A Port Management System
Edward Jones
Computing and Management
(Industry)
Session (2009/2010)

The candidate confirms that the work submitted is their own and the appropriate credit has been given where reference has been made to the work of others.

I understand that failure to attribute material which is obtained from another source may be considered as plagiarism.

(Signature of student)__________________________
Summary

This project looks at how software can be used to aid the allocation of vessels to quay space at the port of Southampton. The report details analysis of the allocation process at the port and software currently used to aid this as well as software available off the shelf to purchase. The report also details the design and implementation of an application that aids the allocation process by making automatic allocations for vessels including evaluation from the port of Southampton.
Acknowledgements

I would first like to thank my project supervisor Dr Kevin McEvoy for his time and help during this project. I would also like to thank Associated British Ports for the help they provided me, Eddy Hooper helped me gain access to the information and personnel I needed at the port of Southampton. Knowledge of the port and feedback was given by David Stewart-White who I would also like to thank.

Furthermore I would like to thank Dr Sarah Fores for her feedback of my mid-project report and the feedback given during my progress meeting.
1 Introduction ............................................................................................................................. 1
   1.1 Project Aim ......................................................................................................................... 1
   1.2 Objectives .......................................................................................................................... 1
   1.3 Minimum Requirements ...................................................................................................... 1
   1.4 Deliverables ....................................................................................................................... 1

2 Port of Southampton ................................................................................................................ 3
   2.1.1 Overview .......................................................................................................................... 3
   2.1.2 Berths ............................................................................................................................... 3
   2.1.3 Port and Vessel Information System (PAVIS) .................................................................. 5
   2.1.4 Voyage States ................................................................................................................... 6
   Summary Screens ....................................................................................................................... 6
   2.1.5 Allocation Process ............................................................................................................ 7

2.2 Tilbury Container Servicers .................................................................................................. 8

3 Existing Off the Shelf Software .............................................................................................. 11
   3.1.1 Harbour Mastery .............................................................................................................. 11
   3.1.2 IBS ..................................................................................................................................... 11
   3.1.3 Zebra Enterprise Solutions ............................................................................................. 11
   3.1.4 Tideworks Technology .................................................................................................... 11

4 Developing a model port ......................................................................................................... 12
   4.1.1 The Port ........................................................................................................................... 12
   4.1.2 Berth Information .............................................................................................................. 13
   4.1.3 Tides ................................................................................................................................. 13
   4.1.4 Vessels .............................................................................................................................. 13
   4.1.5 Measuring the results ....................................................................................................... 14

5 Project Planning ...................................................................................................................... 16
   5.1 Methodology ....................................................................................................................... 16
   5.2 Schedules ............................................................................................................................ 16
   5.2.1 Original Schedule ............................................................................................................ 16
5.2.2 Actual Schedule .................................................................................... 16

5.3 Choice of Programming Language ................................................................. 17

6 Background Reading .............................................................................................. 19

6.1 Scheduling Algorithm Methods ........................................................................ 19

6.2 Critical Path Scheduling (CPM) ............................................................................. 19

6.3 Constraint Based Scheduling (CBS) ..................................................................... 20

6.4 Scheduling Algorithms Outcome ....................................................................... 20

7 Solution .................................................................................................................. 21

7.1 Information required within the system ............................................................ 21

7.2 Designing the Allocation Algorithm .................................................................... 22

7.3 Iteration 1 ............................................................................................................. 24

7.3.1 Design .................................................................................................................. 24

7.3.2 Implementation .................................................................................................... 26

7.3.3 Testing .................................................................................................................. 27

7.4 Iteration 2 ............................................................................................................. 28

7.4.1 Design .................................................................................................................. 28

7.4.2 Implementation .................................................................................................... 28

7.4.3 Testing .................................................................................................................. 29

7.5 Iteration 3 ............................................................................................................. 30

7.5.1 Design .................................................................................................................. 30

7.5.2 Implementation .................................................................................................... 30

7.5.3 Testing .................................................................................................................. 31

7.6 Iteration 4 ............................................................................................................. 32

7.6.1 Design .................................................................................................................. 32

7.6.2 Implementation .................................................................................................... 32

7.6.3 Testing .................................................................................................................. 33

8 Evaluation .............................................................................................................. 34

8.1 Self Evaluation...................................................................................................... 34
8.2 Evaluation from Southampton .................................................................................................. 34
  8.2.1 Criticism I received ........................................................................................................... 34
  8.2.2 Vessels sharing buffers ..................................................................................................... 35
  8.2.3 Positive feedback received ............................................................................................... 36
9 Conclusion .................................................................................................................................. 37
10 Future Extensions ..................................................................................................................... 39
  10.1 Improvements suggested during feedback ........................................................................... 39
  10.2 Integration of previous berthing records .......................................................................... 39
  10.3 Replace model port with the port of Southampton ............................................................ 39
  10.4 Improvement of allocation algorithm from testing against port data ............................... 40
  10.5 Integrations with other port systems .................................................................................. 40
11 Bibliography ............................................................................................................................. 41
12 Appendix .................................................................................................................................. 42
  A. Personal reflection .................................................................................................................. 42
  B. PAVIS software ..................................................................................................................... 44
  C. Scheduler software ............................................................................................................... 45
  D. SPARCS software ................................................................................................................. 46
  E. Email correspondence with existing software companies ...................................................... 47
  F. Email correspondence with Eddy Hooper ........................................................................... 48
  G. Map of the port of Southampton .......................................................................................... 50
  H. Map of the model port ......................................................................................................... 51
  I. Original schedule .................................................................................................................. 52
  J. Actual schedule ..................................................................................................................... 53
  K. Initial user interface design for main form ............................................................................ 54
  L. Initial user interface design for maintaining vessels and visits ........................................... 55
  M. Initial user interface design for maintaining allocations .................................................... 57
  N. Initial database design .......................................................................................................... 58
  O. The user interface ................................................................................................................. 59
<table>
<thead>
<tr>
<th></th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>P</td>
<td>LINQ generated classes</td>
<td>60</td>
</tr>
<tr>
<td>Q</td>
<td>Code snippet of additional functionality added to generated classes</td>
<td>61</td>
</tr>
<tr>
<td>R</td>
<td>Main form showing data</td>
<td>62</td>
</tr>
<tr>
<td>S</td>
<td>Unit test code logic</td>
<td>63</td>
</tr>
<tr>
<td>T</td>
<td>Revised database design</td>
<td>64</td>
</tr>
<tr>
<td>U</td>
<td>Schedule output user interface design</td>
<td>65</td>
</tr>
<tr>
<td>V</td>
<td>Schedule output</td>
<td>66</td>
</tr>
</tbody>
</table>
1 Introduction

1.1 Project Aim
The aim of this project is to design and implement an algorithm for use in the port of Southampton that will aid the allocation process of vessels to berths in the port. A system is already in place for managing port operations; this existing system aids the port staff by collecting information about port operations together in one place. It also allows shipping agencies to enter voyage information directly into the system via a website reducing the workload on staff. However it does not make any port management decisions on behalf of the user. The system relies on users to input berth allocations for vessels using the port.

1.2 Objectives
- Research current available port management systems, including any currently in place at the port of Southampton.
- Obtain requirements by discussing the port operations with the port of Southampton (this is dependent on staff members agreement).
- Design and implement a new management algorithm that would make efficient use of the ports facilities with regards to vessels requiring port services.
- Compare algorithm outcomes with current port operations.
- Reflect on algorithm effectiveness and real world applications.
- Self evaluation of the algorithm and evaluation from staff at the port of Southampton (this is dependent on staff members agreement).

1.3 Minimum Requirements
- Report into current available port management systems.
- Report showing requirements analysis for the port of Southampton.
- Implementation of an algorithm to improve an aspect of port management.
- Report showing algorithm testing and results.

1.4 Deliverables
- Report on background research into the problem and possible solutions.
- Algorithm that could be used within the port of Southampton to aid the automation of berth allocations.
• Report on evaluation of algorithm and its effectiveness in aiding berth allocations at the port of Southampton.
2 Port of Southampton

2.1.1 Overview
The port of Southampton is owned and operated by Associated British Ports (ABP). It was opened in the 1920’s. It is one of the countries busiest and most successful deep-sea ports [1]. With 6 maximum berth lengths of over 400m and a draft of 11.7m it allows unrestricted access to the world’s largest vessels [2]. Southampton is the UK’s principle automotive port it is also home to the second largest container terminal in the UK handling over 42 million tonnes of cargo each year [3]. Due to its unique double tide, vessels benefit from 17 hours of high water each day resulting in long operational windows for ships to arrive and leave the port.

2.1.2 Berths
Vessels using the port are required to come "alongside" to load and offload cargo and or passengers. A vessel will enter the port and move alongside the quay or a jetty. The quay or jetty will be split into a series of berths. These splits may be physical dividers or simply markings on the quay. Berths may be designed to handle a particular type of ship or handle a certain type of cargo. All berths will have an advertised maximum draft; this indicates the depth of water under the berth and the max depth of a vessel that can use the berth. Vessels will have an advertised draft, a safety buffer will then be added to ensure there is enough water under the vessel, this is referred to as the under keel clearance.

Berths will also have an advertised maximum length; this indicates the maximum length of the vessel that can use the berth (often referred to as LOA). The port has a range of river and dock berths. River berths are on the River Test and the River Itchen. These are subject to tidal variations, this means the depth of a vessel they can accommodate changes with the tide, the advertised draft of the berth is the minimum depth of water during low tide. Dock berths are behind dock gates meaning that the berths have a fixed draft. This adds an extra limitation due to the fact the vessel must fit into the dock gates in order to use the berth.

Vessel Owners (Agents) notify the port that they wish to bring a vessel into the port (referred to as a visit). This is done through a website at www.abpnotify.co.uk. A visit is then created in the ports management software program the port uses (section 2.1.3). Based on when the vessel will arrive at the port a Vessel Traffic Services Officer (VTSO) will allocate the vessel a berth. This is a manual process done by the VTSO; they will take into consideration the type of vessel, the services it requires, its cargo and how long they require at the port. This information is then combined with the VTSO’s own experience to decide which berth would be best suited for the vessel.
Vessels using the port will often require different services based on the type of vessel (e.g. Cruise ferry, RO-RO ferry, container carrier, vehicle carrier) and the cargo they are carrying (e.g. passengers, containers, steel, bulk cargo, vehicles, food). Different berths in the port can provide different services to vessels, it is important to make sure that the berth will provide adequate services for any vessels assigned to it. Berths may also be classed as common use indicating they can be used for a multiple array of vessel services. There are additional requirements that need to be considered when making berth allocations, different cargo is loaded and unloaded from the vessel by different means. Some cargo requires specialised cranes and staff e.g. containers. Other cargo such as steel requires certain weather conditions to load or unload outdoors. Vessels may also use the port for different reasons such as repairs. The port has a dedicated dry dock, which vessels may need to use when making repairs. They may also wish to send divers down to inspect the vessels beforehand.

Not all berths may be in service at certain times. Some berths may be out of service for maintenance. It is important that vessels are not permitted to use a berth that cannot leave before scheduled maintenance has begun. All these factors need to be considered when making the berth allocation. It should also be noted that some vessels that use the port regularly might have specialised dedicated berths.
2.1.3 Port and Vessel Information System (PAVIS)

The port of Southampton currently uses a piece of software called Ports and Vessel Information System (PAVIS). This was developed in house by a group of developers based at Hull. This software is written in Microsoft .Net 2.0 and is designed to run on Windows based systems. It is an MDI container application allowing the user to have several windows open at once displaying information about current port operations. A screenshot of the main form can be seen in appendix B.

The application works on a server and client architecture. There is a central server for the port which runs Microsoft SQL Server. This is then accessed by all clients at the port using a wide area network.

Users are required to log into the system and access to features is restricted using task permissions based on the users role at the company.

It manages certain port operations including vessels and resources allocation, pilot allocation and billing. Details about each vessel entering the port are entered into the system. The system stores information about a collection of ships. This is every ship which has had any interaction with the port in the past.

The PAVIS application stores information about different locations of ports around the world. Locations are either local or foreign. Any local location is within the port of Southampton or surrounding areas. All other locations are deemed as foreign.

Each time a ship requires services from the port a visit is created. This visit is made from one or more voyages which can be an arrival, departure or movement. This is determined using the departure and destination locations.

- Arrival – Ship is moving from a foreign location to a local location.
- Departure – Ship is moving from a local location to a foreign location.
- Movement – Ship is moving from a local location to another local location

Additional information is stored about each voyage for a ship. This information includes:

- Estimated time of arrival (ETA)
- Actual arrival of arrival
- Estimated time of departure (ETD)
- Actual time of departure
- Ship length (LOA)
- Ship Width (beam)
• Ship draft
• Ship weight
• Cargo on board (what it is and if it is dangerous)
• Services required when in port, e.g. fresh water, cranes, divers

Most of the information about a voyage is not entered by staff. This information is entered directly by the agent (owner of the ship) via the website www.abpnotify.co.uk. Agents register and are given a user name and password to log into the site. From there they can create a new visit for one of their ships and enter all the other details required by ABP.

Once the agent has entered this information and submitted the visit, it can then be seen in the PAVIS application. Users with sufficient privileges can view voyages created by agents via the website (also known as notifications). The PAVIS user may change any or all of the information on the voyage and then accept it. Once the notification has been accepted a voyage will be created in the PAVIS application that can be viewed by anyone with privileges to view voyages. This voyage created in PAVIS remains linked to the voyage created by the agent via www.abpnotify.co.uk. Changes made to the voyage in PAVIS are reflected on the website so that the agent is notified.

2.1.4  Voyage States
All voyages within the PAVIS applications have a state. This can be any of the following. The voyage can only be in one state at a time.

• Planned – The voyage has been created in the PAVIS and can be seen by any user that has an interest in the voyage. The voyage is not confirmed and any changes or cancellation will not result in the agent being charged.
• On Passage – The ship has left the departure location and is on route to its destination.
• At Destination – The ship is within the limits of the port and has yet to berth.
• Anchored – The ship has anchored itself outside the port awaiting entry for some reason.
• Berthed – The ship is at the berth alongside.
• Completed – The voyage has been completed. Visit details become locked and cannot be edited.
• Archived – The voyage has been completed and archived.

Summary Screens
The PAVIS application allows user to view current information via summary screens. These are forms that display grid controls showing information about vessels; they are also continuously updated as
information in the system is changed by other users.

Reports
The PAVIS application contains an extensive reporting tool allowing the user to display reports relating to port activities, this functionality is beyond the scope of my project.

2.1.5 Allocation Process
The PAVIS application aids the allocation process by allowing the user to assign a berth allocation to a visit, a user can manually select the quay and berth that the vessel should use. This process is entirely manual. The software makes no attempt to automatically allocate the visit to a berth or to suggest possible suitable allocations.

Within the PAVIS software the user makes this allocation by setting the origin and destination attributes of a visit. If either of these is for a local location then the software will expect the user to select a berth within the port. For all foreign locations the user is expected to just select a port.
2.2 Tilbury Container Servicers

I visited Tilbury Container Services (TCS) on the 1\textsuperscript{st} of December 2009. This is a dedicated deep-sea container terminal serving mainly western routes for large container ships importing foodstuffs into the UK. It was opened in 1976 and was the first purpose built container terminal in the UK [4]. It is based on the River Thames approximately an hour from London.

I initially made contact with Neale Banfield who is the company’s Finance Director & Company Secretary. After discussing my project with him he suggested I visit the port for the day and speak to Mark Viner who is the Planning Coordinator and Alan Kinney who is the IT Manager. The purpose of this visit was to gain a perspective of another port outside the Associated British Ports Group. During discussions with the port it also became apparent that the software they use to manage the port was off the shelf as opposed to-the in house built software used by the port of Southampton.

Tilbury Container Services is a small port in comparison to the port of Southampton. It has three berths, two on the River Thames (called river upper and lower) and one inside the port of Tilbury, which is inside a dock [5]. This means the two river berths are primarily used due to the levy applied to vessels to enter the dock. As the River Thames is tidal, vessels are restricted as to when they can enter and leave the port based on the draft of the vessel; they are not restricted on the amount of time they can spend at the port once there, similar to the port of Southampton. There is very little difference between the two river berths other than river upper has slightly larger cranes meaning some vessels with a larger beam may have to be serviced by the river upper berth, but this is normally an exception.

The port sells contractual windows of time when vessels can use the port to load and unload container boxes. These are sold to shipping companies, which operate shipping routes around the world. It will be agreed with the shipping company when the vessel will use the terminal and their expected quantity of containers load or unload. TCS will then give the shipping company a contractual window of time for when they guarantee to load and unload the containers. These schedules are normally set several months in advance. If the vessel cannot make the agreed window they will be offered the next available time slot when TSC can provide the services the vessel requires. As vessels are restricted to when they can leave the port based on the tide, it was explained that the port would often try to save costs when loading or unloading containers by using the minimum amount of cranes and staff to load and unload the vessel. This is due to the fact that the berth space cannot be used again until the tide changes, so makes little sense to rush when loading and unloading the vessel.
After meeting with both Mark Viner and Alan Kinney we discussed the different software the port uses for port management. They explained they used an off the shelf package called SPARCS (section 3.1.3) to automate the loading, unloading and storage of cargo containers for vessels using the port. Berth scheduling was done with a piece of software developed in house they referred to as 'Scheduler'. This software had been developed by a company employee over 10 years ago and stores the schedule information for vessels visiting the port and their berth allocations. Screenshots of this software can be seen in appendix C.

This software performs similar tasks as PAVIS does for the port of Southampton with regards to berth allocations. Vessel schedules are entered into the software by administrative users, this includes the details:

- Vessel Name
- Vessel ID
- Estimated time of arrival (ETA)
- Estimated time of departure (ETD)
- Expected containers to load and unload
- Berth allocation

This information can then be seen by any other users able to login to the system from around the port. The software makes no attempt to automate the process of berth allocation; it is up to the user to specify which berth the vessel will use. The two river berths will be allocated first then the dock berth. It would not matter which river berth is used except if the vessel had a beam exceeding 27 meters in which case it would be required to use the upper river berth. It was also pointed out to me that this system was developed to replace a large whiteboard located in the office. While staff members use the 'Scheduler' software the whiteboard is still in use and it is often the whiteboard that is updated first then this information would be copied over to the ‘Scheduler’ software. ‘Alan Kinney’ the IT Manager explained that they have looked into improving the ‘Scheduler’ software to automate the process, they determined this would be beneficial to the company especially with regards to allocating cranes to the vessels. The reason this has not been done it due to time and funding constraints. The company does not have any full time developers and would need to outsource this task to a third party which makes it unrealistic when considering the costs involved. Alan also expressed an interest to move to a completely electronic system and remove the whiteboard entirely as it provides no error checking or validation to the information it contains.
I was unable to determine from the makers of the SPARCS software exactly how much automation it was capable of with regards to berth allocation. Both Mark Viner and Alan Kinney agreed they were under the impression that the SPARCS software was capable of automating the berth allocation process. They informed me that the SPARCS software was sold on a license module basis and a license to use the software for berth scheduling was too expensive when compared with the benefits in relation to the software they already have.

They informed me that the port of Antwerp, Netherlands uses a product called SPARCS N4, which is a full port management system. Alan estimated that from his experience with NAVIS that a license to use SPARCS N4 would cost in the region on £1.5m plus support costs. This made it out of reach for Tilbury Container Services. Screenshots of the user interface for the SPARCS software were obtained for the features that TSC owned licences for, these can be seen in appendix D.
3 Existing Off the Shelf Software

I looked into existing software packages that may already provide automation of the allocation process at a port. This began by searching the internet for companies offering port management software available to purchase off the shelf. I also spoke to staff members at ABP I had met during my year in industry.

3.1.1 Harbour Mastery

This company produces a software management package called i-Seaports. It is a web based product that provides management for port operations, traffic management, security and accounting. (http://www.harbourmastery.com/)

3.1.2 IBS

This company produces a software management package called iPort. This product seems the closest match to PAVIS (currently used within the port of Southampton) available to purchase off the shelf. It uses a client and server architecture with an internet portal for staff to use when away from an office. It also provides an internet portal for customers to enter shipping information directly into the system. It provides management for marine operations, services and security. (www.ibsplc.com)

3.1.3 Zebra Enterprise Solutions

This company produces a software management packages called SPARCS Terminal Operating System. This product is aimed at container terminals and is a full management suite. It is used by an extensive number of ports around the world, including those operated by Asia Container Terminals who run some of the largest ports in the world. (http://zes.zebra.com/index.jsp)

3.1.4 Tideworks Technology

This company produces several software packages for the marine industry. They produce a package called Mainsail which is a terminal operations system. Some of the software appears to have some impressive visualisation features. (http://www.tideworks.com/)

Despite efforts additional information could not be obtained about any of the software mentioned above. Emails were sent to all companies asking for additional information and help with the project, these are shown in appendix E. No replies were received.

During the visit to Tilbury Container Services (section 2.2) which use the SPARCS software developed by Zebra Enterprise Solutions (section 3.1.3) it was not possible to observe the software making automatic allocations for vessels as Tilbury Container Services did not have a licence for that part of the software package.
4 Developing a model port

After contacting Eddy Hooper who is the head of IT for ABP (email correspondence shown in appendix F) it was suggested that spoke to David Stewart-White who is the Vessel Traffic Services Officer for the port of Southampton. After discussions it was decided to develop a model port based on the port of Southampton. When looking at the port of Southampton in detail it became apparent some of the variables would be difficult to model in a first attempt. A model port could be used to eliminate some of these variables such as reducing the number of berths, and eliminating berths inside the dock to leave only river berths.

4.1.1 The Port

The model port is based on the port of Southampton, it has similar characteristics such as long quays with berth markers and it is tidal. The port consists of three quays named quay 1, 2 and 3, these are all on a tidal river and are subject to changes in depth with the tide. There are no physical berth separators along the quay. There are berth markers along the quay, each berth has a unique identifier which is the quay number followed by the berth number, e.g. 103 for berth three on quay one. The quays for my model port are based on the Western Docks and Container Berths from the port of Southampton, this area can be seen on a map of the port shown in appendix G.

There are a total of 15 berths along the three quays; each berth provides connection points for fresh water and power. Not all berths are the same length along the quay. This can be seen in a map of the model port shown in appendix H.

All berths are common use, meaning that a vessel entering the port can use any berth providing there is sufficient draft for the vessel.

Cargo can be loaded and unload from vessels by cranes, there are cranes that run along the quay; these cranes can be moved by the operator. The cranes cannot move between quays and cannot pass each other. Cargo can also be loaded and unloaded by other means such as vehicle containers with side loaders where vehicles can be driven on and off the vessel.
4.1.2  **Berth Information**

The table below shows the berths contained within the port along the three quays;

<table>
<thead>
<tr>
<th>Berths</th>
<th>Depth (m)</th>
<th>Length (m)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Quay 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>101</td>
<td>10.2</td>
<td>370</td>
</tr>
<tr>
<td>102</td>
<td>10.2</td>
<td>370</td>
</tr>
<tr>
<td>103</td>
<td>10.2</td>
<td>310</td>
</tr>
<tr>
<td>104</td>
<td>11.7</td>
<td>340</td>
</tr>
<tr>
<td>105</td>
<td>11.7</td>
<td>200</td>
</tr>
<tr>
<td>106</td>
<td>11.7</td>
<td>260</td>
</tr>
<tr>
<td>107</td>
<td>11.7</td>
<td>110</td>
</tr>
<tr>
<td>Quay 2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>201</td>
<td>10.5</td>
<td>360</td>
</tr>
<tr>
<td>202</td>
<td>9.3</td>
<td>130</td>
</tr>
<tr>
<td>203</td>
<td>8.7</td>
<td>172</td>
</tr>
<tr>
<td>Quay 3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>301</td>
<td>9.1</td>
<td>274</td>
</tr>
<tr>
<td>302</td>
<td>13.6</td>
<td>310</td>
</tr>
<tr>
<td>303</td>
<td>13.6</td>
<td>310</td>
</tr>
<tr>
<td>304</td>
<td>13.6</td>
<td>310</td>
</tr>
<tr>
<td>305</td>
<td>13.0</td>
<td>420</td>
</tr>
</tbody>
</table>

4.1.3  **Tides**

As the port is on a tidal river vessels are restricted as to when they can enter and leave the port. Once at the port they can remain there as long as they require irrespective of the tide. They must however enter and leave the port during high tide. The exact time for this will change day to day and will also depend on the vessels draft.

4.1.4  **Vessels**

Vessels use the port to load and unload cargo/containers. When a vessel makes a visit to the port it will come alongside the quay at a given berth/berths depending on its size. It will then load and unload cargo. The vessel may also receive other supplies such as freshwater, as each berth has a freshwater connection point this is not a determining factor for deciding berth allocations.

As there are no berth separators the vessel may use more than one berth. Providing there is physical space on the quay for the vessel and sufficient clearance at either end of the vessel for it to be
moored, the vessel can be placed at any point along the quay. The berth markers are there simply to aid this process.

Vessels schedules are normally known well in advance, typically 3-4 weeks. These schedules can change at short notice due to weather conditions and mechanical faults.

Shipping agents will inform the port of the estimated time of arrival for a vessel (ETA). The port will then give the shipping agent a date and time when they are allowed to enter the port. This will normally be when the vessel is expected to arrive as these times are agreed well in advance. If a vessel's estimated time of arrival changes the port operators will attempt to accommodate the vessel when required. This may not always be possible and a vessel may have to change its schedule to meet to the schedule of the port. The vessel also has the option to arrive and wait outside the port until its agreed time.

When shipping agents notify the port of a visit they will also notify the port of the cargo to be loaded and unloaded. Based on this information the port operators will estimate the time required to load and unload the cargo.

Based on the vessel, and estimated time required to load and unload cargo the vessel will be allocated space on the quay, the port operator and the shipping agent will also agree an estimated departure time for the vessel.

Vessels will normally arrive at the port, load and unload cargo and depart within the estimated departure time. It may be necessary for the vessel to overstay its estimated departure time due to unexpected events such as problems loading and unloading cargo or mechanical faults. In this scenario the port operators will need to manage other vessels entering the port to ensure they are not kept waiting.

4.1.5 Measuring the results

There are several measurements that could be taken in order to determine the effectiveness of allocations made at the port.

- Vessel requests – Vessels wishing to use the port will inform the port of their estimated arrival time. During busy periods when port services are fully utilised the port may not be able to accommodate these requests. Vessels may be offered an alternative arrival time.
- The effectiveness of the allocations could be measured based on the number of requests that can be satisfied. This measurement would only be of interest during busy periods when
there are more vessels than the port can handle. During quiet periods vessel requests are likely to be satisfied even if the berth allocations are not made efficiently.

- Vessel waiting times – Vessels wishing to enter the port but are unable to due to insufficient quay space would have to wait until the port could accommodate them. The average waiting time for vessels could be used to determine the effectiveness of the allocations. The drawback to this method is that in reality a vessel would be unlikely to try and enter the port until it had a confirmed booking and time of arrival.

- Quay utilisation – Quay utilisation could be used to measure the effectiveness of the allocations. During busy periods if quay space is not utilised, vessels wishing to use the port may not be able to even though not all quay space is used. This measurement would be of particular interest during busy periods when quay space is at a premium, it would also be of interest during quiet periods to ensure the quay space is used efficiently in case of a sudden change in traffic volume.

- Ship throughput – Ship throughput could be used to measure the effectiveness of the allocations; this could be a simple measure of the number of ships during a given time. This is an important measurement, one which the port management would be interested in. It would however not prove the berth allocations are being made efficiently. It would be hard to gauge if there was room for improvement using this measurement alone.
5 Project Planning

5.1 Methodology

After research of software development methodologies and knowledge from both COMP2670 Object-Orientated Software Engineering and COMP3440 Issues in Business Computing an iterative approach was used. Knowledge was collected from personal experience and the port of Southampton to gain a better understanding about the allocation process. Reading was also completed to gain a further knowledge of algorithm development.

A solution was then designed and feedback gained from the port of Southampton. For implementation a series of iterations were completed building on the complexity of the software each time. Developer testing was completed for each iteration. Feedback from Southampton was attempted during the course of implementation.

5.2 Schedules

5.2.1 Original Schedule
Appendix I shows the original schedule for the project. The schedule was designed so all problem analysis and background reading would be completed before the Christmas holidays. It was intended to visit the port of Southampton as early as possible after the aims and minimum requirements were submitted.

Late December 2009 to January 2010 would be reserved for exam revision then solution design and implementation would commence after this. It was intended that feedback would be gained from Southampton towards the end of the solution design, allowing time to make any changes resulting from the feedback.

Implementation and evaluation would consume a large part of the project, it was intended that outcomes of the software would be compared with historical or real time data from the port. Feedback would also be obtained from the port of the implemented solution. The write up would be completed over a 14 day period, this time was chosen as it would cover the end of the Easter break before revision for summer exams.

5.2.2 Actual Schedule
Appendix J shows the actual schedule that was followed; this differed from the original schedule due to several reasons. This began at the beginning of the project when a visit to the port of Southampton could not be arranged to gain a better understanding of the problem. It was hoped that the day could be spent at the Vessel Traffic Control Centre for the port observing how the
allocations where made. Unfortunately this was not possible and no solid date for this could be arranged. I was however able to communicate with David Stewart-White at the port of Southampton via email and over the telephone.

The opportunity was presented to visit the port of Tilbury after researching available existing port management packages and this was done at the beginning of December 2009. The mid-project report was also completed as scheduled in December 2009.

Design of the solution began earlier than scheduled as the user interface could be designed before the background reading had been completed, although the main development of the solution did not occur until later in the project.

The problem and analysis was started on time but was not completed until after Christmas and ran into the second semester which was not planned. Historical data for port operations could not be obtained and the solution outcomes could not be compared against port data. This became less of an issue due to the way the solution was designed.

Implementation was completed as scheduled and involved four iterations, feedback was obtained from the port of Southampton but this was completed late which limited how the feedback could be used. Both the draft chapter and progress meeting were completed as scheduled.

5.3 Choice of Programming Language

Several choices of programming language were considered when implementing the solution. The first considered was Java which is an object orientated language developed by Sun Microsystems. Both the Java Development Kit and multiple IDE’s are available to download from the internet free of charge. During my first and second years I completed several projects using Java meaning I had previous experience with the language. Microsoft VB.NET was also considered; this is another object orientated language which I had extensive experience with from my year in industry. This is a rapid application development platform that is used in conjunction with Microsoft Visual Studio.

Microsoft VB.NET was chosen to implement the solution; this was mainly due to previous experience with the programming language and IDE. The solution would be similar in some aspects to PAVIS which I had worked on during my year in industry; PAVIS was also written using VB.NET.

It also became apparent that a method to store the information required about the port and vessels would be needed. Both using a database and a flat file system were considered for this. The first considered was Extensible Mark-up Language (XML), VB.NET has excellent support for reading and
writing to XML files. XML however does not support entity relationships; this was expected to cause problems with entities such as berths which must be related to a specific quay.

Therefore a database was considered that would support entity relationships. Due to its availability and compatibility with VB.NET SQL Server was chosen for this.

Both Microsoft SQL Server 2005 and Microsoft Visual Studio 2008 were made available from the School of Computing through the MSDN Academic Alliance.
6 Background Reading

6.1 Scheduling Algorithm Methods

Initially it was believed that the solution was based on an efficiency problem. From talks with the port of Southampton it became apparent that the problem was to do with scheduling. The main objective would be to ensure that vessels wishing to use the port could be allocated a space on the quay when they required it. As the schedule would normally be known in advance (but subject to change at short notice) any algorithm would be required to determine a fair schedule for the vessels using the port. How efficient the allocations were became less of a priority as long as all vessels could be accommodated. According to [6]

“Scheduling is concerned with the allocation of scarce resources to activities with the objective of optimizing one or more performance measures”.

This would seem to relate to the problem at hand, the scarce resources are quay space and we are attempting to improve the allocation of vessels to this quay space.

6.2 Critical Path Scheduling (CPM)

Critical Path Scheduling is used in the construction industry as well as for software development.

[7] Defines it as “It produces the cheapest way of executing the project or alternatively the cheapest way of doing the job as quickly as possible”

It is used when a series of activities need to be completed and dependencies exist between the activities. It involves constructing a model of all the activities to be completed and how long each will take. Using this information a critical path through the project is determined; this is the shortest time the project will take. Any delay in a critical activity will result in a delay of the whole project. Therefore delays in a critical activity should be corrected before any delays in a non-critical activity.

This method has the disadvantage of requiring a complex model of the activities to be calculated beforehand. It works well when the activities are known well in advance and are not subject to change.

When applied to this problem each activity could be considered a vessel allocation for quay space. The critical activities would be allocations that affect other allocations. An Example would be a series of vessels using the port with a large draft, this would limit the berths the vessels could use, and therefore any delay in these activities would have a knock on affect for the vessels after it. Other
allocations for vessels with smaller drafts may not have the same effect as there may be more quay space available for the vessel, these could be classed as non critical activities.

Critical path scheduling is useful when the solution is trying to complete a project in a given time. It would not help the allocation process of determining where to place vessels on the quay. It could be used to give priority to vessels using the port. Trying to avoid allowing a critical allocation to be become delayed would avoid a delay of other allocations in the port.

6.3 Constraint Based Scheduling (CBS)

“The idea of constraint based-programming is to solve problems by simply stating the constraints” [8]

Constraint Based Scheduling is used to solve scheduling problems using constraint programming (CP). According [9] “Constraint Programming is a paradigm aimed at solving combinatorial optimisation problem”. This means it’s used to solve optimisation problems where the set of feasible solutions is discrete.

[9] Also states that “Combinatorial optimisation problems are solved by defining them as one or several instances of the Constraint Satisfaction Problem (CSP)”. They go on to say that CSP is used when you have a set of variables each with a set of possible values; variables are then related via constraints. Using this method I would define a set of variables that are used when making port allocations and a set of possible values for each of these variables. The constraints in my problem would be the relations between these variables.

[8] States that “a solution of a constrain problem is a valuation of the variables such that all constraints are satisfied”. This means a solution would be found to the problem of allocating a vessel when each variable has a value that satisfies all the constraints in the problem.

6.4 Scheduling Algorithms Outcome

Critical Path Scheduling would be useful to give priority to vessels using the port to ensure other vessels are not delayed. However it would not be useful for making allocation to quay space.

Constraint based scheduling makes more sense when applied to the problem, there are constraints that must be considered when making the allocations. E.g. draft and length restrictions, an allocation can only be made once all these constraints have been met.
7 Solution

7.1 Information required within the system

The following items of data would be required for the system to make automatic allocations for a vessel, this was determined from the information in section 4 which outlines the constraints for the system.

Quays

- Name (some user friendly reference to the quay)
- Length (meters)

Berths

- Name (some user friendly reference to the berth)
- Length (meters)
- Draft (meters)
- Quay which the berth is on
- Start and end positions on the quay (meters)

Vessels

- Name
- Length (meters)
- Beam (meters)
- Draft (meters)

Visits

- Vessel that is making the visit
- Estimated time of arrival (ETA) and departure (ETD)

Allocations

- Visit the allocation is for
- Berth on which allocation is made
- Allocation start and end times
- Allocation state – this would be used to determine if the allocation could be altered
### 7.2 Designing the Allocation Algorithm

According to [9] “An algorithm is a procedure to accomplish a specific task” this project aims to accomplish the task of allocating vessels to quay space in the port of Southampton. Therefore any algorithm designed should be a procedure to allocate vessels to quay space in the port of Southampton.

When designing an algorithm for the automatic allocation of vessels using the port sufficient information about the allocation process had been obtained from personal experience and directly from the port of Southampton. Although existing scheduling methods had been explored and deemed unsuitable for this solution (discussed in section 6) a constraint based approach was taken to designing an allocation algorithm.

The constraints to making an allocation were defined;

- Vessels should be allocated space on the quay in order of their estimated time of arrival (ETA). Earlier ETA’s should be given priority.
- Any part of a berth that a vessel covers must have sufficient draft. A vessels draft plus its under keel clearance must be less than or equal to the advertised draft of the berth.
- The space required on the quay is equal to the vessels length plus a defined safety buffer.
- Vessels should be allocated space on the quay in order to conserve wasted draft and length.
- If no space on the quay is available then an attempt should be made to make space by moving other allocations.
- An allocation can be moved until a defined amount of time before the start of the allocation. Once a vessel is berthed the allocation cannot be moved under any circumstances.

Using these constraints to make an allocation, an algorithm was designed that would make allocations. Allocations are made using the following process;

- The system first determines which visits should have allocations, e.g. any visits in the future, there is no point making an allocation for something that has already happened.
- These visits are then sorted by the estimated arrival time to the port with the earliest first.
- Starting with the visit at the top of the list the system attempts to make an allocation for the visit.
  - It first calculates a list of quay straights for the visit; these are sections of quay that are free for the entire time of the visit with sufficient depth for the vessel (including the under keel clearance).
• If no quay straight can be found for the visit (all space is taken for the time of the visit) then an automatic allocation cannot be made for the visit.

• Using this list of quay straights an allocation attempt is made. If any of the quay straights have sufficient length (including the safety buffer) then an allocation is made to the quay straight with the smallest draft and shortest length. This is done to reduce wasted draft and length.

• If none of the quay straights have sufficient length then the system will attempt to extend each of the quay straights in order to make space for the visit.

• For each quay straight the system will work out what allocations either side of the quay straight would need to be moved in order to create enough space for the vessel.

• After working out which allocations would need to be moved the system will attempt to find alternative allocation positions for the vessels using the same process as above. If the allocations that need to be moved can be, then the system will move them in order to create space for the visit.

• If after attempting to extend all the quay straights found and none can be, then the system is unable to make an automatic allocation for the vessel.

• The system will then move to the next visit in the list.

Any visits that the system is unable to make allocations for will be logged and this information displayed to the use in order for a manual allocation to be made. This process is conducted any time a visit is created or edited, when a visit is edited the existing allocation will be removed and the system will attempt to make another automatic allocation for the visit. The allocation will also need to be updated if details for the vessel are changed as this could affect the draft and length required for the allocation. Implementation of this is discussed in the implementation of second iteration of the solution (section 7.4.2).
7.3 Iteration 1

7.3.1 Design

I found when beginning to design my solution that the software split into two sections, the actual software where ships, visits and allocations could be viewed and manipulated and the algorithm to perform automate the allocations.

7.3.1.1 The Application

I already had observed PAVIS (discussed in section 0), which is an application used at the port of Southampton that allows users to maintain a list of ships, visits and assign berth allocations to the visits. This product was developed and owned by Associated British Ports and access could not be gained to the source code. There were also concerns that I would become distracted with adapting the software for use with the model port if I attempted to use it.

After the decision not to use PAVIS, work began on designing a user interface for the solution, this would be similar to PAVIS due to the fact that users must be able to maintain a list of vessels, visits and berth allocations. It would however be focused on aiding the allocation process and therefore the way it would be used would be significantly different from PAVIS. Work on designing the user interface could begin ahead of schedule as shown in section 5.2.2.

The PAVIS software is based around an MDI container; this allows the user to have multiple screens open at once. As the software being produced was likely to display less information than PAVIS it was decided the use forms for the application. A main form would be created that would display the information the user needed frequently; other functionality could then be handled with child forms.

7.3.1.2 The User Interface

7.3.1.2.1 The Main Screen

User interface designs for the application can see seen in appendix K. Usability and HCI has been studied in COMP3440 Issues in Business Computing. Experience of user interface design had also been acquired from my year in industry.

It was decided that the application should show a visualisation of the state of the port. This was given priority on the screen allowing the user to clearly see the state of the port at all times. This visualisation of the port can be seen in appendix K. This shows the three quays of the port aligned vertically, along the quays are markers for the berth separators and labels to show the berth numbers. If there is a vessel berthed at a section of the quay this is displayed with a black bar to the
right of the berth. This allows the user to see the state of the entire port; the user would then be able to view more information about a vessel and its allocation by clicking on the vessel.

The next logical piece of information the user would require is a list of visits; this is displayed in a grid below the visualisation of the port, users would be able to select the day that they wished to view the visits for using a date selector above the grid. They could also jump to today using a button to the right. Visits could be created or deleted using plus and minus buttons to the right.

Information showing vessels berthed at each berth is displayed on the left hand side of the form. This information can be seen from the visualisation of the port but it was decided that allowing the user to quickly see which vessel is covering part of a berth would be useful. This information would be displayed in a treeview control that would show the three quays of the port along with the berths on the quay. If a vessel was currently berths at the quay, it would show the vessel name to the right of the berth name. The user could then see at a glance which berths currently had vessels berthed and which where free.

7.3.1.2.2 Maintaining Vessels and Visits

Users would be required to create and edit vessels stored within the system; this was achieved using a child form that would show a list of all vessels. A user interface design for this form is shown in appendix L. Users can create and remove vessels using the plus and minus buttons at the top right of the form. Creating a new vessel would display another form asking the user to enter information about the vessel; a user interface design for this form can also be seen in appendix L.

7.3.1.2.3 Maintaining Allocations

Although the software was intended to make automatic allocations for vessels it would need to display information about the allocations it had made and allow the user to change these allocations if required. A child form that would display and allow the user to edit the information about a single allocation was designed. A hand drawn sketch of this can be seen in appendix M. The top of the forms displays information about the visit the allocation is for. This information is read only and there simply so the user can be certain about which visit they are working with.

Below this the user can edit information about the allocation, first the quay and berth the vessel is allocated to. These are both drop downs to aid validation. When the user changes the quay selection the list of berths is refreshed with berths for the selected quay. Under this the user can edit the start and end times for the allocation. The user can select a date using a standard datetime picker control, and then select the time using drop down controls. This will reduce the need for validation on the user’s inputs.
7.3.1.3  Storing the data

It became apparent that the software would need to store information about the quays and berths in the port along with any vessels and visits planned. As I discussed section 5.3 this is achieved using a database.

Using knowledge from both DB11 and DB21 studied with the School of Computing an Initial database structure was designed and a UML diagram of this can be seen in appendix N. Six tables were created for the entities discussed in section 7.1.

The tables ‘Quay’ and ‘Berth’ store data about the ports structure, berths are related to a single quay and quays can have multiple berths. The tables ‘Vessel’ and ‘Visit’ store data about vessels within the system and any planned and previous visits made.

7.3.2  Implementation

Implementation began by creating the forms using Microsoft Visual Studio 2008 that had been designed in section 7.3.1.2. Screenshots of the main form can be seen in appendix O.

Once the forms were created focus turned to how the application would store the data it required. The database (section 7.3.1.3) structure had been designed and decided on using Microsoft SQL Server 2005. One of the benefits of using Visual Studio 2008 was I could take advantage of Microsoft Language Integrated Query (LINQ), this is a Microsoft .NET component that adds native data querying capabilities to .NET languages. It generates entity classes from tables within an SQL Server database. These classes were then extended to implement additional features required. This was a huge benefit as designing and implementing a data access layer is a time consuming task. As this was not the main concentration of the project it made sense to use the generation tools available.

Another benefit of using this technique was that if any changes had to be made to the structure of the database, all the data access layer code could be regenerated.

A database was created using SQL Server Management Studio based on the structure discussed in section 7.3.1.3. Then LINQ was used to generate a set of classes based on this database. These could then be used to retrieve and edit data stored in the database. Appendix P shows the classes that were generated. Extra functionality was then added to these generated classes to support the other features required by the application, appendix Q shows a code snippet of the additional functionality that was added to these classes.

The functionality to add and remove vessels, visits and berth allocations was implemented. The database was designed to support this functionality and the LINQ generated classes allowed easy
This iteration was completed ahead of schedule and before all the background reading was completed on the allocation algorithm design. It was therefore mainly concerned with getting the software into a state where vessels and visits could be created, manual allocations could be created for a visit and the visualisation of the port would show the current port state to the user. The automatic allocation process was not implemented until iteration 2.

7.3.3 Testing
Testing of the implementation was completed with a combination of manual and unit tests. Visual Studio 2008 had built in unit testing functionality that allows the generation of unit tests for classes. This was used for both classes that had been generated with LINQ and also hand written classes. This became very useful to quickly ensure the data access layer was functioning correctly. Unit tests were created for each entity in the database allowing the testing of:

- Inserting a record with known values, and then attempting to retrieve this record and checking against the known values.
- Inserting a record with known values, and then attempting to retrieve this record. Then making an update to this record and retrieving it again checking the updated values.
- Inserting a record with known values, and then attempting to retrieve this record. Then deleting the record.

This meant inserts, updates and deletions could be checked for each database entity with a single click. Using the rollback functionality of SQL Server, the database could be restored after each test avoiding polluting the database with unit test data. Unit tests for the additional methods on the classes were also written. Appendix S shows the unit test code for the ‘quay’ and ‘berthAllocation’ entities.

Manual testing was also used to ensure the functionality of the forms, checking that vessels, visits and allocations could be created deleted and edited without errors in the software. Other features such as the visualisation where also tested manually. I did not create any formal test routines other than using unit tests, the reasons for this is discussed in section 5.1.
7.4  Iteration 2

7.4.1  Design
The second iteration saw the implementation of the automatic allocation functionality, the design of this algorithm is discussed in section 7.2. This resulted in an allocation process the application should follow when attempting to make allocations for vessels. The algorithm had already been designed and therefore little design was needed with this iteration as the majority of the work would be during the implementation phase.

7.4.2  Implementation
Implementation of the allocation algorithm was contained within a single class that would handle all of the automatic allocations. This would expose a single method that would perform allocations on any visits in the database where required. The main application form could then call this method after a visit was created or any changes were made to existing visits. Encapsulating the entire automatic allocation process in a single class was done so that as future implementations of the algorithm were created the software could run multiple algorithms side by side to test their effectiveness.

As soon as work began on the automatic allocation process it became apparent an error had been made in the database design. The initial design had related allocations to specific berths along the quay. When automatic allocations were to be made, the system required the ability to allocate a vessel to a specific point on the quay. This point could be across berths, therefore a change in the database design was required. Appendix T shows a revised database design allowing an allocation to specify the start and end points in meters on the quay where the ship should berth.

Once the changes to the database structure had been completed implementation of the automatic allocation process began. The allocation process follows the process outlined in section 7.2. It first retrieves a list of visits from the database where the estimated time of departure for the visits is greater than the current system time. This is sorted by ascending ETA. For each of these an allocation attempt is made. This begins by calculating the quay straights for the time the vessel will be in the port.

This is done by retrieving a list of quays in the port; beginning with the first quay it retrieves a list of berths for the quay. It then iterates through the berths checking if they have sufficient draft and are free of allocations for the time the vessel will be in the port. If the berth has sufficient length and is free then the berth length is recorded and the system moves to the next berth along the quay.
If there is insufficient length or the berth is not free then this data is recorded and used to create a quay straight for the time the vessel will be in the port. This quay straight will hold the following information:

- Start Time
- End Time
- Quay
- Start position on quay
- End position on quay
- Min draft along the straight.

When the system has reached the end of the list of berths it moves on to the next quay in the system.

If any of the determined quay straights have enough length then an allocation will be made to the shortest to preserve long quay straights. The allocation will be made at a point on the quay straight to also preserve quays straights.

If none of the determined quay straights have sufficient length then the system will chose the longest one and attempt to extend it. This is done by determining if the allocations to the left or right of the quay straight can be moved to another position in the port.

The allocation is first checked if it is in a planned state and therefore can be moved. If so the same process will be used to determine quay straights for the allocation and if any are suitable.

If allocations can be moved to create space they are and the system commits the changes to the database. If the allocations cannot be moved the system will attempt this process with the other quay straights until one can be extended to create space for the vessel. If none of the quay straights can be extended then the system is unable to make an automatic allocation and the user is notified.

7.4.3 Testing
The testing for this iteration although for only a single feature was far more time consuming than for the first iteration. It was important the automatic allocation process was tested thoroughly as this formed the main feature of the software. Testing from the first iteration has showed that the software was functioning correctly and vessels, visits and allocations could be made and updated. Therefore testing would concentrate on checking that automatic allocations where made when required and they were made using the algorithm designed.
I also completed the same unit tests that I created in iteration 1 to ensure that I had not changed any part of the software I had not intended.

7.5 Iteration 3

7.5.1 Design
For this iteration the automatic allocation process was improved. It had been noticed during testing of the previous iteration that the software was not performing the automatic allocation as intended. There was little design needed for this change as this was an error in the software as it was not behaving as designed in the previous iteration.

A berth summary information pane was also added to the main form to provide additional information about the selected berth on the treeview above. The reason for this was the user had no way of finding out information such as the length and draft of a berth even though it was stored within the system. Although an experienced user is likely to know this information already it made sense to have the software display this to the user. A grid was also added to show any future allocations for the berth again this was because the information was stored in the system but could not be viewed by the user.

With the addition of showing future allocations for a berth it became apparent that the visualisation of the port would only show the current state of the port, not any future states. A slider was therefore added to the top of the visualisation so the user could view the state of the port over a period of time. A slider was chosen for this instead of asking the user to enter a specific time because it would be quicker to use. It could also then be used without the use of a keyboard. A label next to the slider would display the date and time the visualisation is showing the state of the port for.

7.5.2 Implementation
The main bulk of this implementation was concerned with bug fixing as opposed to adding new features. Testing had showed that software was not making allocations as expected. This was fixed by stepping through the code to determine how it differed from the algorithm that had been designed. It soon became apparent that the software was failing to determine the correct quay straights for a vessel. This meant the options for locations to allocate the vessel where reduced and allocations where not being made as expected. This fault was corrected.
The opportunity was also taken to rewrite sections of the allocation code to make it easier to understand, the process was broken down into smaller steps and the code was refactored to reduce the possibility of errors.

Implementing the two other changes for this iteration were considerably easier, to show berth information all the data was currently stored in the database and could be retrieved and displayed when a berth was selected on the main form. Implementation of the slider was again a simple task; all the code to display the visualisation of the port was contained in a single method which took a datetime as a parameter. The slider would alter this datetime and the visualisation would update with the port state at the selected time.

7.5.3 Testing
Tests for this iteration were carried out in a similar way to the previous iteration. Testing of the berth information and slider to select the port state were done manually by verifying the information shown in the software with that in the database.

As the system had failed the testing of the automatic allocation process during the second iteration, the same tests were performed again to ensure the errors had now been fixed.

I also completed the same unit tests that I created in iteration 1 to ensure that I had not changed any part of the software I had not intended.
7.6 Iteration 4

7.6.1 Design
This iteration began close to the end of the time reserved for implementation; therefore no attempts were made to improve the automatic allocation process. Instead it was noticed that the software did not have a simple output of the schedule of allocations. A way to view all the upcoming allocations was added, a user interface design for this is shown in appendix U. This is a grid showing the vessels, its ETA and ETD and where on the quay it has been allocated. This allows the user to view this information then print or email it as they wish.

Two other problems that were identified during the progress meeting were also corrected, it was raised that when editing an allocation, the current system would allow the user to place a vessel at a berth where there was insufficient draft or where it would clash with another allocation in the system.

The system would not make this mistake when making automatic allocations as it checked these parameters, however this could be done when a user edited an allocation. Therefore checks should be made and the user should be prompted if they attempted to change the allocation into a state that was invalid.

This would be a message prompt on the screen informing the user of the error, they could be required to correct this before the edit could be made. It was decided for clashes of allocations that the system should warn the user but allow them to proceed if they wished. This was to avoid the system preventing the user from changing the allocation times if no other times were free.

It was also raised that the functionality to mark vessels at the berth and departed for the berth had not been completed. Once a vessel is at the berth the user would need a way to inform the system as the vessel should not then be moved to make space for another allocation. Although this had been planned from the first iteration and the database supported this feature (as it is used during automatic allocations) the system did not support a method for the user to update this information in the system.

Menu items were added to the context menu of the visits grid on the main form allowing the user to mark the vessel at berth and departed.

7.6.2 Implementation
Implementing the schedule output involved creating another form that would display the information. This form could then be accessed from a menu item on the main form. The form
contained a single grid that is populated with any allocations in the database where the departure time has not passed. The schedule output can be seen in appendix V.

Both changes for this iteration were relatively simple compared with previous iterations, validation code was added to the ‘berthAllocation’ object that would throw an error if the allocation did not pass length and draft checks, it would also throw an error if the allocation clashed with another allocation. This would then be displayed as a message to user informing them of the error.

Allowing the user to mark the vessel at the berth and departed from the berth involved adding a menu item that would allow the user to change the state of the allocation. If the allocation was currently planned then the user could mark the vessel at the berth. If the vessel was at the berth then the user could mark the vessel as departed. As the database already supported this the state of the allocation could be changed without any changes to the database or data access layer.

7.6.3 Testing
The testing of the changes for this iteration was done manually. These could be tested easily by attempting to allocate a vessel to a section of quay that did not have sufficient draft or sufficient quay space to accommodate the vessels LOA. Tests were also conducted to check that the system would warn the user if they attempted to make an allocation to a section of quay that would not be free. I also completed the same unit tests that I created in iteration 1 to ensure that I had not changed any part of the software I had not intended.
8 Evaluation

I perform evaluation of the software myself using my experience from my year in industry using software that aiding managing ports. From my experience I was able to ensure the software was both useable and the system was capable of storing information required to make allocations. I could not however evaluate the allocation process myself, for this I required the help from the port of Southampton. For this I requested the help of David Stewart-White who is the Vessel Traffic Services Officer (VTSO). It is his responsibility to oversee all traffic within the port of Southampton. He was kind enough to meet with me at the port for a demonstration of the software to provide feedback.

8.1 Self Evaluation

Self evaluation was conducted on the project throughout its development. This was intentionally done as part of the methodology to test during development.

As discussed in section 7 unit tests and developer tests where used to ensure the software was reliable and to reduce the number of errors in the code. Unit tests provided and excellent way to quickly and repeatedly check that the data access logic along with the allocation logic was consistently reliable. These unit tests were used during each iteration to ensure code had not been unintentionally changed.

The usability was constantly reviewed during the development process; my personal skills of user interface design were used to ensure the usability of the application.

8.2 Evaluation from Southampton

For feedback of the software from the port of Southampton a demonstration of the software was given focusing on the automatic allocation process. Implementation of the allocation algorithms and the process it uses was also explained.

8.2.1 Criticism I received

8.2.1.1 Spring tides

Each berth has an advertised draft, this the minimum amount of water under the quay. There will often be more water than this during a normal tide. During spring tides that occur twice a month this amount of water will increase. When making allocation at the port currently they would take this into consideration. The increased water depth may mean a vessel could be allocated to a berth that would not be possible during a normal tide.
My software currently is unaware of spring tides and does not take this into consideration when making and automatic allocation

8.2.2  **Vessels sharing buffers**

The software currently takes into consideration that a vessel requires space at the bow and stern for safety reasons and to allow space for mooring lines. This is a system wide setting of an integer value that is added to the required berth length when making allocations. This value need to be calculated for each vessel in the system. The required space between vessels may be determined by the vessels length or it may be a minimum value set by the shipping agent.

Vessels may also share this safety space allowing mooring lines to cross, this was not implemented in the software meaning automatic allocations will have wasted space between vessels.

8.2.2.1  **Draft clearance needs to be stored against a ship**

Vessels are required to have a minimum amount of clearance between the draft of the berth and the draft of the vessel; this is referred to as under keel clearance. This is currently a system wide setting in my software. This should be related to the vessel and is normally set by the shipping agent.

As the software is currently not calculating this correctly vessels may be allocated to berths with insufficient under keel clearance causing potential safety concerns.

8.2.2.2  **Fending**

Fenders are required to protect the side of vessels against the quay wall; these are large floating bumpers positioned against the side of the quay. At the port of Southampton these are moved up and down the key as needed as there is insufficient fendering to cover the entire quay. When allocations are made currently staff at the port will check there are sufficient fenders free for an allocation.

The software is currently unaware of fenders and does not take them into consideration when making allocations. In its current state, automatic allocations may be made where insufficient fenders are available for the vessel.

8.2.2.3  **Too simplistic model to be adopted by the port**

It was decided that the simplified model used meant the software could not be adopted by the port in its current state even with the other faults fixed. The software would need access to far more information about the ports layout and the cargo on each vessel. By using the model port as a basis for allocation these extra constraints where not taken into consideration. David explained a few of these constraints but this list is not exhaustive;
• Certain shipping agents will pay the more to have priory access to a berth, when making allocations berths should be kept free for expecting vessels with priority even if another vessel will arrive before it.

• The port is used by vehicle carriers; these vessels carry hundreds of vehicles at a time. These are driven onboard and off the vessel. Each car will have a maximum mileage that it can reach during transport (called its delivery mileage). It may be required to take this mileage into account when making an allocation as this would affect the distance the vehicles have to travel.

### 8.2.2.4 Berthing Officers database should be used to aid allocations

It was mentioned that the Berthing Officers who are responsible for overseeing the berthing process of a vessel keep records of all previous vessel allocations for the port. It was suggested that these records could be used when making an automatic allocation.

### 8.2.2.5 Implementation may be better suited at another port

It was noted during feedback from the port of Southampton that the software may be better suited to a smaller green field’s port where there is more structure in the port layout. Some of the criticism in section 8.2.2.3 could be satisfied at other ports that have more set structures in the allocation process.

### 8.2.3 Positive feedback received

Along with criticism, positive feedback was also received from the port of Southampton. The solution was based on a model port that resembled the port of Southampton; this meant it was unable to attempt allocations for the port of Southampton. However for the simplified model they were impressed the application could make automatic allocations. It was therefore suggested that for a smaller more standardised port the software may be able to attempt automatic allocations for vessels in its current state.

Positive comments were also made about the visualisation of the port displayed on the main form and the ability to view the allocations on the quay for a given point in time.
9 Conclusion

It seems clear from the report that the project succeeded in some areas and struggled in others. A better understanding of the problem was achieved and research was conducted into how vessels are allocated space on the quay at Southampton currently. This involved a better understanding of how their current system PAVIS is used to aid this and its areas where it could be improved. Research was also conducted on available port management software packages that can be purchased off the shelf and a visit to a port using the software to see it in action was made.

I successfully designed and developed a proof of concept to show software can be used to aid the allocation process and even make automatic allocations based on a simplified model of the port. I visited the port of Southampton and demonstrated the software to gain feedback.

The area that the project struggled was when it came to getting feedback from Southampton. This should have happened much sooner in the project; attempts were made to gain feedback earlier in the project but could not be arranged due to the busy work schedule of the staff. Feedback was obtained for the solution and fair criticism was made about failings with the software. It is my belief that if the time was available most of these failings could be rectified with another iteration during development.

An original aim of the project was to compare the outcomes of the algorithm with historical or live data of port allocations. An agreement could not be made to obtain access to previous visit records; therefore this could not be done. A simplified model of the port was developed to eliminate certain variables that were considered too advanced for a proof of concept. This simplified model port also meant it was not possible to compare the algorithm outcomes with historical allocations.

It was hoped that the model port would be developed into a more sophisticated version that would more accurately represent the port of Southampton, due to time constraints this was not achieved as intended.

It was also hoped that the algorithm would be developed to make improved allocation when space on the quay is limited. The current implementation does attempt to move other vessel allocations that have not yet berthed in order to create space. Further development could be made to improve this process.

Section 5.2.2 shows how the original schedule was not stuck to, understanding of the problem took longer than expected and therefore delayed the development of the solution. Although considerable time was spend of designing and implementing the solution it became clear that to achieve a
solution that could make fully automatic allocations for the port of Southampton would take considerably more time than scheduled.

Improvements to the algorithm and software are covered in section 10; these would be completed given more time to work on the project.
10 Future Extensions
Given extra time and help from the port of Southampton the following extensions are recommended for the software;

10.1 Improvements suggested during feedback
Several improvements where suggested during feedback from the port of Southampton (discussed in section 8.2);

- Awareness of tide data to detect spring tides and adjust draft clearance for allocations
- Ability to share buffers between vessels along the quay
- Under-keel clearance dependant on the vessel
- Awareness of fending and ability to map fender positions along the quay

These improvements could be made to the system and demonstrated once again to the port to gain additional feedback.

10.2 Integration of previous berthing records
It was suggested during feedback from the port of Southampton that the Berthing Officers database of previous allocations could be used to improve the allocation process. The Berthing Officer at the port is responsible for overseeing the berthing of a vessel on the quay after an allocation is made. A record is kept of where every vessel that has previously visited the port has been berthed. This information could be used to improve the allocation process. The software could be adapted to check if a vessel has made a previous visit to the port, if so it could attempt to allocate a vessel to a section of quay that it has previously berthed at. If a vessel has previously berthed at a section of quay and has been recorded by the Berthing Officer, it would be safe to assume that the vessel could be berthed there again. How feasible this would be would depend on how the berthing records are kept and if they could be converted into a format that could be used by the software. Use of this information may help to solve the criticism discussed in section 8.2.2.3 regarding the simplistic model used.

10.3 Replace model port with the port of Southampton
A model port was developed based on the port of Southampton; this was done to eliminate certain variables. This model could be extended and other port variables could be introduced. This could include;
• Inclusion of quays which are behind a dock; currently the system is not capable of handling quays behind a dock where extra restrictions on vessel length and draft may be to be enforced.

• Vessel cargo could be recorded in the system and used when making automatic allocations. For instance if the port wished to group certain trades together on the quay.

10.4 Improvement of allocation algorithm from testing against port data
If the model port was adapted to better represent the port of Southampton, historical or current port data could be used to test the allocation algorithm. This would be dependent on gaining access to port data which was not possible during this project. Testing of the allocation algorithm alongside current allocations that are made at the port could lead to improvements in the allocation process.

10.5 Integrations with other port systems
The port currently has both radar and transponder systems to allow the tracking of vessels in the port. These systems could be integrated so the system was aware of the actual real time location of vessels within the port. This data could be used to aid the allocation process if making allocation changes for vessels already in the port. If the system had access to real time location data for a vessel, it could use this information to make a new automatic allocation that kept the vessels travelling distance down to a minimum. This information could also be used to improve the visualisation of the port. It could be improved to show the real time location of vessels as well as their allocations. It could also be improved to show better poisoning information about the vessel when berthed, such as its direction against the quay.
11 Bibliography

http://www.southamptonvts.co.uk/ [20th April 2010]

[2] Associated British Ports Southampton Overview
http://www.abports.co.uk/custinfo/ports/soton.htm [15th April 2010]

http://www.southamptonvts.co.uk/com/bandf.htm [20th April 2010]

[4] Tilbury Container Services Background
http://www.tcsonline.co.uk/aboutTCS.asp [20th April 2010]

[5] Tilbury Container Services Terminal Equipment and Services
http://www.tcsonline.co.uk/equipment.asp [20th April 2010]


12 Appendix

A. Personal reflection

I originally decided to undertake a project about the allocation process in the port of Southampton as it became of interest to me during my year in industry when I visited the port. It is for this reason why I found the project incredibly interesting, I had spent the past year working on the PAVIS software they currently use and considered this an improvement to it.

I enjoyed talking to Southampton and gaining a better understanding of the allocation process they used. I also thoroughly enjoyed the visit to the port of Tilbury; this was an interesting experience that allowed me to see a port outside of the Associated British Ports group.

After my year in industry a career developing software for the ports industry seemed an interesting option and this project allowed me to continue my experience and knowledge of the industry. I had hoped that I could produce something that the port of Southampton would continue to develop, and whilst this is a possibility it seems unlikely.

Another area of knowledge I improved was that of scheduling, this is something I had little or no experience of before the project. It was interesting to research existing scheduling methods then applying these to designing my own for the problem at hand.

Looking back at the project there are areas I wish I had done differently, the first being attempting to get support from the port of Southampton much earlier. I under estimated how long this would take, even with the contacts from my time there. I chose the port of Southampton due to the complications of the allocation process and the interest that had been expressed about improving the process at the port. Due to its proximity to Leeds it expensive and difficult to visit the port, as travel costs and accommodation had to be paid for. I was willing to pay this for the project but it limited the time I could spend at the port. Had I chosen another location closer I may have had the opportunity to spend more time at the port gaining a better understanding of the problem.

I expected to get more help from existing software companies that produces port management software packages. I attempted to contact every company I found and received no help, there was no surprise at the port of Tilbury or Southampton when this was mentioned, indicating that I was not likely to. Although I do not think I could have done much more to obtain help it would have been beneficial to gain at least one company’s help.

If asked to give advice to any students considering a project I would advise them to ensure it is based on a subject they are interested in. It becomes difficult to manage the project alongside other final
year commitments and having something you are interested in makes this a lot easier. I would also advise them that if they are dealing with an external company to ensure they secure the help, software and any data they need upfront and in plenty of time. Even with contacts at the company it’s very easy to underestimate the time this will take and overestimate the time people are able to set aside for you.
B. PAVIS software
C. Scheduler software

Screenshots of the scheduler software used at Tilbury Container Services
D. **SPARCS software**

Screenshots of the SPARCS software observed at Tilbury Container Services
Email correspondence with existing software companies

The following email was sent to Harbour Mastery, IBS, Zebra Enterprise Solutions and Tideworks Technology on the 10/11/2009 but no reply was received from either company.

Dear Harbour Mastery,

I am currently studying Computing and Business Management at the University Of Leeds, United Kingdom. I am performing research into how Artificial Intelligent could be used within a port to aid services and resource allocation to vessels.

Would you be able to provide any more information on your i-Seaport Management product? I can see from your website that this is a web based product. Do you provide a way for customers to try this software? If so could I please request access to it for my research?

Thank you for your time

Edward Jones
Js06ej@leeds.ac.uk
Tel: 004407545536796
School of Computing
University Of Leeds
Leeds
LS2 9JT
UK
Hi Edward,

I am well thanks and hope that the same is true of you.

Your project sounds like an interesting one. We had looked at this area before in Southampton where berth allocation was, and I think even in these difficult times, still is. (There was a company that we were talking to that specialised in this area, when I recall who they were I send along some details.)

As well as the fundamental, how much time did the vessel need on the berth, there was also the complication of how much water would be under the vessel during the entire load/unload operation, this before we even get to availability of shoreside resources.

There has been a start in this area with the current release of PAVIS which has delivered the graphical berth planner. This functionality is basic and relies on the user to place the vessel where they think best. This decision may be the result of a sophisticated albeit manual process, or just plain old experience.

HIT could be considered. HIT is a simpler 2 berth operation at Immingham which does suffer from demurrage claims, but I am not sure that we need an algorithm for this one, but you would be welcome to take a look.

Looking at your requirements I would recommend Southampton. Let me know and I will get you introduced to the Harbour Master.

Regards

Eddy
I am back at university now and have decided to do a final year project. I would like to do a project about optimising port management and was hoping you would be able to help me.

I would like to look at how berth allocation and resource allocation could be improved using artificial intelligence / scheduling algorithms. Would you be able to suggest an ABP port that would be suitable for this, perhaps a port with high traffic and limited berth space?

I release there are restriction on sensitive data / the data protection act but if I could be allowed access to any traffic information (even if it is historical) that would be a big help?

Thanks Edward
G. Map of the port of Southampton
This map shows the model port that was designed. Blue area denotes water.
# I. Original schedule

Gantt chart showing the initial schedule for the project

<table>
<thead>
<tr>
<th>Task Name</th>
<th>Start</th>
<th>Finish</th>
<th>Duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Project initiation</td>
<td>1/1/2020</td>
<td>1/1/2020</td>
<td>1d</td>
</tr>
<tr>
<td>2. Define the problem</td>
<td>2/1/2020</td>
<td>2/1/2020</td>
<td>1d</td>
</tr>
<tr>
<td>3. Research and Literature</td>
<td>3/1/2020</td>
<td>3/1/2020</td>
<td>1d</td>
</tr>
<tr>
<td>4. Develop software prototype</td>
<td>4/1/2020</td>
<td>4/1/2020</td>
<td>1d</td>
</tr>
<tr>
<td>5. Software testing</td>
<td>5/1/2020</td>
<td>5/1/2020</td>
<td>1d</td>
</tr>
<tr>
<td>7. Software evaluation</td>
<td>7/1/2020</td>
<td>7/1/2020</td>
<td>1d</td>
</tr>
<tr>
<td>8. Software maintenance</td>
<td>8/1/2020</td>
<td>8/1/2020</td>
<td>1d</td>
</tr>
<tr>
<td>9. Software update</td>
<td>9/1/2020</td>
<td>9/1/2020</td>
<td>1d</td>
</tr>
<tr>
<td>10. Project closure</td>
<td>10/1/2020</td>
<td>10/1/2020</td>
<td>1d</td>
</tr>
</tbody>
</table>
J. Actual schedule

Gantt chart showing the actual schedule followed.
K. Initial user interface design for main form
L. Initial user interface design for maintaining vessels and visits

Form to allow user to view a list of vessels in the system
Form to allow users to edit the details of a vessel stored in the system
M. Initial user interface design for maintaining allocations
N. Initial database design

```plaintext
<table>
<thead>
<tr>
<th>Table</th>
<th>PK</th>
<th>Type</th>
<th>Columns</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vessel</td>
<td>VESSEL_ID</td>
<td>INTEGER</td>
<td>VESSEL_NAME, VESSEL_LOA, VESSEL_BEAM, VESSEL_DEPTH, DATE_ADDED</td>
</tr>
<tr>
<td>Visit</td>
<td>VISIT_ID</td>
<td>INTEGER</td>
<td>VESSEL_ID, ESTIMATED_TIME_OF_ARRIVAL, ACTUAL_TIME_OF_ARRIVAL, ESTIMATED_TIME_OF_DEPARTURE, ACTUAL_TIME_OF_DEPARTURE, VISIT_NOTES</td>
</tr>
<tr>
<td>BerthAllocation</td>
<td>BERTH_ALLOCATION_ID</td>
<td>INTEGER</td>
<td>VISIT_ID, BERTH_ID, ALLOCATION_START_TIME, ALLOCATION_END_TIME, STATE</td>
</tr>
<tr>
<td>Port</td>
<td>PORT_ID</td>
<td>LONG</td>
<td>PORT_NAME, PORT_DESCRIPTION</td>
</tr>
<tr>
<td>Quay</td>
<td>QUAY_ID</td>
<td>INTEGER</td>
<td>QUAY_NAME, QUAY_LENGTH, QUAY_DESCRIPTION</td>
</tr>
<tr>
<td>Berth</td>
<td>BERTH_ID</td>
<td>INTEGER</td>
<td>QUAY_ID, BERTH_NAME, BERTH_LENGTH, BERTH_DEPTH, BERTH_START_POS, BERTH_END_POS</td>
</tr>
</tbody>
</table>
```

Diagram:

- Vessel
- Visit
- BerthAllocation
- Port
- Quay
- Berth
0. The user interface

This shows the initial main form design
P. LINQ generated classes

Screenshot of the LINQ generated classes in Visual Studio 2008
Q. Code snippet of additional functionality added to generated classes

```
Partial Public Class BerthAllocation

    Public Function AllocationLength() As Integer
        Return Me.QUAY_POSITION_END - Me.QUAY_POSITION_START
    End Function

    Public Function _Berths() As List(Of Berth)
        'Loop through the berths on the quay and see if the allocation covers any
        Dim objBerths As New List(Of Berth)
        For Each objBerth As Berth In (From b In g_objDBConnection.Berths Where b.QUAY_ID = Me.QUAY_ID)
            If DoesAllocationCoverBerth(objBerth, Me) Then
                objBerths.Add(objBerth)
            End If
        Next
        Return objBerths
    End Function

    Public Function IsValid() As Boolean
        Me.m_objValidationErrors = New List(Of ValidationError)
        If Not Me.QUAY_ID > 0 AndAlso Me.Quay Is Nothing Then
            m_objValidationErrors.Add(New ValidationError("Please select a quay."))
        End If
        If Me.QUAY_POSITION_START = Integer.MinValue Then
            m_objValidationErrors.Add(New ValidationError("Position Start is required"))
        End If
        If Me.QUAY_POSITION_END = Integer.MinValue Then
            m_objValidationErrors.Add(New ValidationError("Position End is required"))
        End If
        If Me.QUAY_POSITION_END < Me.QUAY_POSITION_START Then
            m_objValidationErrors.Add(New ValidationError("Position end cannot be before position start."))
        End If
        If Me.ALLOCATION_START_TIME = Date.MinValue Then
            m_objValidationErrors.Add(New ValidationError("Start time is not valid"))
        End If
        If Me.ALLOCATION_END_TIME = Date.MinValue Then
            m_objValidationErrors.Add(New ValidationError("End time is not valid"))
        End If
        If Not Me.ALLOCATION_START_TIME = Date.MinValue AndAlso Not Me.ALLOCATION_END_TIME = Date.MinValue Then
            If Me.ALLOCATION_END_TIME < Me.ALLOCATION_START_TIME Then
                m_objValidationErrors.Add(New ValidationError("End time cannot be before start time."))
            End If
        End If
        Return m_objValidationErrors.Count = 0
    End Function
```
R. Main form showing data
**S. Unit test code logic**

```vbscript
<TestMethod()> _
Public Sub InsertUpdateDeleteBerthAllocationTest()
    Dim objDbConnection As New PADataDataContext("Data Source=SERVER-SS2;Initial Catalog=PORT_ALLOCATOR;Integrated Security=True")
    Dim dtmAllocationStartTime As Date = Now
    Dim dtmAllocationEndTime As Date = Now.AddHours(12)
    Dim intPositionStart As Integer = 10
    Dim intPositionEnd As Integer = 100
    Dim intQuayId As Integer = 2
    Dim intVisitId As Integer = 65
    Dim enuState As g_enuBerthAllocationState = g_enuBerthAllocationState.Berthed
    Dim intCreatedID As Integer
    Dim target As BerthAllocation = New BerthAllocation
    target.ALLOCATION_START_TIME = dtmAllocationStartTime
    target.ALLOCATION_END_TIME = dtmAllocationEndTime
    target.QUAY_ID = intQuayId
    target.VISIT_ID = intVisitId
    target.STATE = enuState
    objDbConnection.BerthAllocations.InsertOnSubmit(target)
    objDbConnection.SubmitChanges()
    intCreatedID = target.BERTH_ALLOCATION_ID
    Dim expected As BerthAllocation
    expected = (From a In objDbConnection.BerthAllocations Where a.BERTH_ALLOCATION_ID = intCreatedID).FirstOrDefault
    Assert.IsNotNull(expected)
    Assert.AreEqual(expected.ALLOCATION_START_TIME, dtmAllocationStartTime)
    Assert.AreEqual(expected.ALLOCATION_END_TIME, dtmAllocationEndTime)
    Assert.AreEqual(expected.QUAY_ID, intQuayId)
    Assert.AreEqual(expected.VISIT_ID, intVisitId)
    Assert.AreEqual(expected.STATE, enuState)
    dtmAllocationStartTime = Now.AddHours(-5)
    dtmAllocationEndTime = Now.AddHours(24)
    intPositionStart = 50
    intPositionEnd = 250
    intQuayId = 3
    intVisitId = 66
    enuState = g_enuBerthAllocationState.AllocationWaiting
    expected.ALLOCATION_START_TIME = dtmAllocationStartTime
    expected.ALLOCATION_END_TIME = dtmAllocationEndTime
    expected.QUAY_ID = intQuayId
    expected.VISIT_ID = intVisitId
    expected.STATE = enuState
    objDbConnection.SubmitChanges()
    expected = Nothing
    expected = (From a In objDbConnection.BerthAllocations Where a.BERTH_ALLOCATION_ID = intCreatedID).FirstOrDefault
    Assert.IsNotNull(expected)
    Assert.AreEqual(expected.ALLOCATION_START_TIME, dtmAllocationStartTime)
    Assert.AreEqual(expected.ALLOCATION_END_TIME, dtmAllocationEndTime)
    Assert.AreEqual(expected.QUAY_ID, intQuayId)
    Assert.AreEqual(expected.VISIT_ID, intVisitId)
    Assert.AreEqual(expected.STATE, enuState)
    objDbConnection.BerthAllocations.DeleteOnSubmit(expected)
    objDbConnection.SubmitChanges()"
```
T. Revised database design

```
Vessel
PK VESSEL_ID INTEGER
VESSEL_NAME TEXT(50)
VESSEL_LOA INTEGER
VESSEL_BEAM DECIMAL(10,2)
VESSEL_DEPTH DECIMAL(10,2)
DATE_ADDED DATETIME

Visit
PK VISIT_ID INTEGER
FK1 VESSEL_ID INTEGER
ESTIMATED_TIME_OF_ARRIVAL DATETIME
ACTUAL_TIME_OF_ARRIVAL DATETIME
ESTIMATED_TIME_OF_DEPARTURE DATETIME
ACTUAL_TIME_OF_DEPARTURE DATETIME
VISIT_NOTES TEXT(0)

Port
PK PORT_ID LONG
FK1 VESSEL_ID INTEGER
PORT_NAME TEXT(50)
PORT_DESCRIPTION TEXT(0)

Quay
PK QUAY_ID INTEGER
FK1 PORT_ID INTEGER
QUAY_NAME TEXT(50)
QUAY_LENGTH TEXT(50)
QUAY_DESCRIPTION TEXT(500)

BerthAllocation
PK BERTH_ALLOCATION_ID INTEGER
FK1 VESSEL_ID INTEGER
FK2 QUAY_POSITION_START INTEGER
QUAY_POSITION_END INTEGER
VISIT_ID INTEGER
ALLOCATION_START_TIME DATETIME
ALLOCATION_END_TIME DATETIME
STATE INTEGER
QUAY_ID INTEGER

Berth
PK BERTH_ID INTEGER
FK1 QUAY_ID INTEGER
BERTH_NAME TEXT(50)
BERTH_LENGTH INTEGER
BERTH_DEPTH DECIMAL(10,2)
BERTH_START_POS INTEGER
BERTH_END_POS INTEGER
```
Schedule output user interface design

<table>
<thead>
<tr>
<th></th>
<th>Date</th>
<th>Time</th>
<th>Duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Event</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Event</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Event</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Event</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Event</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Event</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Event</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Event</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Event</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Event</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Note: All events in schedule.
V. Schedule output

This is the schedule output from the application