singulation operations are both concurrent and asynchronous). However, the system implemented in this thesis occurs at a lower level than typical ADACOR applications and therefore this system will not implement some of ADACOR's features, in particular disturbance handling by the task holon. A few further adaptations were made to the reference architecture for implementation in the feeder station and are described in the following sections.

In this implementation, services are used at different stages of production. Two holons can represent physically different machines, but supply the same services. For example, two different robots can transport the same component type and therefore their holons appear to the other holons to be the same. This allows a level of modularity as hardware and software can be replaced, but if the corresponding holons supply the same service, the change will not affect the remainder of the system.

ADACOR's generic holon types are the supervisor, task, product, operational and staff holons. Their roles are summarised in Section 2.2.2. In this case study, the control architecture includes a supervisor holon, one or more task holons (determined by the number of pallets), subtask holons (explained in Section 4.5) and three types of operational holons (namely SU, transport and quality inspection holons). These holons will communicate in a bidding procedure for resources in the form of services. The feeding station controller does not contain product holons, since its task holons are launched using the product information received from the cell controller. The following sections give more detail about the holon types.

4.2. Coordinator Holon

The coordinator holon is an implementation of ADACOR's staff holon. It is responsible for communication with the cell controller, commissioning of new tasks and logging of information. The coordinator holon commissions new task holons based on information communicated from the cell controller. It also logs global events such as the launching and completion of tasks.

4.3. Supervisor Holon

The supervisor holon is responsible for optimization in its coordinated holon group in the ADACOR architecture and in the research presented here, the supervisor holon coordinates the subtask holons and OHs, by proposing the execution order to the subtask holons to optimize the production schedule and throughput.

In this thesis, the execution order is implemented by means of a booking system, whereby the supervisor holon reserves the use of OHs for a specific subtask. Formal optimization may be required in larger systems, but would have little benefit in the current system, since the robot is expected to be the limiting resource. The supervisor holon is responsible for booking the list of services (supplied by OHs) needed to execute the completion of a subtask (or feeding of a single component). Booking of OHs ensures that all services are available before a subtask begins. This also ensures OHs cannot be double booked, thus avoiding confusion about which subtask can employ an OH.

4.4. Task holon

According to the ADACOR architecture, the task holons (THs) are responsible for managing execution and storage of dynamic information in each production order launched to the floor. This is usually applicable to a shop floor level and not applicable to such a low level of the manufacturing system as is being considered in the feeder station. Therefore, task management in this implementation deviates from that specified in ADACOR.

Each task holon in this implementation is associated with a pallet. Hence, a task is launched for every pallet entering the feeding station. The task possesses product information in the form of components to be fed, components already fed, component placement positions relative to the pallet and the current pallet position. It does, however, not drive production as in the case of the ADACOR task holon. Task holons sub-contract subtask holons to feed specific parts, with each subtask holon representing the process of feeding a single component. The subtasks are therefore responsible for driving the production.

4.5. Subtask Holon

Subtask holons are associated with components in the feeder. They are responsible for feeding a single component and one subtask holon exists for every component type in the feeder cell. Task holons employ or contract the sub-task holons to feed their respective parts. The subtask holon is responsible for driving production by means of feeding a component. This includes subcontracting the supervisor holon (for booking), execution and unbooking of operational holons. To the task holon, however, the subtask holon appears to be an OH as it is launched at start-up and is employed as a resource to perform a specific action.

4.6. Operational holons

As introduced in Section 3.2.1, operational holons are associated with the physical devices. To summarise the discussion there:

SU (Operational) holons are responsible for singulating and supplying components with their collection pose.

Transport (Operational) holons are employed to transport components from the SU to the Pallet. In the eye-in-hand configuration, this holon is expanded to have the additional functions of inspection, i.e. firstly, inspecting whether components are collectable from SU's, and secondly, the quality inspection described below.

Quality Inspection (Operational) holons perform quality checks, e.g. inspecting the complete pallet for defects in the form of missing or incorrectly placed components.

4.7. Operation flow

The sequence of events during production is explained here with the aid of the flow diagram in Figure 16. This process applies to both the FC and the EIH configurations.

1. The cell controller informs the feeding station's coordinator holon when a pallet arrives at the feeder substation. The cell controller also communicates the appropriate product information of the specific task, including pallet position, components needed and component placement positions.
2. The coordinator holon then uses this product information to launch the appropriate task holon. Multiple tasks can exist (implying multiple pallets are present), within the feeding station at any stage.
3. Task holons then employ subtask holons to feed components. One subtask exists for every component in the feeder station and a subtask can only negotiate with one task holon at a time. The subtask holons handle requests from the task holons on a "first-in-first-out" basis.
4. The employed subtask holon then contacts the supervisor holon in an attempt to book the list of operational holons (further referred to as the **service list**) required for feeding of the specific component-type. The service list contains (1) a SU holon that can supply the required component, (2) a transport holon to transport the component from the SU to the pallet and (3) a quality inspection holon. It should be noted that in the EIH configuration the transport holon performs the post placement inspection.
5. The supervisor holon will book the OHs if all the services required to place a component are available. If one or more required services are not available, the

supervisor holon will not book any of the OHs. Instead, it will inform the subtask and the subtask will relay this message to the task holon. The task holon then re-requests the service from the subtask holon restarting the negotiation. Supervisor and subtask holons process requests on a “first-in-first-out” basis, thereby allowing the next task its turn to request a booking.

6. Once the OHs are booked, the supervisor informs the subtask holon and passes the service list to the subtask holon.

7. The subtask holon then sequentially requests the OH to perform the services previously booked.

8. Once all the services have been successfully completed, the subtask holon informs the task holon of successful completion.

9. The “unbooking phase”, described below, is then performed by the subtask holon.

10. The task holon then either begins the process to feed the next component or informs the supervisor holon that all the required components have been fed successfully.

11. When all the required components have been fed, the supervisor holon destroys the task holon and informs the cell controller that the feeding station completed the cell controller’s request.

Steps 4 to 9 above are grouped into 3 phases, namely booking, execution and unbooking. Figure 17 illustrates the associated holon interactions during production for a particular component. Since the Coordinator holon does not take part in production, it is not shown in the figure. The **booking phase** comprises steps 4, 5, and 6 while 7 and 8 constitutes the **execution phase**. After the completion of the placing operation, the **unbooking phase** commences (step 9). In this final phase, the subtask holon informs the OHs that they can resume operation. This phase is necessary to inform the SU holons that the component previously singulated, has been collected and that they may resume singulation operations. The phase also allows OHs to be used more than once by the same subtask holon during the execution phase (e.g. in the EIH configuration, the transport also performs quality inspection).

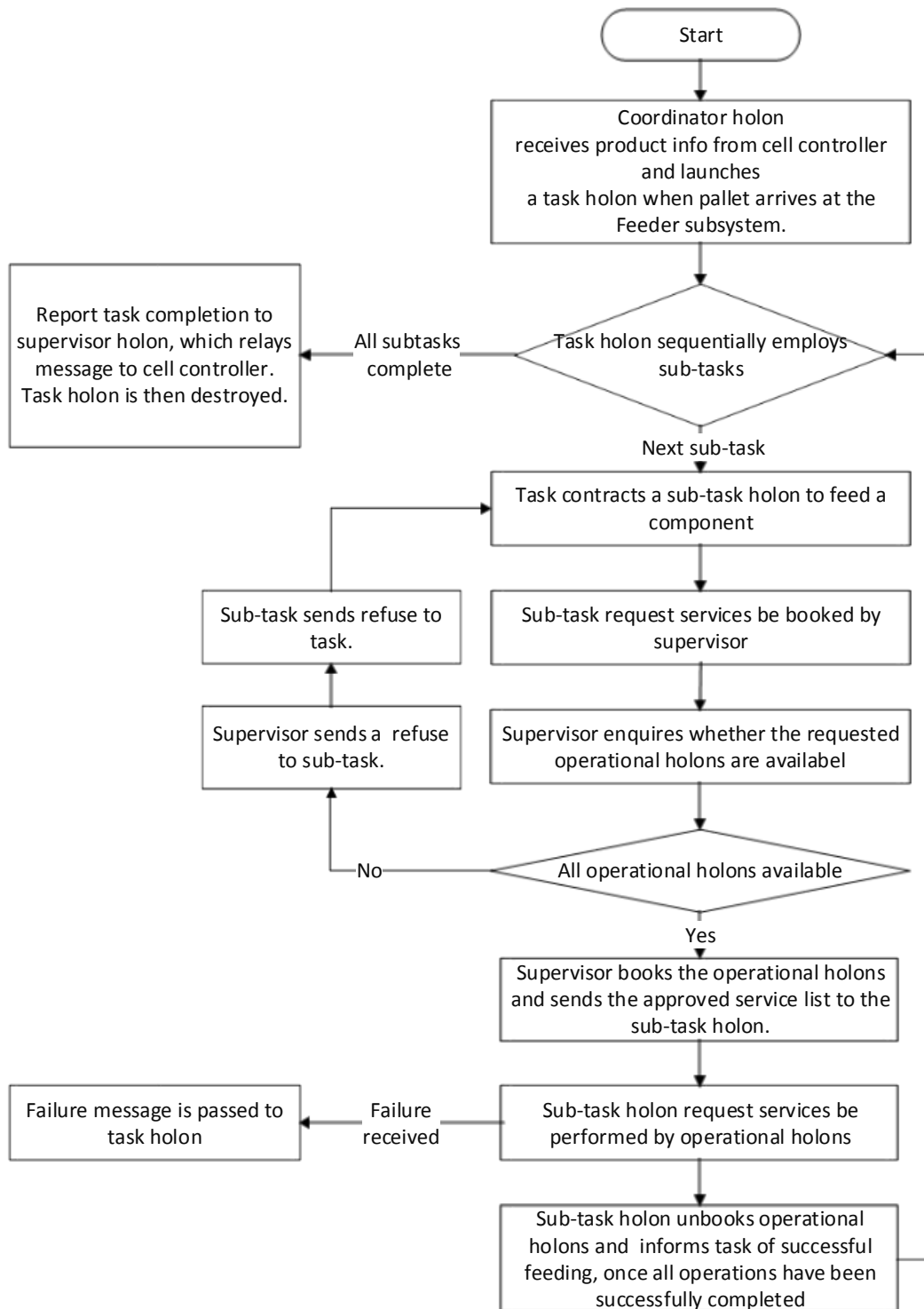


Figure 16: Flow diagram of operational sequence.

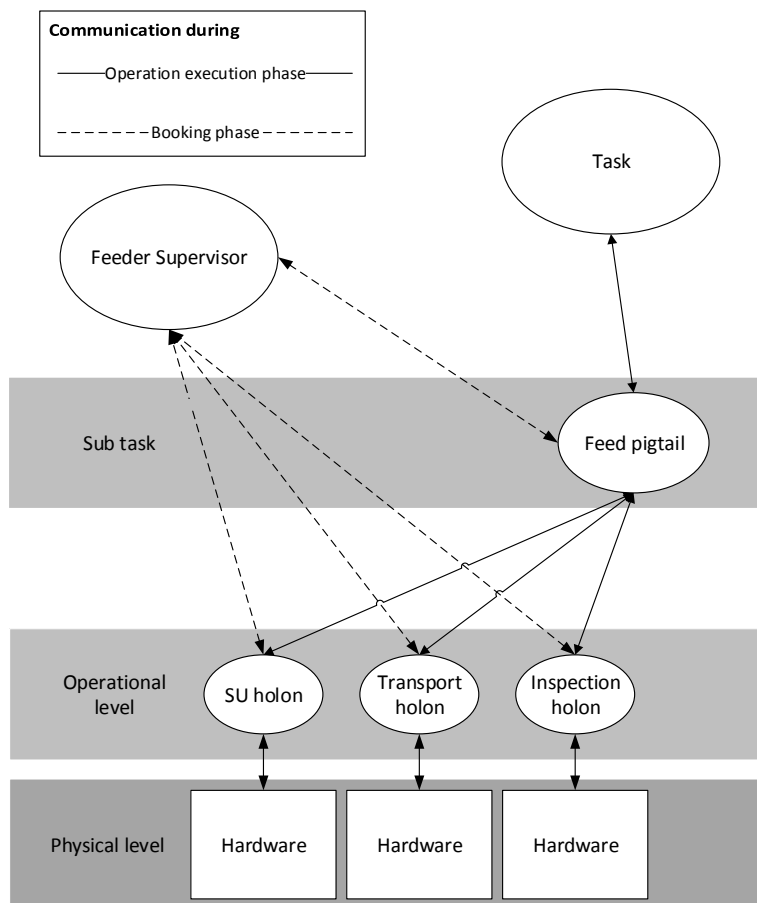


Figure 17: Holon communication production.

4.8. Scalability considerations

The research presented here assumes that there is only one pick-n-place robot in the feeding station. This assumption does still allow the evaluation of FC and EIH configurations, but does constrain the number of variations that need to be considered. Even though the assumption appears to limit the scalability (an important characteristic of RMSs) options, it is a reasonable limitation since the robot can be expected to be the most expensive single device in the feeder substation. Adding or removing feeding stations, each with one robot, to or from the cell is therefore a reasonable approach to achieve scalability.

The design of the feeding station does allow internal scalability by allowing the addition or removal of SUs, and by the use of one or more pallets in the feeding station simultaneously, as illustrated in Figure 18.

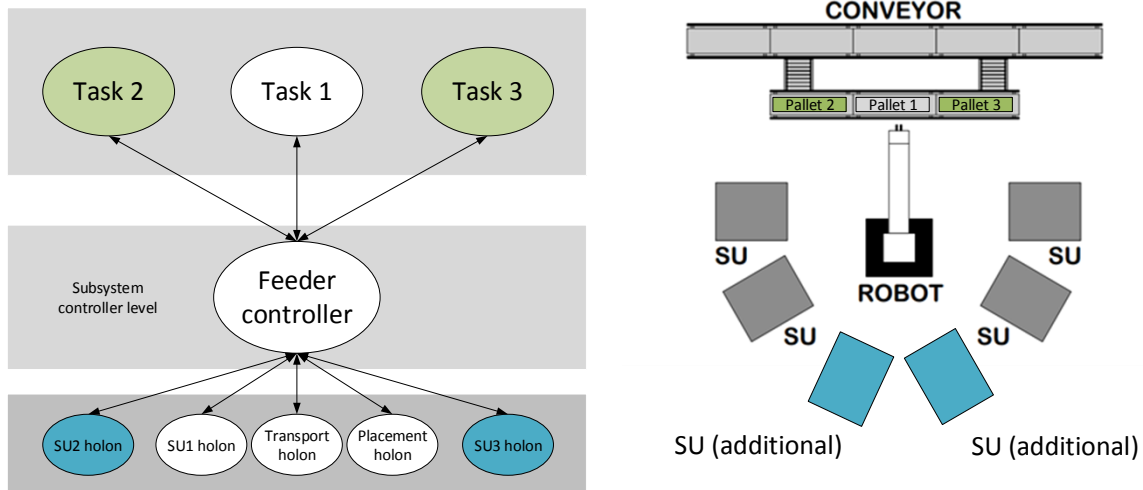


Figure 18: Scalability illustration.

Scalability was therefore implemented in terms of a variable number of task holons, which allows multiple pallets to be within the feeding station at the same time. The task holons can negotiate for resources simultaneously. Two immediate advantages of this approach are apparent. Firstly, one pallet's work could be continued while another is waiting for a component to become available at a SU. Secondly, by having multiple pallets waiting in the feeder substation, the time taken by the pallet to move into and out of the feeding station need not be added directly to the overall cycle time.

Scalability was also implemented on the OH level by allowing the addition and removal of SU holons. The immediate advantage of this is that additional SUs can be added, which increases the chance of a component being available for feeding.

4.9. Storage of component position coordinates

An architectural choice had to be made on how the position coordinates (component collection, placement and pallet position) would be handled. Two choices were considered, i.e. keeping this information in the robot's controller or in the station controller.

The conventional (i.e. non-RMS) approach would be to store these position coordinates in the robot's controller program. This would entail storing the component placement positions for each different configuration of pallet. The KUKA KRC controller, used in in this case study, allowed for the storing of 23 coordinate "bases" in the built in memory and more could be added in the executed program. This approach would mean that when a new product or pallet type is introduced, its position information has to be included in the program data on the robot controller. The station controller would also need to be reconfigured to add

the functionality to refer to the newly added coordinate positions. This would increase reconfiguration time.

Storing the position coordinates in the robot controller, however, could increase performance since less information has to be communicated to the robot. The robot could then also likely communicate directly with the vision system, allowing an industrial communication protocol to be used, which could further increase performance and robustness.

In the alternative route, the component placement and machine position coordinates are stored in the station controller. In other words, the task holon would contain the pallet and component placement positions relative to the robot's reference position. This means that, if the SU and pallet position coordinates relative to the robot position are precisely known, reconfiguration can be done on the station controller level alone, thus reducing the need for multiple robot program reconfigurations. Positioning machines with such accuracy is, however, not practical at this stage and calibration of machine positioning was used.

In this case, the robot control program is generic, and will not require modification with reconfiguration of the pallets or SUs or the addition of new products, as long as the base reference positions remain unchanged.

4.10. EIH open loop vs close loop

Open loop and closed loop vision based control (described in Section 2.4.2) were considered for the control system of the EIH configurations. Table 1 shows the advantages and disadvantages of each approach. An open loop control system was selected for this implementation because it could be expected to yield better performance from the robot, which is a high priority resource. The control system was developed assuming a well-structured environment with little effects such as temperature fluctuations and machine reference position (SU position or pallet position) shifting during operation. Hardware adaptations and additions would likely occur during system downtime in a system reconfiguration. The environment is also not expected to have substantial temperature fluctuations that will affect the robot accuracy. An open loop control system does ease time constraints on the vision system used, noting the vision system performance was not evaluated for a closed loop vision controller.

An important point not shown in the table is that the open loop vision control is easily adapted from the FC system, in which the same inspection routine can be used if the calibration is adjusted. This eased the adaption of the feeding station control system from FC to EIH.

Table 1: Open loop vs. closed loop vision based control.

Open Loop	Closed Loop
-Blind grasping errors	+Can account for grasp errors and some disturbances
+High speed, taking advantage of commercial robot repeatability	- The image processing sampling period could be larger than the robot-sampling period. This may slow the robot movement.
-Requires calibration, usually timely and costly, although automated calibration are available	+ Less setup/ reconfiguration time
-Environments must be structured. Reference points are used for coordinate transformation, these reference points must not shift as slight changes due to factors like temperature fluctuation can result in a mis-grasp. Recalibration is required when some disturbances occur.	- Requires high speed image processing (technology may not be available, therefore limiting robot movement speed)
	+No need for calibration, temperature effects are also compensated

5. Implementation

This chapter describes the implementation of the feeder control system. It describes the choices made, the detailed structure of holons and hardware used. Error handling was not considered in detail in this implementation since errors may have multiple solutions, being recoverable, critical to holon operation, critical to station operation and some self-curable. Instead, error messages would be passed up to task level and even to cell controller level, where a decision can be made on how to proceed.

In the EIH configuration, the EIH transport holon performs inspections for EIH SU holons and also transports components, while in the FC configuration, FC SUs singulate components autonomously. This allows the robot in FC configurations (or FC transport holon) to spend more time transporting components. Therefore, it is expected that the FC configuration will have a lower feeding time and the SU performance characteristics would influence the performance of FC and EIH configurations differently. It should also be noted that a FC SU holon costs considerably more than an EIH SU holon, since each FC SU will require its own vision system.

5.1. JADE as the HLC Software platform

The JADE software platform was selected to implement the HLC of the holonic control system. The software platform supports multi-threading, asynchronous communication and dynamic instantiation out of the box. Agent orientated programming and JADE specifically, was selected for the use of the high-level holonic control in this subsystem for the following reasons:

- JADE ACL messages simplify communication between holons allowing a simple means of asynchronous communication, the embedding of data within ACL messages and the use of out of the box conversation/communication protocols. These were used as the communication component (as described in Section 2.2.2).
- Each agent runs in its own thread and, therefore, concurrency is easily achieved. Java also allows for manipulation of threads. The decision component (as described in Section 2.2.2) of holons was implemented in Java.
- JADE was used in the original implementation of ADACOR by Leitao & Restivo (2005) and has been used extensively in research for multi-agent manufacturing systems.
- Further functionalities of JADE are well suited for multi-agent manufacturing systems. These include the use of services, and the directory facilitator to find services.
- The JADE directory facilitator allows for ease of reconfigurations since every agent does not need to know the current state of the system (which agents are available or not). This adds to the system's scalability, since

agents can be added with no modification to other agents, and robustness, as non-critical agents that malfunction can be removed at runtime without the system stalling.

- Java sockets were used for communication with LLCs. This provides a flexible and modular physical interface that was achieved using physical Ethernet connections.

Since JADE agents can fulfil the role of the logical control device mentioned in Section 2.2.2, they were selected as the feeder station controller.

5.2. Coordinator holon

5.2.1. Architecture

The coordinator holon is responsible for commissioning task holons, based on their specific product type. This would normally be done using information received from the cell controller. The cell controller was, however, not implemented in this case study and the coordinator holon therefore launched task holons at the user's discretion. The coordinator holon also logs information within the feeder station. The coordinator holon consists only of a coordinator agent.

5.2.1. Behaviours and lifetime

The coordinator agent has little to do with production, allowing it to be a thread used exclusively for logging and external communication.

The coordinator agent launches the other agents at start-up and shuts down the JADE platform, informing other agents to perform the takedown sequence, which, is used for shutting down holons and their external communication. It has also been given the ability to destroy agents if need be, for example if an agent is not responding or needs to be shut down externally while keeping the system running, but this was not implemented.

The coordinator has a task log listener behaviour, which is an extension of the JADE “AchieveREResponder” behaviour and uses the task log ontology. The task log listener receives a request when a task has been completed. It then logs the event.

5.3. Task holon

5.3.1. Architecture

The task holon manages the feeding of components for one pallet. Upon launch, it contains the product information of its specific pallet task. The product information for a pallet refers to the component-types and component placement positions

(relative to pallet) of each component to be fed. One task holon exists for every pallet inside the feeder substation.

The task holon comprises of three entities, as shown in Figure 19, with the first being the physical pallet shown in Figure 11. The pallets have modular fixtures in which components are placed at the feeding station. Upon the pallets' arrival at the feeding station, a lifting mechanism on the conveyor ensures they are kept in a repeatable position for the duration of the feeding process.

The second entity that makes up the task holon is the pallet agent. It represents the intelligent and decision making component of the holon. It drives production indirectly. The pallet agent also manages the third entity in the task holon, i.e. the product information.

The product information of the pallet refers to the list of components to be fed and the placement position of these components. The pallet agent stores the product information using two data structures. The first data structure is the list of subtask services, specifying the component types, for example "Feed Load terminal, Feed Pigtail". This list is used to schedule a series of conversational behaviours that will employ subtasks.

The second data structure is the part data. This structure contains all coordinates required for feeding one component. One part data instance exists for every component being fed. Table 2 depicts this data structure. It initially contains the component type (partName), component placement position (placeCoord) and pallet position (placeBase). The other fields are populated during the execution phase mentioned in Section 4.7.

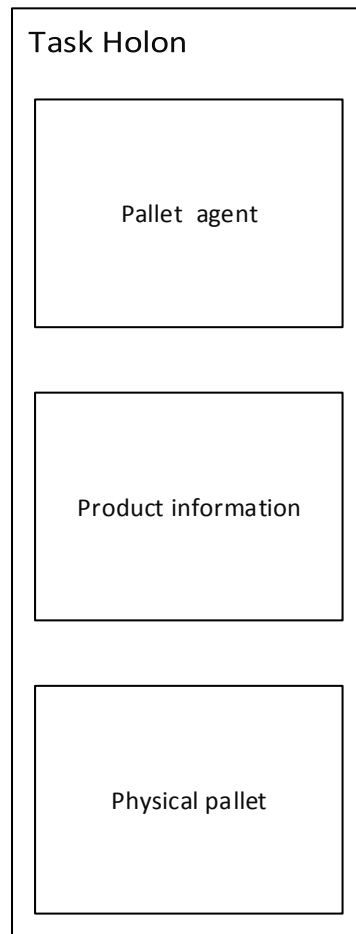


Figure 19: Task holon architecture.

Table 2: Part data class.

Variable Name	Data type	Description
pickCoord	XYCoord	Instance containing the collection coordinates of the component relative to the SU platform (or “pickBase”)
placeCoord	XYCoord	The placement coordinates of the component relative to the pallet.
pickBase	XYCoord	The global coordinates of the SU platform where the part will be collected from
placeBase	XYCoord	The global coordinates of the pallet
partName	String	A string describing the part type

5.3.2. Behaviours and lifetime

Pallet agents are launched during runtime when a pallet arrives at the feeding station. The product information and the pallet position are received from the cell controller and used by the station controller to launch the pallet agent. As mentioned in Section 4.8 the feeding substation can accommodate multiple task agents (thus, pallet agents) negotiating simultaneously.

Figure 20 shows the flow diagram of the pallet agent. The figure shows that the pallet agent is relatively simple, and executes multiple Subtask Initiator behaviours, manages product data and stores the production data. Once all the Subtask Initiator behaviours have completed, the pallet agent informs the cell controller and the agent is destroyed. The cell controller is then responsible for the transport of the pallet to the next substation in the cell.

As mentioned previously, the pallet agent drives production indirectly by employing the subtask agents. This negotiation is handled using the Subtask Initiator behaviour, which is a subclass of the JADE behaviour “AchieveREResponder”. In this implementation, components are fed sequentially in a specific order. If the order of part feeding was of no consequence, the behaviours could simply be launched in parallel, as oppose to in series. This could easily be achieved in JADE.

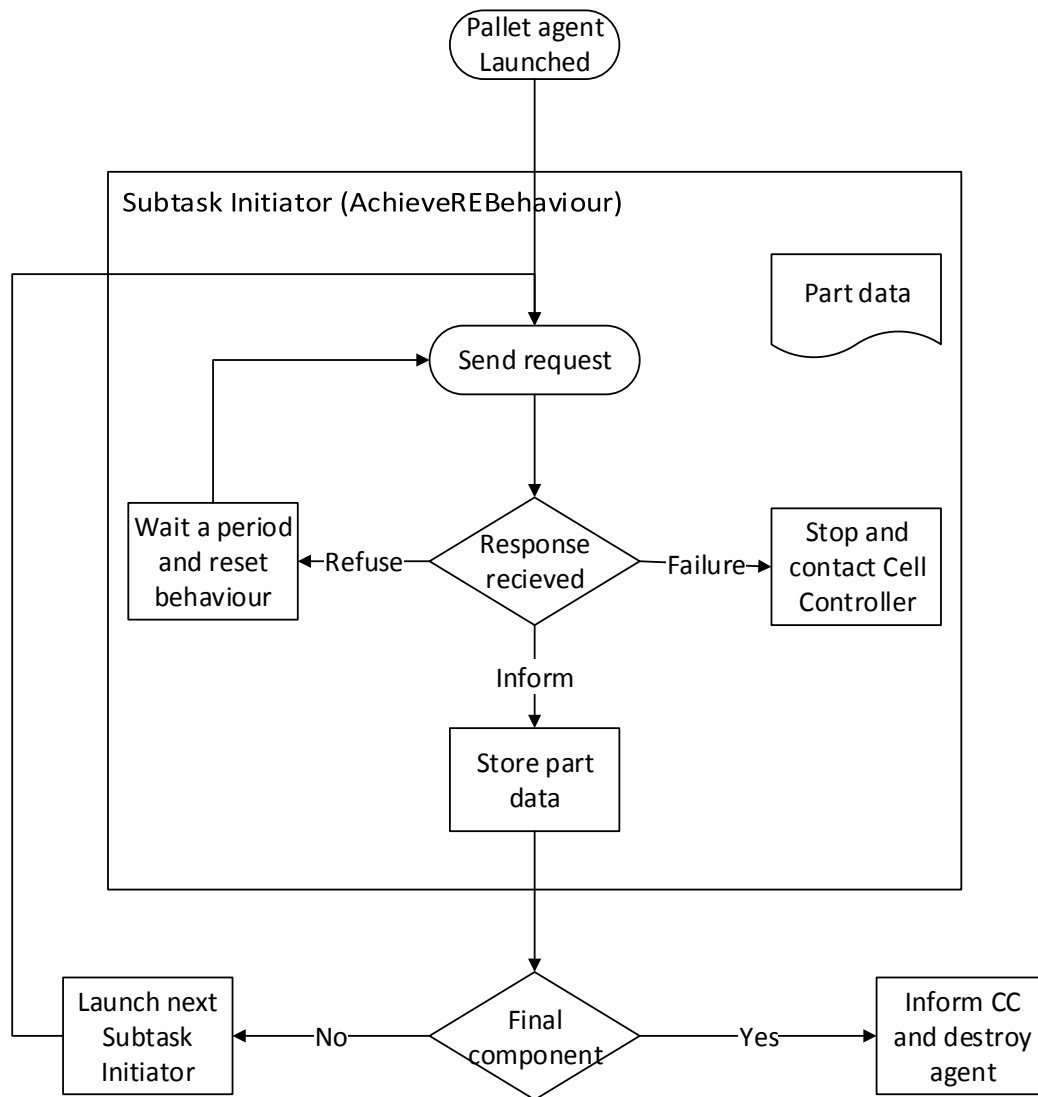


Figure 20: Pallet agent flow diagram.

5.3.3. Task subtask communication

Figure 21 illustrates the communication sequence between the task agent and the subtask agent. The process starts when the agent requests the subtask agent to feed a part. The request contains the product data for the specific component in the form of an instance of the part data structure, shown in Table 2. Once a request is sent, the pallet agent is bound to the subtask agent and must await a response. The subtask agent now handles the feeding of the component. Three possible responses could be generated depending on the outcome of production, as illustrated in the Figure 21.

If the subtask could not procure the services (OHs) needed for the feeding of the component, it responds with a **refuse**. The pallet agent then waits a period and restarts the process with a new request. Negotiations are then restarted.

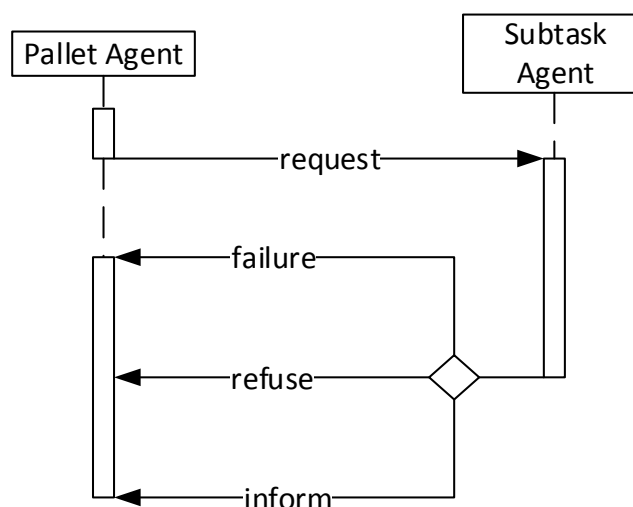


Figure 21: Task-Subtask conversation.

If a critical error occurred anywhere during production, the subtask responds to the task with a **failure**. At this point production will stall and an operator will be required to decide how to proceed. The failure message will contain diagnostic information assisting the diagnosis of the problem. As stated earlier, error handling was not considered in depth in this implementation.

If a part is successfully fed, the subtask responds with an **inform**. The inform contains the populated part data, so that this data could be logged to track errors further down the production process, for example, a damaged components can be traced back to a malfunctioning singulation unit. At this stage, the task can continue to the next subtask. If all the components of the task have been fed, the task agent informs the cell controller. The pallet agent is destroyed at this point.

5.4. Supervisor Holon

5.4.1. Architecture

The supervisor holon only consists of a supervisor agent, since it has no physical components. The supervisor holon is responsible for optimising and coordinating operations within the station controller. This is done by proposing an execution order to each subtask holon and the corresponding devices, in the form of a service list introduced in Section 4.7.

5.4.2. Service list

The service list is the list of services required to complete a subtask. It was implemented using an array of the data structure shown in the Table 3. The service list contains the list of services that are required to complete a subtask, in the field “serviceName”. The other fields are populated during the booking phase.

When the service list is transmitted between holons, it is embedded in ACL messages using Java serialization, which is a method of transforming Java objects into text. Once the list is approved by the supervisor holon, all the OH’s supplying the services are booked and recorded in the service list. The service list is also used during the production phase for recording conversation IDs and agent names.

Table 3: Content of “ServiceList” class.

Variable Name	Data type	Description
serviceName	String	Specifies the service supplied by the operational agent
agentID	String	Specifies the name of the operational agent booked to supply the service
serviceAvailable	Boolean	Used in the booking phase to identify whether or not the operational agent being queried is available
ConversationsID	String	Used as the conversation ID of the CNP conversation used during booking

5.4.3. Behaviours and lifetime

The supervisor agent is launched at system start up. Upon launch, the supervisor agent registers its service with the directory facilitator. Then it launches the “Contract List Responder” behaviour, shown in Figure 22, which is a subclass of the JADE behaviour “ContractNetResponder” and implements the FIPA specified contract net protocol. The Contract List Responder is the main loop of this agent and all subsequent behaviours are sub-behaviours of it. It has two child sequential behaviours, “Inquire Service List” and “Book Service List”, in which it executes conversations with the OHs using grandchildren behaviours (“Inquire Service” and “Book Agent”).

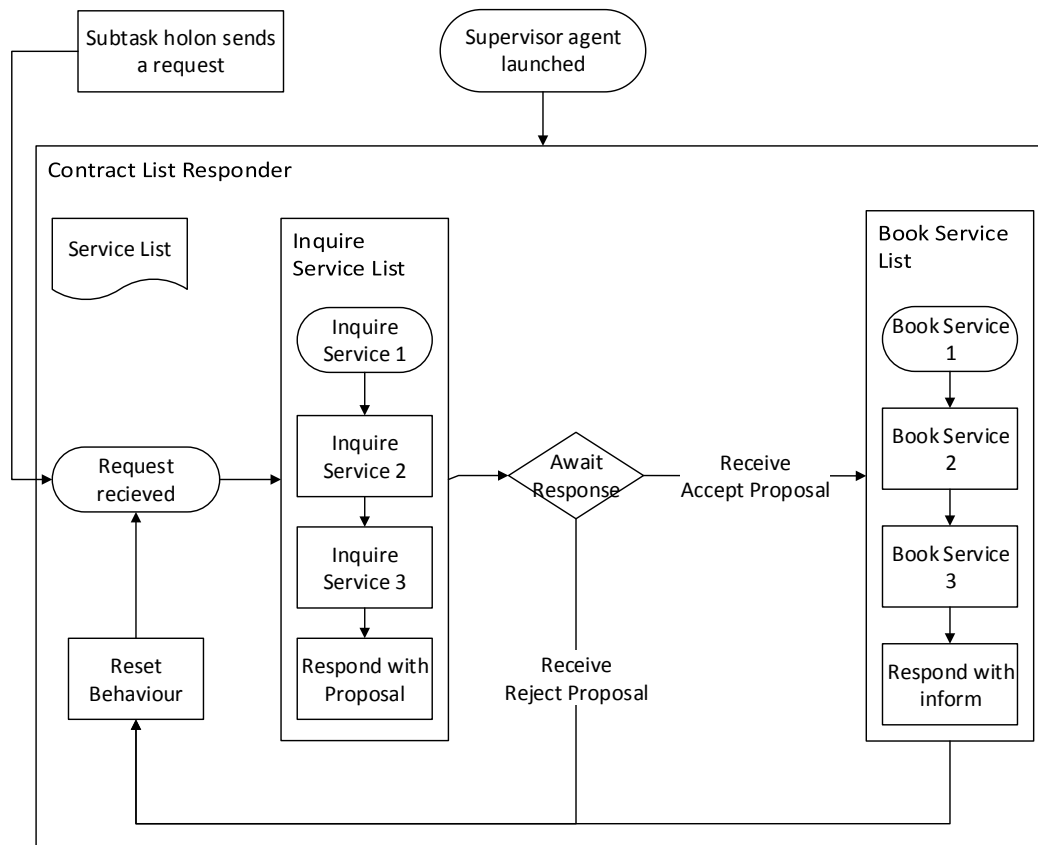


Figure 22: Supervisor agent flow diagram.

5.4.4. Booking phase

The booking system was implemented using nested CNPs.

The following negotiations are illustrated Figure 23:

- [A] The subtask agent sends a CFP containing the service list (containing the required services) to the supervisor agent.
- [B] The supervisor agent then inquires whether OHs that supply the required services are available. Availability in this case implies that the OH is not booked, is functioning correctly and ready (for example, a SU has a component ready for collection). The supervisor agent inquires whether all the services are available before responding.
- [C] The supervisor now responds with a proposal to the subtask agent. This proposal contains an updated service list with the available services, names of the respective OH agents and the respective conversation IDs. At this point, if any of the services that are unavailable were not crucial for production (i.e. quality inspection), the subtask could still accept the proposal. If the available

services are sufficient to complete the task, the subtask agent sends an accept-proposal (AP) to the supervisor agent.

If one or more of the services required by the subtask agent is not available, the subtask agent will respond to the supervisor agent with a reject proposal and the behaviour will be complete.

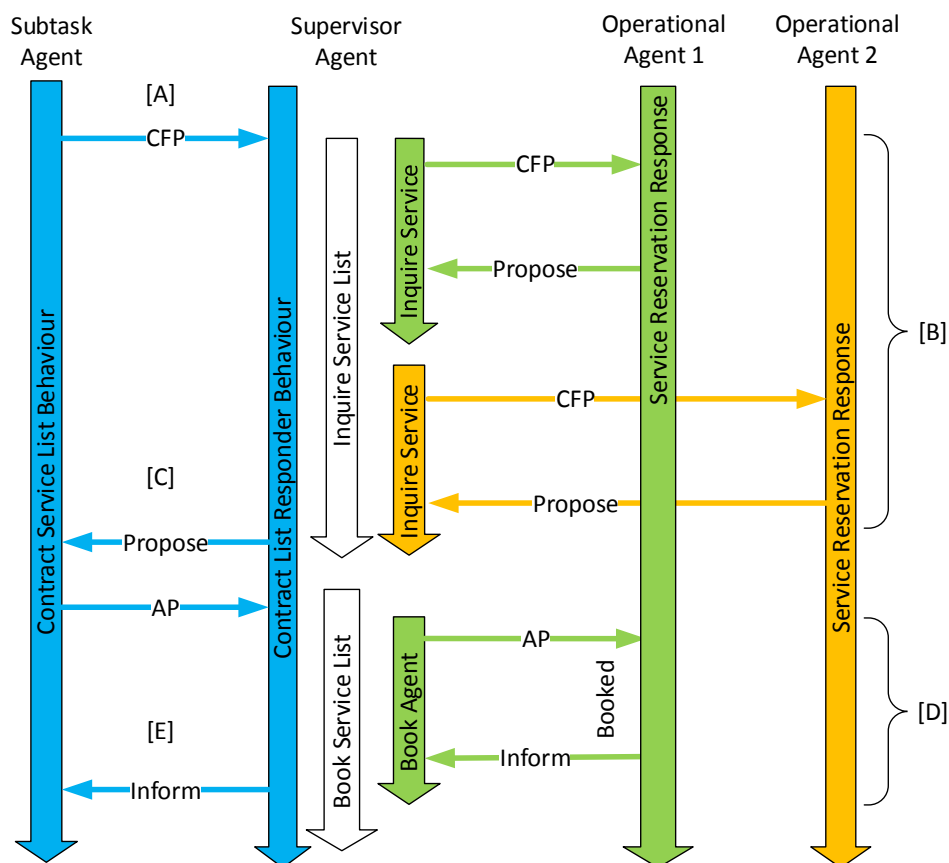


Figure 23: The booking of services.

- [D] When the supervisor agent receives an accept-proposal, it books the OHs and updates the corresponding Boolean entries in the service list.
- [E] Once the services have been booked the supervisor agent responds to the subtask agent with an inform containing the populated service list.

In the thesis' implementation, the first service to respond was selected for the service. For instance, if OH1 and OH2 supplied the same service in phase [B], and OH1 was the first to respond with a proposal, then OH1 is recorded in the service list. Further optimization can be done if the proposing OH responds with

information that could be evaluated for its performance in this job. The colours used in Figure 23 are used to group conversations between agents.

5.5. Subtask holon

5.5.1. Architecture

As with the supervisor holon, a subtask holon does not contain any physical devices and therefore only comprises of a subtask agent. Subtask agents represent the feeding of a single component (e.g. “Feed Contactor Agent” and “Feed Pigtail Agent”). They are employed by task holons to negotiate with the supervisor holon and to perform and monitor the operation during feeding by communicating with OHs. Once the feeding of a component is complete or has failed to complete, the subtask communicates its result to the task agent with either an *inform* or a *failure*. Subtask agents are launched at the start-up of the JADE platform. Only one subtask agent exists per component type in the feeding station since only one component can be fed at a time.

Once the services have been booked, as described in Section 5.4.3, the supervisor agent responds to the subtask agent with an *inform* containing the populated service list. The subtask agent then sequentially requests the OHs to perform the services. A FIPA Request Interaction Protocol behaviour namely “*RequestService*” is used to execute operations. The relevant “*partData*” is passed back and forth within these messages. The component collection coordinates are not initially known by the subtask agent and are received by the SU agent. On the other hand, the component placement coordinates are received by the subtask agent from the task agent.

5.5.2. Behaviours and lifetime

The subtask holons are launched at the start of the agent platform. Each subtask is specialized by registering its service with the DF (for example “PTTask”, referring to the feeding of a pigtail) and setting the subtask service list, which is the services needed to complete the feeding of one component, for example:

- “SUPT” - referring to the singulation or supplying of a singulated pigtail
- “Transport” - referring to the transportation of the singulated component by the pick-and-place robot
- “Inspection” - referring to the inspection of the completed pallet

As illustrated in Figure 24, after specialization, the Subtask Responder behaviour is launched. This behaviour is essentially the main loop of this agent and all other behaviours used for production are children of this behaviour. This allows the sharing of data between child behaviours as mentioned in Section 2.3.3. If a request is received from the pallet agent, the “Contract Service List” behaviour is

scheduled. This behaviour is, in FIPA terms, a contract net initiator that negotiates with the supervisor agent, as explained in step [A] of Section 5.4.4.

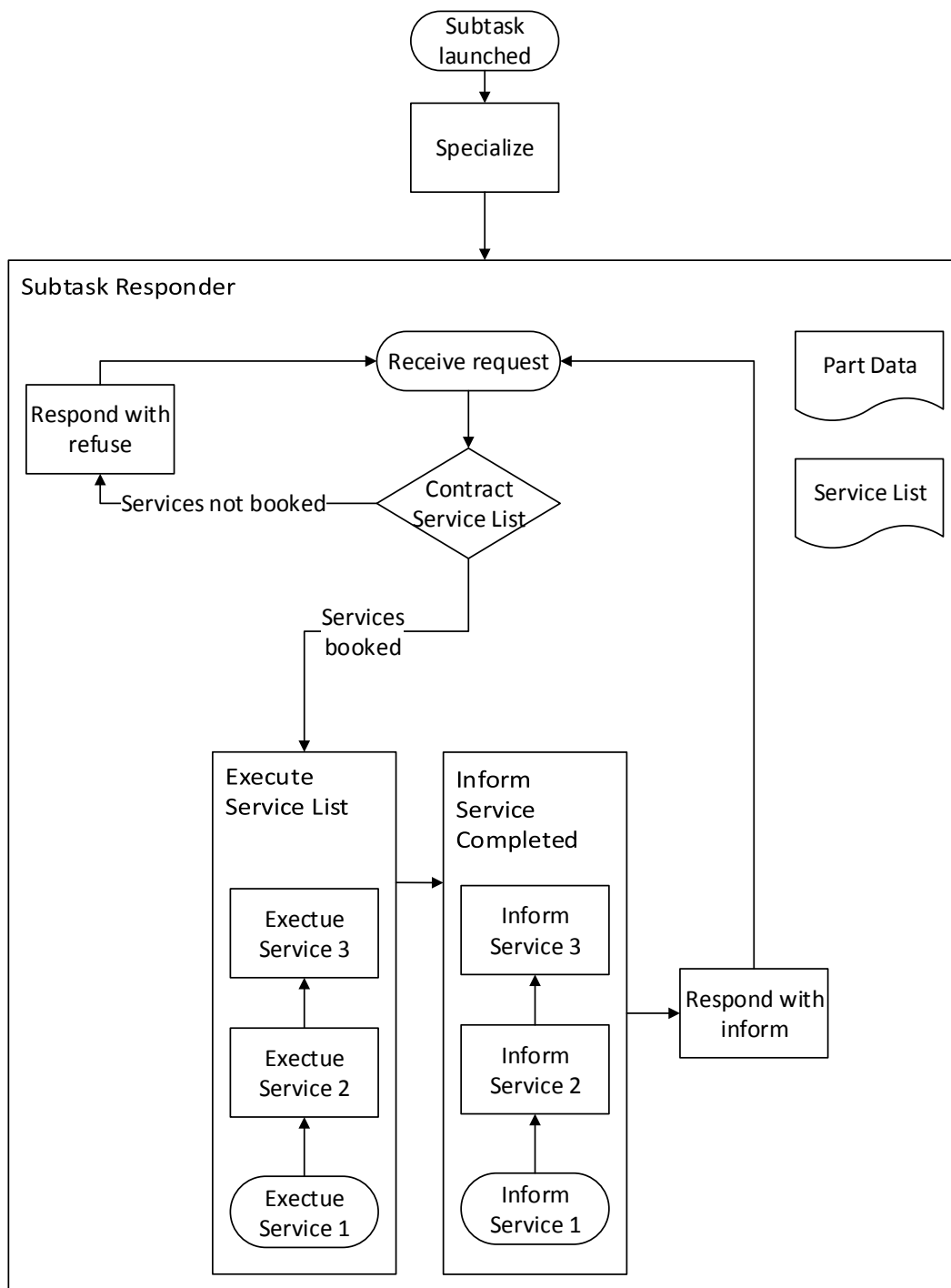


Figure 24: Subtask Agent flow diagram.

Once the service list is approved and the OHs booked, the execution phase begins. In this phase the physical production occurs. The messages exchanged between agents during the execution phase are illustrated in Figure 25. The execution phase is implemented by the “Execute Service List” behaviour (a sequential behaviour). It schedules children behaviours, Execute Service (an implementation of “AchieveRE Initiator”), which sequentially request the services be performed by the OHs. JADE ontologies are used to group conversations so that requests at different phases will not be incorrectly interpreted.

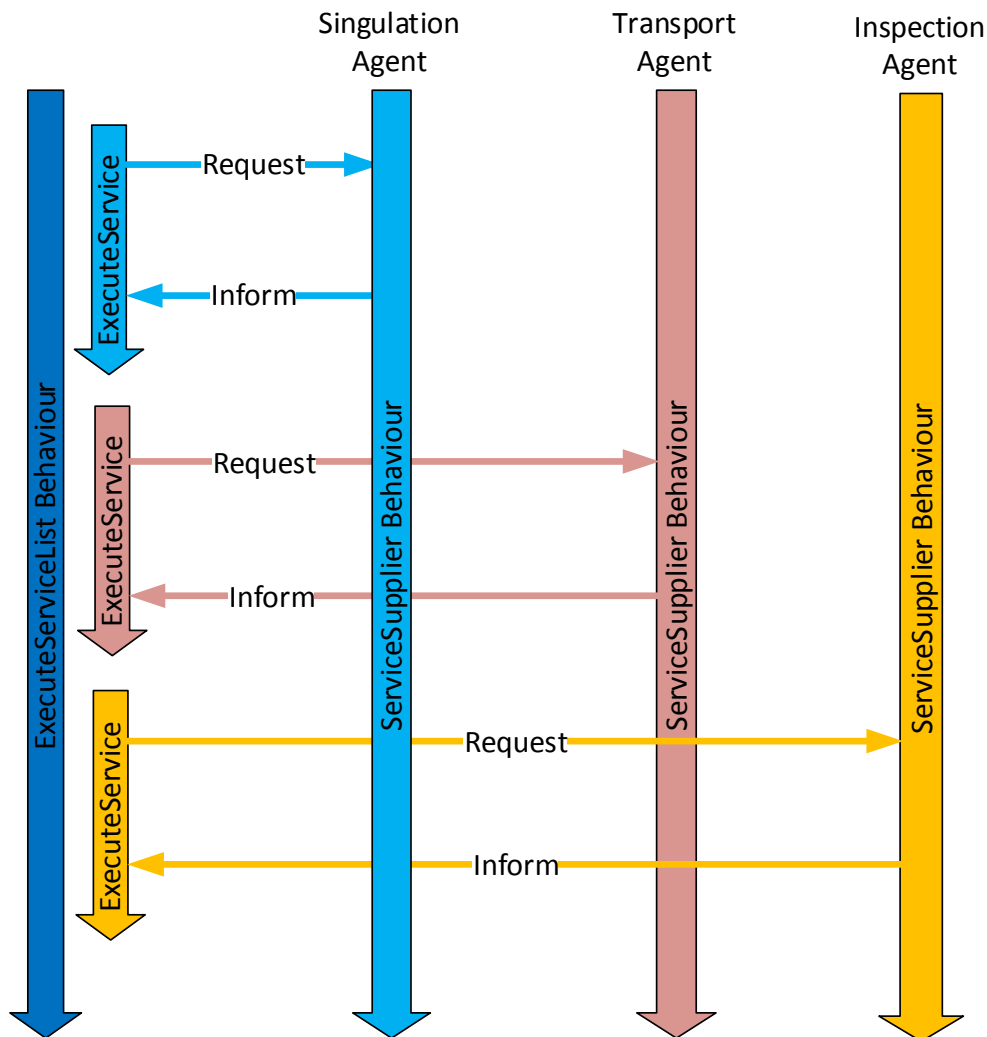


Figure 25: Execution phase, interaction between subtask and OHs.

Each request sent by the subtask agent contains the part data (Table 2) for the specific component. The part data is modified with each consecutive conversation. The part data is first received from the pallet agent, during the conversation shown in Figure 21, at which time it contains the part names, placement coordinates and placement base of the parts. These coordinates are sent to the SU agent where the part data is modified to include the “pickcoords” and “pickbase” fields. These

represent the SU collection platform and the component coordinates relative to it, respectively. The part data sent to the transport agent now contains the necessary information to transport the component. The transport holon collects the component from the SU platform and places it in the pallets fixture. The subtask holon then requests a quality inspection agent to inspect the pallet to determine whether the component was correctly placed. Once a reply is received, the execution phase is complete.

Note that if a critical error occurs during the execution phase, a failure message is sent to the pallet agent. At this point, it was assumed that an operator would be contacted to inspect the pallet. Methods of correcting errors during production were not explored in this thesis.

Only when the subtask is complete can OH agents be unbooked and allowed to resume operation. If a SU where to resume operation immediately after supplying the component collection coordinates, the robot would not have time to collect the part. For this reason, it is necessary for the SU to wait until the part is collected. The unbooking phase is very similar to the execution phase in that it is implemented by a sequential behaviour containing a number of child “Archive Responder” behaviours.

5.6. Common aspects of operational holons

5.6.1. Architecture

Within ADACOR, the operational holons represent physical resources and their management during production. Figure 26 is an adaption of that used by Vrba *et al.* (2011) (Figure 2, page 6) and it shows that JADE agents were selected as the HLCs. A Java object was developed as the control interface between the HLC (JADE agent) and LLC (machine specific controller or controllers). The control interface objects use TCP/IP sockets through physical Ethernet connections, for convenience and modularity.

The operational agents are responsible for the communication and decision making of the holon. The agent also manages the holon data, which includes local data as well as the communicated data.

The agent also communicates with LLC by invoking methods in the control interface. Control interface classes were written for the specific LLCs.

5.6.2. Overview of behaviours and lifetime

Operational agents have a generic set of behaviours that are specialized for their specific purpose. As illustrated in Figure 27, they also possess common variables that are used to change the holons state. Two Boolean variables are used to

differentiate the holons states. The first being the “agentReady” Boolean, which is false on launch and is then set to true when the holon is ready and can be booked. The holon cannot be booked before the agent is ready.

The second variable, “agentBooked” involves the booking of the agent. The variable is false when the agent is not booked and will be true once the agent is booked. An agent cannot supply a service if it not booked.

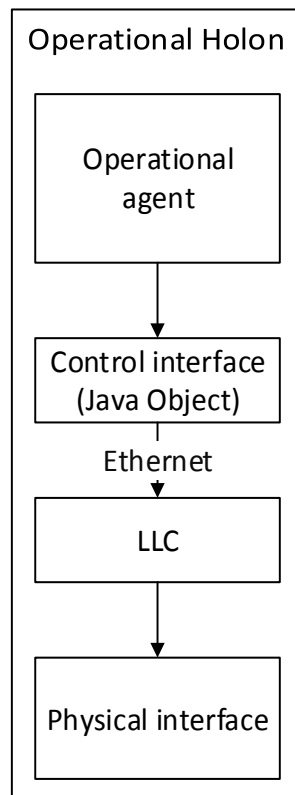


Figure 26: Operational Holon.

Upon launch, the operational agent is specialized (not shown Figure 27). This usually involves instantiating the control interface and setting the IP address of the LLC. In addition, any variables that are needed for the specific operational holon are declared and set (for example the singulation base).

After specialization, the three conversational behaviours are scheduled. An optional operate behaviour may also be scheduled if necessary. All of these behaviours are scheduled in parallel. The ontologies determine how the operational agent interprets messages received. The Service Reservation Response and Service Complete Listener behaviours are generic and are described in the paragraphs that follow. The Service Supplier and Operate behaviours are described under their respective OH sections.

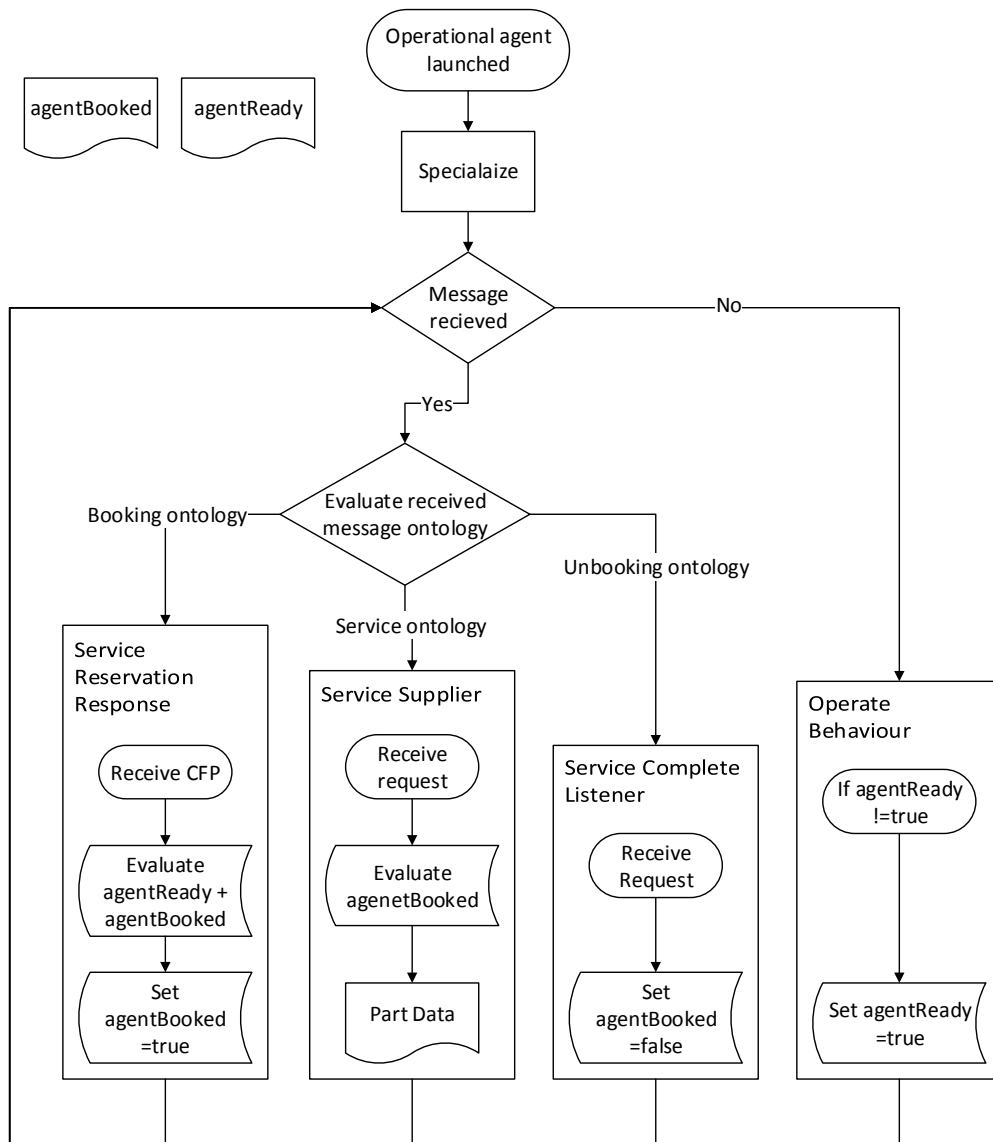


Figure 27: Generic Operational Agent flow diagram.

5.6.3. Service reservation response (booking phase)

The booking of operational agents is handled by the “Service Reservation Response” (SRR) behaviour, an implementation of the FIPA “Contract Net Responder”. This conversation, described in Section 5.4.4, uses the booking ontology. Figure 28 expands upon the behaviour.

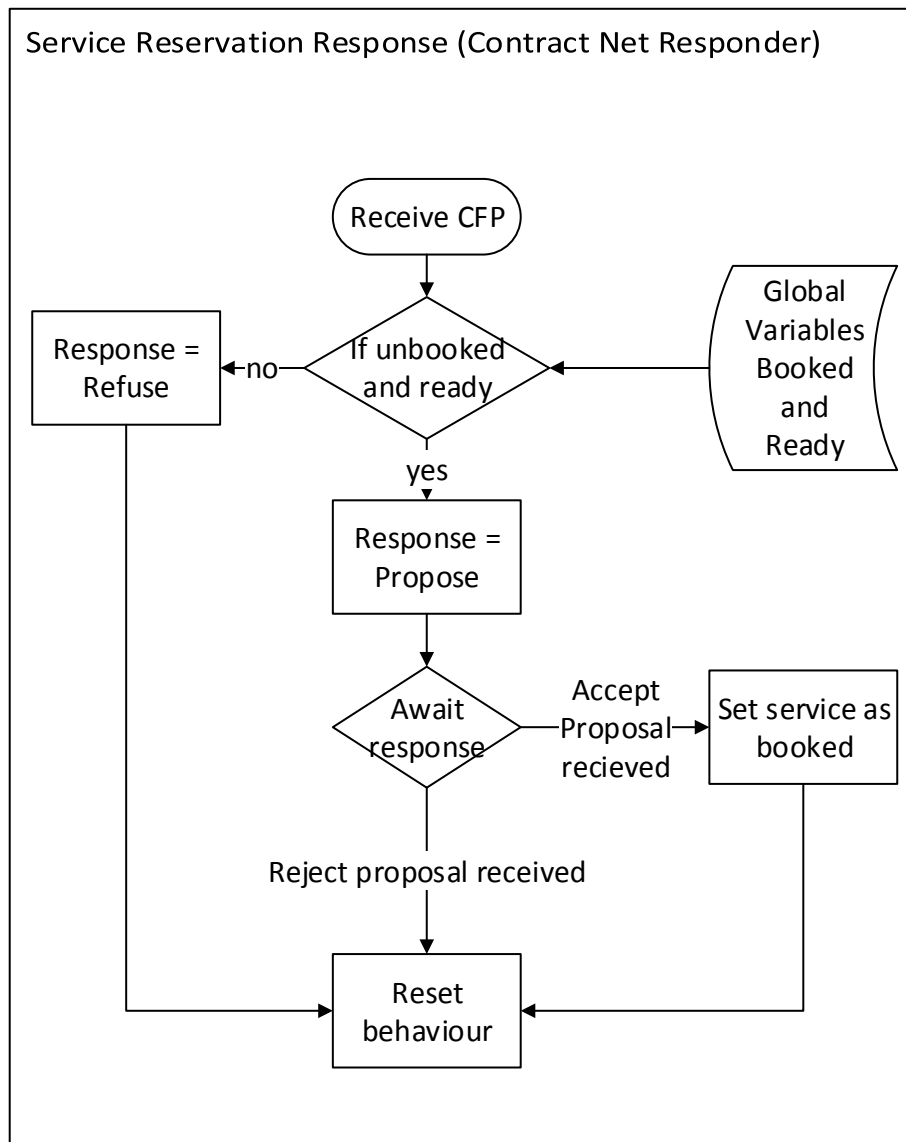


Figure 28: Flow diagram of Service Reservation Response behaviour.

The behaviour is resumed when a CFP of the booking ontology is received from the supervisor agent. The operational agent will then be evaluated as to whether it is ready and not booked. If so, the operational agent responds with a **propose**. The operational agent now awaits a response, if this response is a **reject proposal** then the behaviour is reset and the conversation ends without booking. If the response is an **accept proposal**, then the operational agent is now booked. The Boolean “agentBooked” is now set to true.

If the agent is either booked or not ready, then the response is a **refuse** and the behaviour is reset, as the conversation is now complete.

5.6.4. Service complete listener

Once the service has been completed, the subtask agent informs the operational agent that it is done and can be unbooked. The unbooking is handled by an “AchieveReResponder” behaviour called the Service Complete Listener (SCL) shown in Figure 29. This conversation, described in Section 5.5.2, uses the unbooking ontology.

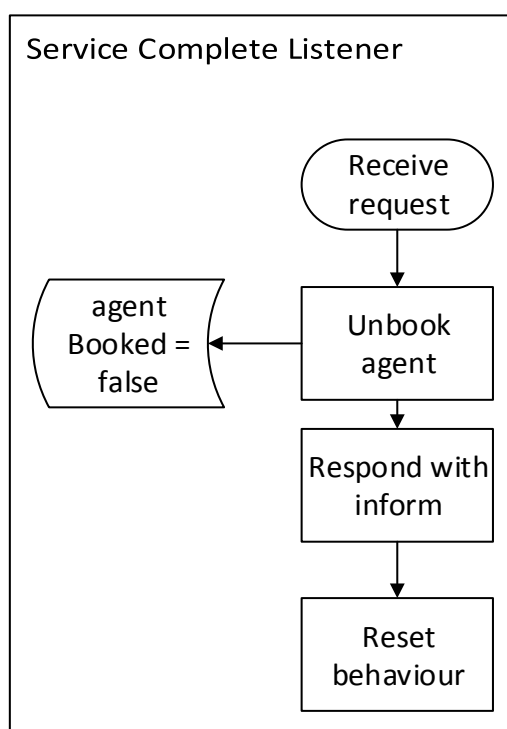


Figure 29: Flow diagram of Service Complete Listener behaviour.

The behaviour is resumed when a request of the unbooking ontology is received from the subtask agent. The operational agent will then evaluate whether the operational agent is ready and not booked. If so, the operational agent responds with a **proposal**. The operational agent now awaits a response. If the response is a **reject proposal**, then the behaviour is reset and the conversation ends without booking. If the response is an **accept proposal**, then the operational agent is booked. The Boolean “agentBooked” is now set to true.

5.6.5. EIH inspection considerations

By definition, each EIH transport holon contains a vision sensor, while EIH SUs do not. Therefore, the EIH SUs will require the visual inspection services to be supplied by the EIH transport holon. This is different from other conversations in the system as it is a service being supplied by one OH to another. The EIH

configuration's transport holon is able to supply inspections to multiple SUs. Two interactional methods of achieving OH-to-OH services were considered, namely (1) subservices and (2) nested services. A subservice interaction was implemented for the EIH inspection service conversation, as it potentially reduces the robot idle time. The nested services interaction is explained in Appendix C 2.

In this subservice system, the SU holon autonomously requests an “EIH Inspection” service from the transport holon, when it is required. The request contains the coordinates at which the inspection should occur. The transport holon will supply inspections if it is not booked. This will make good use of the robot resource, reducing robot idle time, since the robot can supply inspections when no other tasks are requesting its service.

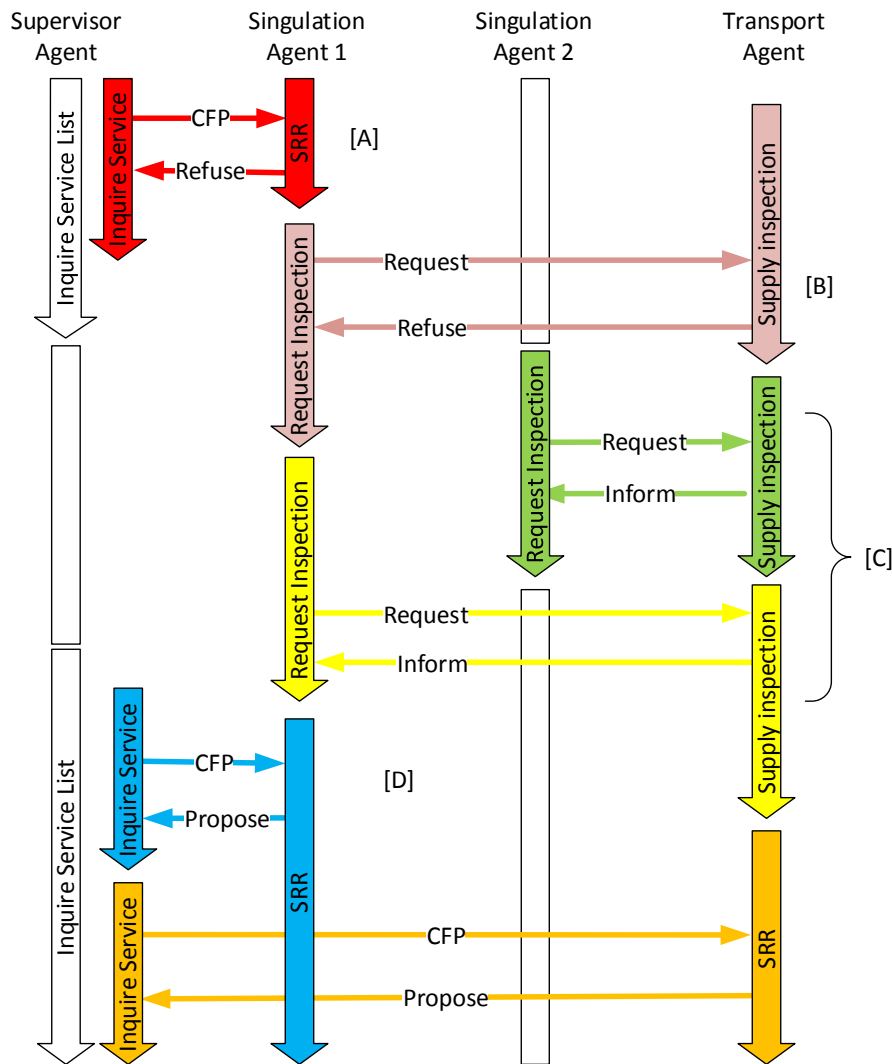


Figure 30: Sub services architecture.

Figure 30 shows a typical subservice interaction implemented when SU holons request an EIH inspection.

- [A] The supervisor agent sends a CFP inquiring whether the SU agent is available for booking. The SU agent refuses, as it does not have a collectable part at this time.
- [B] When ready, the SU requests an EIH inspection from the transport agent. The transport agent enquires at the directory facilitator as to which service this specific SU supplies. This information is used to determine for which component the inspection should search. If the inspection is unsuccessful, the transport agent refuses.
- [C] If successful, the response is an inform containing the component coordinates. The SU is now ready to supply its service, i.e. the part and its pose coordinates.
- [D] When the Supervisor agent receives a CFP from a subtask (not shown in this figure), the Supervisor inquires whether the SU agents are available for booking.

It is worth noting that the SU holons in the EIH and the FC configurations supply the same service. The SU holons communicate the SU collection base and component coordinates relative to the collection base. This allows for a hybrid system with both FC SUs and EIH SUs.

5.7. Singulation unit (SU) holon

5.7.1. Architecture

The SU holon is an operational holon responsible for supplying components and their pose. Please refer to Section 5.6 for a description of the aspects common to all operational holons.

The SU holon was implemented as a JADE agent to facilitate higher-level communication between holons. The service supplied by the SU holon is specific to the component type it supplies. For example, “SULT” refers to supplying a load terminal, similarly, “SUPT” for pigtail. The method of singulation can be either manual labour or machine driven. Numerous architectural configurations can exist for SU holons. In this case study, three hardware configurations were used. The first is a component magazine, in which parts are presented in fixtures in a known pose, thus eliminating the need for a vision system. Figure 15 shows three component magazines. The second SU configuration is the FC configuration, in which parts are singulated in a random pose (no fixtures in the holon) and the camera is a component of the SU holon. The third configuration is the EIH configuration, in which parts are singulated in a random pose, but the camera is not a component of

this holon; instead, inspections are requested externally from another operational holon. Figure 31 shows a photograph of the FC SU used for testing on the left, and an EIH SU holon on the right. Since no functioning SUs were available for the tests, components were manually placed on a collection platform (the white rectangular board in the photographs) to simulate the SUs.

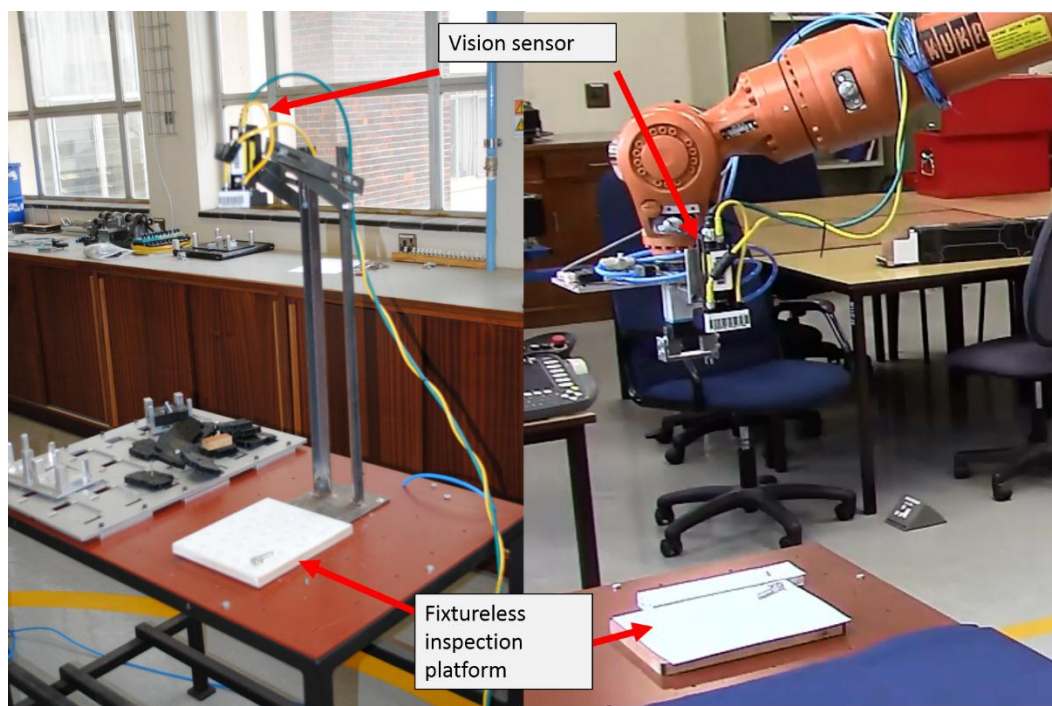


Figure 31: Simulated FC and EIH SUs used for experiments.

In the **fixed-camera configuration**, the Cognex smart-camera and its accompanying breakout board (In-sight CIO-micro) are components of the SU holon, as shown on left-hand side in Figure 32. The breakout board was not used in this implementation. Instead, all communication was done via Ethernet. Telnet, a text-based communication session layer built on the TCP/IP protocol, was used for communication between the agent and the smart camera. The Apache Commons Net Java API was used for Telnet communication. A Java class was developed for the control interface, in which methods invoked executed tasks and returned data. For example, “TelnetCamera.trigger(void)” would trigger the sensor resulting in an inspection by the camera. These classes serve as a modular layer allowing components (in this case, the camera) to be swapped out without affecting the remainder of the controller. This class is later also used in the EIH configuration's transport holon. The camera's breakout board was used as a controller for the simulated SU.

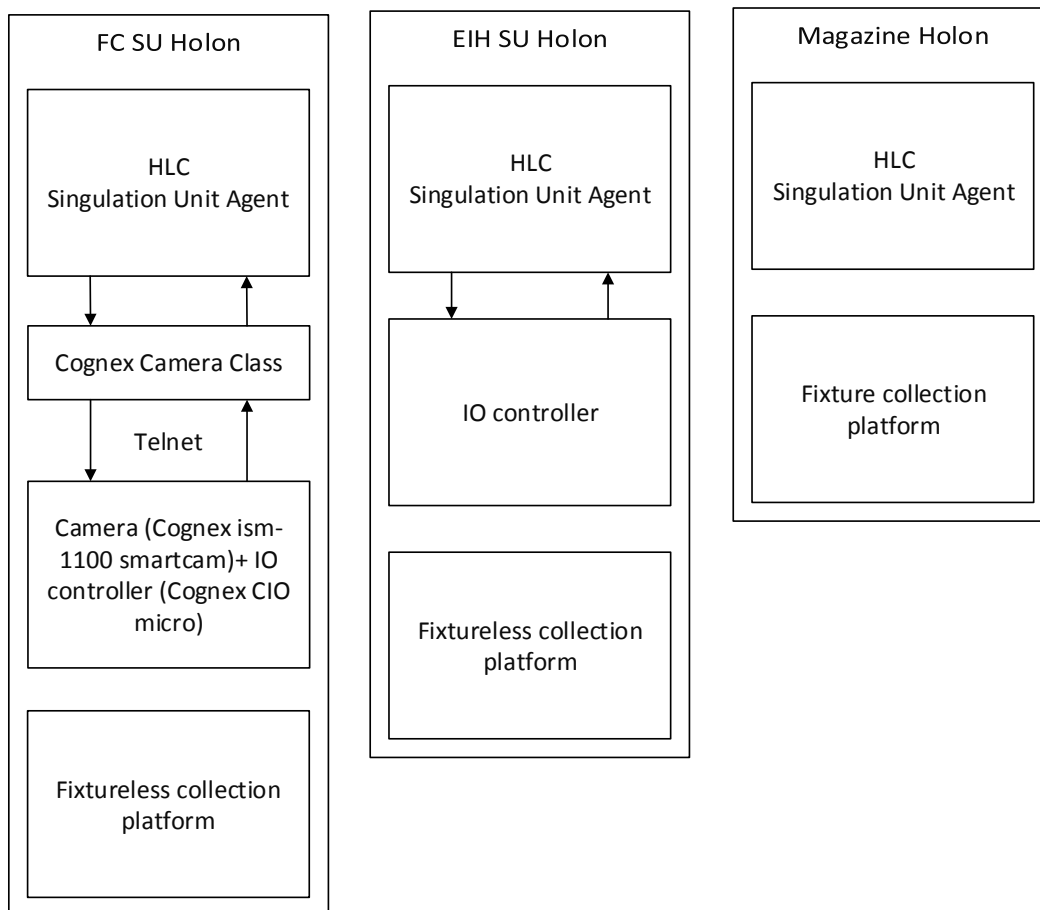


Figure 32: SU holon architecture configurations.

In the **eye-in-hand** configuration, no vision sensor is present in the EIH SU holon. Instead, the SU employs the vision system as a service from another operational holon, namely the EIH transport holon, to locate components. A SU holon using machine driven singulation would, however, have a controller that communicates with the HLC. Figure 32(middle) shows the EIH SU holon architecture assuming that it has an IO controller. In the implementation of this thesis, however, no controller was used.

In the case of a component **magazine**, the holon was adapted to neither have a vision sensor, nor require it as a service. The magazine would instead have components in fixtures where they can be collected in a known and repeatable pose. The components positions are stored in an array in the SU agent. This type of SU could be used to simulate manual singulation. Due to the simplicity of the magazine holon, it does not require further explanation.

5.7.2. Coordinate system calibration

The vision system is calibrated to supply the part coordinates relative to a base that is common to both the robot and the vision systems coordinate system. Figure 33 shows the calibration of the vision system to such a point, namely the SU Base origin. A vision system calibration was used for each collection platforms inspected by the sensor.

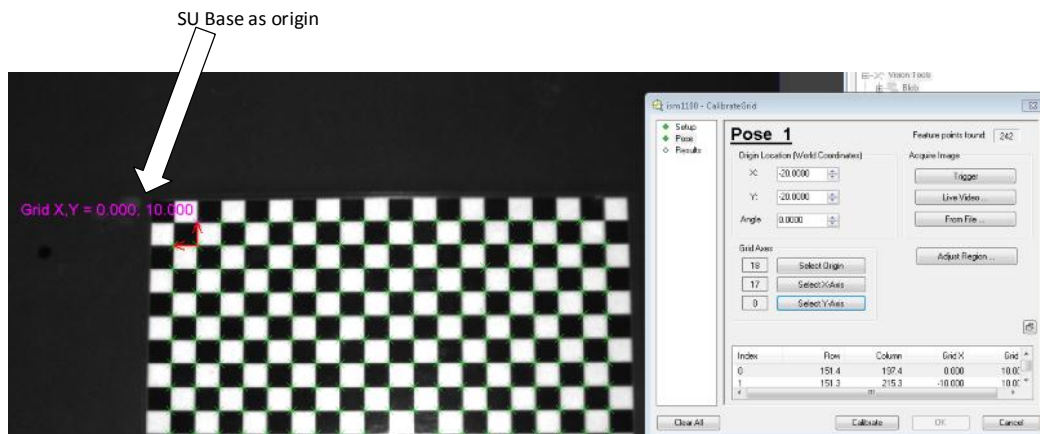


Figure 33: Cognex calibration showing SU base.

Figure 34 shows the Cognex in-sight explorer software locating the components. In the figure, the load terminal is located on the left and the pigtail on the right. The cross hairs in Figure 34 show the coordinate returned for the component identified by the vision system. This point was conveniently selected as the point at which the robot gripper grasps the component. The coordinate communicated to the robot is relative to its SU base (shown in Figure 33). For this implementation, inspections only identified components in the orientation shown in the figure.

Figure 35 assists in the explanation of these relative bases. The robot base position is selected as the reference point of the world coordinate system. In the figure, A shows the SU base position relative to the robot, noting the figure does not show rotations. The SU base coordinate is a property of the SU holon and is therefore stored in the SU agent as the “myCollectionBase” field. The vision system communicates the [B] part coordinates relative to the SU base and [A] its base position. The robot controller then does the transformation, finding [C] the part pose relative to the robot base. This is used by the robot for collection of the component.

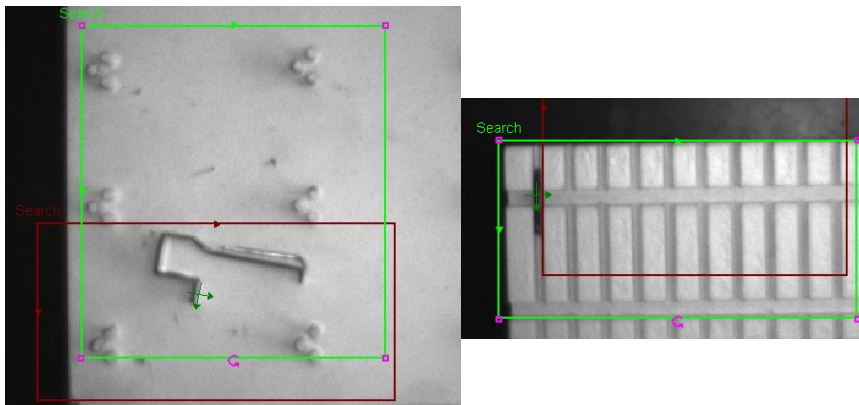


Figure 34: In-sight software Locating components.

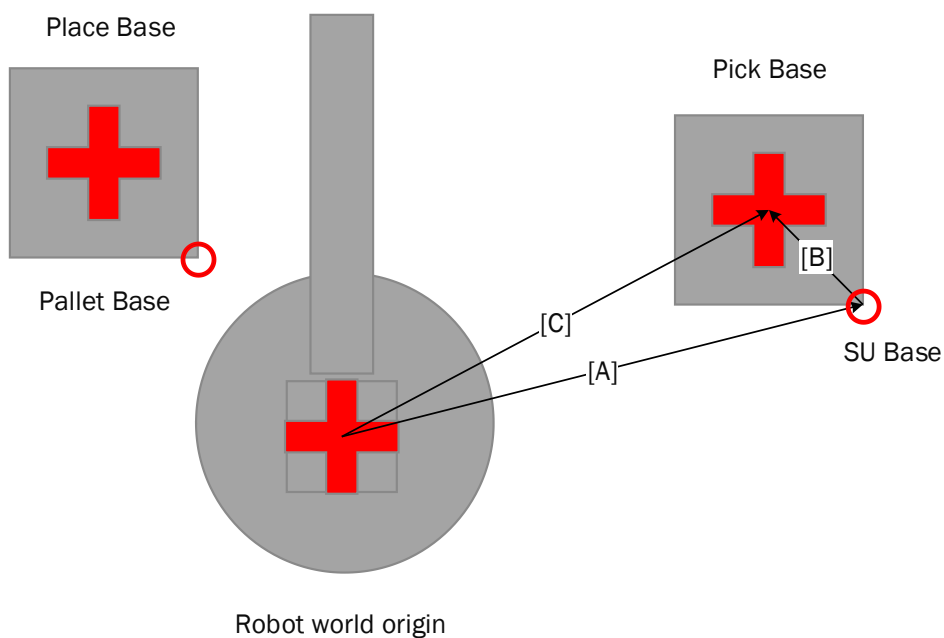


Figure 35: Component collection coordinate transformation.

5.7.3. Behaviours and lifetime

The SU agent facilitates communication and manages the holon data, as stated in Section 5.6. The SU agent is a subclass of the operational agent class and contains additional properties, namely the part location coordinates and collection base coordinates.

5.7.3.1. Fixed camera singulation unit

The SU agent class is further sub-classed to a fixed camera SU (or “FCSU”) agent. Figure 36 illustrates the behaviours of the specialized FCSU Agent. The behaviours shown are running in parallel with the SRR and SCL behaviours shown in Figure 27. This specialized agent class (FCSU) also contains an instance of the Telnet camera object which is used to interface with the vision system. After launching, the agent specializes, setting the camera IP and SU collection base coordinates.

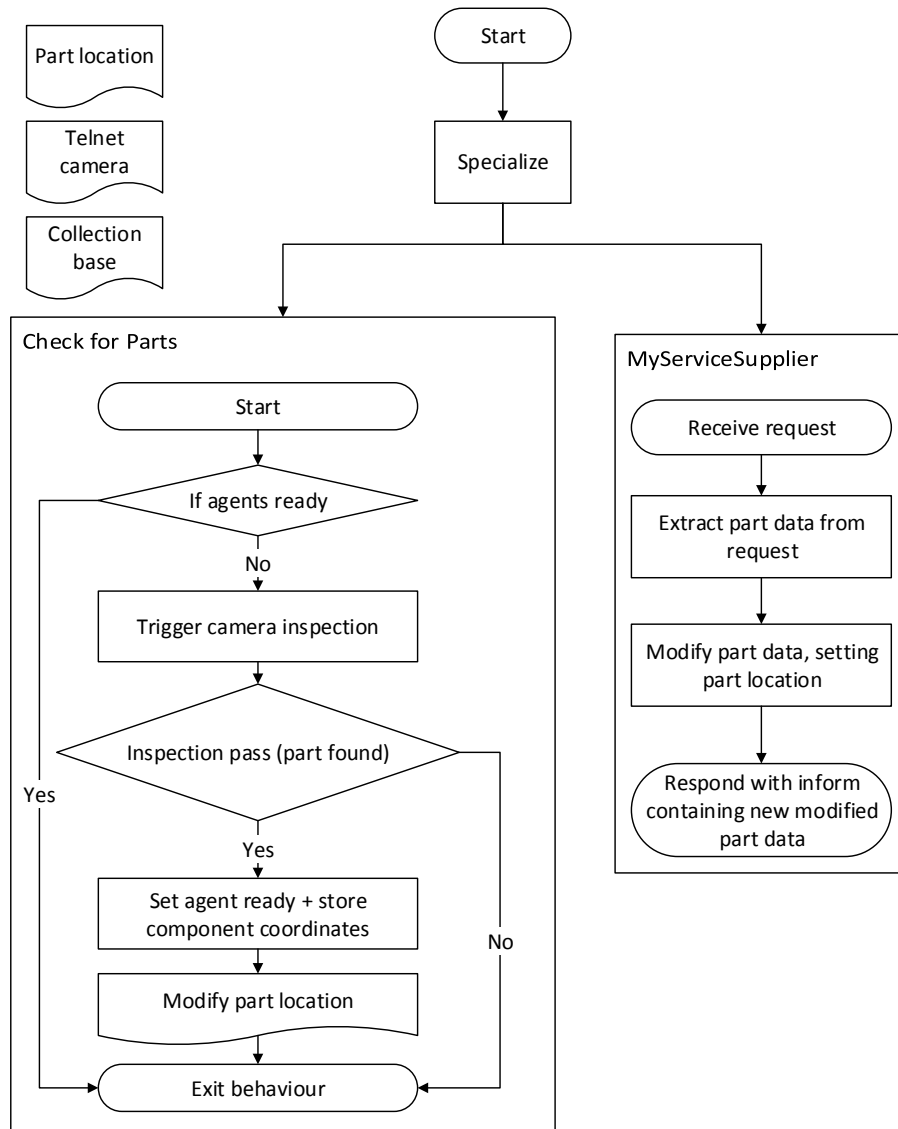


Figure 36: Fixed Camera Singulation Unit Agent Behaviours.

The “Check for Parts” behaviour is an implementation of the Operate behaviour shown in Figure 27. It simulates the singulation of components by the SU holon with periodic sensor inspections, since a physical SU was not available for the

research. If a component is identified by the camera, the component pose is stored and the agent is ready for booking. This behaviour was implemented using a wake timer to trigger inspections.

The “Service Supplier” behaviour in Figure 27 is also replaced with “MyServiceSupplier” shown in Figure 36. This behaviour receives a request, from the subtask agent and extracts the part data from the received request. It then modifies the “pickBase” and “pickCoord” fields in the received part data variable. This supplies the SU base and part pose coordinates respectively. The modified part data is then sent back to the subtask agent.

5.7.3.2. EIH singulation agent

The EIH singulation agent is another subclass of the SU agent. It inherits the part location and collection base field from its superclass. It has an additional property, specifically the inspection point. This is the point at which the EIH transport holon will supply the visual inspection.

The “operate” behaviour in Figure 27 is again implemented with the check for parts behaviours (Figure 37). The “check for parts” behaviour in this case will schedule a request EIH inspection. The service supplier behaviour is the same behaviour used in the FCSU agent.

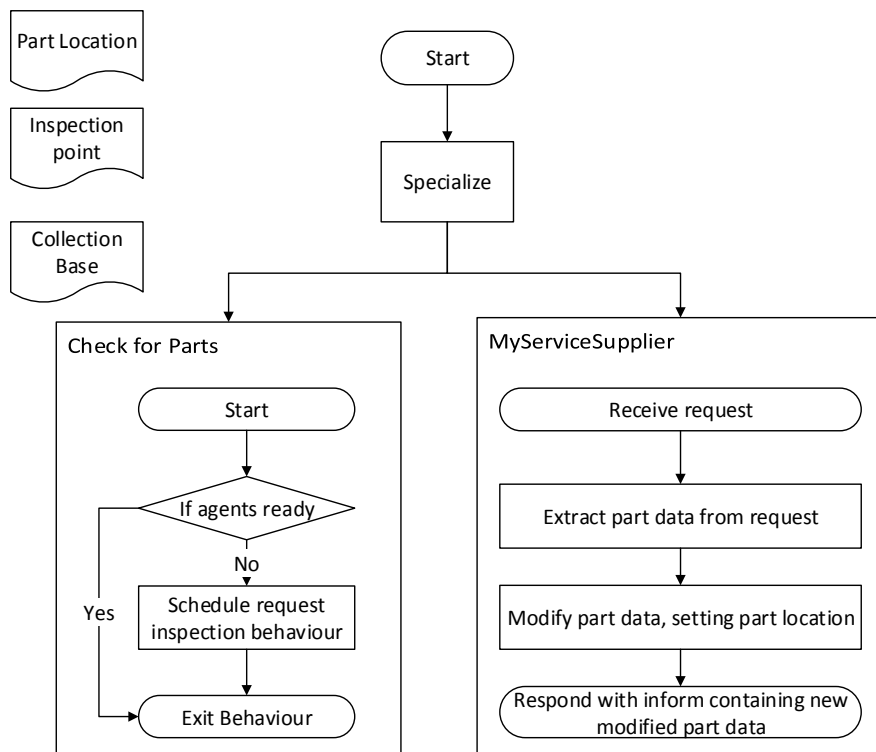


Figure 37: EIH singulation agent behaviours.

5.8. Transport holon

5.8.1. Architecture

The transport holon is responsible for transporting components from a singulation unit to a fixture on the pallet. Please refer to Section 5.6 for a description of the aspects common to all operational holons. The two architectural configurations of the transport holons used in this thesis are shown in Figure 38.

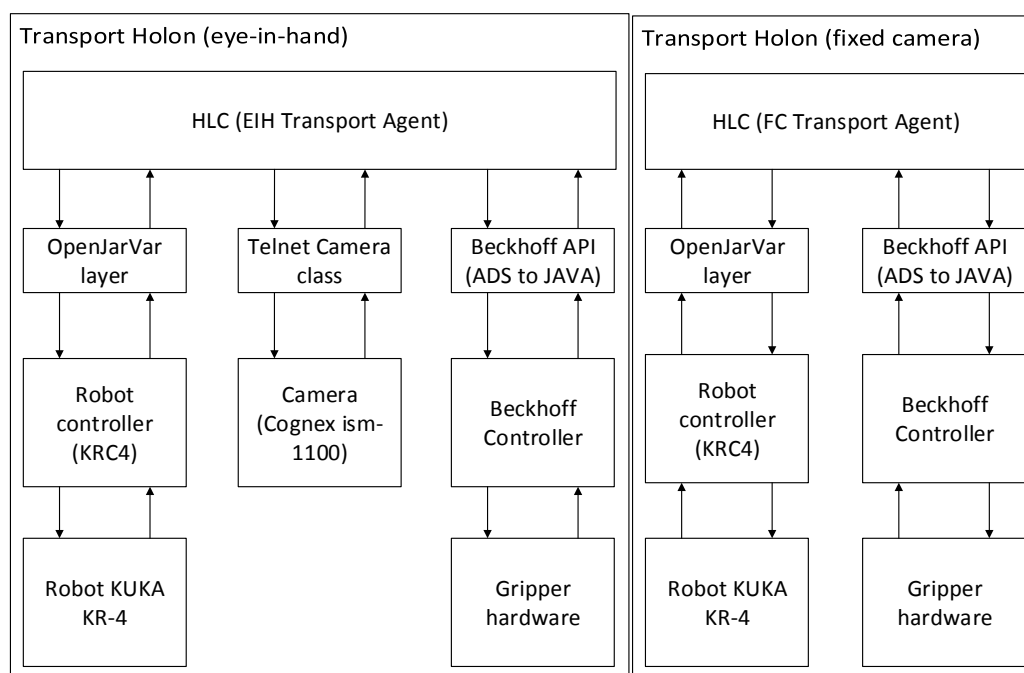


Figure 38: Transport holon components.

The transport holon used for fixed camera feeding is the simpler of the two configurations and is shown on the right. This holon consists of a JADE agent as the high-level controller, a KUKA KR-16 robot, the KUKA Robot controller (KRC4), a Beckhoff Embedded PC controller and the pneumatic gripper module and sensor used for the robot end effector. In the EIH configurations, the transport holon contains these elements in addition to a vision system, the same vision system used in the FC SU holon.

The HLC communicates with the KRC by modifying Boolean variables and coordinates in the KRC memory. “JOpenShowVar”, an open source third party API written for academic purposes, was used for communication between Java and the KRC. The “JOpenShowVar” API communication was implemented using socket communication through physical Ethernet. It should be noted that KUKA XML is a product that supplies the same capabilities and may be better suited for industrial production systems. A Beckhoff embedded PC was used for the gripper controller.

The Beckhoff ADS to Java API was used for communication between the HLC and the embedded PC. Beckhoff ADS web service was also investigated for the above-mentioned communication, but was found to have longer lead times and therefore it was not used.

A common question that arises is why not make the camera a separate holon. This gives rise to the bigger question of when to split a holon, which is discussed in Section E 2.

5.8.2. Robot control

The robot control and communication was written to allow all robots controllers in the station to have the same program running on it, thereby easing the reconfiguration process and eliminating the need for multiple robot programs.

KUKA robot language (KRL) was used for the robot control. Two types of movement functions were employed in this control implementation, namely point-to-point (PTP) and linear (LIN) movements. PTP is defined by KUKA as the “*the quickest way of moving the tip of the tool (Tool Center Point: TCP) from the current position to a programmed end position.*” (KUKA Robot Group, 2002). The linear movement “*calculates a straight line from the current position (the last point programmed in the program) to the position specified in the motion command*” (KUKA Robot Group, 2002) . The KRC uses a “computer advanced run”, which performs arithmetic and logic calculations while the robot is moving, allowing the KRC to queue up to five robot movements. This affects the Java thread interpretation of what the robot is doing, as the KRC memory has already been modified. For example a movement that is already planned (calculations completed but motion is still in progress) by the KRC will be interpreted as complete by the Java thread potentially actuating the gripper at the wrong time. The wait() command was used to halt advanced run mode.

Figure 39 is a flow diagram of the KUKA robot language (KRL) program running on the KRC on the right and the Java method used to invoke a robot movement on the left. The KRL program is essentially a control loop that waits for an external trigger.

The two programs communicate using three variables, namely, the move-start (“MOVE_START”) Boolean, the coordinate buffer (“MOVE_COORDS”) and the control integer (“CONTROL_INT”). The machine base (“MACHINE_BASE”) coordinates is another variable that is used in the KRL program. Its purpose is to store the base coordinates of machine so that component coordinates can be specified relative to it.

The move-start variable acts as a trigger to change the state of the KRL program from idle (when move-start is false), too active (when move start is true). In addition, it is used by the Java thread to check whether the KRL program is idle and

ready for the next command to be sent, or whether the KRL program is active viz. being currently busy with a previous command.

The move coordinates buffer serves as a coordinate buffer between the two programs. It is used (i) to move the robot to a global (or absolute) coordinate, (ii) to set the machine base coordinates or (iii) in conjunction with the machine base to move relative to a base coordinate.

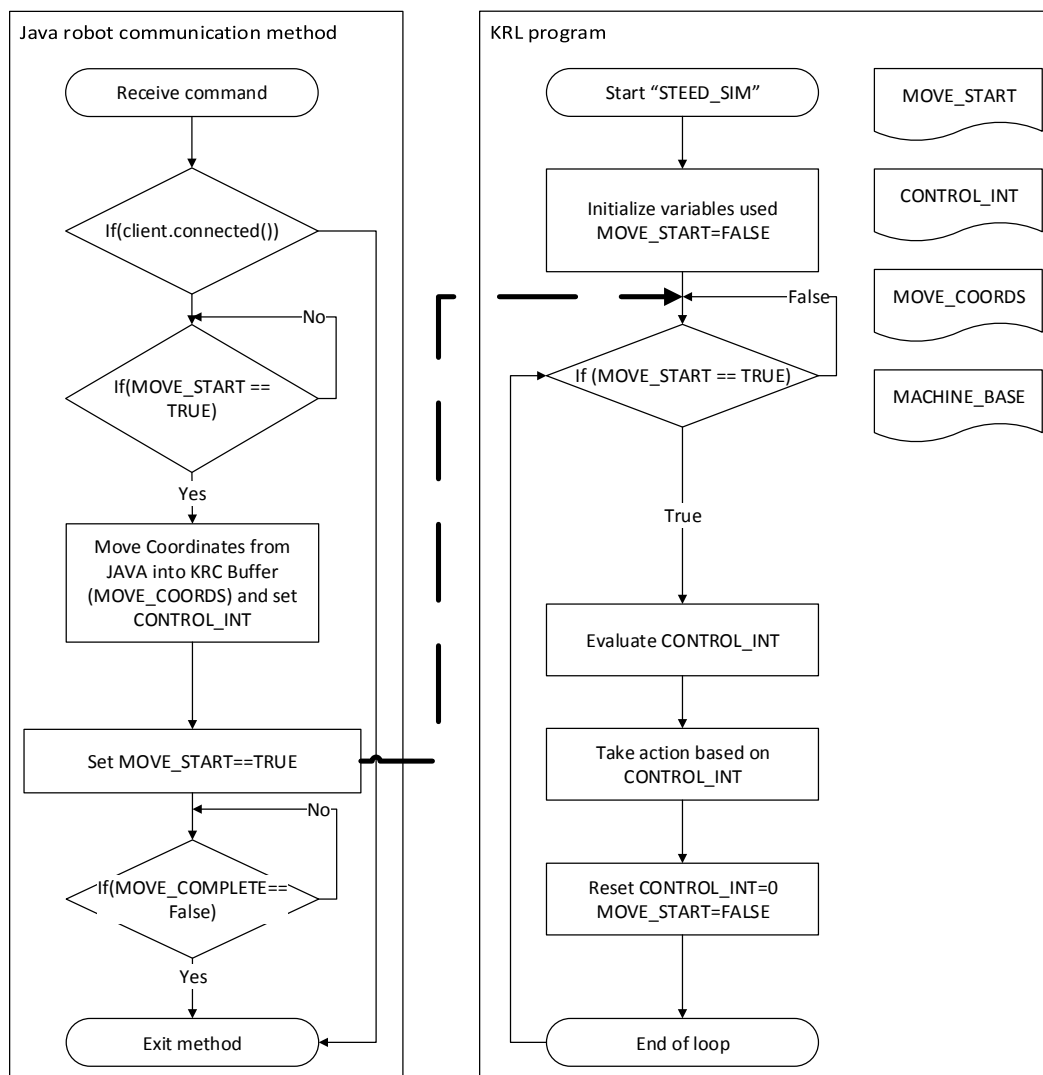


Figure 39: Robot communication between Java and KRL program.

The control integer determines the action to be taken by the KRL thread when triggered. Table 4 shows the values of the control integer and the action performed. The wait command is used when an action needs to happen at a specific point between movements, i.e. gripper actuation. This disables the KRC computer advanced run for one movement and the KRC will not calculate ahead. The “set machine-base” command moves the coordinates from the coordinate buffer into the

machine base variable within the KRC. The “move home” command moves the robot to its default position. The “move PTP absolute” command moves the robot to the position specified in the coordinate buffer using a PTP movement. The “move PTP relative” command moves the robot to a position of move-coords relative to the current set machine base using a PTP movement. This is typically used in part placement and part collection where the robot travels to part position (move-coords) relative to the SU base (machine base). The “move LIN absolute” and “move LIN relative” commands are similar to the “move PTP absolute” and “move PTP relative” respectively, performing the same action using Linear movements instead of PTP.

Table 4: Control integer description.

Control Integer Value	Description
-2	Wait
-1	Set Machine-base
0	Move Home
1	Move PTP Absolute
2	Move PTP relative to Machine-base
3	Move LIN Absolute
4	Move LIN Relative to Machine-base

A typical relative robot movement invoked by the Java thread is explained with the aid of Figure 39 in this paragraph. After invoking the movement method, the Java program checks whether it is connected to the KRC. If there is no connection, it throws an exception. If the programs are still currently connected, the Java thread assesses whether the KRC is active or idle by evaluating the move-start Boolean to be true or false respectively. If the KRL program is idle, the Java thread proceeds to set the coordinates in the move-coords buffer and sets the control integer. Once the buffer is populated and the control integer is set, the KRL program state is then set to active by switching the move-start Boolean from false to true.

The KRL program is now active and the control integer is evaluated. The appropriate action is taken resulting in either: (i) a robot movement, (ii) the setting of a machine base or (iii) the program halting until the queued movements have completed. Once the KRL control command is complete, the move-start variable is set to false and the loop is repeats.

5.8.3. Gripper control

The KUKA robot used for the work presented here does not have IO control capabilities. Consequently, an external IO control was used for the gripper control, i.e. a Beckhoff embedded PC. The Beckhoff controller communicates with Java using Beckhoff's in-house Java API.

The implementation of a separate holon for the robot end effector was considered, but decided against. A separate gripper holon would allow an additional level of modularity. However, if a more complex end effector was used, for example a welding or machining device, the communication between the robot and end effector should have a low latency and be of a flexible nature, for which holon level communication is not suited. For this reason, the gripper was implemented as a component in the holon and not a separate holon.

5.8.4. Behaviours and lifetime

The transport agent facilitates communication and manages the holon data, as stated in Section 5.6. It is a subclass of the operational agent class and it has no operate behaviour because it is always ready to supply its service, unless it is currently busy (with either an EIH inspection or transport supplying its service). The SRR and SCL behaviours are inherited from the operational agent class and implemented verbatim.

The “service supplier” behaviour is specialized and shown in Figure 40. This behaviour is used for the transport holons of both the FC and the EIH configurations. The “service supplier” behaviour resumes when the transport holon receives an inform. The inform message contains the fully populated part data. This data is used to set the machine base to the SU base coordinates in the KRC. Then the robot is moved to collect the part. Once the robot is positioned to collect the part, the gripper jaws are closed. At this point, the machine base is set to the pallet base coordinates and the robot then proceeds to place the component in the fixture. Once in position, the gripper jaws are opened and the robot moves away from the pallet. At this time, the transport holon now responds with an inform.

In EIH configurations, a telnet camera object is also a component of the transport holon and this holon has the supply inspection behaviour mentioned in Section 5.6.5.

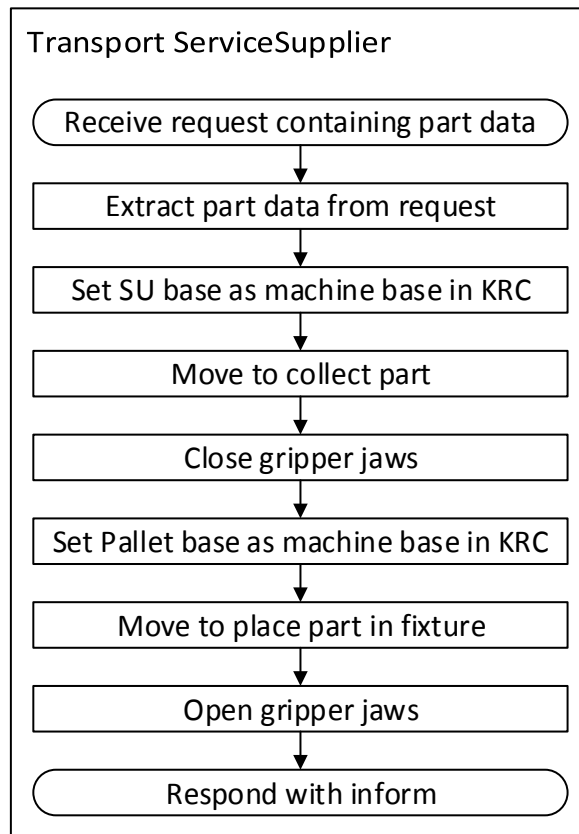


Figure 40: Transport holon service supplier behaviour.

5.9. Quality inspection holon

5.9.1. Architecture

The quality inspection holon is an operational holon that inspects the presence of the component and whether the components have been correctly placed on the pallet. Please refer to Section 5.6 for a description of the aspects common to all operational holons.

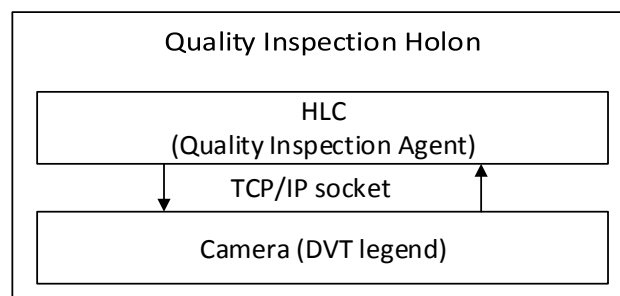


Figure 41: Quality inspection holon architecture.

The quality inspection holon architecture is illustrated in Figure 41 and consists of a quality inspection agent as an HLC, and a DVT Legend 540 smart camera as the LLC and physical entity. The camera's breakout-board has eight digital inputs and eight digital outputs, but it was not used in the implementation. The HLC and LLC communicate via Ethernet, sending strings through TCP/IP sockets.

5.9.2. Behaviour and lifetime

The quality inspection agent is a subclass of the operational agent and inherits its SRR and SCL behaviours. It does not have an operate behaviour as the agent is constantly ready to be booked.

The quality inspection agents' service supplier behaviour is specialized and shown in Figure 42. The agent receives a request containing the part data, which includes the component type. The inspection is done based on the component-type, evaluating the presence of the component. If the inspection passes, the quality inspection agent responds to the subtask agent with an inform message, while if the inspection fails, meaning components are not located in the appropriate pose, a failure is sent.

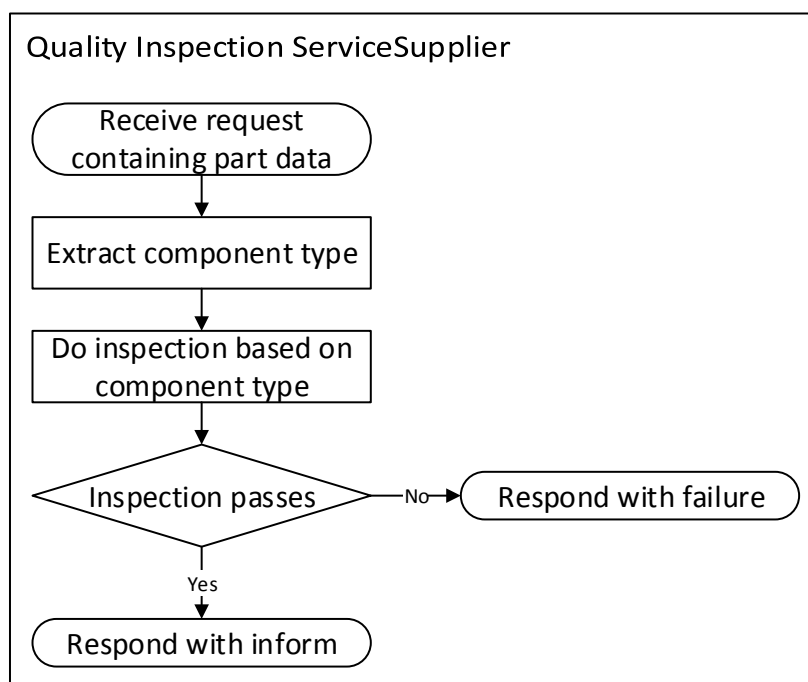


Figure 42: Quality inspection agents service supplier behaviour.

The quality inspection holon performed a visual inspection of the pallet from the side to avoid hindering robot movements. The quality inspection holon locates correctly placed components. This is done by locating components within a limited

inspection region and with a limited angle of rotation. Figure 43, shows the quality inspection of the holon identifying the load terminal.

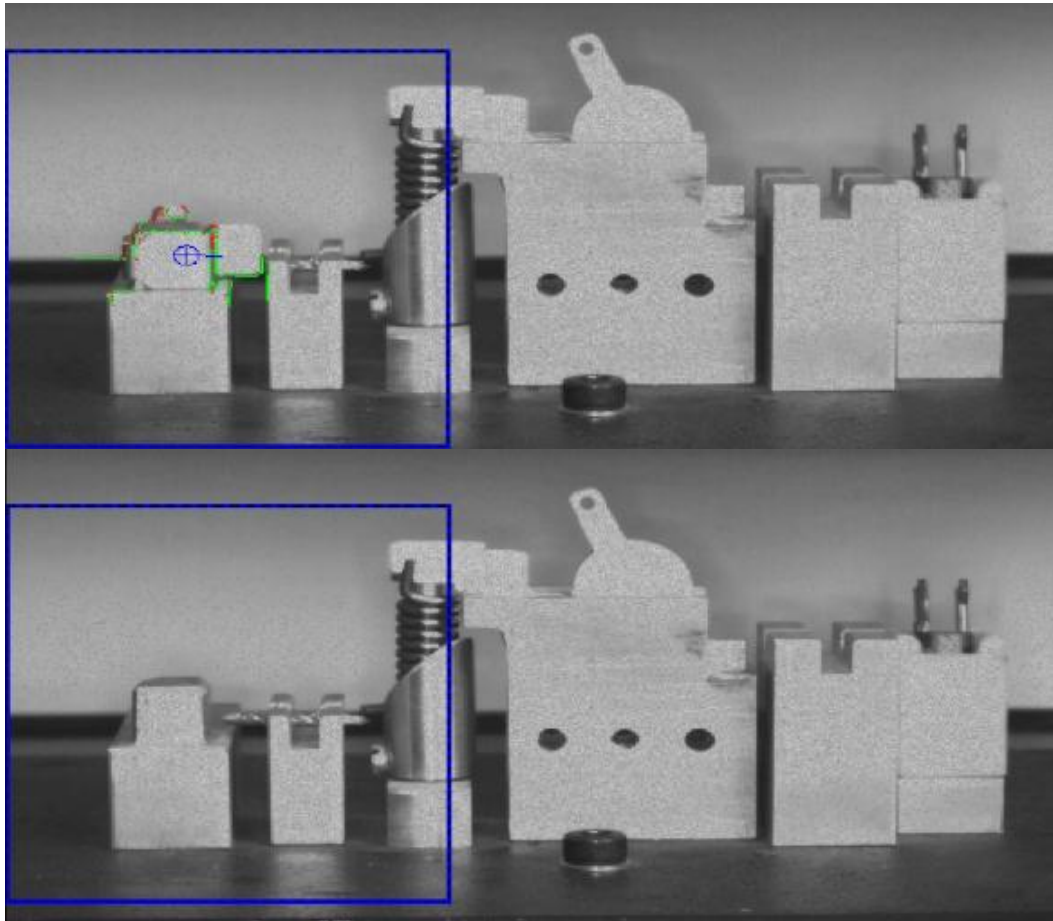


Figure 43: Quality inspection of planet.

6. Simulation model

6.1. Overview

One objective of this thesis was the development of a simulation model that could reasonably predict the throughput rate of a specific configuration. For the feeding station case study, a configuration constitutes, in addition to the FC and EIH hardware configurations, the number of singulation units, the layout of machines, the number of pallets and the singulation properties (singulation rate and successful singulation probability). This chapter considers the simulation model itself and its ability to assess FC and EIH alternatives in the context of RMSs.

In the simulation model, the physical KUKA robot was replaced with a software emulator packages, KUKA Office Lite and KUKA Simpro. These packages were designed to simulate the robot movements accurately. A test was performed whereby the robot performed simple movements and the difference in time take between the physical robot and the simulation software was found to be less than 1%. Gripper actuation time was simulated by pausing execution for a period. Vision sensors were simulated by a function returning random coordinates, within a reasonable range. SU properties were simulated mathematically, using paused execution periods for singulation time and computer generated percentage chances of collection. The simulation model allows scaling the system by adding or removing SU and task holons.

The remainder of the chapter describes various experiments that were performed using the simulation model. The first experiment was conducted to validate the simulation model, by comparing the average throughput rate of the physical test system to that of the simulation. The subsequent simulations were performed to compare FC and EIH configurations.

6.2. Experiment 1: Validation of simulation model

The first experiment was performed to validate the simulation model. In this experiment, the average feeding time for the physical and simulation configurations were compared. Task (or pallet) holons required two component be fed, namely a load terminal and a pigtail. The experiment measured the time taken for a task or pallet to complete and compared the results of the physical system to the simulation model. The operational holons used in this experiment were (i) one SU supplying the load terminal component, (ii) one SU supplying the pigtail and (iii) one transport holon.

6.2.1. Fixed camera test

The tests evaluated the feeding of a load terminal and pigtail as a task. The FC load terminal singulation was simulated by manually singulating components on the

inspection platform where the smart-camera (Cognex) inspected for collectable components. The FC pigtail singulation was simulated using a component magazine that presented the pigtails in fixtures, thereby not needing a smart-camera.

As shown in Table 5, three configurations were employed during this experiment. The first test configuration used all the physical entities. In the second test, the KUKA (KR-16) robot and the KRC (4) were replaced with the KUKA Simpro package, whereas the other hardware components were used as is. In the final test, the robot, as well as the camera and gripper controller, was replaced. The camera was simulated by supplying random part coordinates within a small range. Opening and closing of the gripper was simulated by waiting a specified period. During all three tests, the robot moved at 30 percent of its maximum speed.

Table 5: Hardware configurations fixed camera validation experiment.

	Physical test	Simulation with hardware test	Simulation with no hardware test
Transport holon	KUKA robot	Simulated	Simulated
Camera	Cognex ism-1100	Cognex ism-1100	Simulated
Gripper controller	Beckhoff embedded PC	Beckhoff embedded PC	Simulated

The results from the experiment are tabulated in Table 6. The outcome shows that the average feeding time for the simulation resembles that of the physical setup, with a resulting error below 5%. Java logging was used to measure execution time.

Table 6: Fixed camera validation experiment results.

	Average feeding time (seconds)	Percent error
Physical	16.0	-
Simulation with hardware	15.4	3.4
Simulation with no hardware	15.7	1.8

6.2.1. EIH system test

The same experiment done above was repeated for the EIH configuration and the resulting error was found to be 7.25%. The difference in error between the EIH and FC tests may be because of the movement commands used since the FC robot program used more movement commands as it had to avoid collision with the camera bracket.

6.2.2. Robot movement scaling

Initial testing revealed that the error between simulation and physical experiments scaled with robot movement speed. This section describes the investigation into this effect.

The feeding test mentioned above was repeated, scaling the robot movement speed from 30% to 75%, where 75% is assumed the eventual operating speed. The simulation was repeated using different computer hardware configurations, shown in Table 7. In the initial test (simulation A) the feeding station controller (JADE agent platform), the KRC emulator (KUKA officelite) and the KUKA robot simulator (KUKA Simpro) ran on the same PC (Machine A). It was suspected that because of fewer physical (Ethernet) connections, less physical encoding and decoding took place compared to the physical test. For this reason, these physical connections were simulated by having the software platforms running on distinct machines as in simulation B. Here, the feeding station controller was moved to a different machine, and this reduced the 75% speed error from 19.4% to 15.9%.

Table 7: PC hardware specification during experiments

	Simulation A	Simulation B	Simulation C
Machine A Intel i7 3770 (4x3.2GHZ) 8GB ram (1300mhz)	KUKA office lite (KRC simulator)	KUKA office lite (KRC simulator)	KUKA Simpro
	KUKA Simpro	KUKA Simpro	
	JADE agent platform		
Machine B Intel i7 2670m (4x2.6Ghz) 8GB ram (800mhz)	N/A	JADE agent platform	JADE agent platform
Machine C Intel Pentium D (2x3.4Ghz) 512 Mb RAM (500Mhz)	N/A	N/A	KUKA office lite (KRC simulator)

Next simulation C was performed in which the KRC emulator (KUKA office lite) was relocated to a considerably slower PC (Machine C). The resulting simulation

had reduced error, from simulation A's 19.4% to simulation C's 9.8%. This showed that the KRC host machine's computational power could substantially affect the simulations performance.

The simulation results are illustrated in Figure 44. The figure additionally contains data-points for a 75% robot-movement speed test for Simulation B and C in which different PC hardware configurations were used.

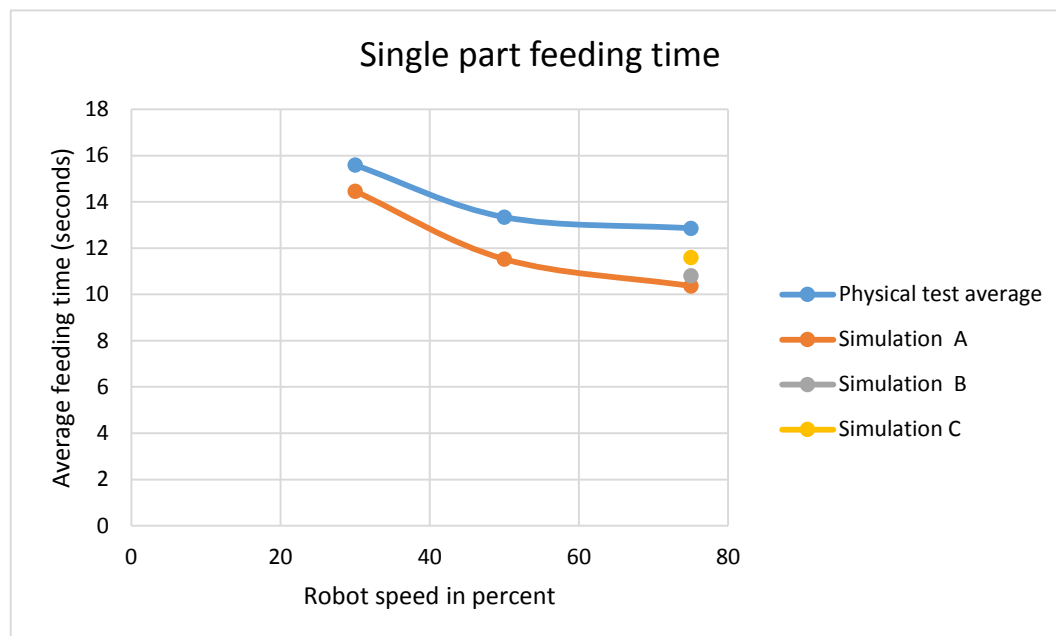


Figure 44: Graphical results from scaling the robot speed during simulation validation experiment.

6.2.3. CPU effect on KRC controller

Another experiment was performed investigating the effect of a burdened KRC CPU on the simulations' feeding time. Simulation C was run again measuring the feeding time and the PC running the KRC emulator (KUKA officelite) was burdened at random intervals. Memory intensive programs were started at random periods to investigate the effect of a burdened CPU on the simulation performance.

Figure 45 shows the feeding time for simulation C with a burdened CPU. It can be seen that burdening has a substantial effect on the performance of the simulation. This was noted when all simulation were performed, by keeping processes running on the simulation PC to a minimum.

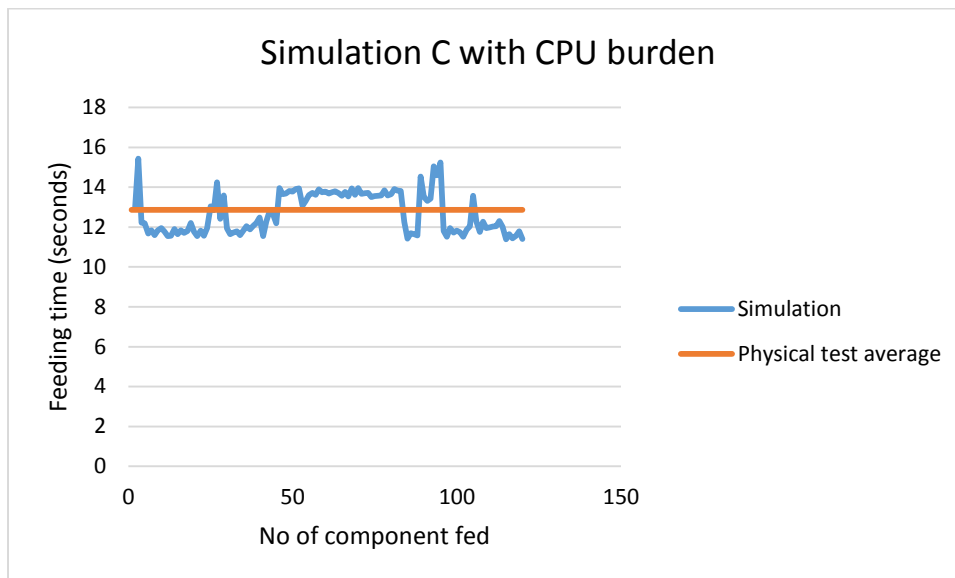


Figure 45: Graph showing simulation C with CPU burden.

The findings in this experiment show that an error ranging from approximately positive 10% to negative 10% was found (in simulation C) and that the effects of a burdened CPU can substantially affect the simulation results. However, the simulation model can still be used to compare EIH and FC hardware configurations, as it is expected that the error will scale with both configurations.

6.3. Simulation 2: Influence of SU performance characteristics

In this simulation, the simulation model predicts the influence of the SU performance characteristics on the system's throughput rate. SU performance characteristics investigated were the singulation rate and the probability of a singulated component landing in a collectable pose (or probability of a successful singulation). It can be assumed that the SU performance characteristics are a function of the singulation method and component being singulated. It is expected that different methods of singulation (e.g. manual, stepped conveyor or tumbling barrel) will have different results for different component geometries. In addition, components with many non-collectable landing-orientations will have a lower probability of successful singulation. Reconfigurable singulation methods have not been extensively investigated and the only available data (singulation period and probability of successful singulation) was that of Kruger (2013) from tests on a stepped conveyor feeder. The data is shown in Table B.2 in Appendix B.

In this test, task (or pallet) holons again required two components be fed, namely a load terminal and a pigtail. The operational holons used in this simulation were (i) three SU supplying the load terminal component, (ii) three SU supplying the pigtails and (iii) one transport holon. Figure 46 shows the layout of the holons, noting that only Task 1 was used in this simulation. The coordinates are tabulated in Appendix B 1.

This simulation assumed one task is always present in the system and the next one is initiated immediately after the previous one is completed. This could be the case if more than one task is present in the system, as pallets need to be transported in and out of the feeding station.

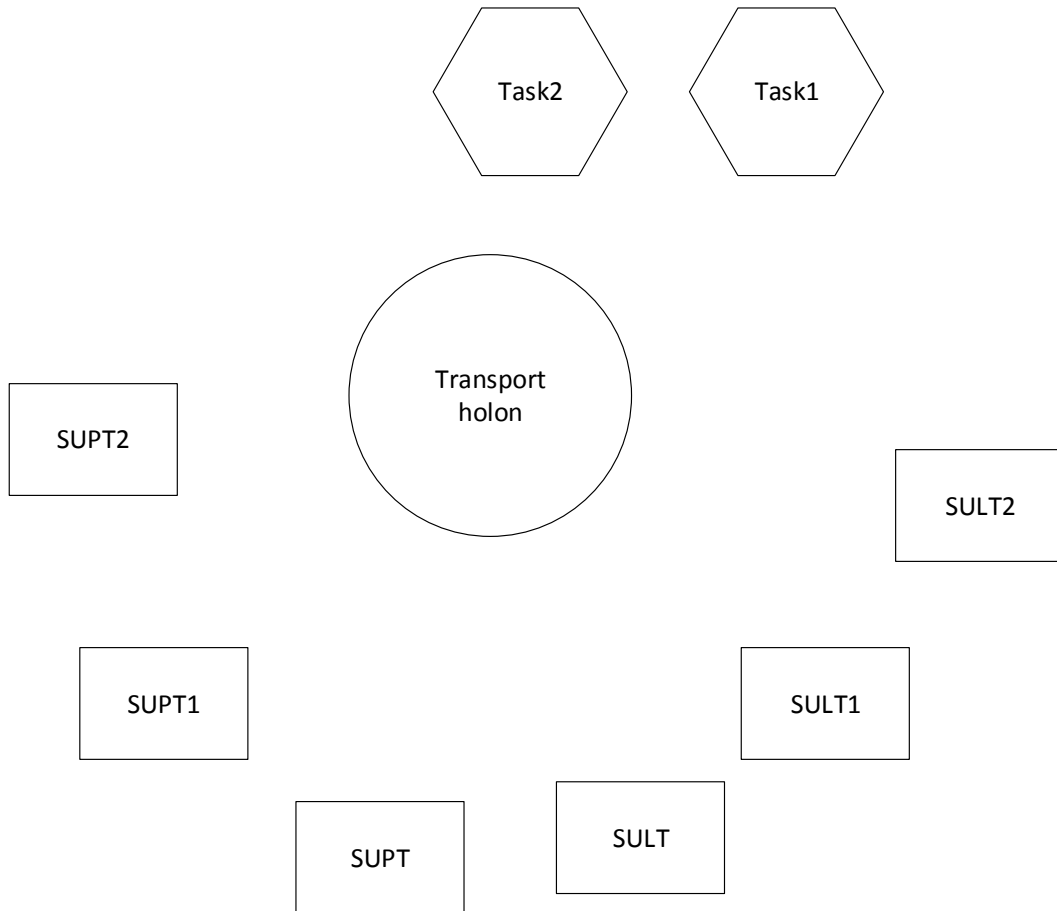


Figure 46: Layout of holons Simulations 2 and 3.

Figure 47 shows the throughput rates measured in this simulation for various combinations of time per singulation and probability of the singulated part being in a collectable pose. Note that for singulation rates of 1.2 s and 1.8 s the FC test were not performed since the throughput can be assumed to be saturated at these points, meaning at least one component of every component-type is always ready for collection when needed.

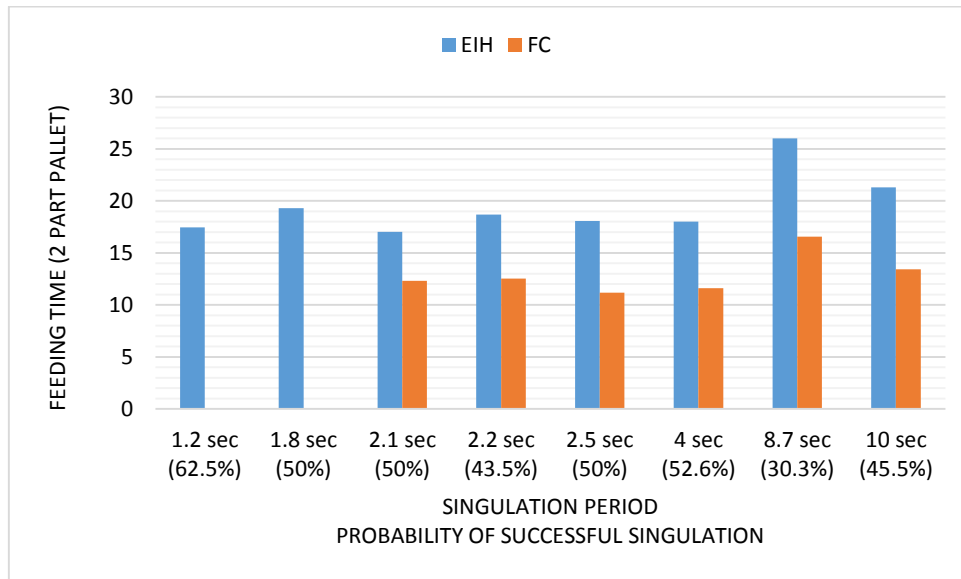


Figure 47: Throughput rate of system for various SU properties.

The simulation showed little dependence of SU performance on feeding time for the most part, with small variations in feeding rates in all FC cases, except at 8.7 s, and all EIH cases, except 8.7 and 10 s. This could be due to the system being saturated. The FC configuration has a noticeably lower feeding time, as would be expected for a saturated system, since the EIH transport holon still has to perform inspections while the FC transport holon does not. The observed longer feeding time at an 8.7 s singulation rate, compared to the 10 s singulation rate, is likely due to the lower probability (30.3 %) of successful singulation. This indicates that the tested configuration may be sensitive to the average effective singulation rate. The condition of saturation is investigated in Section B 2

The system cost was estimated, as shown in Table B.3. This allowed a comparison of the system price to throughput ratio for the FC and EIH systems. Figure 48 shows the throughput rate per unit cost for the system configuration described in Simulation 2. It can be seen that for this simulation's, the FC configuration appears to perform better than the EIH configuration with regard to throughput per unit cost comparison, but only slightly. This is due to the additional cost of the vision systems required for each SU and for quality inspection in the FC configuration, which does not occur in the EIH configuration. Consequently, this may not be a worthwhile investment considering the FC system cost 30% more than the EIH system.

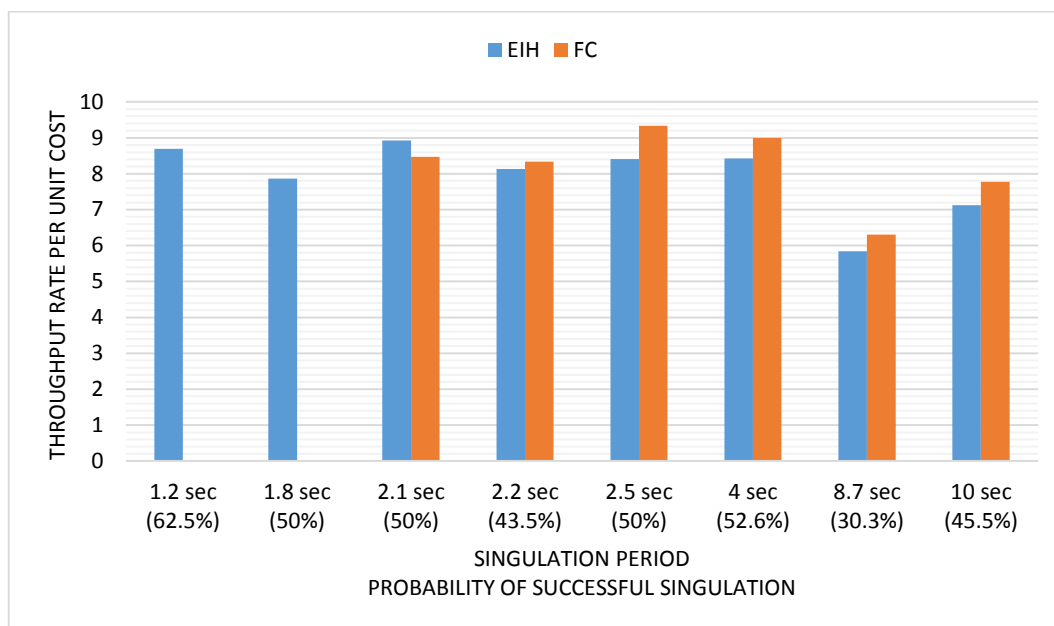


Figure 48: Throughput rate per unit cost for system Simulation 2.

The results of Simulation 2 are specific to the SU properties extracted from Kruger's (2013) simulations and these results may be highly dependent on the properties of the SU. During most of the simulation, the system appears to be saturated, yielding little meaningful variation between tests. This simulation does reveal that in this case, the EIH system configuration may be a better solution due to its similar throughput per unit cost and considerably lower capital cost.

6.4. Simulation 3: Task scaling

In this simulation, the model from the previous simulation was used to determine the influence that the number of pallets being fed would have on the system's performance. Both FC and EIH configuration were tested. This simulation was aimed at assessing the simulation's ability to guide such a choice, by investigating the effect of adding a pallet (corresponding to a task holon).

6.4.1. Simulation 3A: Task scaling without implicating transport time

This simulation's schematic layout is also as shown in Figure 46. The simulation tested the pallet feeding time for two components per pallet, evaluating the feeding time when one task was in the system, relative to when two are present. The SU performance of 10 s singulation time and 45 % probability of successful singulation, mentioned in the previous section, was used here. A saturation case with one pallet (also 10 s singulation time, but with a 100 % probability of successful singulation) was also simulated for both FC and EIH systems.

Figure 49 shows the average time taken to complete the feeding for one pallet for each configuration. It can be seen that the EIH (both 1 task and 2 tasks) configurations are far from saturation, whereas the FC configurations appear closer. Both configurations' feeding time benefit little from the additional task in the system.

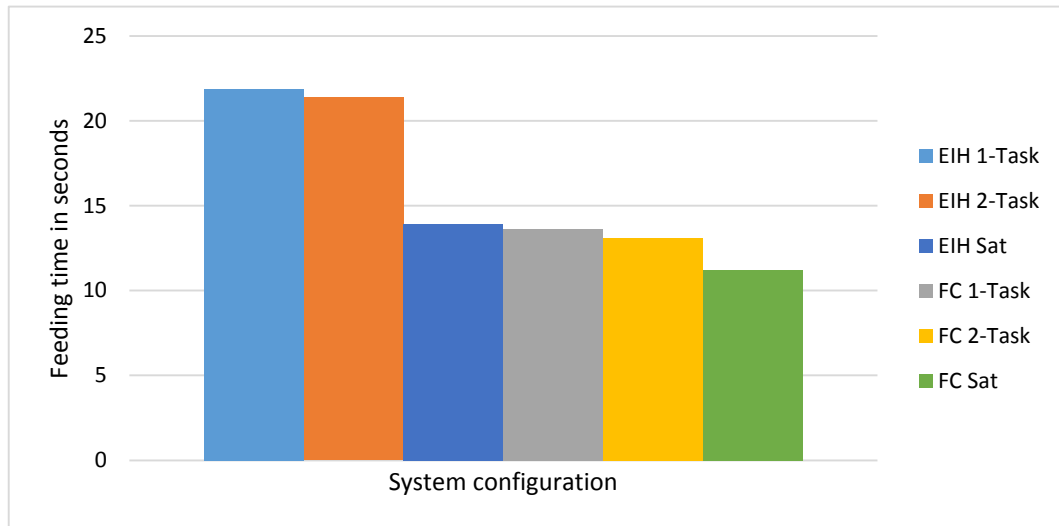


Figure 49: Feeding time for simulation 3 with no transport time.

Figure 50 shows that the EIH configuration benefits slightly, in terms of throughput rate per unit cost, from having an additional task, while the FC configuration does not. Furthermore, the cost of an additional QI camera for an additional pallet negatively affects the FC configurations throughput rate per unit cost. It also becomes evident from this figure that the EIH saturation tests' throughput per unit cost is substantially higher than other configurations. This supports an earlier mentioned (Section 3.2.3) expected result that an EIH system with a high probability of SS would be desirable.

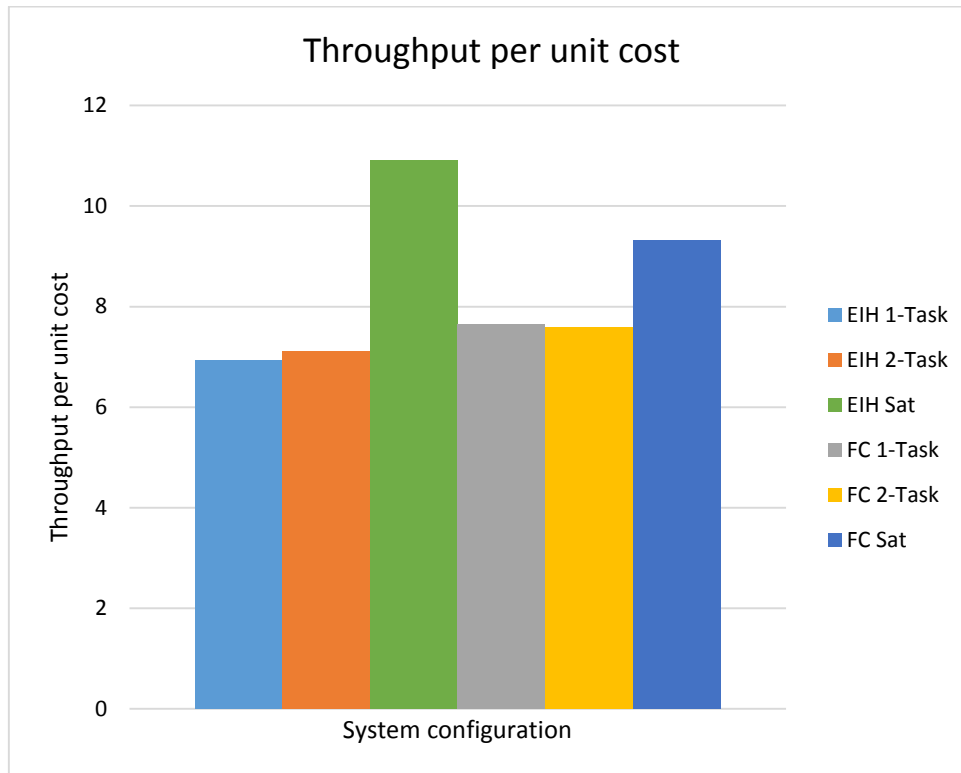


Figure 50: Throughput rate to cost ratio results simulation 3 with no transport time.

6.4.2. Simulation 3B: Task scaling with the implication of transport time

The previous simulation was repeated, but transport time for the pallet was also taken into account. The transport time refers to the time taken to remove a pallet from the feeding station and replace it with a new pallet. Pallet transport time was simulated by introducing a wait period between the completion of a task and the start of another. The occurrence of significant transport times is one of the main motivators for adding additional task. Figure 51 shows the average feeding time for the system with a 12 s transport time. The transport time was selected to be longer than the feeding time for a task in a saturated system.

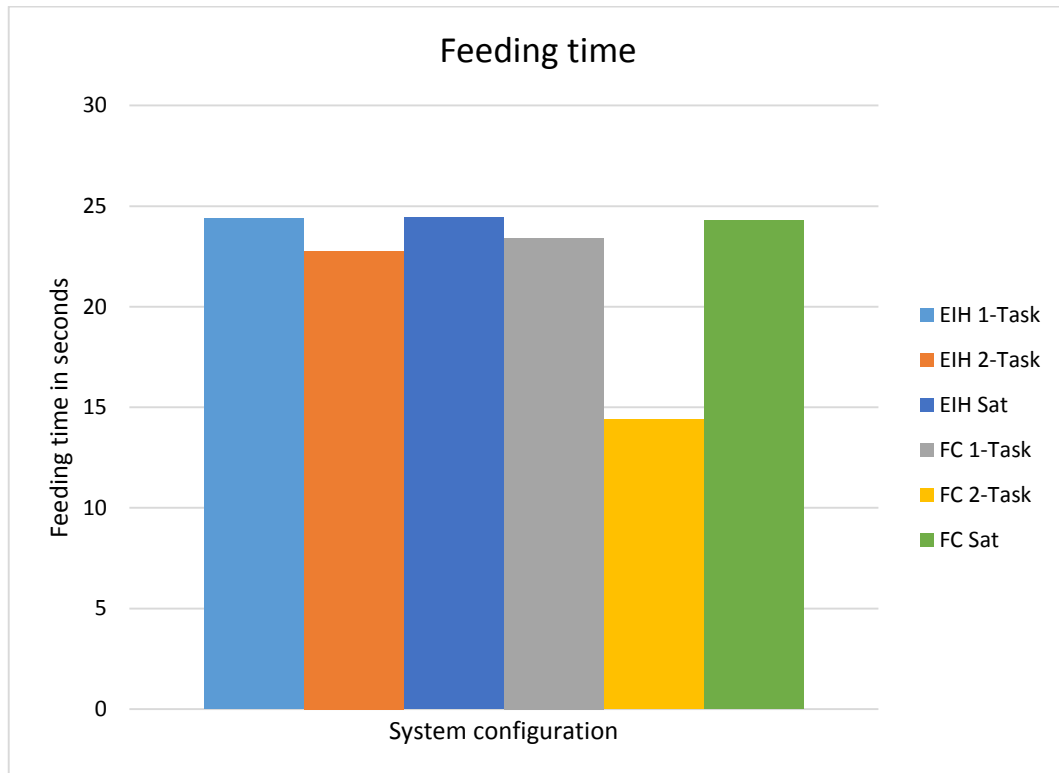


Figure 51: Feeding time for simulation 3 with 12-second transport time.

Comparing the average feeding time for Simulation 3 performed with no transport time (Figure 49) and a 12 second transport period (Figure 52), it is evident that the feeding time of all systems tested, increases. As expected, the FC sat and FC 1 task configurations has the largest relative increase in feeding time, as the transport duration is sequentially added to their feeding time. The FC 2 task feeding duration was only affected slightly, as the second task acts as a buffer.

The EIH sat and EIH 1 task yielded the same feeding time showing that with the added transport time EIH 1 task become saturated. Making use of two tasks benefited both the EIH and FC configurations by decreasing their relative feeding times. This was an expected result as the one task can be attended to while another is in being initiated or completed (pallet being transported).

Figure 52 shows the throughput rate per unit cost of simulation 3B. By comparing the throughput per unit cost from simulation 3A (Figure 50) and 3B (Figure 52), it is interesting to see how the FC configuration benefits from an additional task, where the EIH configuration does not benefit as much. The EIH systems offer lower throughput per unit cost, but are still an option considering their lower capital investment.

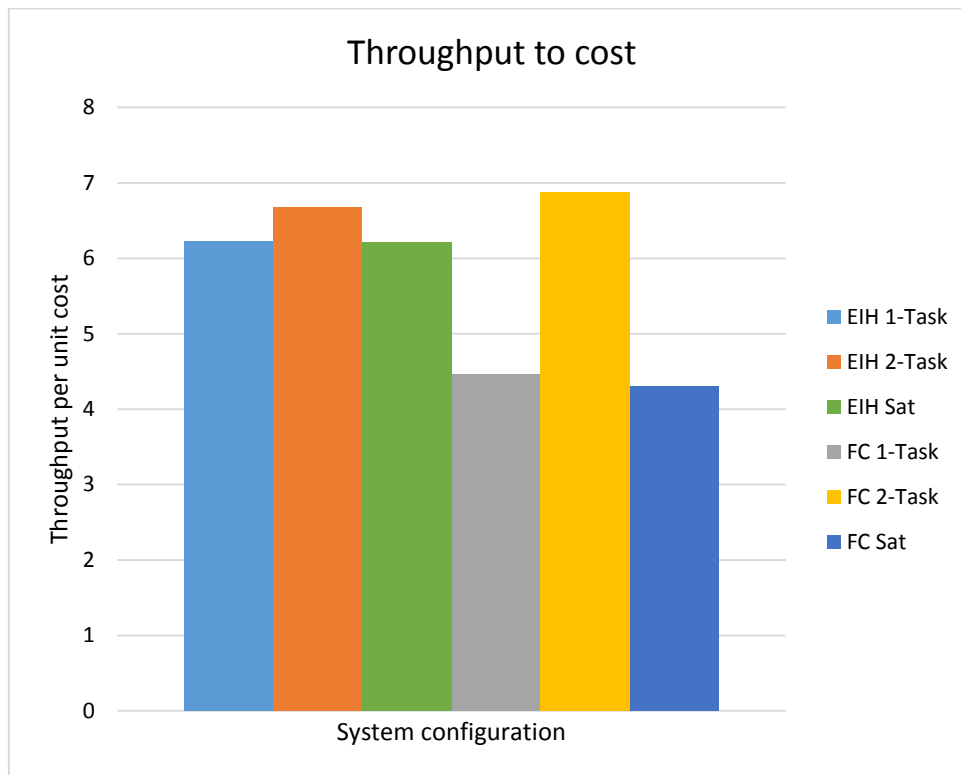


Figure 52: Throughput rate to cost ratio results simulation 3 with 12-second transport time.

6.5. Assessment of experimental results

The experiments shows that the simulation can produce meaningful results that can be explained within reason, and quantify the performance of alternative configurations. Noting that these were simple cases, a complicated simulation could involve numerous components per task and numerous SUs with individual properties. Such a system could benefit in the design stages from pre-implementation simulations, such as this one. The simulation is useful for comparing EIH and FC configurations as it is expected errors in the simulation will scale similarly with the two configurations. The simulation model also allows for scaling of the system by adding additional SU and task holons.

Experiment 1 shows that although the simulation accurately models robot dynamics, feeding times for the physical system differ from those of the simulation. This appears to be due to the KRC simulation being affected by CPU performance. Other environmental factors that cannot be simulated should also be considered, like network communication performance.

Simulation 2 shows FC configurations have a higher throughput rate. This is easily explained by the fact that in the FC configuration the robot only transports components and SUs singulate components autonomously, where in the EIH

configuration the robot also assists in visual inspections. The simulation also showed that the EIH and FC configurations had similar performance (for the SU properties used here) when comparing the throughput rate per unit cost of the system, with the FC configuration performing slightly better. It should be noted that the EIH configuration has a considerably lower system cost. Also, further analysis (Appendix B) reveals that the FC configuration handle low probability of successful singulation and low singulation periods better than EIH SUs do.

Simulation 3 showed both the FC and EIH configuration take advantage of an additional task (pallet) in the system. Similar throughput rate per unit cost was achieved by the two-task EIH and FC configurations. This simulation also revealed that the one-task FC configuration is not a good option as the robot is idle much of the time. A better solution would be a one-task EIH configuration as the EIH transport holon can perform inspection while waiting for the pallet to be transported into the station. This simulation again shows the two-task EIH and two-task FC configurations having similar throughput rate per unit cost with the EIH configuration having considerably lower capital cost.

7. Conclusions

The research documented in this thesis investigates vision aided robotic feeding systems, within the context of reconfigurable manufacturing systems. The research is focused on comparing EIH and FC configurations for such a feeding station. The objective of the research was to develop a simulation tool would allow a reasonable prediction of the systems behaviour. The development of the simulation tool included the creation of an agent-based control system for a reconfigurable feeding station.

As a case study for validating the simulation, the control system was implemented on a feeding station of an experimental assembly cell at the University of Stellenbosch. The feeding station assembles components of circuit breaker. In a laboratory implementation used for validating the simulation model, collection platforms on which parts were placed manually, were used to emulate reconfigurable singulation units. The physical implementation further included two pallets to receive parts, a part magazine and a six-degree of freedom robot. The case study implementation was developed for two configurations, namely an EIH and a FC configuration. For the FC configurations, a fixed camera was provided above the collection platform and the pallets. While for the EIH configurations, a camera was attached to the end effector of the robot arm. The implementation fed only two component types, but the addition of new component types would be simple, provided the components can be collected approaching from the top and the current gripper can be used.

A holonic control system was developed for the feeding station controller, with consideration of the six core characteristics of RMSs, i.e. customisability, configurability, scalability, modularity, integrability and diagnosability. The control system included a multi-agent system as the high-level controller, in which an agent was created for each holon, thus reflecting modularity. ADACOR was used as the reference architecture, with holon responsibilities being modified as needed for the application. This resulted in a control system with the responsibilities divided between the coordinator, supervisor, task, subtask and operational holons.

Integrability was achieved by interfacing the HLC to numerous LLCs using TCP/IP through Ethernet. The control system was further able to scale the number of pallets or tasks and singulation units in the system, thus demonstrating elements of scalability. This was largely made possible by using agents as HLC and the use of the directory facilitator. Rapid integration of new modules, such as additional SUs, was achieved with the aid of JADE agents as a HLC, the use of a modular communication interface and the placement of base coordinates in the agents' code.

Error handling and diagnosability was limited in the present implementation to inspecting (using an EIH or a FC configuration) whether the parts fed by the feeding station were correctly placed. Although the agent platform's communication allows subscribers to be informed of events, which allows diagnostic information to be

communicated and stored, this was not implemented since it does not affect the FC vs EIH comparison. In addition, ADACORs error handling was not implemented here, as the system does not have sufficient redundancy.

In addition to the physical implementation described above, a simulation was also developed in which the physical components were replaced either by external software packages (e.g. Kuka Sim was used to replace the Kuka robot) or by user written functions. The simulation emulated the singulation rate and successful singulation probability of reconfigurable SUs, yielding reasonable results in sample experiments. The ability of the simulation to compare EIH and FC configurations for a number of cases was demonstrated. The results showed that in the particular cases considered, using typical SU performance parameters, a FC station configuration yields a higher throughput rate and a higher throughput rate per unit cost. However, in many cases the EIH configuration offered competitive performance with lower capital investment.

Although this was not explored fully in the thesis, the experiments conducted showed that the simulation model is a useful tool, not only for simulation, but also for development of a physical station. A CAD environment can be used to build a detailed configuration in KUKA Simpro, where the robot movements and collision detection can be programmed. This is especially useful when developing a FC station, since a robot collision with a vision sensor can be very costly. The experiments also showed how hardware in the loop simulations allow the development or integration of other modules (e.g. replacing the camera with another brand or with simulation object) can be done without full station deployment.

The main limitation in the simulation, and therefore an area for further work, is in diagnosing errors and handling those errors. However, the implementation of such measures is highly application-specific. The importance of handling the errors also depends on the frequency and cost of the errors, to that the importance may range from insignificant to critical.

Some functionality offered by EIH configurations and not FC configurations, that should be further investigated, is the use of EIH robots for remote visual monitoring, allowing specialist supervision remotely. This functionality will require many safety measures if implemented using an industrial robot and would require investigation into a means of low-level safety mechanisms involving the robot. In this implementation, low-level robot safety mechanisms are programmed on the robot controller and cannot be edited externally (as with Java or JADE in this implementation).

The use of the smart camera to control a singulation unit should be investigated. This may drive down the cost of FC SU substantially making them more competitive in comparison to EIH SUs

8. References

- Bellifemine, F., Caire, G. & Greenwood, D., 2004. *Developing multi-agent systems with Jade*. Chichester, West Sussex, England: Wiley.
- Bi, Z., Lang, S., Shen, W. & Wang, L., 2008. Reconfigurable manufacturing systems: the state of the art. *International Journal of Production Research*, Vol. 46(4), pp.967-992.
- Borangiu, T., Gilbert, P., Ivanescu, N.-A. & Andrei, R., 2009. An implementing framework for holonic manufacturing control with multiple robot-vision stations. *Engineering Applications of Artificial Intelligence*, Vol. 22(4-5), pp.505–521.
- Braggins, D., 2006. Vision today for assembly automation. *Assembly Automation*, Vol. 26(3), pp.181-183.
- Cognex Corporation, 2013. *COGNEX: In-sight micro systems*. [Online] Available at: <http://www.cognex.com/PBFourColumnWireFrame.aspx?pageid=11474&rdr=true&langtype=1033> HYPERLINK
"http://www.cognex.com/PBFourColumnWireFrame.aspx?pageid=11474&rdr=true&langtype=1033"
<http://www.cognex.com/PBFourColumnWireFrame.aspx?pageid=11474&rdr=true&langtype=1033> [Accessed 06 Augustus 2013].
- Edwards, L. & Golub, S.S., 2004. South Africa's international cost competitiveness and exports in manufacturing. *World Development*, Vol. 32(8), pp.1323-1339.
- Gascon, B.R. & Barraza, R.M., 2012. Six DOF stereoscopic eye-in-hand visual servo system BIBOT. In *Robotics Symposium and Latin American Robotics Symposium (SBR-LARS), 2012 Brazilian*. Fortaleza, 2012. IEEE pp.284 - 289
- Harrison, R., Colombo, A.W., West, A.A. & Lee, S.M., 2007. Reconfigurable modular automation systems. *International Journal of Flexible Manufacturing Systems*, Vol. 18(3), pp.175–190.
- Herakovic, N., 2009. Robot vision in industrial assembly and quality control processes. In A. Ude, ed. *Robot Vision*. Intech. pp.501-535.
- Hu, S.J., 2013. Evolving paradigms of manufacturing: from mass production to mass customization and personalization. *Procedia CIRP*, Vol. 7, pp.3-8.
- Jones, R.E. & Hage, P.M., 1985. Visual feedback control for orientating parts in an assembly robot cycle (chapter 13). In J. Billingsley, ed. *Robots and Automated Manufacture*. Stevenage UK: Iet. pp.173-184.

- Kermorgant, O. & Chaumette, F., 2011. Multi-sensor data fusion in sensor-based control: application to multi-camera visual servoing. *2011 IEEE International Conference on Robotics and Automation*, Vol. 22(3), pp.4518-4523.
- Khan, M.U., Iqba, N. & Jan, I.u., 2009. Vision based system for camera tracking in eye-in-hand configuration. In *Proceedings of the 7th International Conference on Frontiers of Information Technology - FIT '09*. New York, New York, USA, 2009. ACM Press pp.1-4 Article No. 76
- Koestler, A., 1967. *The Ghost in the Machine*.
- Kopparapu, 2006. Lighting design for machine vision application. *Image and Vision Computing*, Vol. 24(7), pp.720-726.
- Koren, Y., Heisel, U., Jovane, F., Moriwaki, T., Pritschow, G., Ulsoy, G. & Van Brussel, H., 1999. Reconfigurable manufacturing systems. *CIRP Annals Manufacturing Technology*, Vol. 48(2), pp.527-540.
- Koren, Y. & Shpitalni, M., 2011. Design of reconfigurable manufacturing systems. *Journal of Manufacturing Systems*, Vol. 29(4), pp.130–141.
- Kruger, K., 2013. *Control of the feeder for a reconfigurable assembly system*. MSc.Eng (Mechatronics) Thesis. University of Stellenbosch.
- Kruger, K. & Basson, A.H., 2014. *A Stepped Conveyor Singulator Design for Reconfigurable*. Internal Report. Stellenbosch University Department of Mechanical and Mechatronic Engineering, Stellenbosch.
- KUKA Robot Group, 2002. KRC Expert programming guide, KUKA software systems (KSS). Germany: KUKA Roboter GmbH.
- Leitao, P., 2009. Agent-based distributed manufacturing control: A state-of-the-art survey. *International Journal of Production Research*, Vol. 46(4), pp.967-992.
- Leitao, P. & Restivo, F., 2005. ADACOR: A holonic architecture for agile and adaptive manufacturin control. *Computers in Industry*, Vol. 57, pp.121-130.
- Oda, N., Ito, M. & Shibata, M., 2009. Vision-based motion control for robotic systems. *IEEE Transactions on Electrical and Electronic Engineering*, Vol. 4(2), pp.176-183.
- Piepmeyer, J.A., Gumpert, B.A. & Lipkin, H., 2002. Uncalibrated eye-in-hand visual servoing. In *ICRA.*, 2002. IEEE pp.568-573
- Scaggs, T.E., 1993. Robotic visual servoing in a flexible manufacturing workcell. In *Electrical Electronics Insulation Conference and Electrical Manufacturing & Coil Winding Conference, 1993. Proceedings., Chicago '93 EEIC/ICWA Exposition*. Chicago, IL, 1993. EEIC/ICWA Exposition pp.413 - 417

Shirai, Y. & Inoue, H., 1973. Guiding a robot by visual feedback in assembly tasks. *Pattern Recognition*, Vol. 5(2), pp.99-108.

Spong, M.W., Hutchinson, S. & Vidyasagar, M., 2006. *Robot modelling and control*. 1st ed. Hoboken: John Wiley & Sons.

Van Brussel, H., Wyns, J., Paul, V., Bongaerts, L. & Peeters, P., 1998. Reference architecture for holonic manufacturing systems. *Computers in Industry*, Vol. 37(3), pp.255-274.

Vrba, P., Tichy, P., Marik, V., Hall, K.H., Staron, R.J., Maturana, F.P. & Kadera, P., 2011. Rockwell automation's holonic and multiagent control systems compendium. *IEEE Transactions on Systems, Man, and Cybernetics-part C: Applications and Reviews*, Vol. 41(1), pp.14-30.

Vyawahare, V.S. & Afzulpurkar, N.V., 2006. Uncalibrated eye in hand visual servoing for fixtureless assembly automation. In *ICIT 2006. IEEE International Conference on Industrial Technology*. Mumbai, 2006. IEEE pp.2244 - 2249

Zhang, B., Wang Jianjun, Rossano, G., Martinez, C. & Kock, S., 2011 [2]. Vision-guided robot alignment for scalable, flexible assembly automation. In *International Conference on Robotics and Biometrics*. Phuket Thailand, 2011 [2]. IEEE pp.944-951

Zhang, B., Wang, J., Rossano, G. & Martinez, C., 2011. Vision-guided robotic assembly using uncalibrated vision. In *Proceedings of the 2011 IEEE International Conference on Mechatronics and Automation*. Beijing, China, 2011. IEEE pp.1384 - 1389

Zorcolo, A., Escobar-Palafox, G., Gault, R., Scott, R. & Ridgway, K., 2011. Study of lighting solutions in machine vision applications for automated assembly operations. *IOP Conference Series: Materials Science and Engineering*, Vol. 26, pp.12-19.

Appendix A Agent services

Each holon and consequentially each agent, supplies a service used during production. These services are shown in Table A.1. From the table it can be seen that some services are component specific, for example singulation and subtask services, while others are not component specific, for example transport and inspection services. This system therefore assumes that the non-component specific services can handle all parts in the system.

Table A.1: Agent services supplied.

Agents	Services
Supervisor	“FeederSupervisor”
Pallet	None
Pig Tail (Subtask)	“PTTask” - Feed pigtail
Load Terminal (Subtask)	“LTTask” - Feed load terminal
Singulation (Load Terminal)	“SULT” - Supply load terminal
Singulation (Pig Tail)	“SUPT” - Supply pigtail
Quality Inspection	“QInspection” - Inspect pallet-type
Transport (fixed camera)	“Robot” - Transport
Eye-in-hand Robot (camera)	“EIHInspection” - supply inspection (component-specific) + “Robot” + “QInspection”

It should be noted that a singulation unit can supply more than one service in terms of part type, but that was not explored in this thesis. Singulation units also supply the same services whether they have a fixed camera or require inspection from the transport (EIH) agent, in which case they request the “EIHInspection” service. This allows for a hybrid configuration with both EIH and FC SU holons, provided there is an EIH transport holon to supply the EIH inspections to the requesting SUs. Collision of the cameras must be considered.

Appendix B Simulation experiment additional information

B 1. Simulation configuration data

Table B.1 shows the base coordinates of the holons used in the simulation experiments. The corresponding layout shown in Figure 46.

Table B.2 shows the SU performance values used in experiment 2 and 3. These values are results obtained by Kruger (2013) through physical testing of a SU.

Table B.1 Holon base coordinates used for simulation experiment.

	X coordinate (mm)	Y coordinate (mm)
Transport holon	0	0
SULT	340.5	-1005
SULT1	759.5	-700
SULT2	1109.5	-250
SUPT	-250	-1050
SUPT1	-740	-700
SUPT2	-900	-100
Task holon1	671.5	690.5
Task Holon 2	90.5	690.5

Table B.2 Singulation unit performance values, extract from Kruger (2013, p.89).

Average singulation time (seconds)	10	8.7	4	2.5	2.1	2.2	1.8	1.2
Singulation success rate (%)	45.5	30.3	52.6	50.0	50.0	43.0	50.0	62.5

Table B.3 shows the cost estimate calculated and used in experiment 3 for the throughput rate to system cost ration. The substantial price difference between the FC SU and EIH SU is due to the FC SU containing a vision sensor where EIH SUs do not.

These were machine estimates at the time of purchase of the equipment and may vary. Note the values in the graph illustrated are scaled for convenience, removing unnecessary zeros.

Table B.3 Cost estimate for simulated system.

	no of units	Cost EIH(each)	Cost FC(each)	Cost EIH all	Cost FC all
SULT holon	3	R 10 000	R 40 000	R 30 000	R 120 000
SUPT holon	3	R 10 000	R 40 000	R 30 000	R 120 000
QI holon	1	R -	R 30 000	R -	R 30 000
Transport holon	1	R 335 000	R 305 000	R 335 000	R 305 000
System cost				R 395 000	R 575 000

B 2. Saturation calculations

A quantitative method was devised to explain and predict when saturation occurs in the configurations used in experiment 2. Data points for experiment 2 were used to test the hypothesis used here.

For simplification, these calculations assume the following:

- all parts in have the same transport time (collection and placement)
- the same amount of singulation units per component exits
- the singulation units have the same properties (probability of successful singulation (SS) and singulation period)
- one task exist in the system at a time

For saturation of a part to occur the follow statement must hold true.

$$\text{Average effective singulation period} < \text{Saturated component feeding period}$$

Now defining the saturation factor as

$$\varphi = \frac{\text{Saturated component feeding period}}{\text{Average effective singulation period}} \quad \text{B. 1}$$

Saturation occurs for

$$\varphi > 1 \quad \text{B. 2}$$

These terms vary slightly for the two configurations (EIH and FC)

B 2.1. FC application

In the FC case:

$$\text{Average effective singulation period} = \frac{\text{Singulation period (s)}}{\text{probability of SS} \times \text{no. singulation units}} \quad \text{B. 3}$$

and

$$\text{Saturated component feeding period} = \frac{\text{Saturated task completion time (s)}}{\text{no of components per task}} \quad \text{B. 4}$$

Where the “average task completion time” for saturated tasks was taken as the data points from experiment 2 save the last two.

Figure B.1 illustrates the saturation factor of the systems at the data points from experiment 2. It clearly shows that the two final data points (point 7 and 8) are not saturated ($\varphi < 1$) in the FC-case and that can be seen by the results in Figure 47.

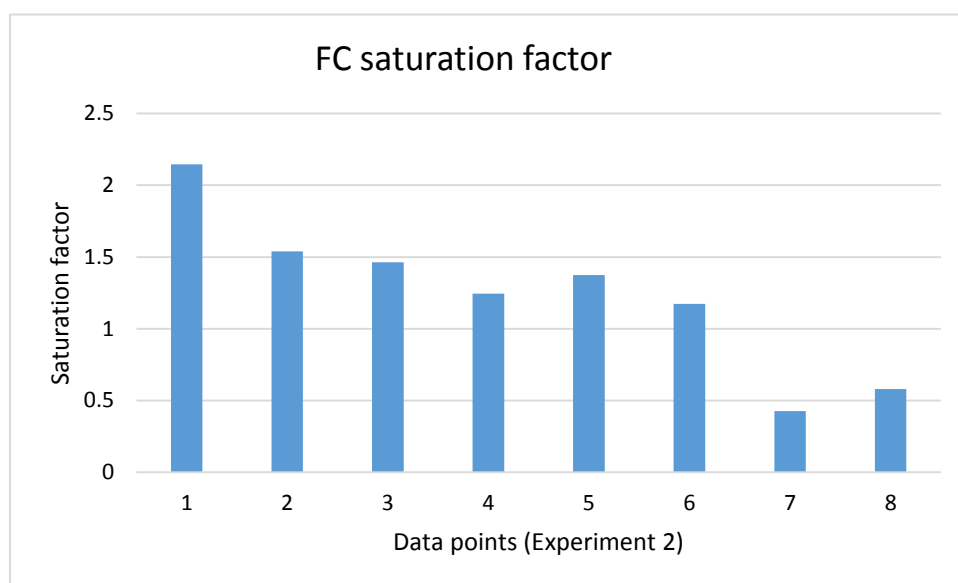


Figure B.1: Saturation factors for experiment 2 FC.

B 2.2. EIH application

The average effective singulation period defined for an EIH configuration differs from that of the FC case and is as follows:

$$\text{Average effective singulation period} = \frac{\text{Singulation period (s)} + \text{inspection time}}{\text{probability of SS} \times \text{no. singulation units}} \quad \text{B. 5}$$

Where the inspection time was calculated as the average difference between FC and EIH feeding times (per component) for saturated data points. The saturated component feeding time was calculated according to Equation B.4 applied to data points for the EIH experiment.

Figure B.2 illustrates the saturation factor for the EIH data points in experiment 2. This figure shows that the final two points (7 and 8) are not saturated, with $\varphi_8 = 0.94852$.

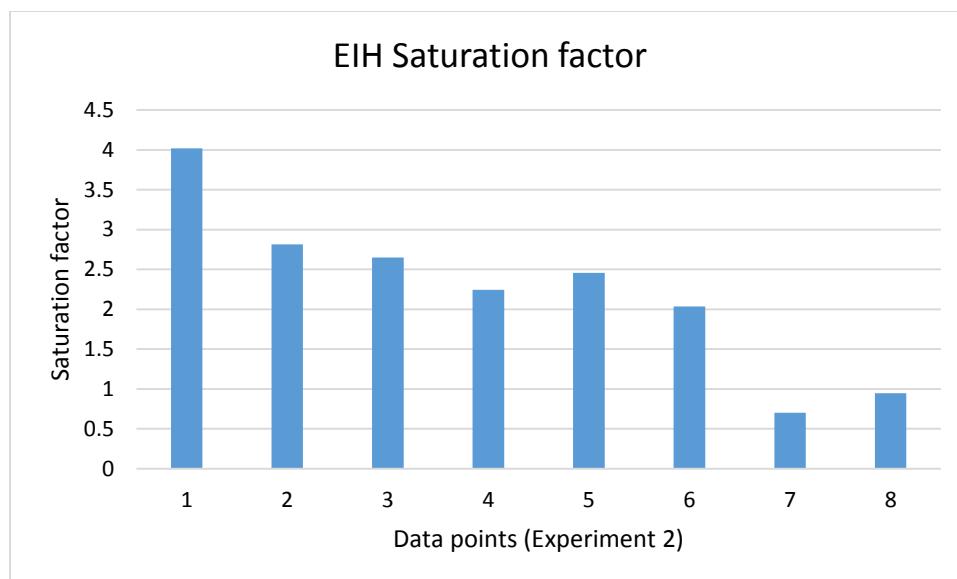


Figure B.2: Effective singulation period vs saturated feeding time for EIH.

The FC (B.3) and EIH average effective singulation rate (B.5) have a profound difference. The FC effective singulation rate is a function of singulation period, probability of successful singulation and number of singulation units, where the EIH is also includes the inspection time.

B 2.3. Example calculations

Consider SUs with a low probability of successful singulation and low singulation period. Assuming the same system (meaning same number of SUs with the same SU properties) is reconfigured it becomes evident that the FC configuration will have a higher average singulation rate. Consider the example data shown in the Table B.4. These values are typical of the system developed in this thesis, with the exception of the low chance of successful singulation.

Table B.4 Example system data.

Number of SUs	3
SU period (s)	2
% Successful Singulation	10
EIH inspection time (s)	5
Number of components per task	2
Saturated task completion time FC (s)	11
Saturated task completion time EIH (s)	14

For this system the saturation factor and the expected task feeding time is calculated. These values are shown in Table B.5. As expected the FC configuration performs better with a higher

saturation factor less than 3 times the expected task feeding time. This shows that FC configurations are better suited than EIH configurations for SU with the characteristics of low chance of successful singulation and short feeding periods.

Table B.5 Example system data.

	FC	EIH
Average effective singulation period	6.6	23.3
Saturated component feeding time	5.5	7
Saturation factor	0.825	0.3
Expected task feeding time (s)	13.3	46.6

Appendix C Conversational behaviours

C 1. Conversations of feeding one component

Figure C.1 displays the behavioural conversations for the feeding of one part, including the three phases of communication. Note that the booking phase shown is incomplete. In the figure, the subtask is employing two operational holons to achieve its goal and all services are available and supplied successfully.

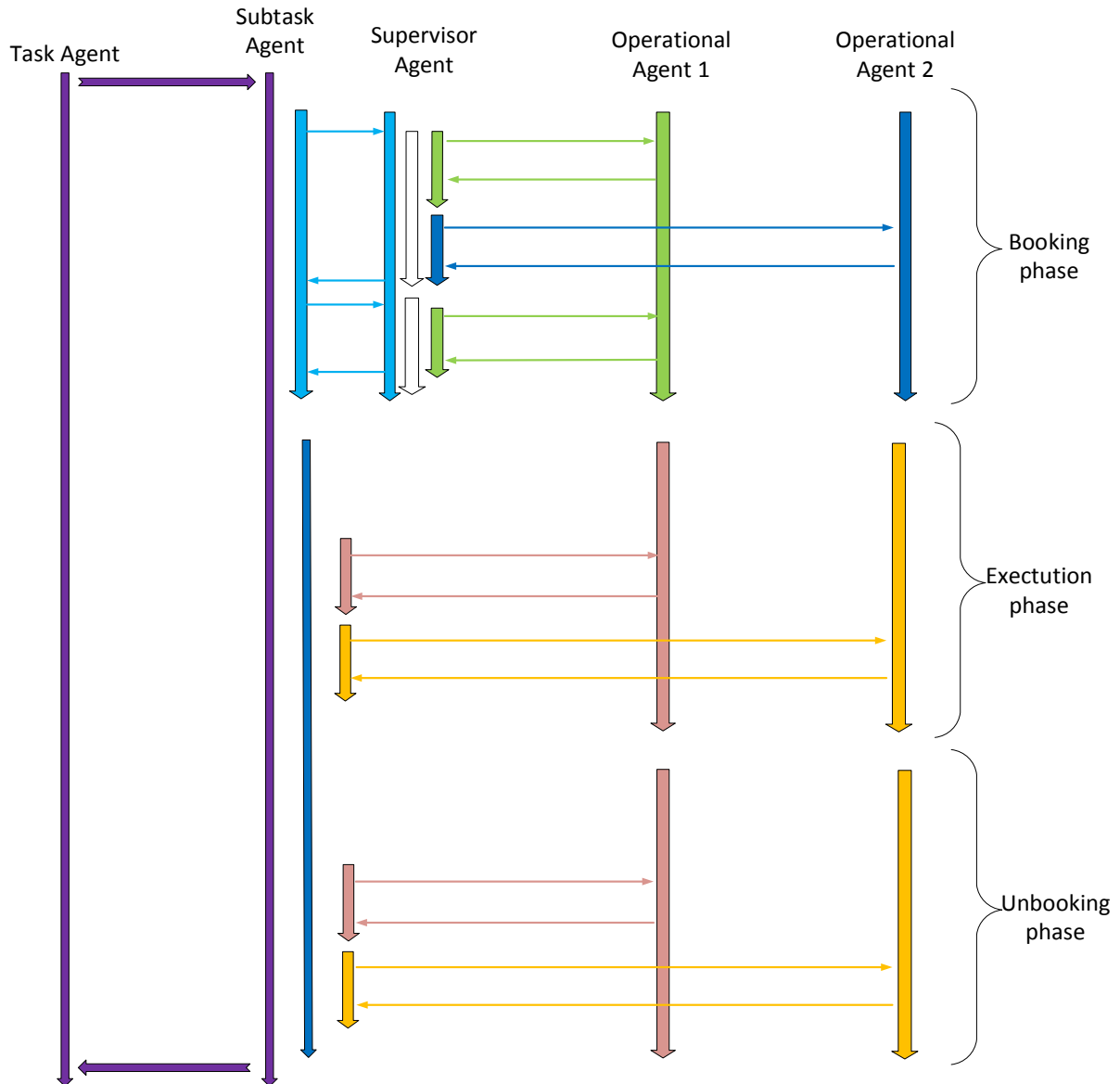


Figure C.1: Behavioural conversation for feeding one part.

C 2. Nested services

Nested services are a means of achieving OH-to-OH communication. It is an alternative to sub-services used for the EIH inspection negotiation, discussed in Section 5.6.5. In the architecture shown in Figure C.2, the SU would wait for a request before it enquires for the service of a transport agent, the transport agent may refuse or accept the request. Nested services were not used in the implementation described in this thesis, since sub-services appeared to be a better use of the assumed to be bottleneck robot resource. Nested services may be better suited in other applications, for example in applications where a process has to be done within a short period after singulation such as furnace heating before moulding.

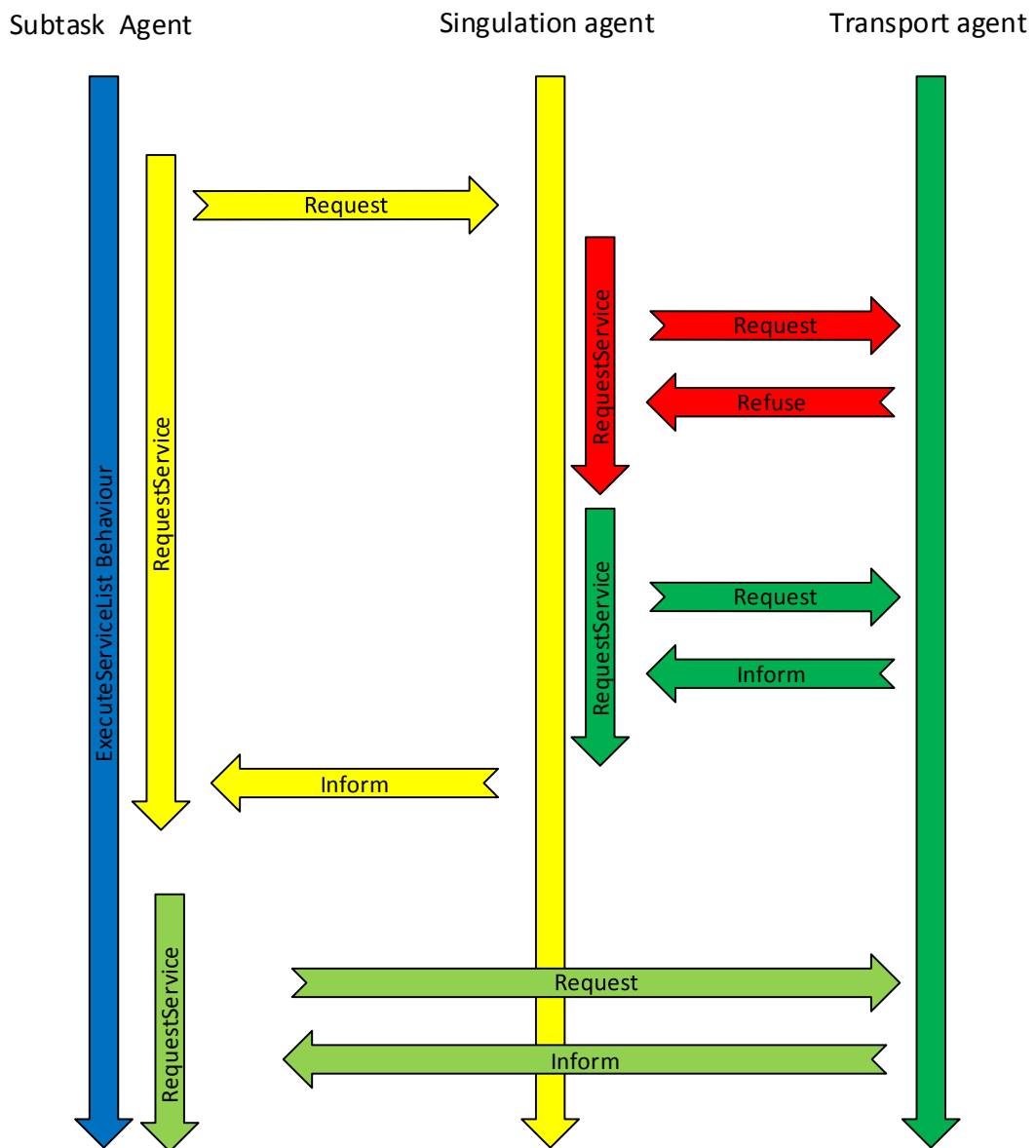


Figure C.2 : Nested services architecture.

Appendix D Practical measures related to computer vision

D 1. EIH Camera Calibration

When calibrating the image for the inspection in the eye-in-hand experiment the camera was directly above the platform. The calibration grid was laminated and this caused a reflection from the flash. Because of the nature of EIH setup, auto exposure settings were used and this resulted in a low quality image as can be seen below.

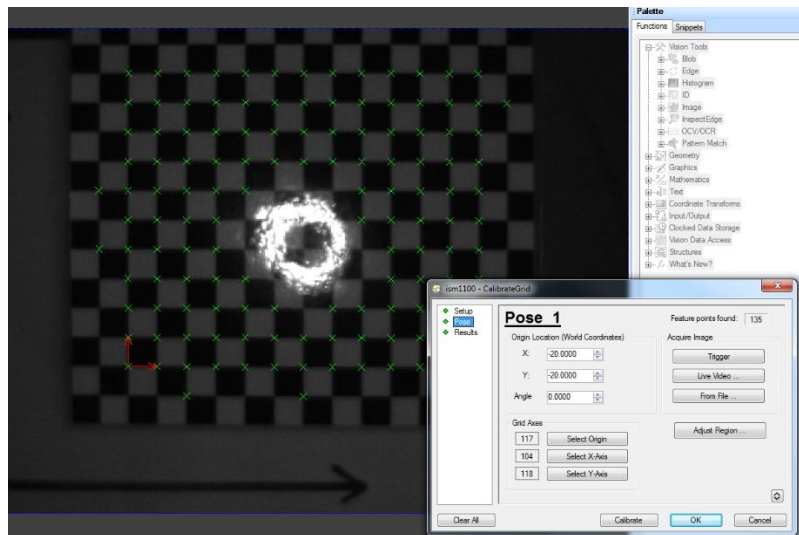


Figure D.1: Vision system calibration on reflective surface.

For this reason a circular matte finished object, paper in this case, was placed in the middle of the image. This resulted in a higher quality picture.

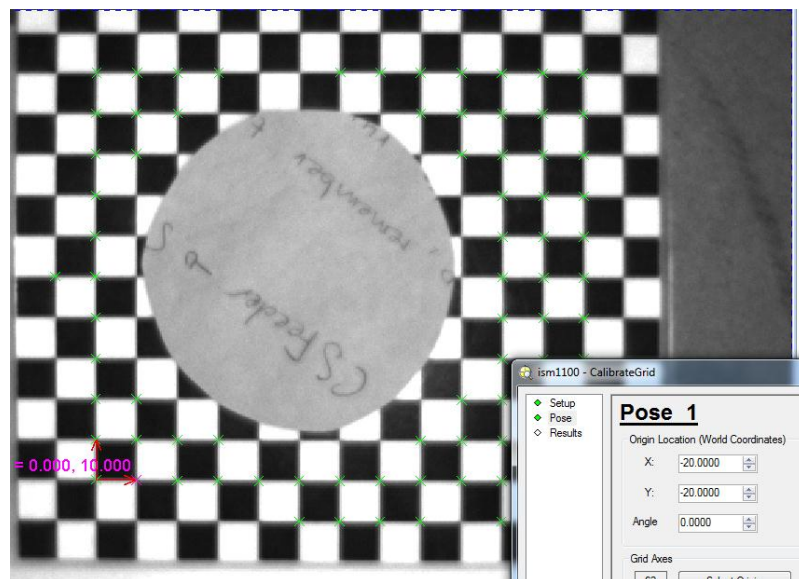


Figure D.2: Vision system calibration using matt reflection blocker.

D 2. Fixed camera viewing angle for component identification

It was noticed during testing that the rotation of parts about the Z-axis has a negative effect on the accuracy of collection of parts. Inaccuracies during collection propagate to placement, as parts need to be accurately placed within fixtures.

Collection was reliably achieved for parts placed at ± 25 degrees from the calibrated position. It is suspected that this error is a cumulative result of tool calibration error, camera pixel loss, camera calibration error and non-linear transformation error. The non-linear transformation error could be mitigated by having multiple trained images for different sectors (or a range of angles).

This would result in successful gripping of parts for all angles. A negative effect of this is that each trained image is considered a separate part and therefore will need to be tested individually. This will increase reconfiguration time.

An alternative method of increasing placement accuracy and robustness is the employment of vision based feedback control to accurately place parts. This may negatively affect feeding times.

D 3. Environmental considerations for visual inspection

Initially the vision system had trouble identifying components on the magazine platform. This was due to the load terminal and pigtailed being made of copper and the component magazine is made of wood. The vision system relies on identifying the contrast between the two objects. The Cognex smart-camera would time out (5-second period) when attempting to locate components in this scene. This indicates inspections performed on scenes with low contrast may reduce performance and these scenes were avoided.

The background of the component magazine was sprayed with a mat white finish to accentuate the contrast between components and background. This can be seen in Figure 15, which shows the component magazine containing the load terminal (right) after painting and the component magazine containing the pigtailed (left) before painting, Figure 34 shows the pigtail magazine after painting. For a similar reason, the component fixtures were painted.

The machine vision tools (or algorithms) are sufficient for the application of robotic feeding of fixtureless components in both FC and EIH systems. This did require the environment be controlled in this application.

Appendix E Further notes

E 1. Agent Class hierarchy

Figure E.1 shows the Java class hierarchy developed for this system. RMS agents inherited simplified methods to execute generic tasks such as registering services and destroying agents. Operational agents were developed to inherit conversational behaviours that could be overridden when needed, for example SRR and SCL behaviours. Operational agents were further specialized to include classes for hardware communication, for example transport agent contain a KUKA KRC object, while an EIH transport agent also contains a Telnet Camera object. This illustrates how the use of object-orientated programming makes the systems more scalable, since new agents can be easily developed by specializing more generic agents.

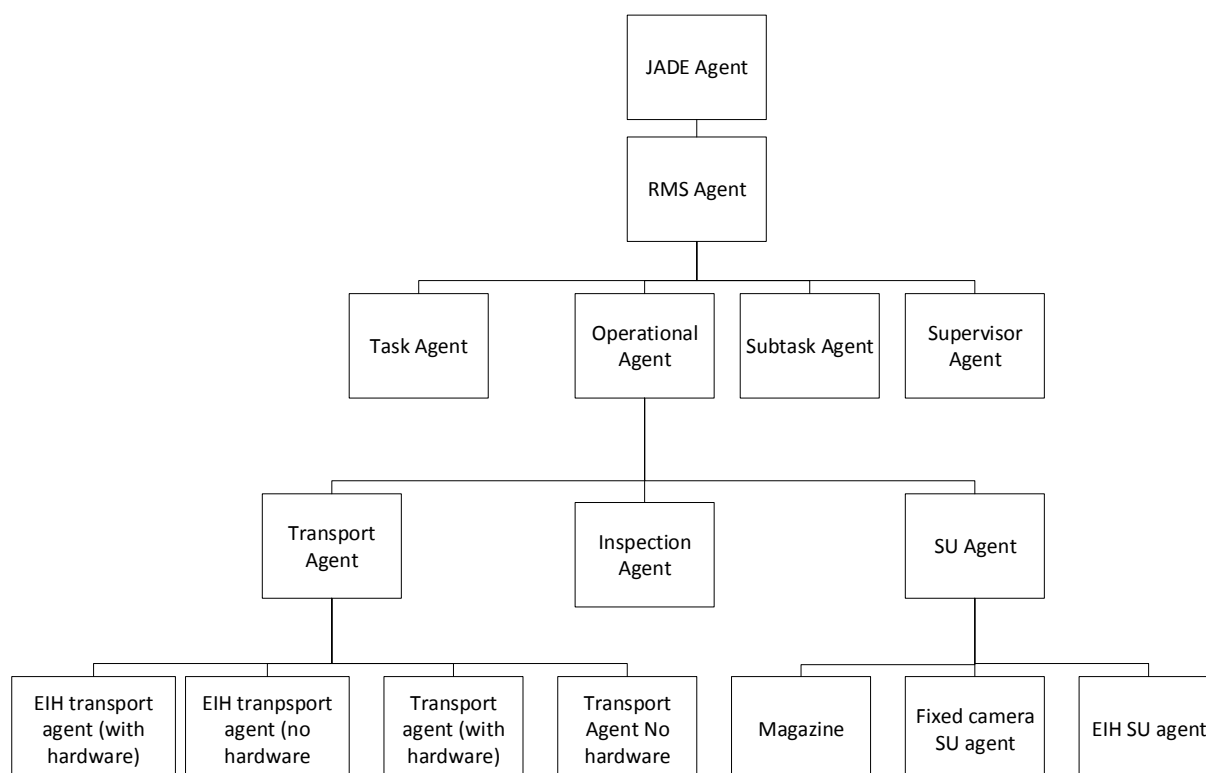


Figure E.1: Class hierarchy used for agents.

E 2. When and how to split a holon

One decision that occurred during the conceptualization of the system was how and when to split holon. Firstly, consider the transport holon shown in Figure E.2, in which the transport holon consisting of the robot and gripper controller could be split into a robot holon (corresponding to the robot controller) and the gripper holon (corresponding to the gripper controller). This notion can be expanded by considering the addition of the camera to the transport holon.

It was decided in this implementation that, since the gripper (and camera) will rely on the robot and cannot perform its function without the robot, these components should be incorporated in

one holon. Additionally, the robot does not supply any service without the gripper. The level of modularity is still preserved by using the control interface (shown in Figure 4), which enables the camera to be easily moved between SU holon and Transport holons.

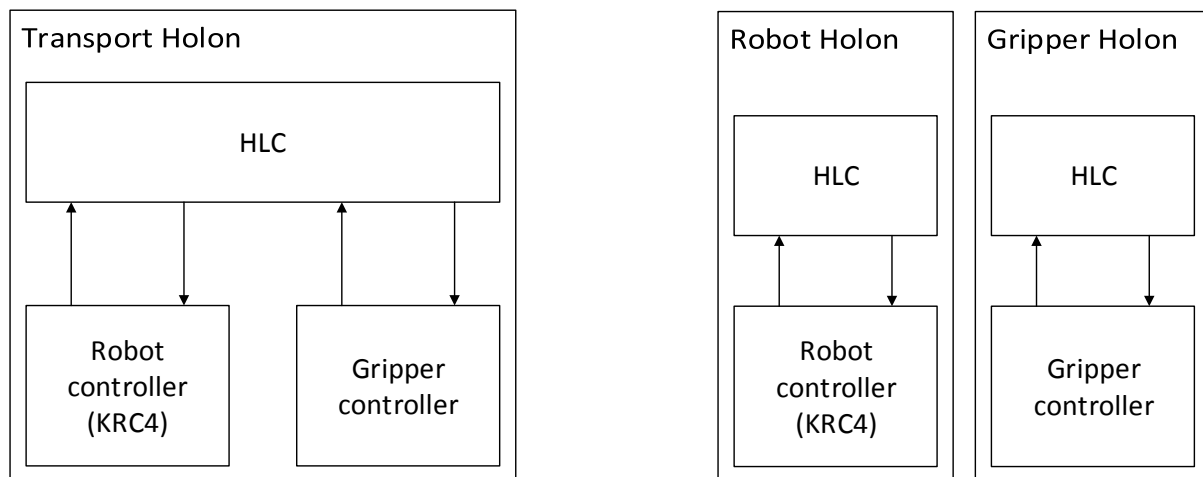


Figure E.2: Transport holon architecture alternatives.